

# Binarization, Synchronous Binarization, and Target-side Binarization

Liang Huang

University of Pennsylvania

(part of this work was done at USC/ISI)

SSST workshop, NAACL 2006, Rochester, NY

# Overview of Binarization

parsing

CFG

$S \rightarrow NP VP PP$

machine translation

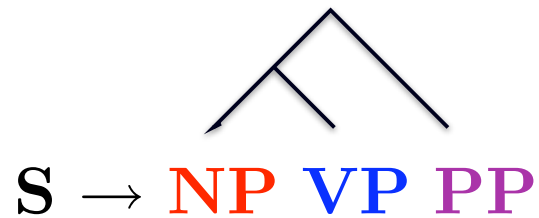
Synchronous CFG

$S \rightarrow \begin{matrix} NP & PP & VP \\ NP & VP & PP \end{matrix}$

# Overview of Binarization

parsing

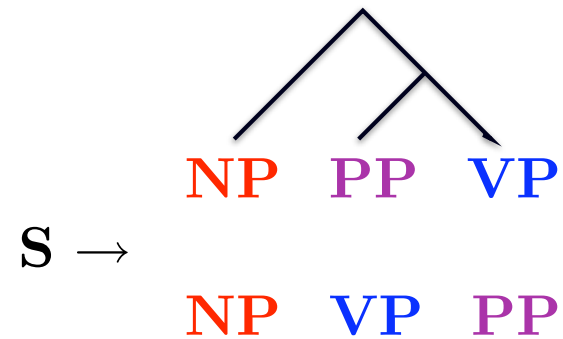
CFG



for cubic time parsing

machine translation

Synchronous CFG

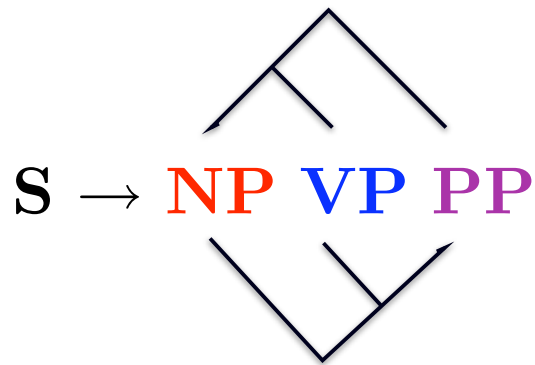


for polynomial time decoding

# Overview of Binarization

parsing

CFG

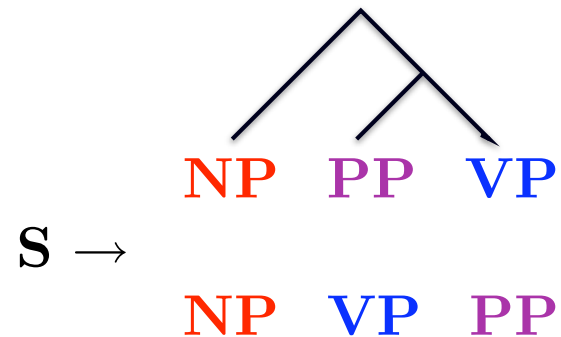


for cubic time parsing

different schemes  
work **equally well**

machine translation

Synchronous CFG

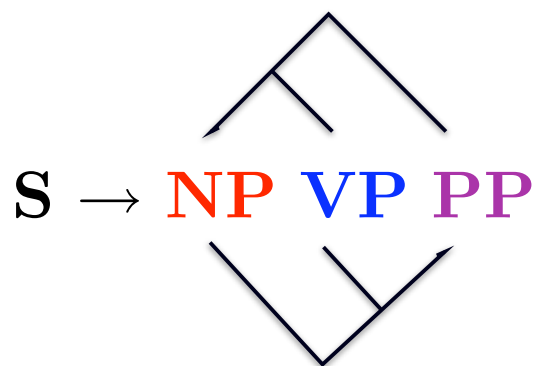


for polynomial time decoding

# Overview of Binarization

parsing

CFG

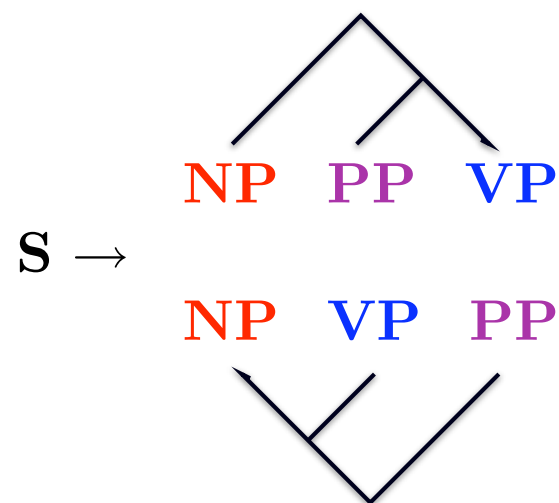


for cubic time parsing

different schemes  
work **equally well**

machine translation

Synchronous CFG



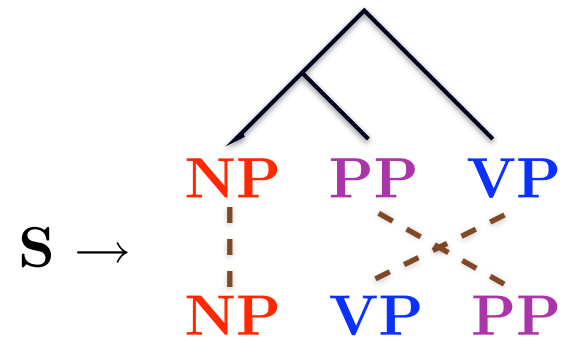
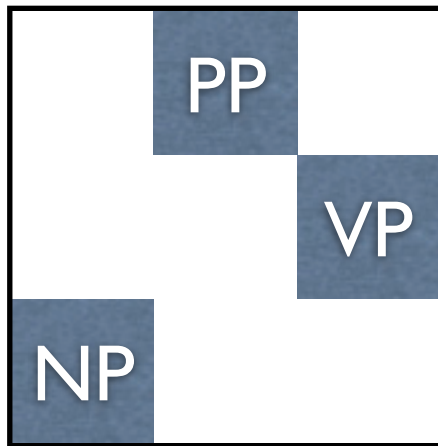
for polynomial time decoding

different schemes  
work **very differently**

# Overview of Binarization

machine translation

Synchronous CFG

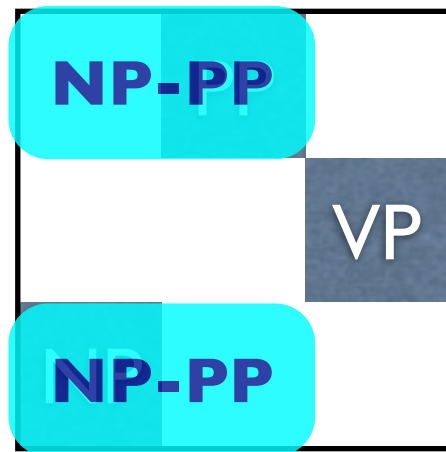


for polynomial time decoding

different schemes  
work **very differently**

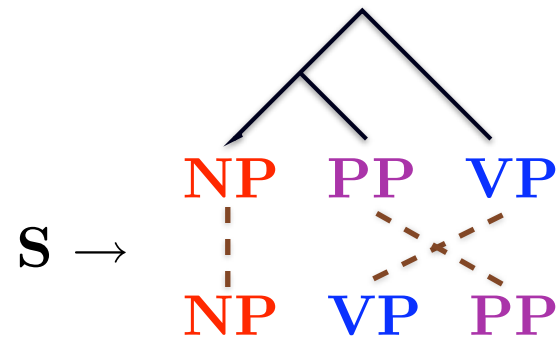
# Overview of Binarization

(naïve) binarization



machine translation

Synchronous CFG



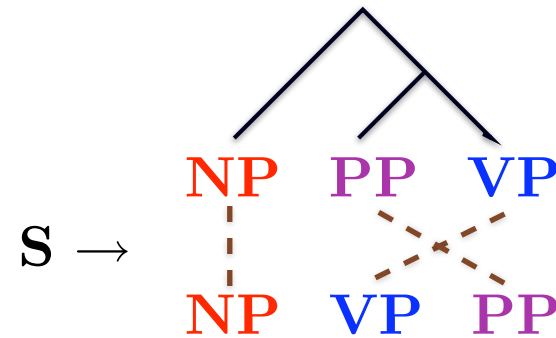
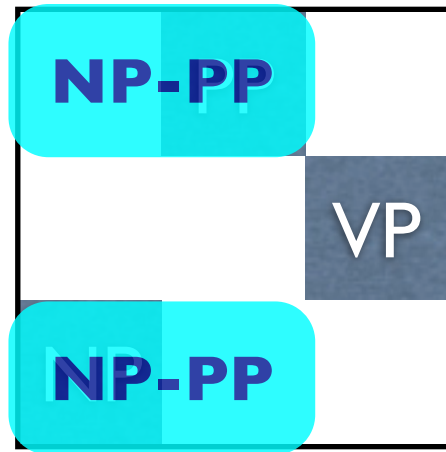
for polynomial time decoding

different schemes  
work **very differently**

# Overview of Binarization

machine translation

Synchronous CFG



for polynomial time decoding

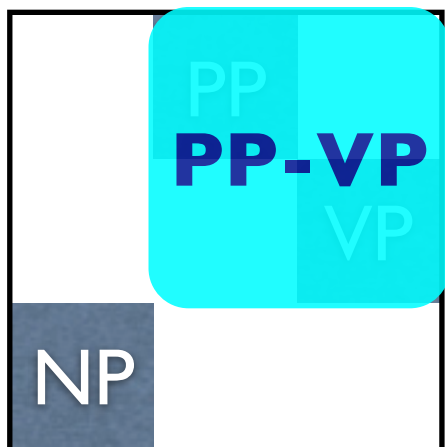
different schemes  
work **very differently**

# Overview of Binarization

synchronous binarization

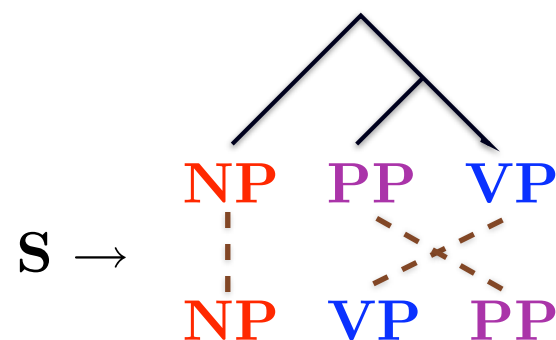
(Zhang, Huang, Gildea, Knight, 2006)

is a principled scheme



machine translation

Synchronous CFG



for polynomial time decoding

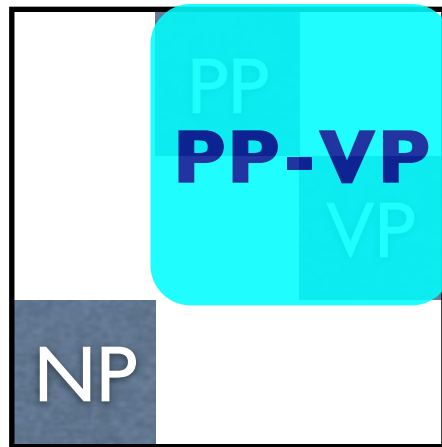
different schemes  
work **very differently**

# Overview of Binarization

synchronous binarization

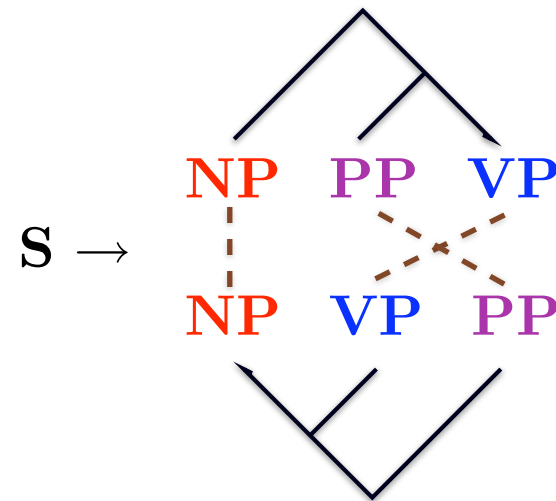
(Zhang, Huang, Gildea, Knight, 2006)

is a principled scheme



machine translation

Synchronous CFG



for polynomial time decoding

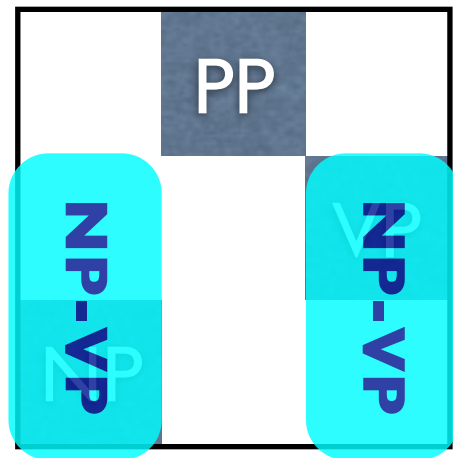
different schemes  
work **very differently**

# Overview of Binarization

synchronous binarization

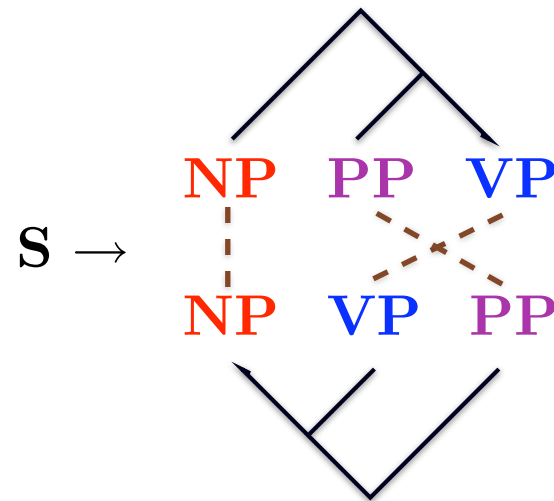
(Zhang, Huang, Gildea, Knight, 2006)

is a principled scheme



machine translation

Synchronous CFG



for polynomial time decoding

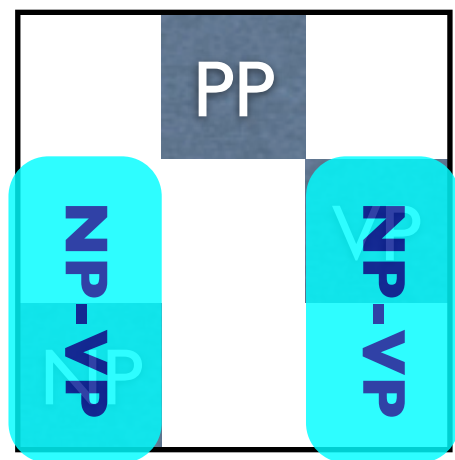
different schemes  
work **very differently**

# Overview of Binarization

synchronous binarization

(Zhang, Huang, Gildea, Knight, 2006)

is a principled scheme

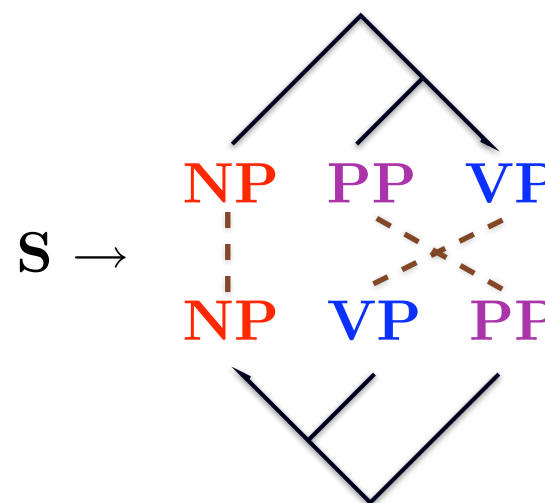


**target-side binarization:**

a simpler alternative  
that works equally well on  
tree-to-string systems

machine translation

Synchronous CFG



for polynomial time decoding

different schemes  
work **very differently**

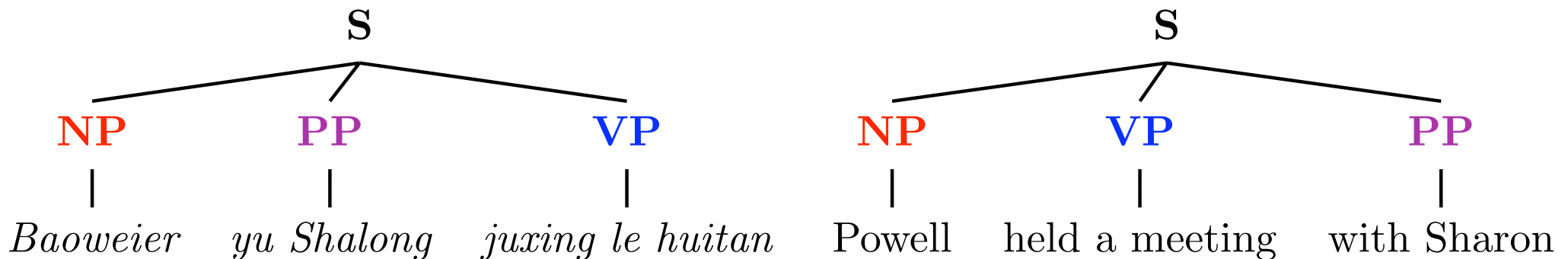
# In this talk ...

- Background: SCFGs and their applications in MT
  - **string-input** and **tree-input** systems
- The Three Binarization Schemes
  - source-side, synchronous, and **target-side** binarizations
- Theoretical Analysis
  - translation as parsing
  - decoding with an integrated language model
- Experiments on Tree-to-String Systems

# Background: SCFGs

- Synchronous Context-Free Grammar (SCFG)
- CFG in two dimensions, generating pairs of trees/strings
- co-indexed nonterminal further rewritten as a unit

**S** → **NP**<sup>(1)</sup> **PP**<sup>(2)</sup> **VP**<sup>(3)</sup>,    **NP**<sup>(1)</sup> **VP**<sup>(3)</sup> **PP**<sup>(2)</sup>  
**NP** → *Baoweier*,    Powell  
**PP** → *yu Shalong*,    with Sharon  
**VP** → *juxing le huitan*,    held a meeting



(Aho/Ullman, 1972)

(Wu, 1997)

(Chiang, 2005)

# Two Uses of SCFGs

**string-input**

(Wu, 1997; Chiang, 2005; Galley et al., 2006)

*Baoweier yu Shalong juxing le huitan*

---

**tree-input** (Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

# Two Uses of SCFGs

string-input

(Wu, 1997; Chiang, 2005; Galley et al., 2006)

**NP**

*Baoweier yu Shalong juxing le huitan*

**NP**

|

Powell

---

tree-input (Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

# Two Uses of SCFGs

string-input

(Wu, 1997; Chiang, 2005; Galley et al., 2006)

NP

PP

NP

PP

*Baoweier yu Shalong juxing le huitan*

Powell with Sharon

---

tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

# Two Uses of SCFGs

string-input

(Wu, 1997; Chiang, 2005; Galley et al., 2006)

NP

PP

VP

*Baoweier yu Shalong juxing le huitan*

NP

PP

VP

Powell with Sharon held a meeting

---

tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

# Two Uses of SCFGs

string-input

(Wu, 1997; Chiang, 2005; Galley et al., 2006)

S

NP

PP

VP

NP

PP

VP

*Baoweier yu Shalong juxing le huitan*

Powell with Sharon held a meeting

---

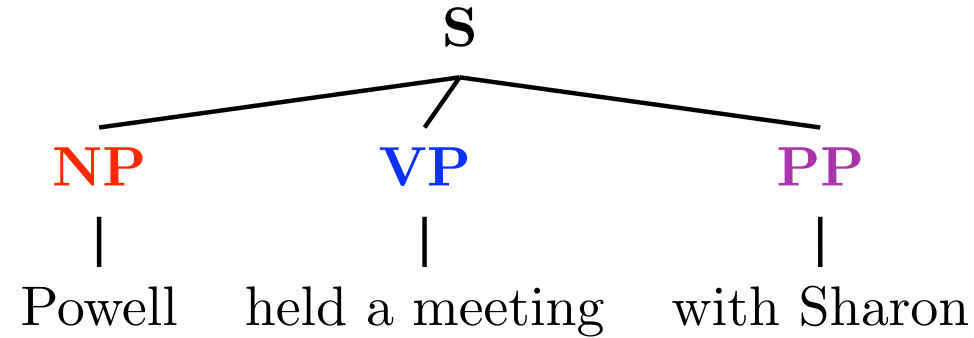
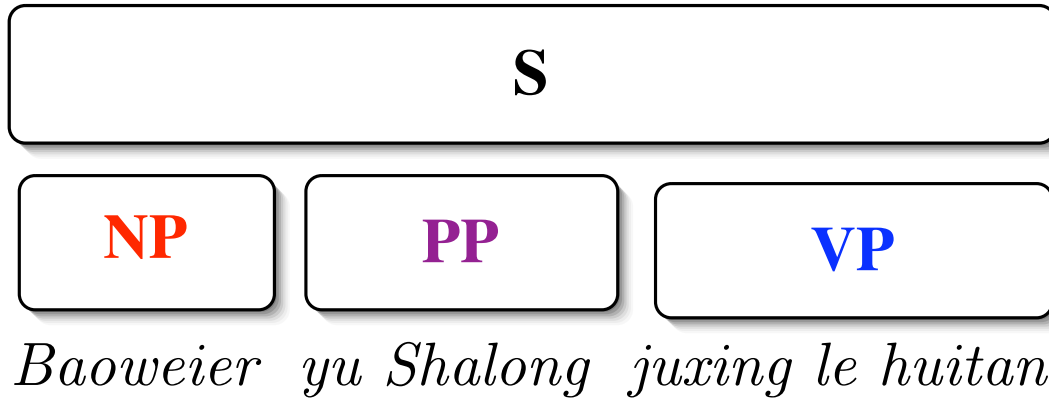
tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

# Two Uses of SCFGs

string-input

(Wu, 1997; Chiang, 2005; Galley et al., 2006)



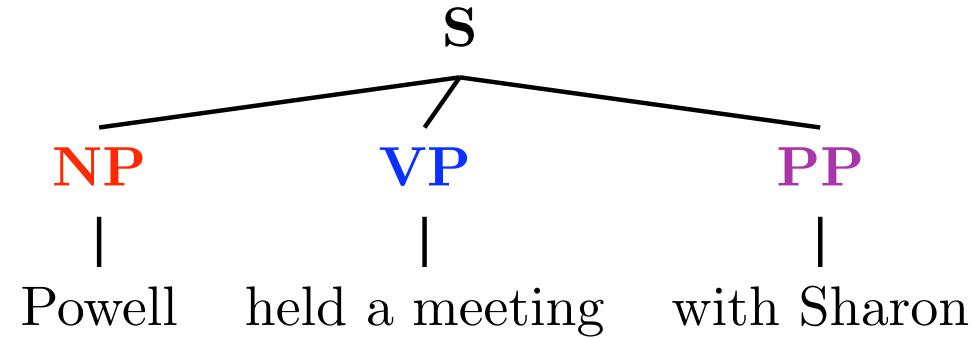
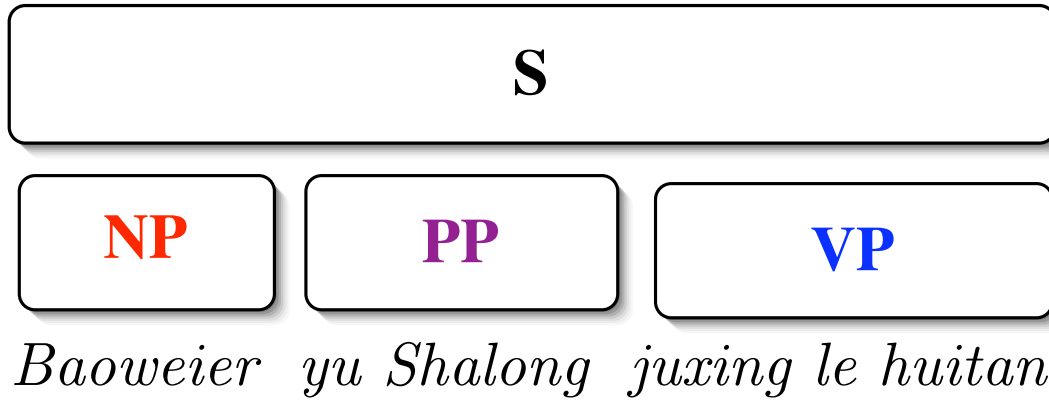
tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

# Two Uses of SCFGs

string-input

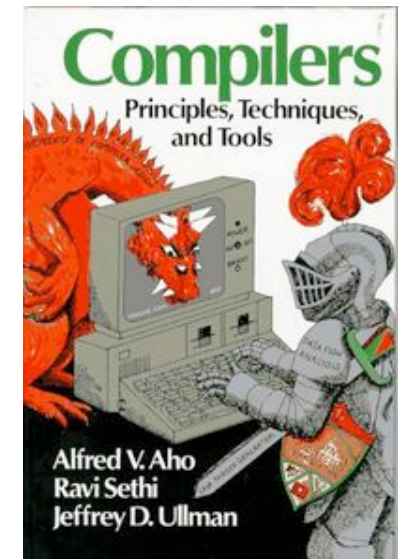
(Wu, 1997; Chiang, 2005; Galley et al., 2006)



tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

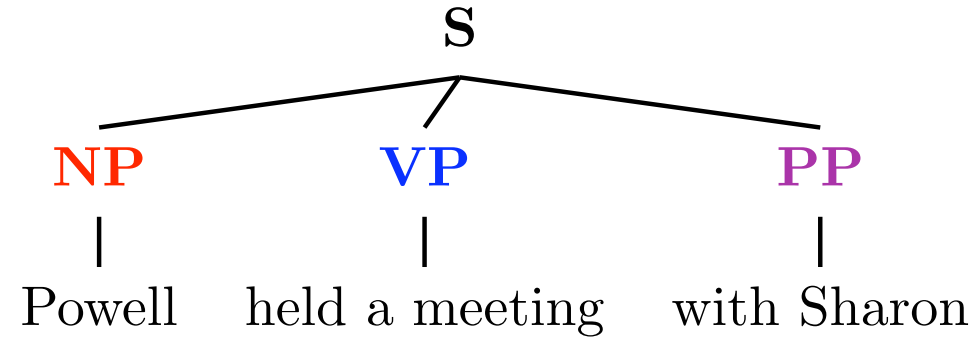
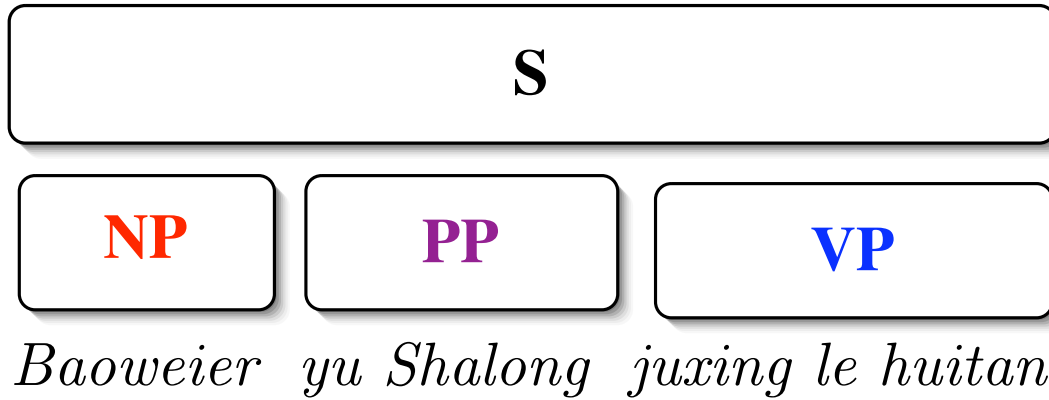
“syntax-directed”



# Two Uses of SCFGs

string-input

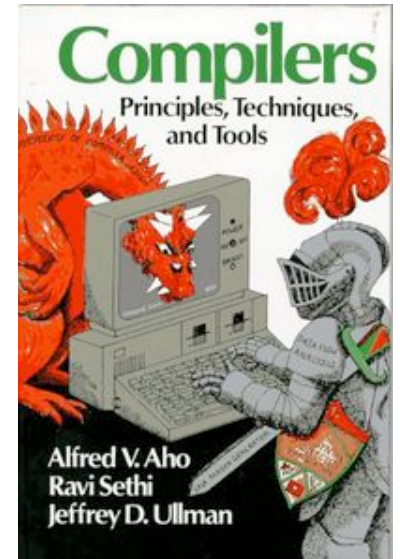
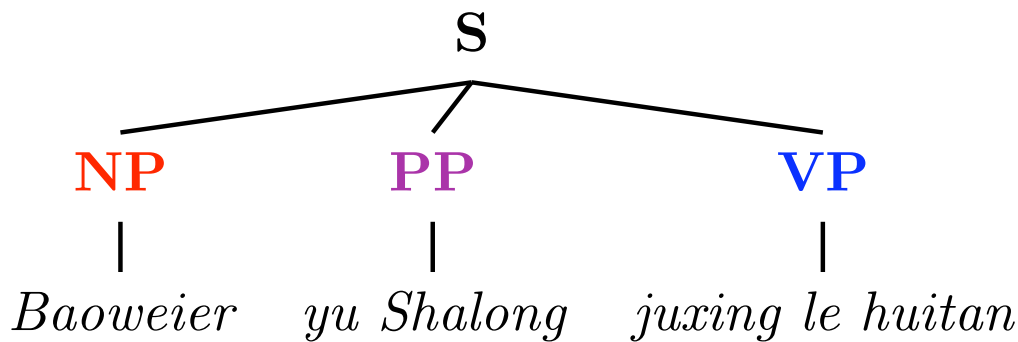
(Wu, 1997; Chiang, 2005; Galley et al., 2006)



tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

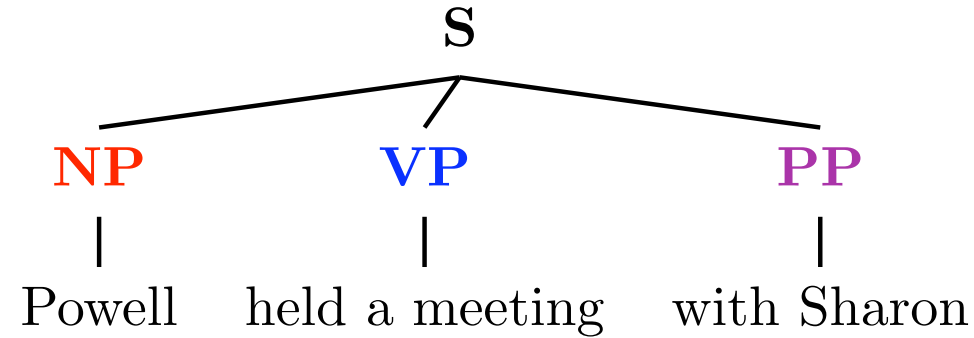
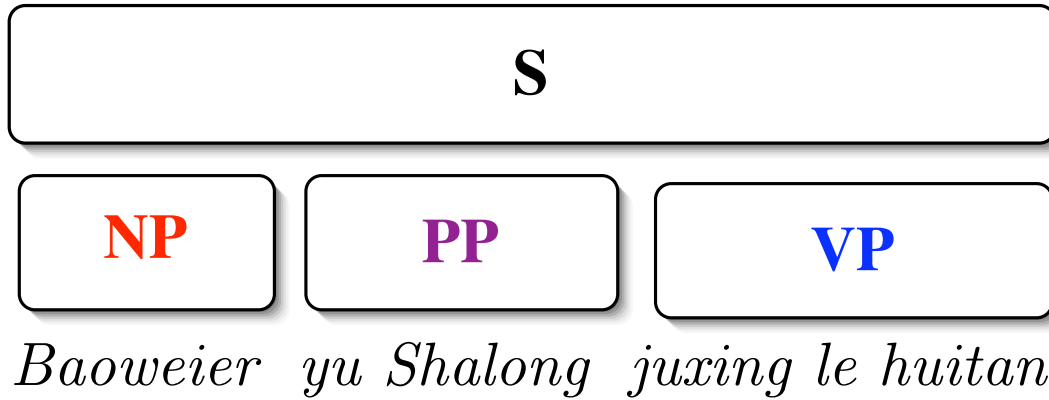
“syntax-directed”



# Two Uses of SCFGs

## string-input

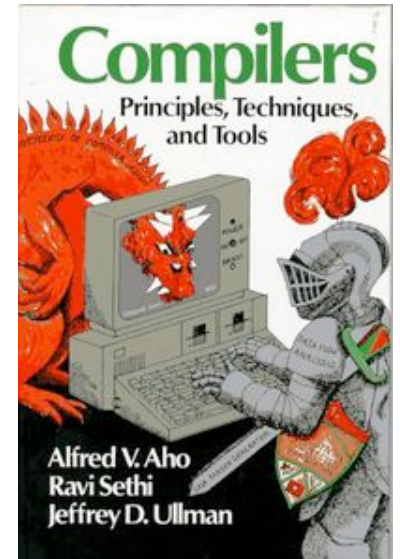
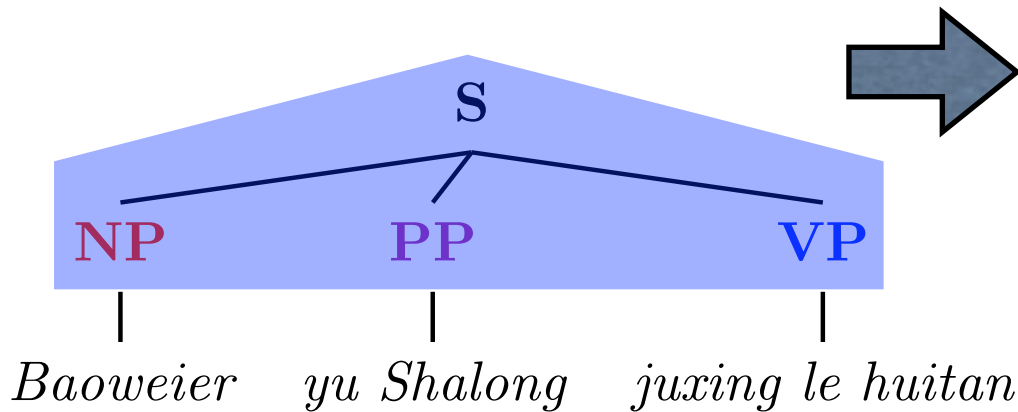
(Wu, 1997; Chiang, 2005; Galley et al., 2006)



## tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

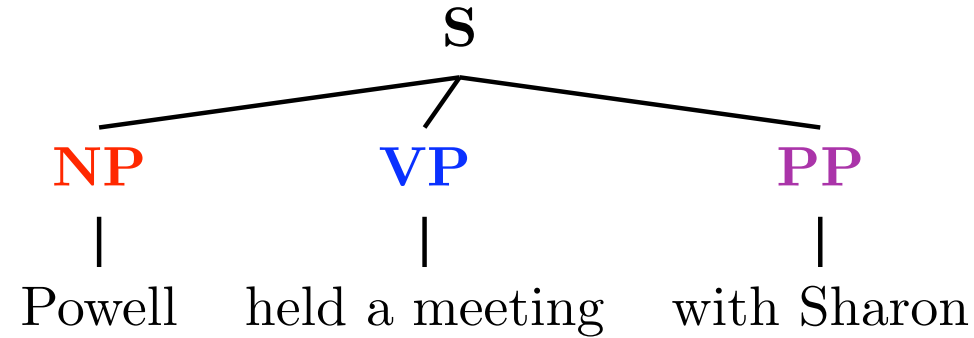
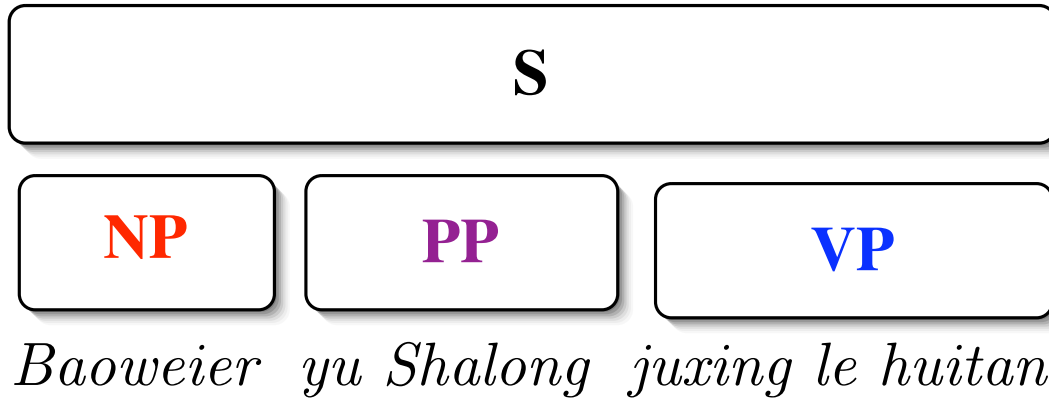
“syntax-directed”



# Two Uses of SCFGs

## string-input

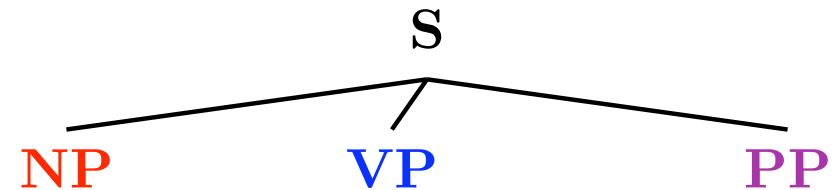
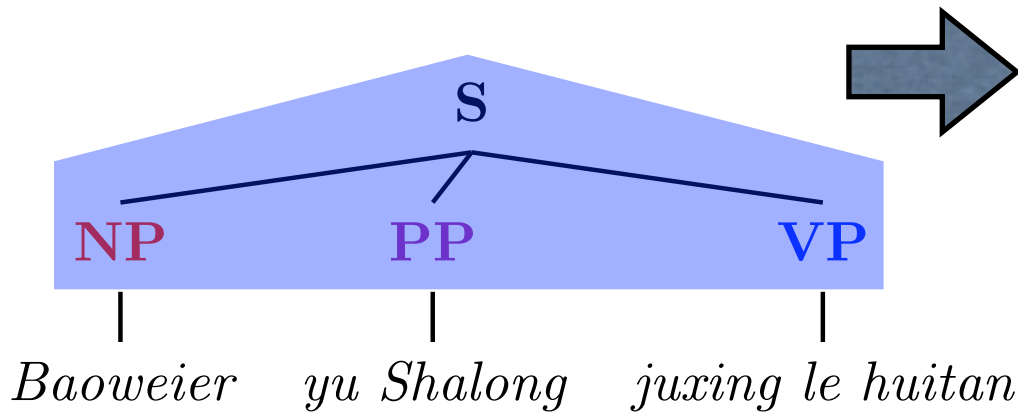
(Wu, 1997; Chiang, 2005; Galley et al., 2006)



## tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

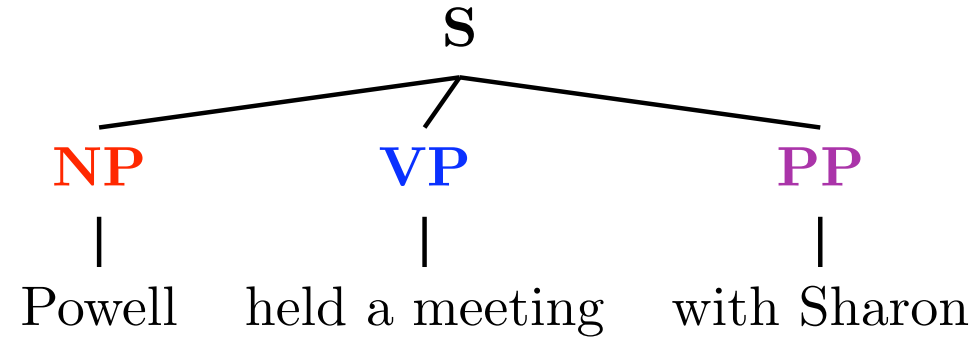
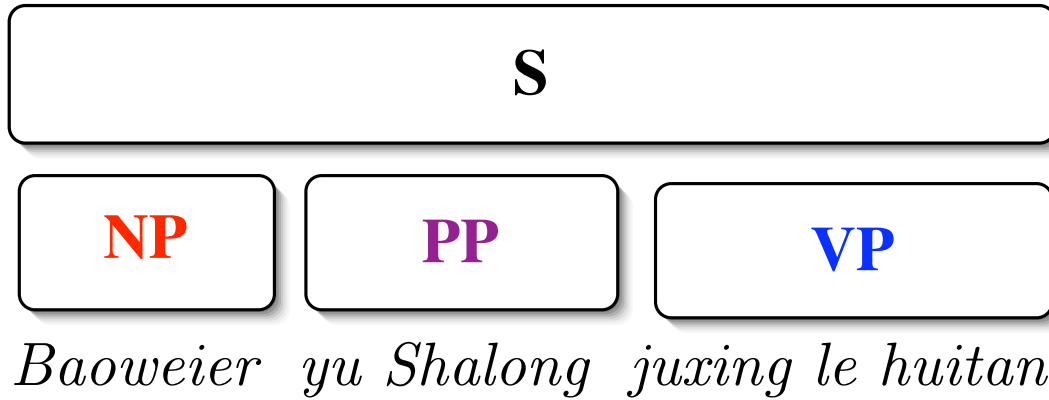
“syntax-directed”



# Two Uses of SCFGs

string-input

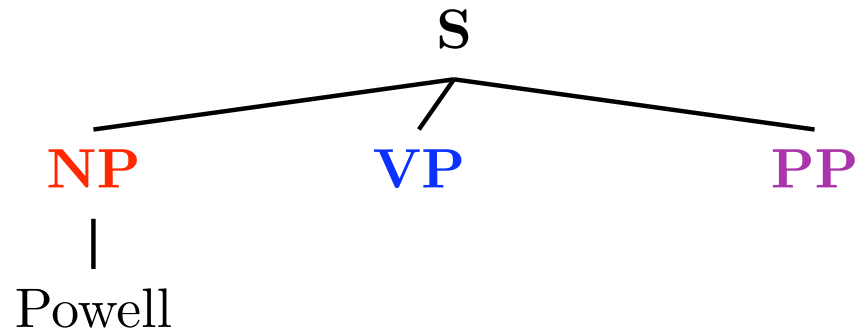
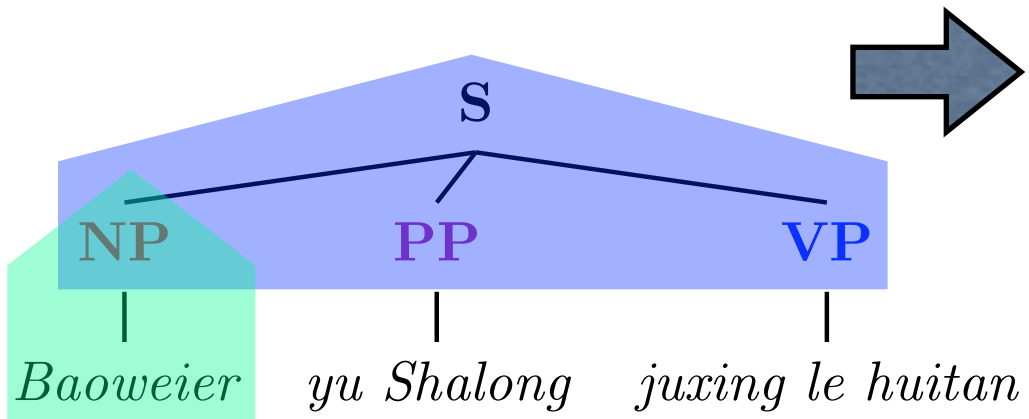
(Wu, 1997; Chiang, 2005; Galley et al., 2006)



tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

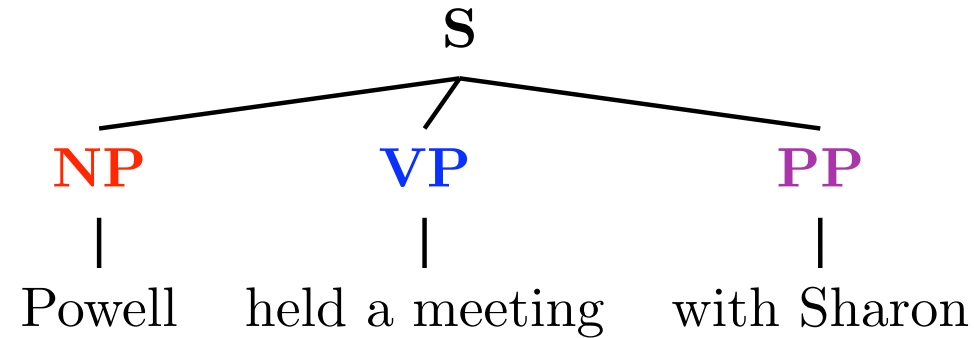
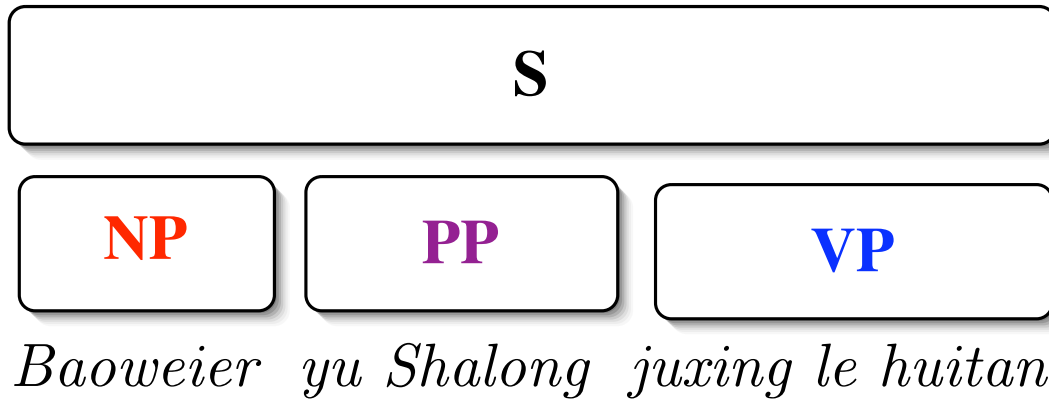
“syntax-directed”



# Two Uses of SCFGs

string-input

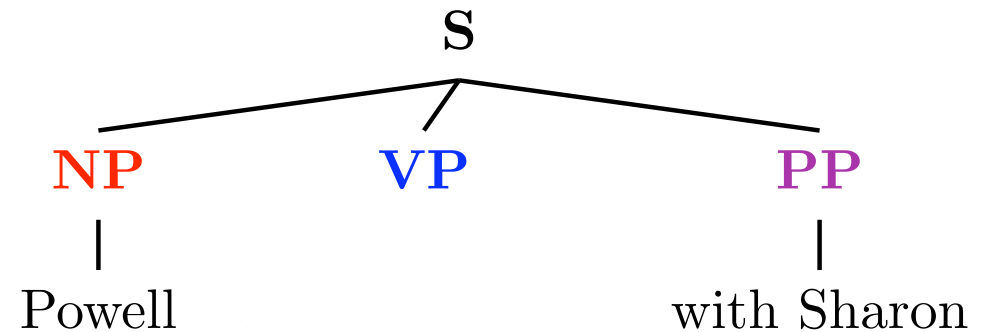
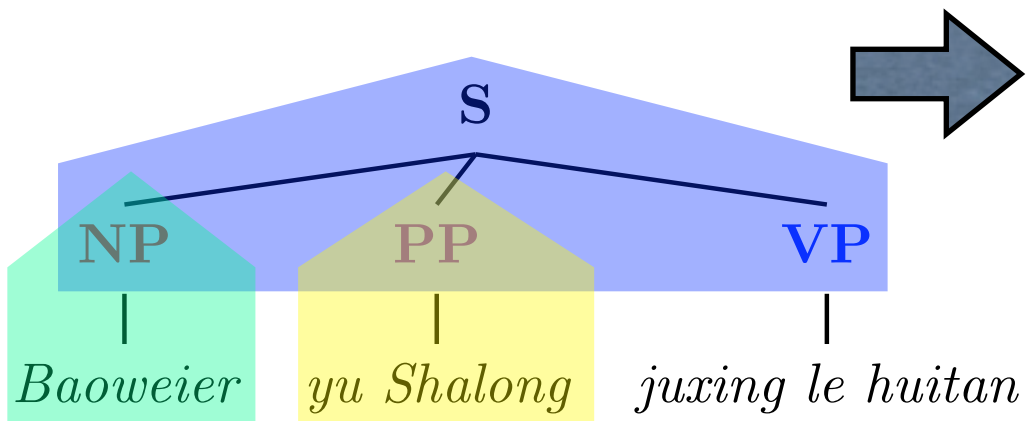
(Wu, 1997; Chiang, 2005; Galley et al., 2006)



tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

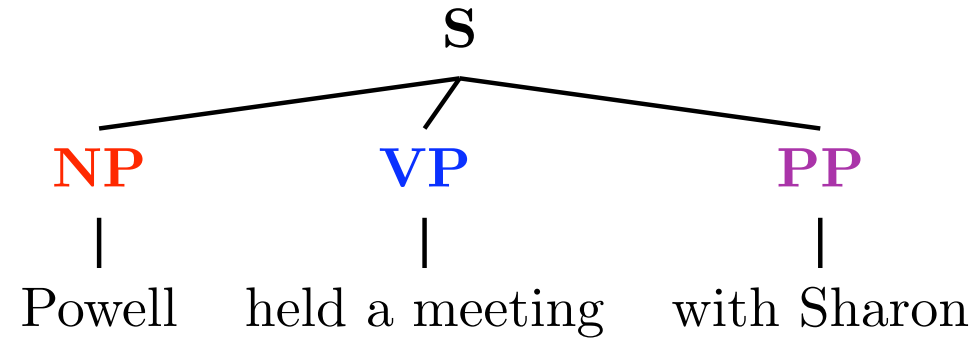
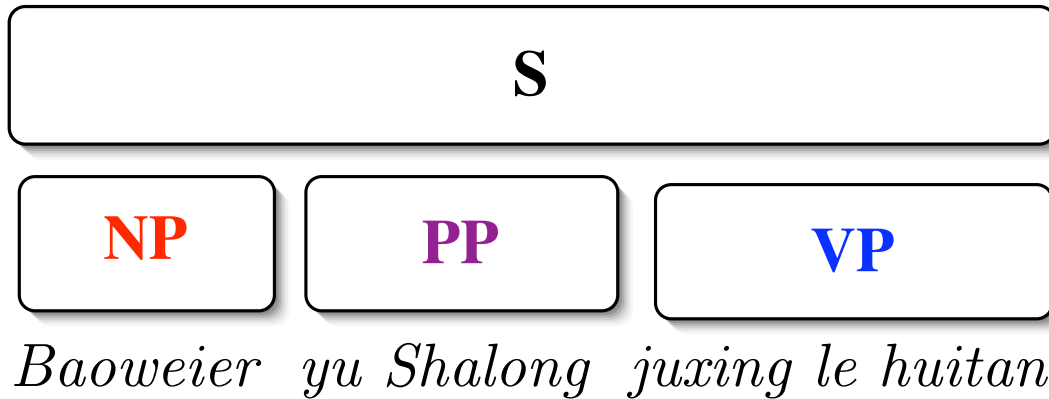
“syntax-directed”



# Two Uses of SCFGs

## string-input

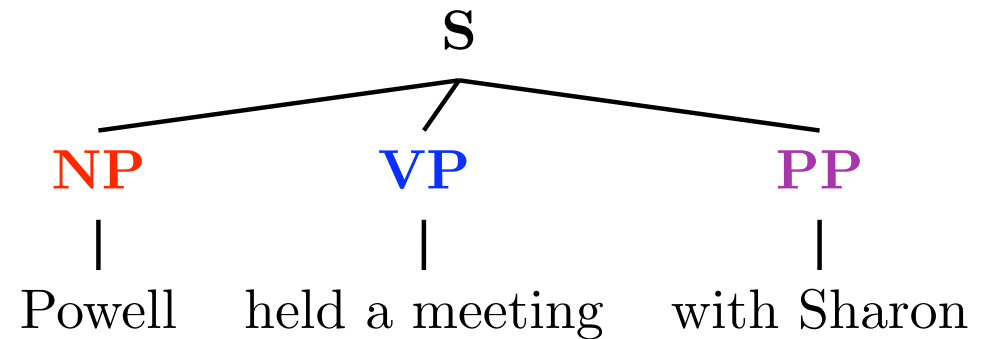
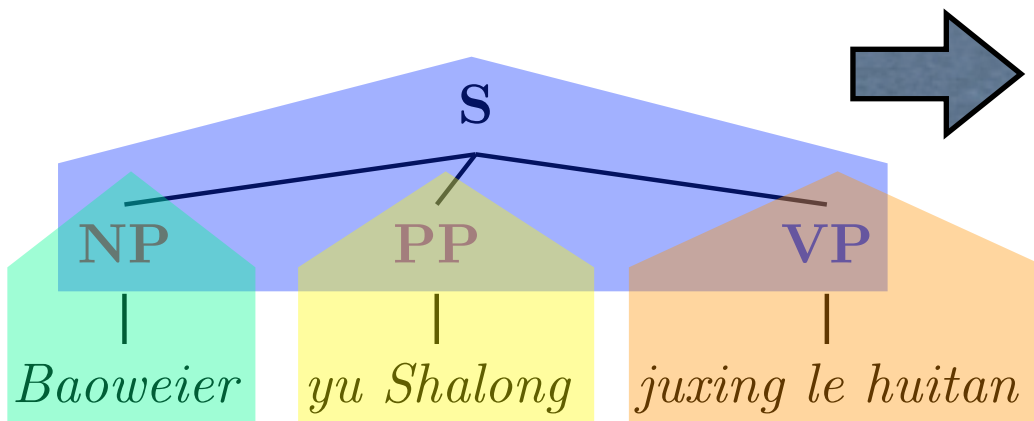
(Wu, 1997; Chiang, 2005; Galley et al., 2006)



## tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

“syntax-directed”



# Two Uses of SCFGs

string-input

(Wu, 1997; Chiang, 2005; Galley et al., 2006)

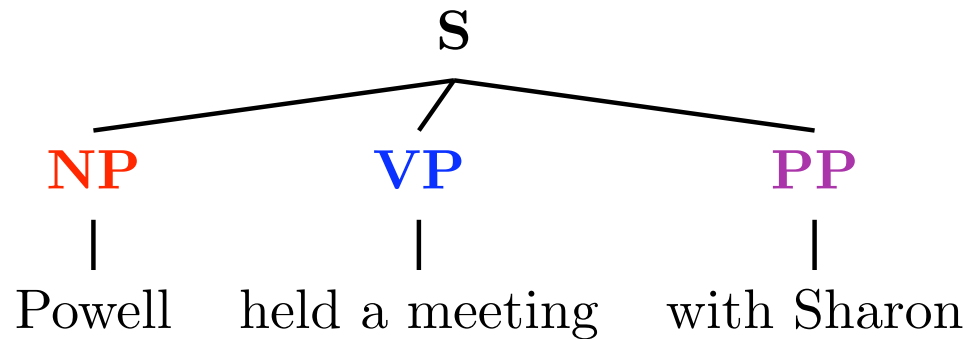
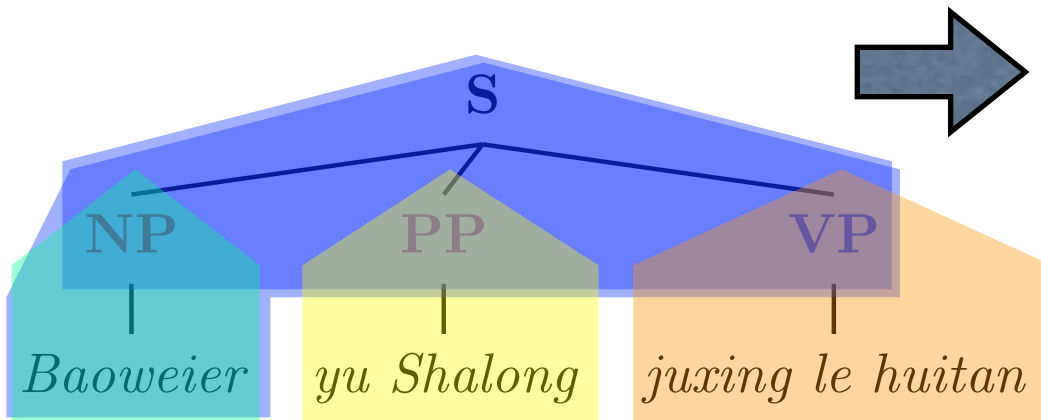


*Baoweier yu Shalong juxing le huitan*

tree-input

(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006)

“syntax-directed”



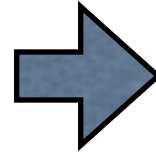
# Two Uses of SCFGs

## string-input

(Wu, 1997; Chiang, 2005; Galley et al., 2006)



*Baoweier yu Shalong juxing le huitan*

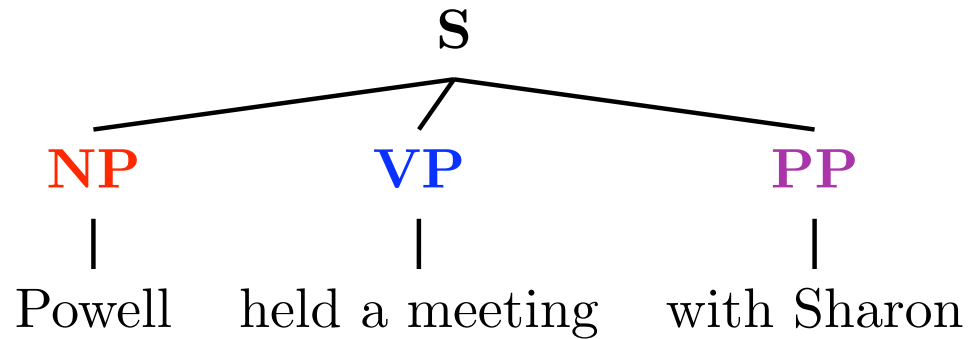
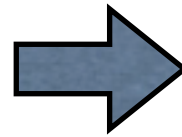
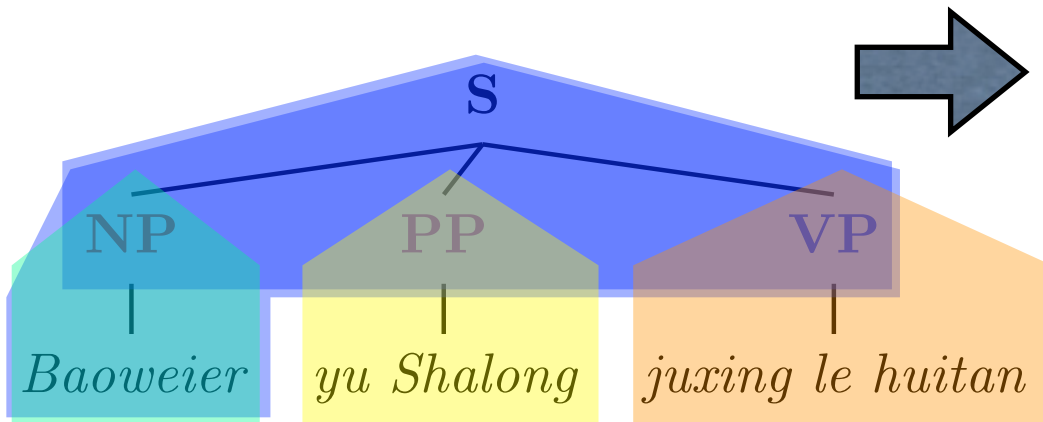


Powell held a meeting w/ Sharon

---

## tree-input

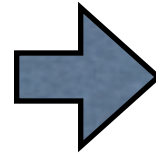
(Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006) “syntax-directed”



# Two Uses of SCFGs

string-input

(Wu, 1997; Chiang, 2005; Galley et al., 2006)

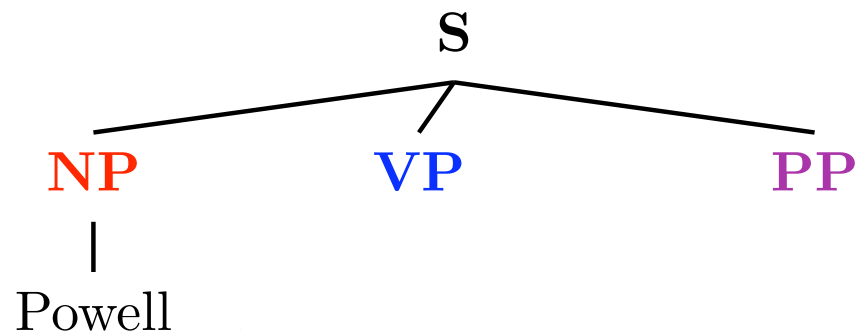
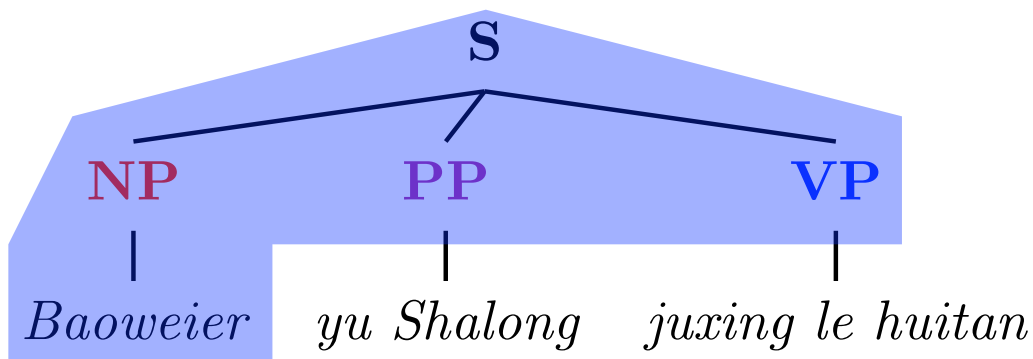


*Baoweier yu Shalong juxing le huitan*

Powell held a meeting w/ Sharon

---

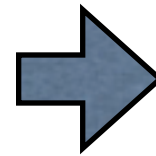
tree-input (Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006) “syntax-directed”



# Two Uses of SCFGs

string-input

(Wu, 1997; Chiang, 2005; Galley et al., 2006)

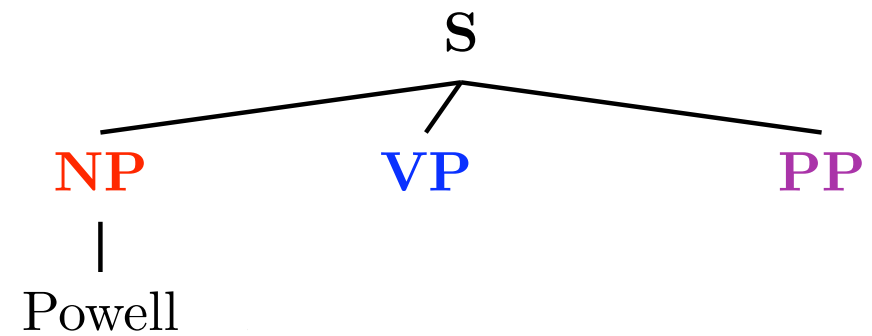
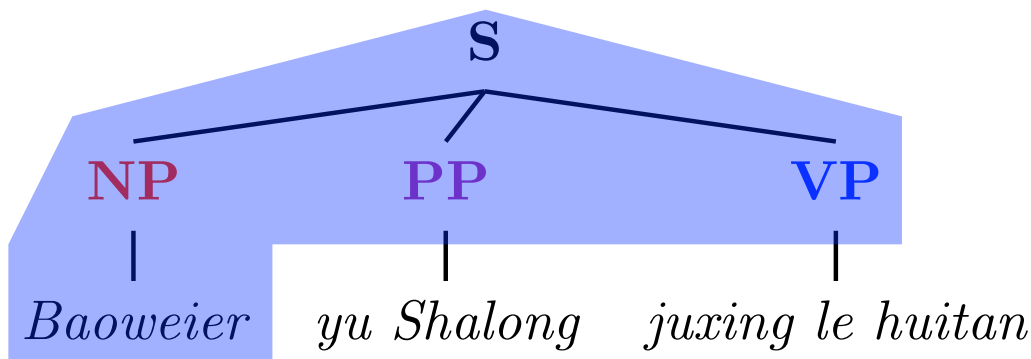


*Baoweier yu Shalong juxing le huitan*

Powell held a meeting w/ Sharon

---

tree-input (Liu et al., 2006; Huang et al., 2006; Cowan et al., 2006) “syntax-directed”



larger locality: STSG, STAG, ...

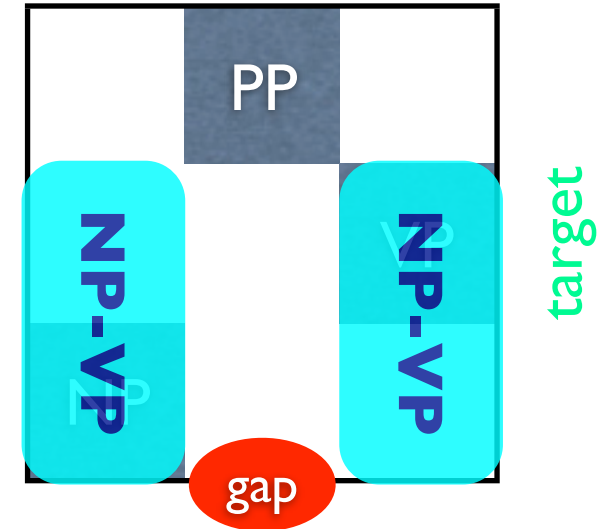
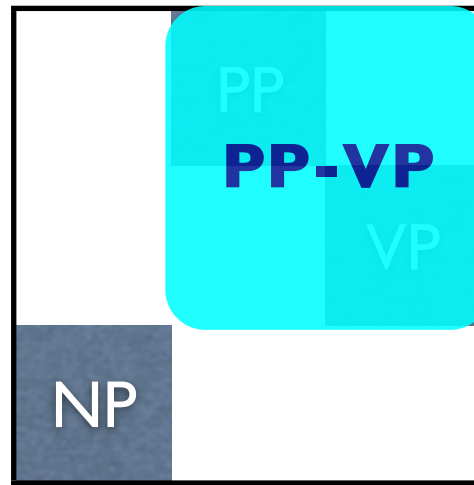
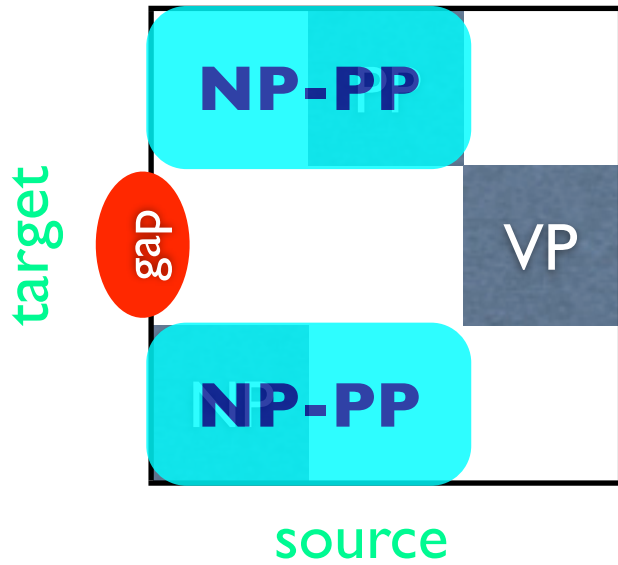
details later...

# Three Binarization Schemes

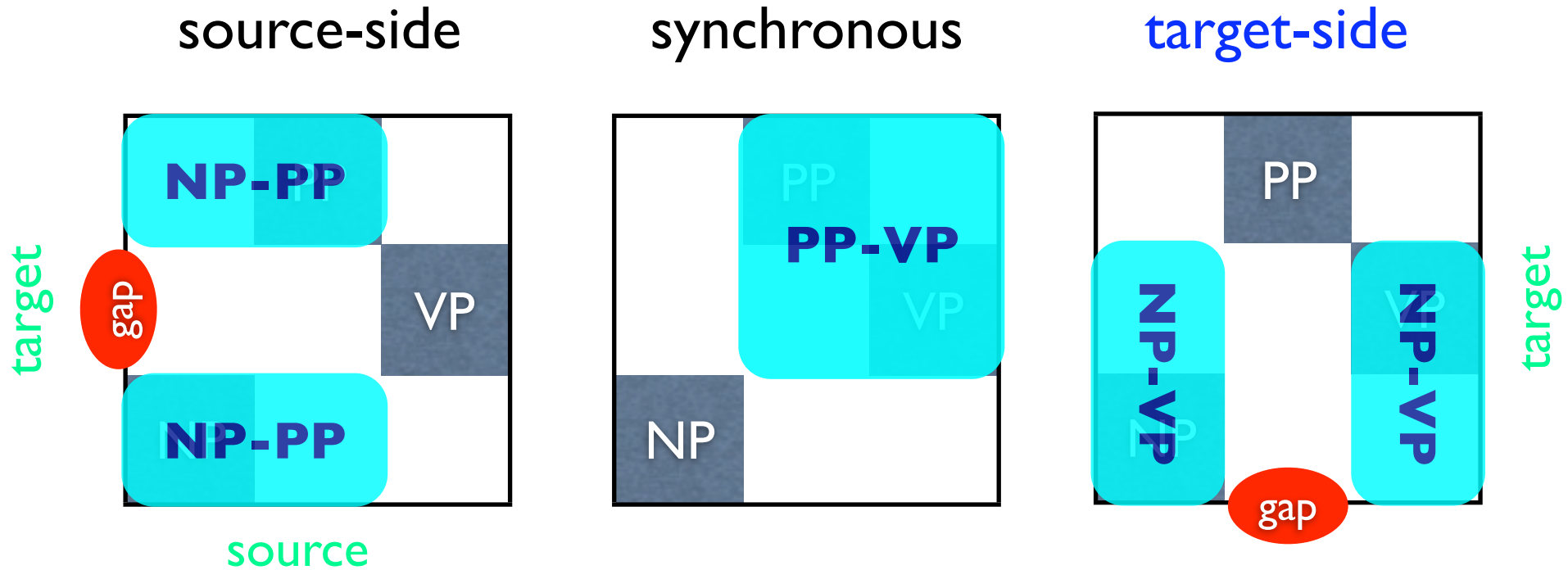
source-side

synchronous

target-side

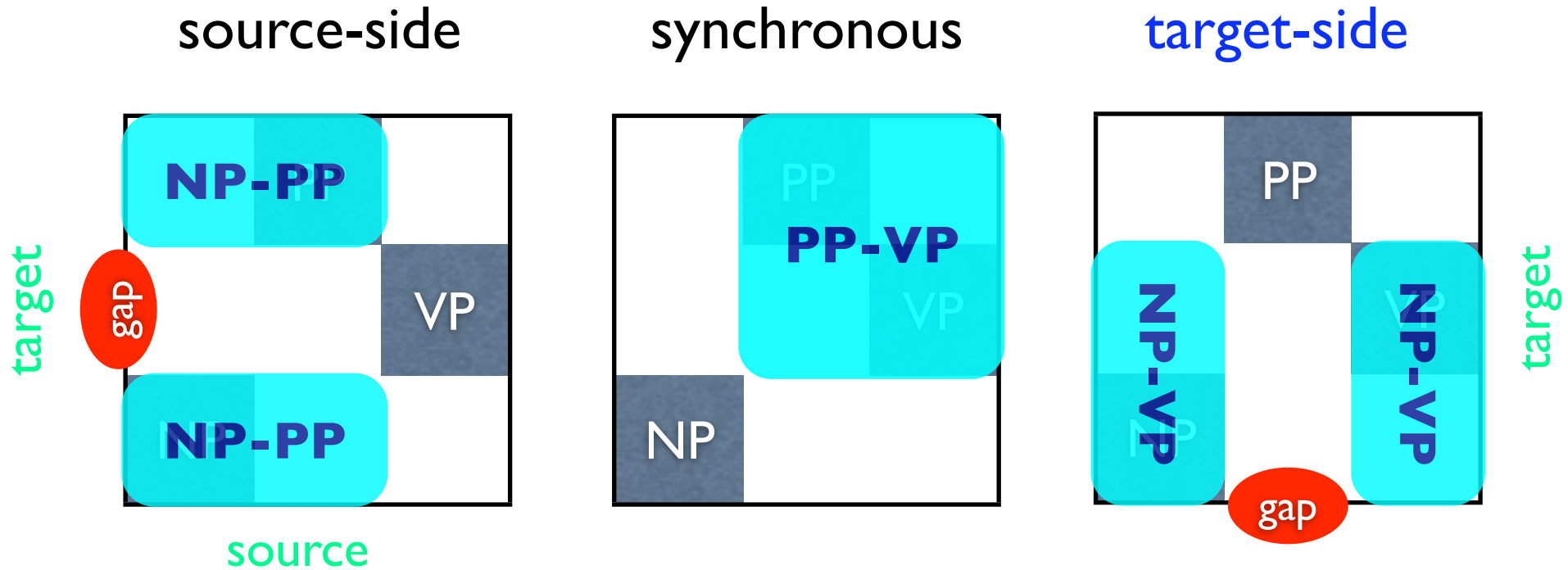


# Three Binarization Schemes



contiguous on:	source only	both sides	target only
applicability:	100%	high, but not always	100%
implementation:	trivial	involved	trivial

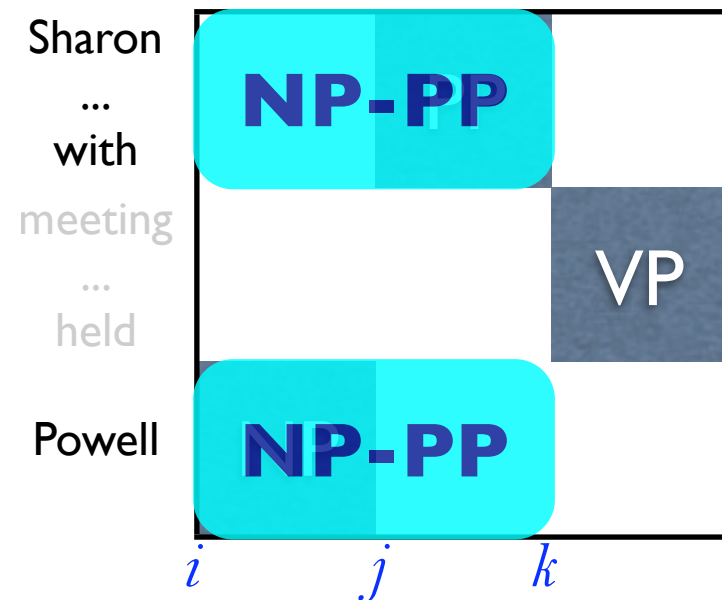
# Three Binarization Schemes



contiguous on:	source only	both sides	target only
applicability:	100%	high, but not always	100%
implementation:	trivial	involved	trivial
string-input:	very bad	good	bad
tree-input:	very bad	good	very good

# Source-side Binarization

- translation model (TM) only
  - same as monolingual parsing
  - cubic on the source side
- with a language model (Wu, 1996)
  - exponential due to gaps (Huang et al., 2005)



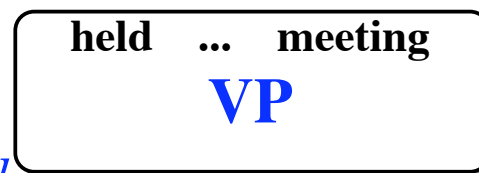
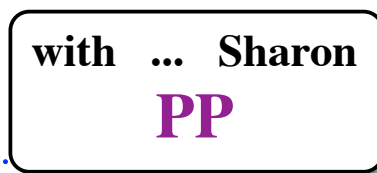
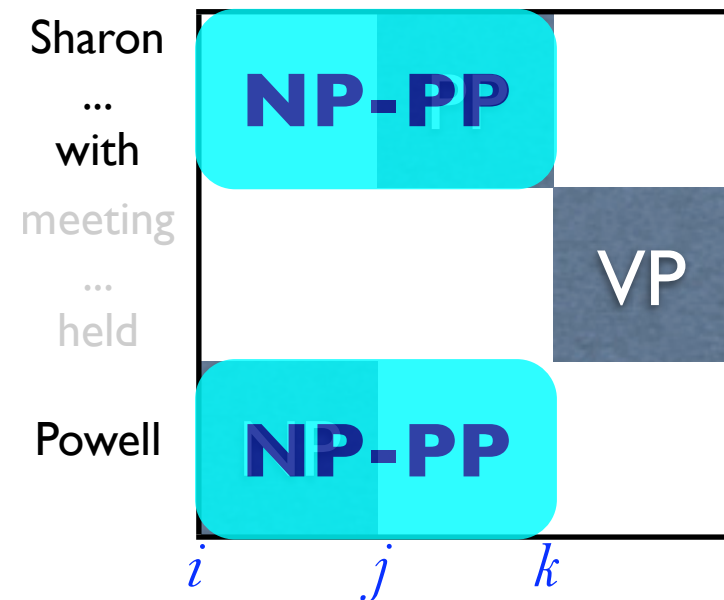
*Baoweier* *yu* *Shalong* *juxing* *le* *huitan*

$$O(|w|^{3+2(t+1)(m-1)})$$

$m$ -gram,  
target discontinuity  $t$

# Source-side Binarization

- translation model (TM) only
  - same as monolingual parsing
  - cubic on the source side
- with a language model (Wu, 1996)
  - exponential due to gaps (Huang et al., 2005)



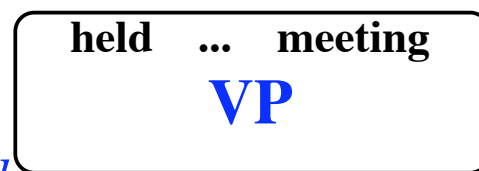
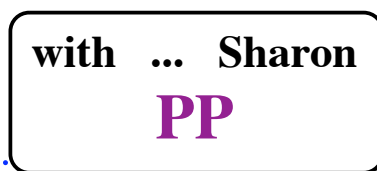
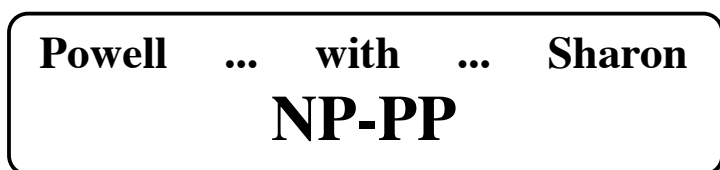
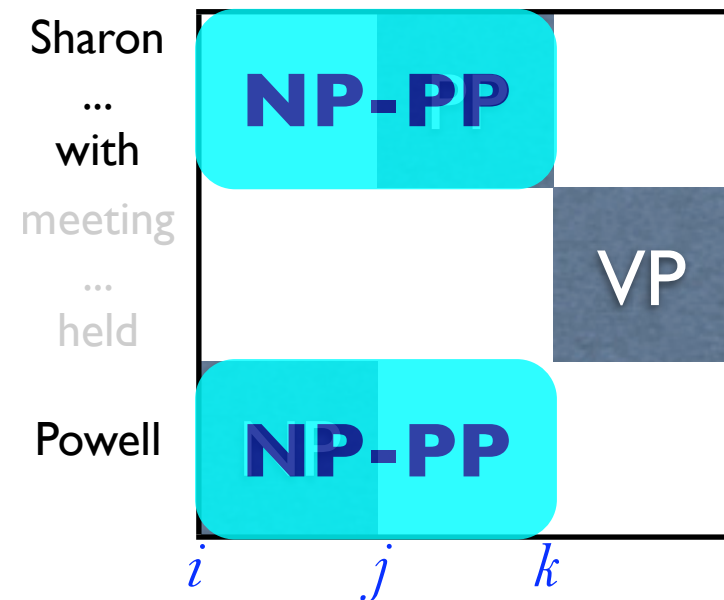
*Baoweier yu Shalong juxing le huitan*

$$O(|w|^{3+2(t+1)(m-1)})$$

$m$ -gram,  
target discontinuity  $t$

# Source-side Binarization

- translation model (TM) only
  - same as monolingual parsing
  - cubic on the source side
- with a language model (Wu, 1996)
  - exponential due to gaps (Huang et al., 2005)



$i$   $j$   $k$

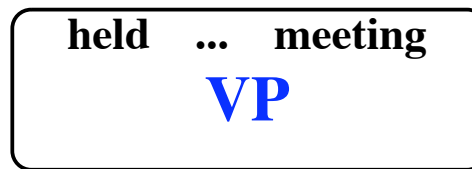
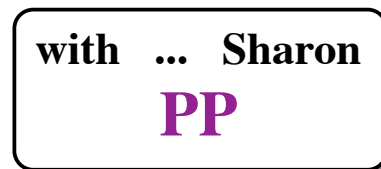
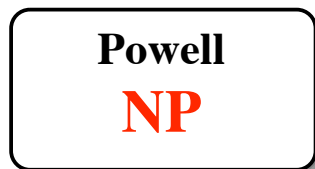
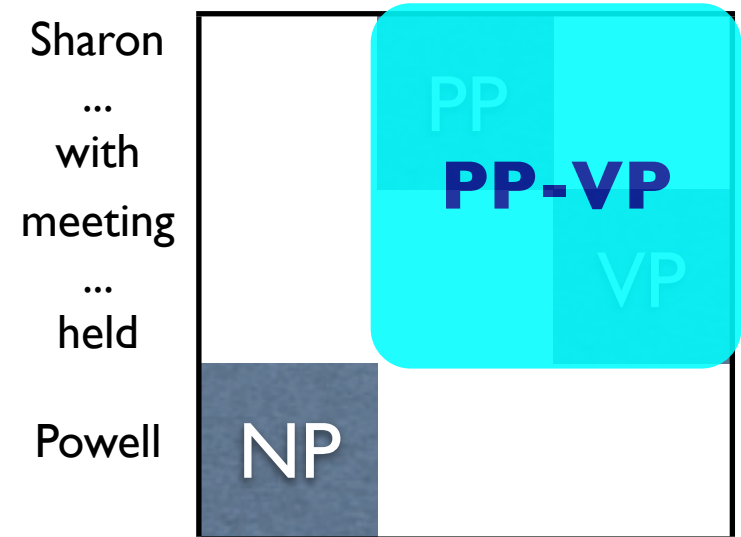
*Baoweier yu Shalong juxing le huitan*

$$O(|w|^{3+2(t+1)(m-1)})$$

$m$ -gram,  
target discontinuity  $t$

# Synchronous Binarization

- contiguous on both sides
- cubic on the source side
- polynomial on the target side
- but ...



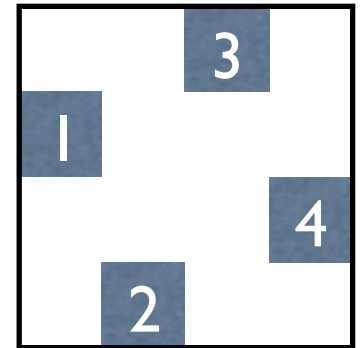
*Baoweier yu Shalong juxing le huitan*

$$O(|w|^{3+4(m-1)})$$

# Binarizability

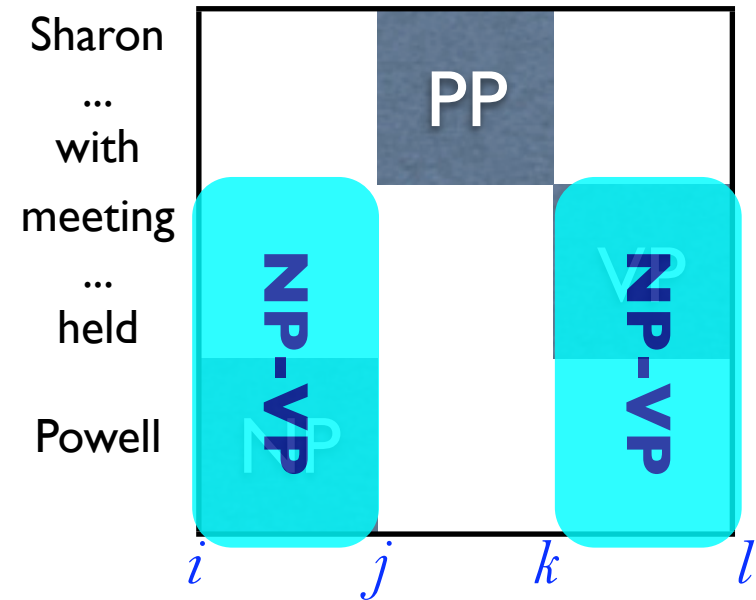
- not all rules are synchronously binarizable
- Chinese-English: vast majority is binarizable
- but binarizability ratio decreases on freer word-order languages (Wellington et al., 2006)

(Wu, 1997)



# Target-side Binarization

- contiguous on target-side
- polynomial for LM
- but gaps on the source-side!
  - can not use CKY
  - exponential for discont. parsing (cf. TAGs)
- trade-off is worthwhile
  - because LM complexity is higher
  - related: (Watanabe et al., 2006)  
(Venugopal et al., 2007)



$$O(|w|^{(2s+1)+4(m-1)})$$

# Summary: string-input systems

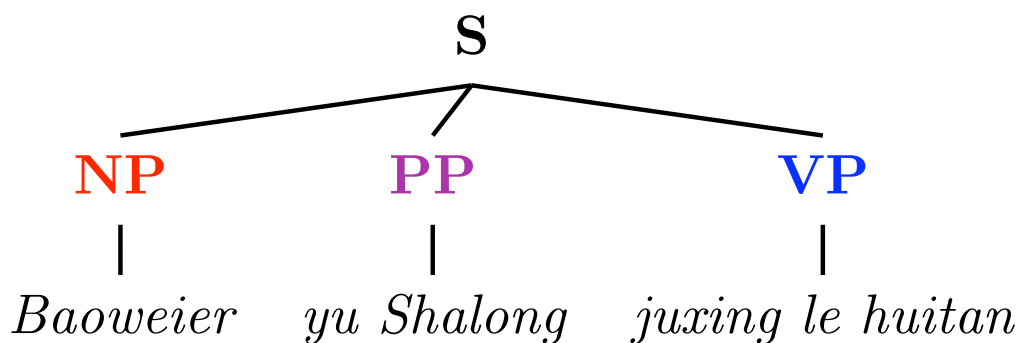
$m$ -gram, source discontinuity  $s$ , target discontinuity  $t$

	TM	LM	overall
source-side	3	$2(t+1)(m-1)$	$3+2(t+1)(m-1)$
synchronous	3	$4(m-1)$	$3+4(m-1)$
target-side	$2s+1$	$4(m-1)$	$(2s+1)+4(m-1)$

# Tree-input Systems

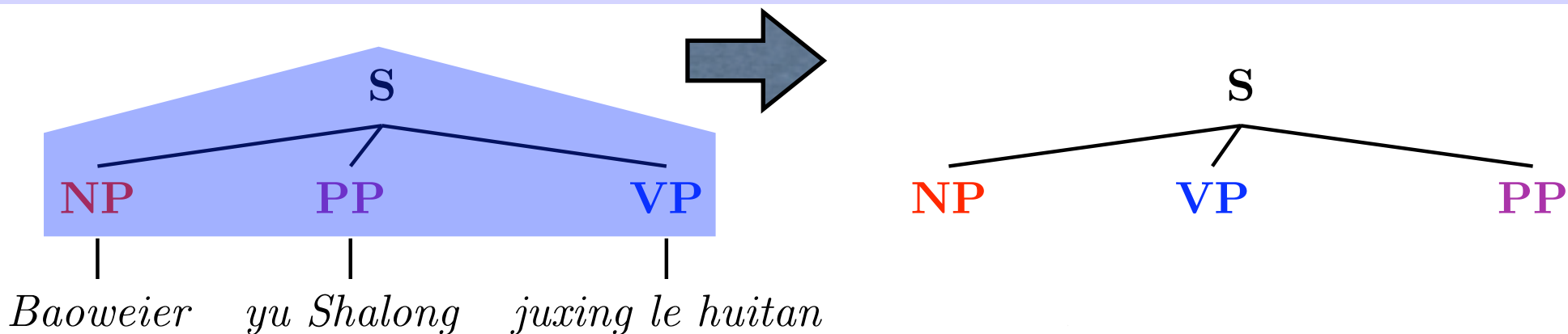
- pattern-match the input tree with the source-proj.
- **key difference**
  - TM-only decoding does **not** need binarization!
  - always **linear-time** decoding regardless of arity
- source-side binarization: still exponential time

# Tree-input Systems



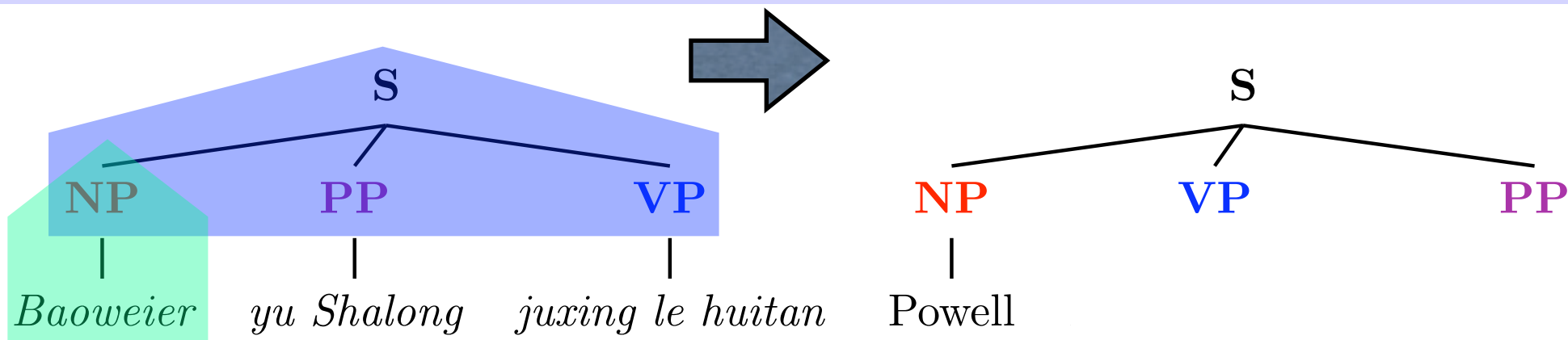
- pattern-match the input tree with the source-proj.
- **key difference**
  - TM-only decoding does **not** need binarization!
  - always **linear-time** decoding regardless of arity
- source-side binarization: still exponential time

# Tree-input Systems



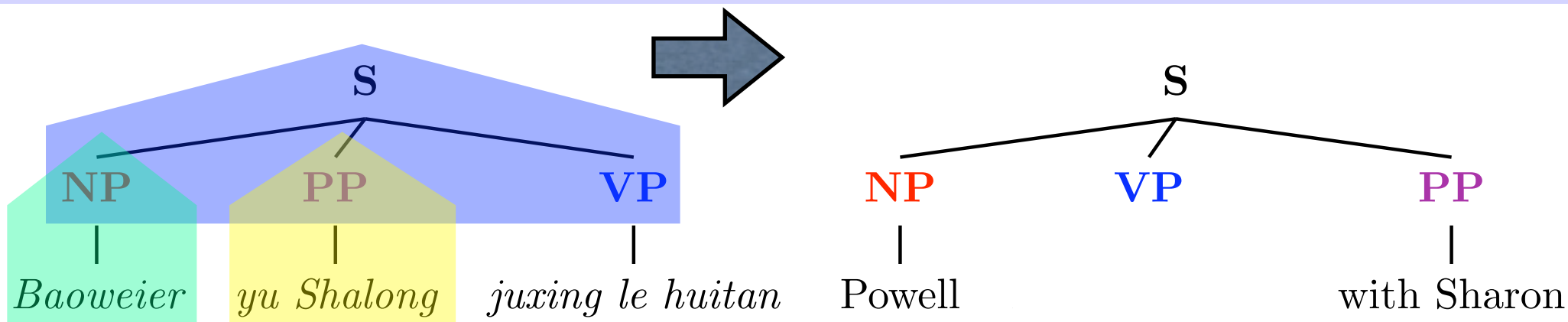
- pattern-match the input tree with the source-proj.
- **key difference**
  - TM-only decoding does **not** need binarization!
  - always **linear-time** decoding regardless of arity
- source-side binarization: still exponential time

# Tree-input Systems



- pattern-match the input tree with the source-proj.
- **key difference**
  - TM-only decoding does **not** need binarization!
  - always **linear-time** decoding regardless of arity
- source-side binarization: still exponential time

# Tree-input Systems

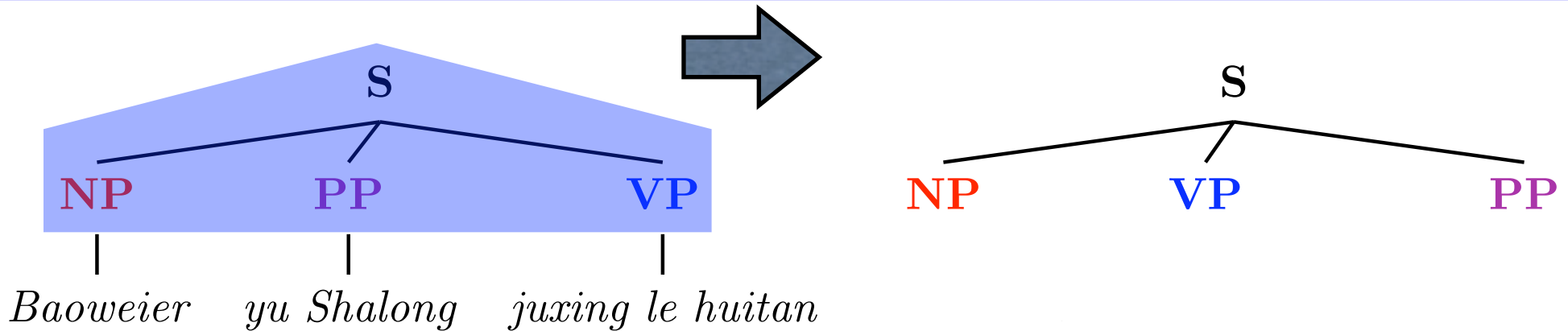


- pattern-match the input tree with the source-proj.
- **key difference**
  - TM-only decoding does **not** need binarization!
  - always **linear-time** decoding regardless of arity
- source-side binarization: still exponential time

# Target-side Binarization

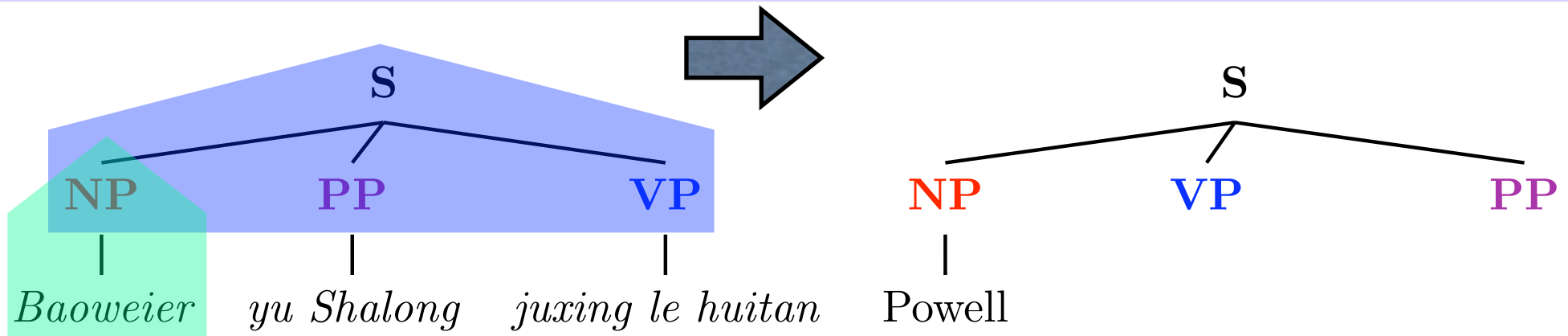
- (on-the-fly) left-to-right generation on the target-side
- also memoize virtual non-terminals
  - => target-side binarization for tree-input systems
- now it does not suffer from discontinuous parsing, and still enjoys contiguous generation

# Target-side Binarization



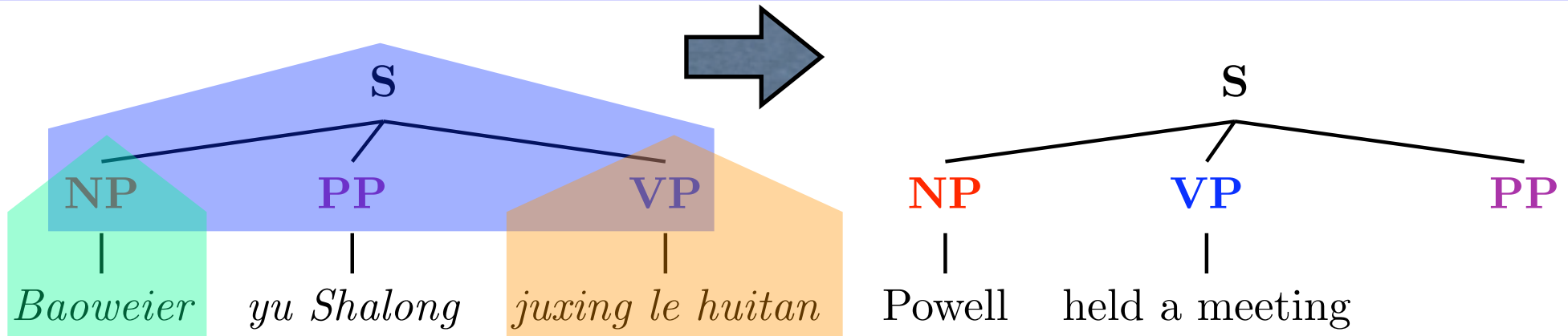
- (on-the-fly) left-to-right generation on the target-side
- also memoize virtual non-terminals
  - => target-side binarization for tree-input systems
- now it does not suffer from discontinuous parsing, and still enjoys contiguous generation

# Target-side Binarization



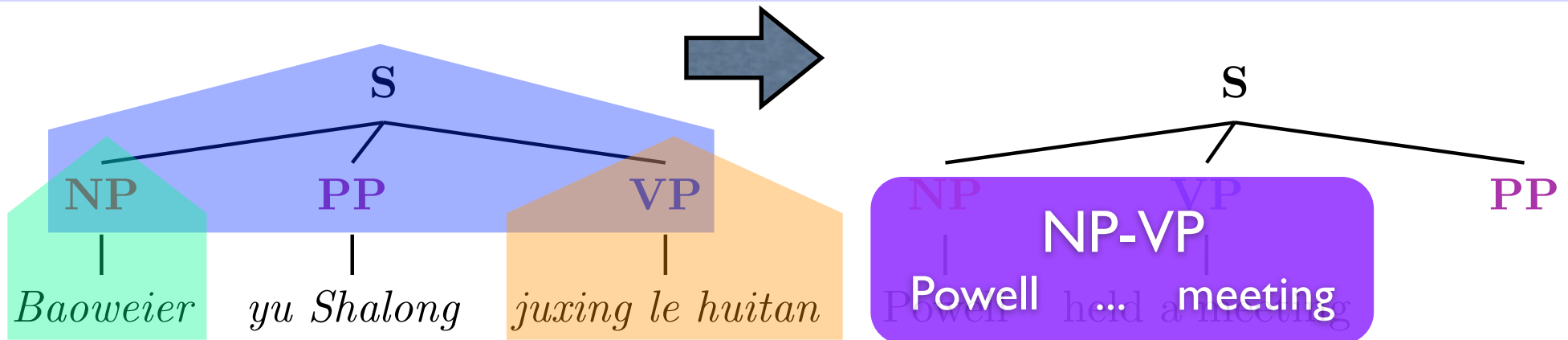
- (on-the-fly) left-to-right generation on the target-side
- also memoize virtual non-terminals
  - => target-side binarization for tree-input systems
- now it does not suffer from discontinuous parsing, and still enjoys contiguous generation

# Target-side Binarization



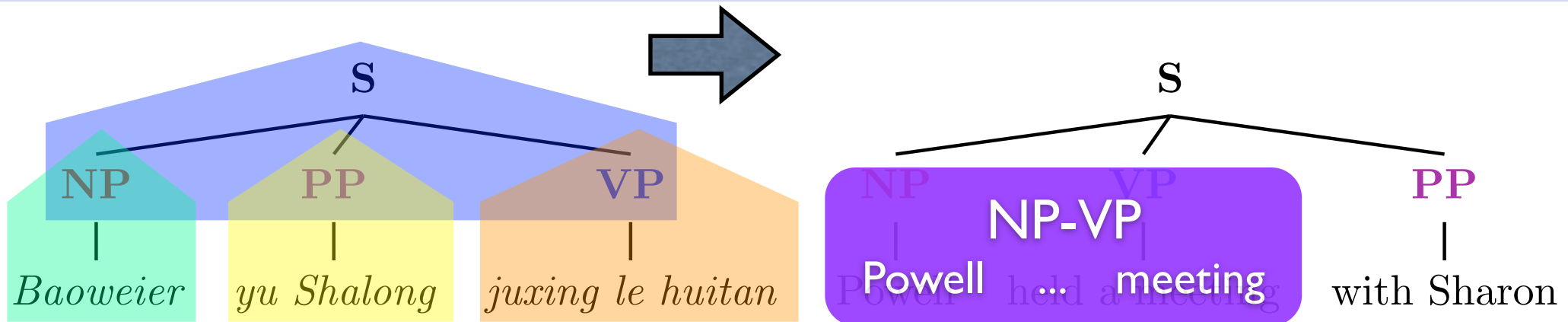
- (on-the-fly) left-to-right generation on the target-side
- also memoize virtual non-terminals
  - => target-side binarization for tree-input systems
- now it does not suffer from discontinuous parsing, and still enjoys contiguous generation

# Target-side Binarization



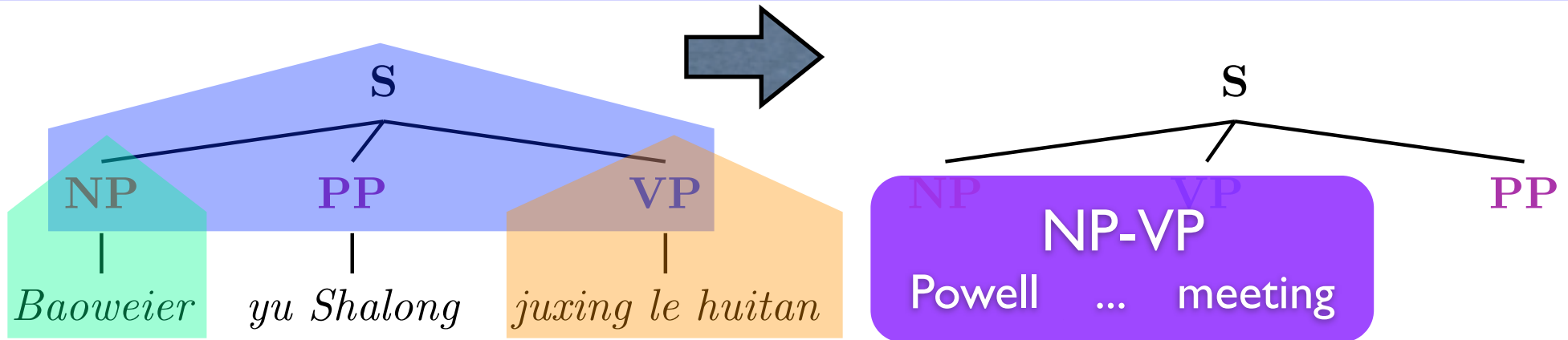
- (on-the-fly) left-to-right generation on the target-side
- also memoize virtual non-terminals
  - => target-side binarization for tree-input systems
- now it does not suffer from discontinuous parsing, and still enjoys contiguous generation

# Target-side Binarization



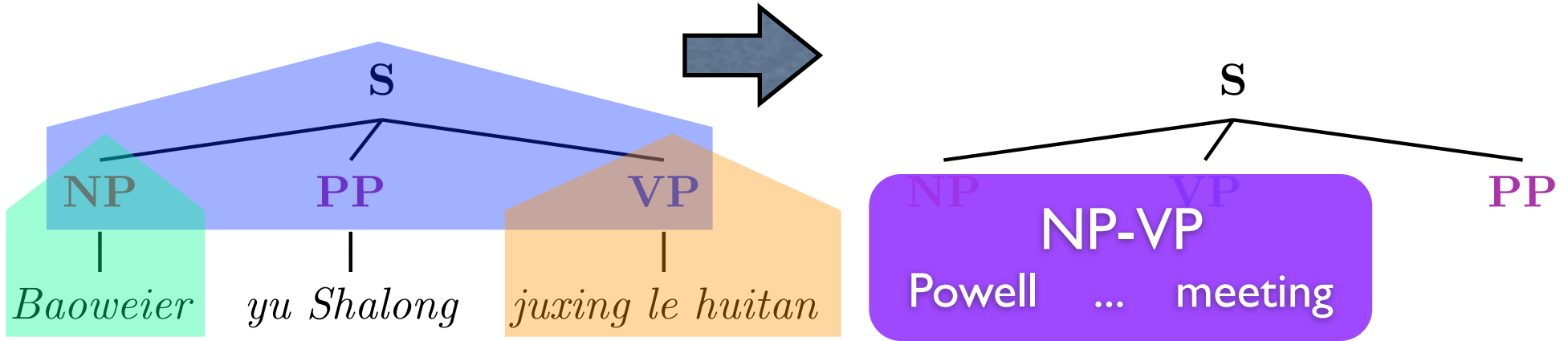
- (on-the-fly) left-to-right generation on the target-side
- also memoize virtual non-terminals
  - => target-side binarization for tree-input systems
- now it does not suffer from discontinuous parsing, and still enjoys contiguous generation

# Target-side vs. Synchronous

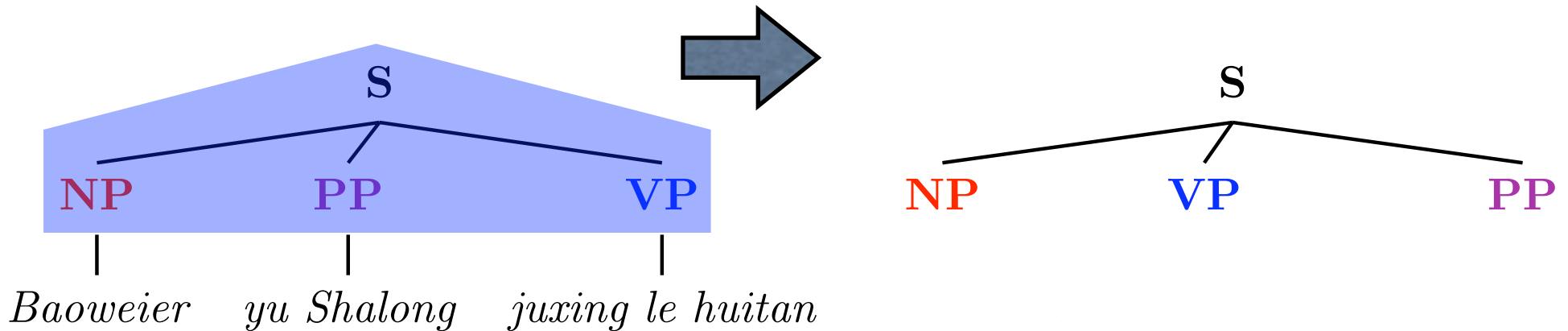


- synchronous binarization: unnecessarily complicated

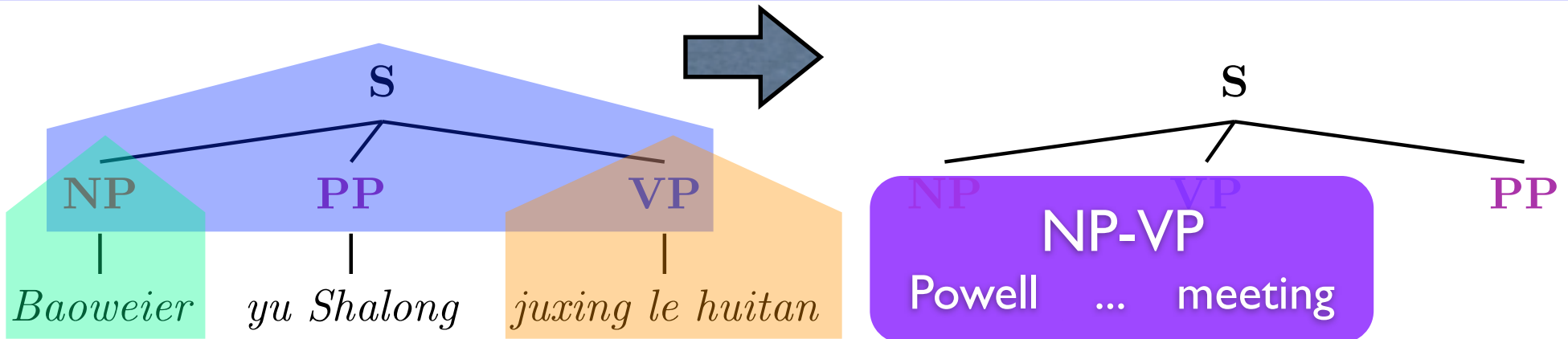
# Target-side vs. Synchronous



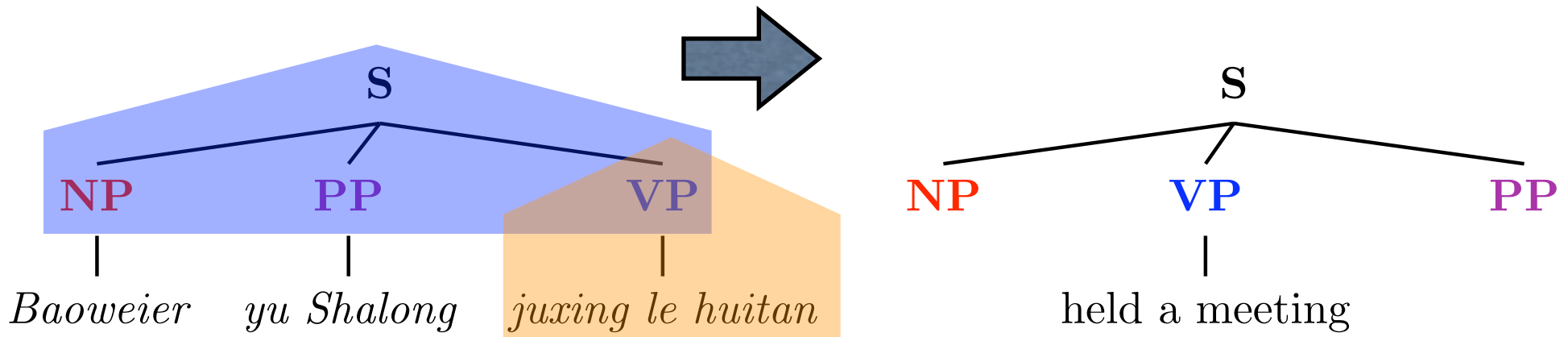
- synchronous binarization: unnecessarily complicated



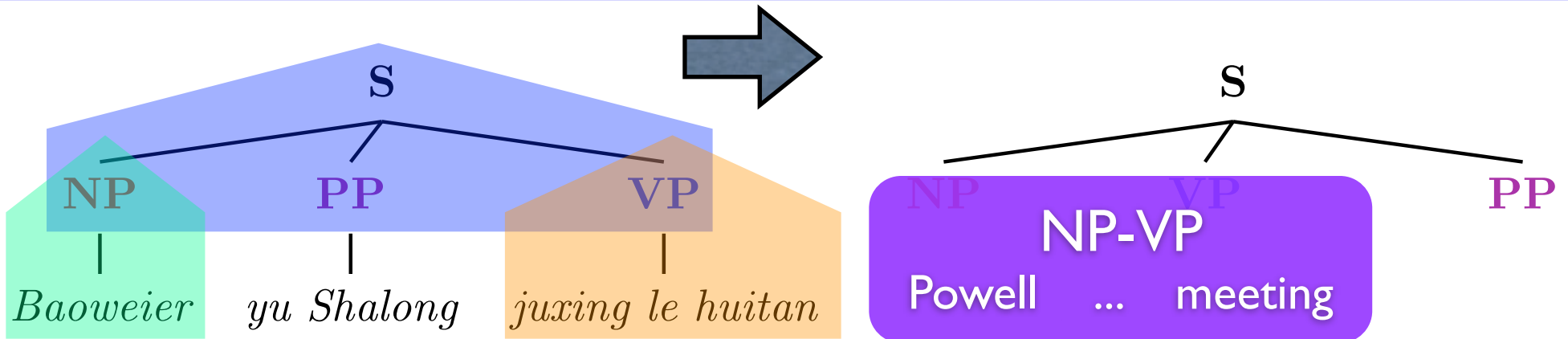
# Target-side vs. Synchronous



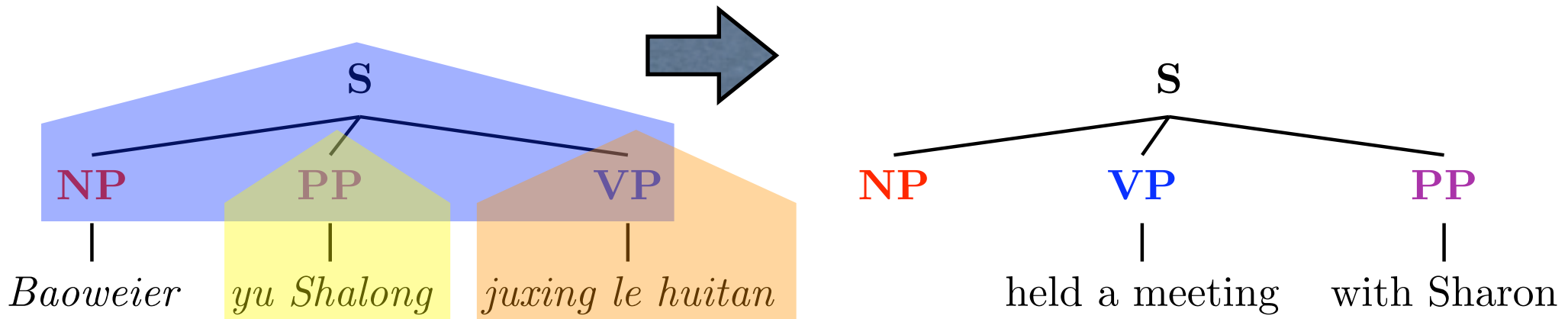
- synchronous binarization: unnecessarily complicated



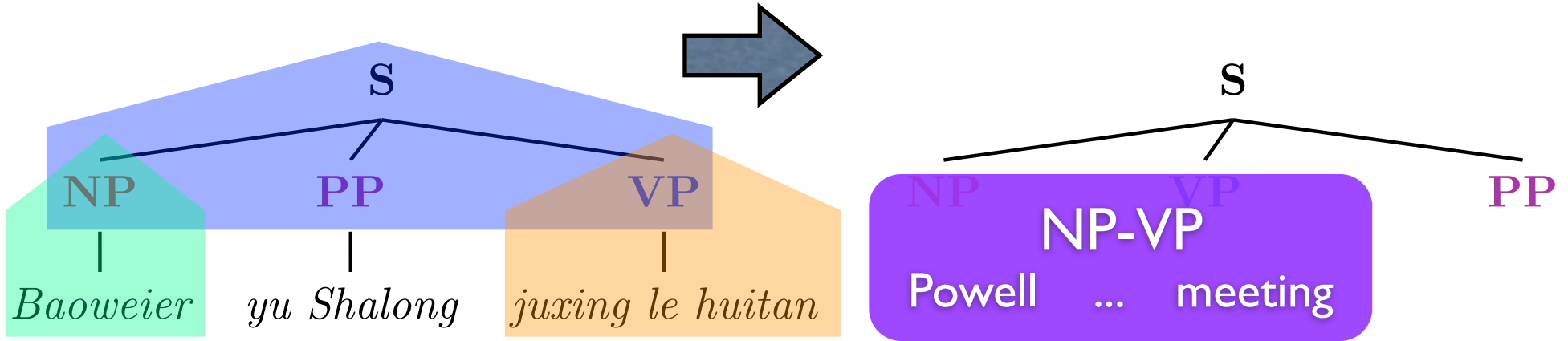
# Target-side vs. Synchronous



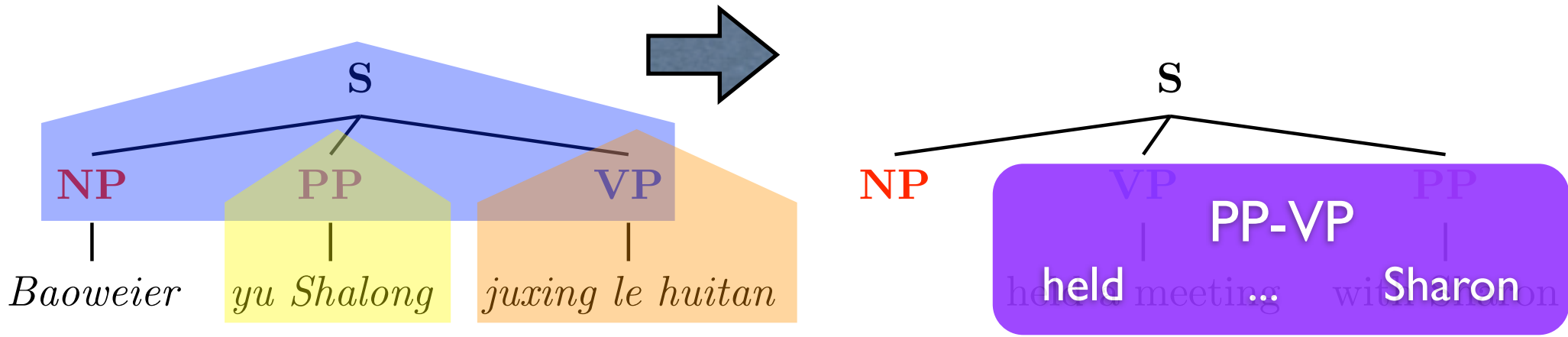
- synchronous binarization: unnecessarily complicated



# Target-side vs. Synchronous



- synchronous binarization: unnecessarily complicated



# Complexity Summary







$m$ -gram, source discontinuity  $s$ , target discontinuity  $t$

	string-input	tree-input *
source-side	$3 + 2(t+1)(m-1)$	$1 + 2(t+1)(m-1)$
synchronous	$3 + 4(m-1)$	$1 + 4(m-1)$
target-side	$2s + 1 + 4(m-1)$	$1 + 4(m-1)$

\* does not include parsing time

# Complexity Summary

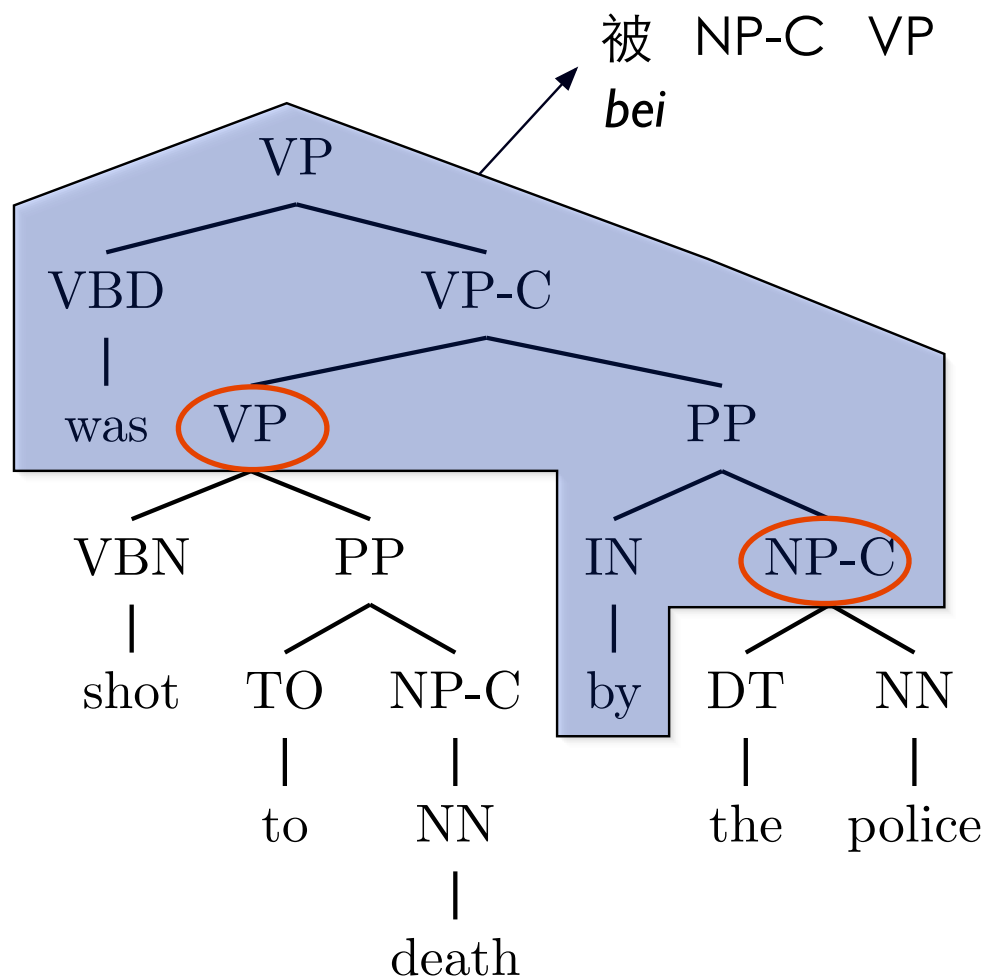
$m$ -gram, source discontinuity  $s$ , target discontinuity  $t$

	string-input	tree-input <sup>*</sup>
source-side		
synchronous		
target-side		

# Experiments

# Tree-to-String System

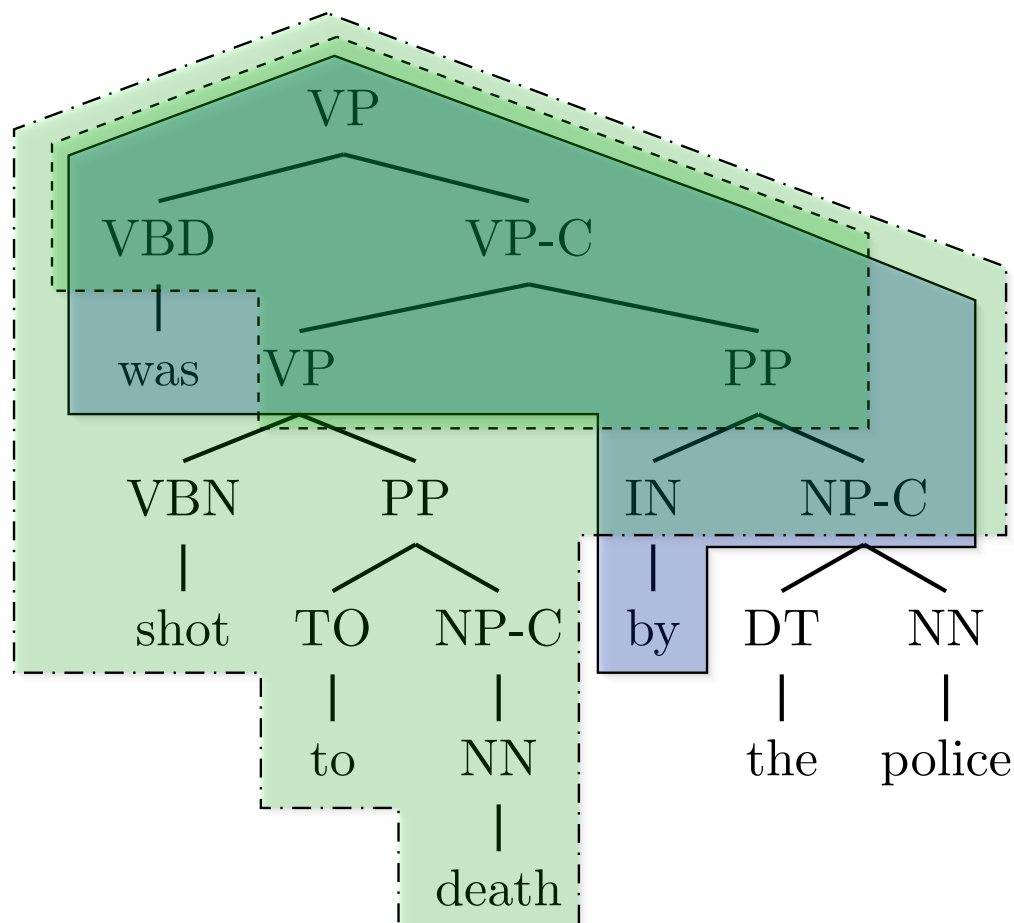
- syntax-directed English to Chinese (Huang, Knight, Joshi, 2006)
- the reverse direction is found in (Liu et al., 2006)



extended domain of locality:  
synchronous tree-  
substitution grammars (STSG)  
(Galley et al., 2004; Eisner, 2003)

related to  
STAG (Shieber/Schabes, 90)  
STIG (Nesson et al., 06)

# TM-only Decoding

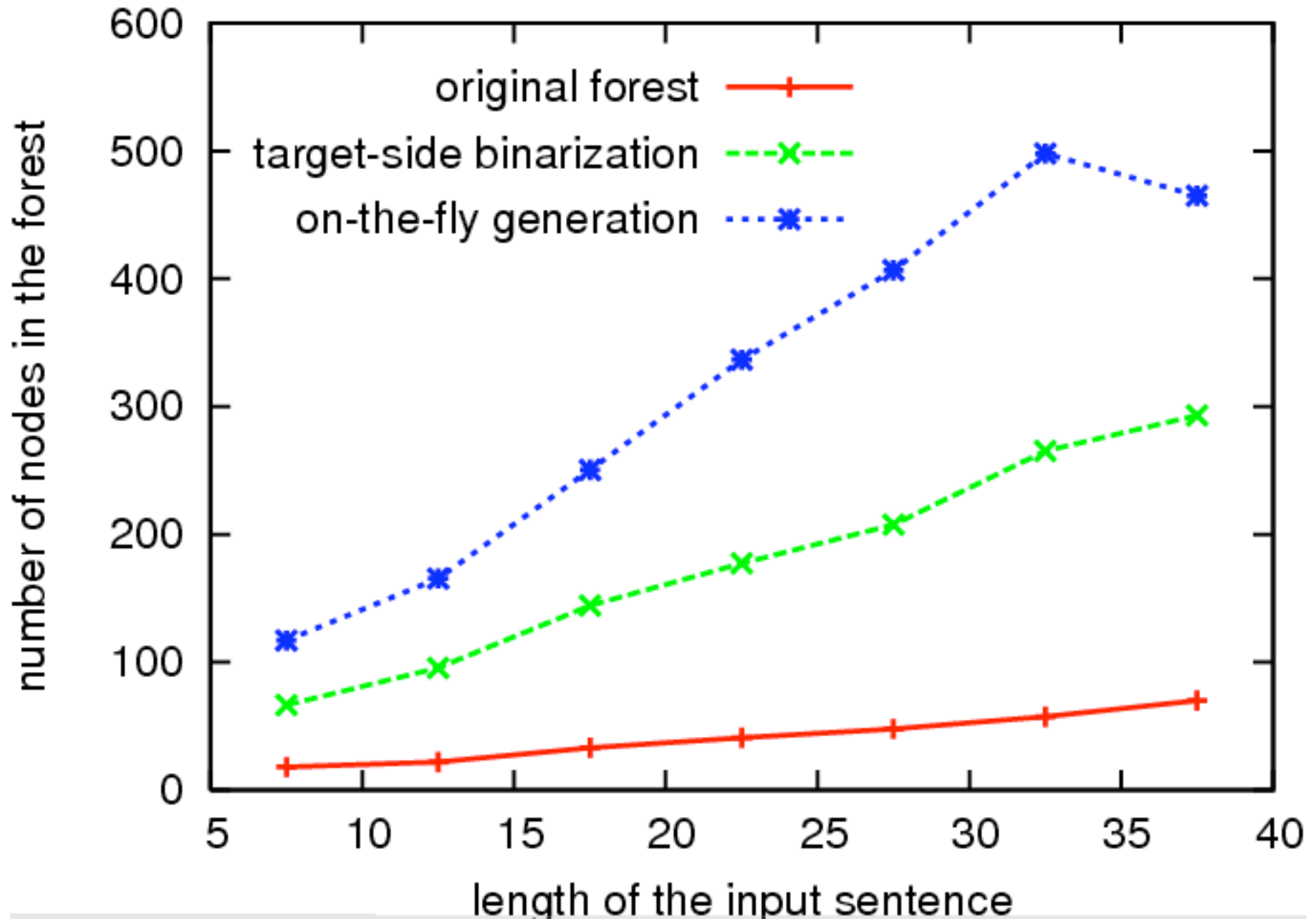


- depth-first-search (DFS)
  - for each tree node
  - try all rules applicable
  - recursion on subtrees
- top-down memoization
  - linear-time
  - dynamic programming
  - soft decisions

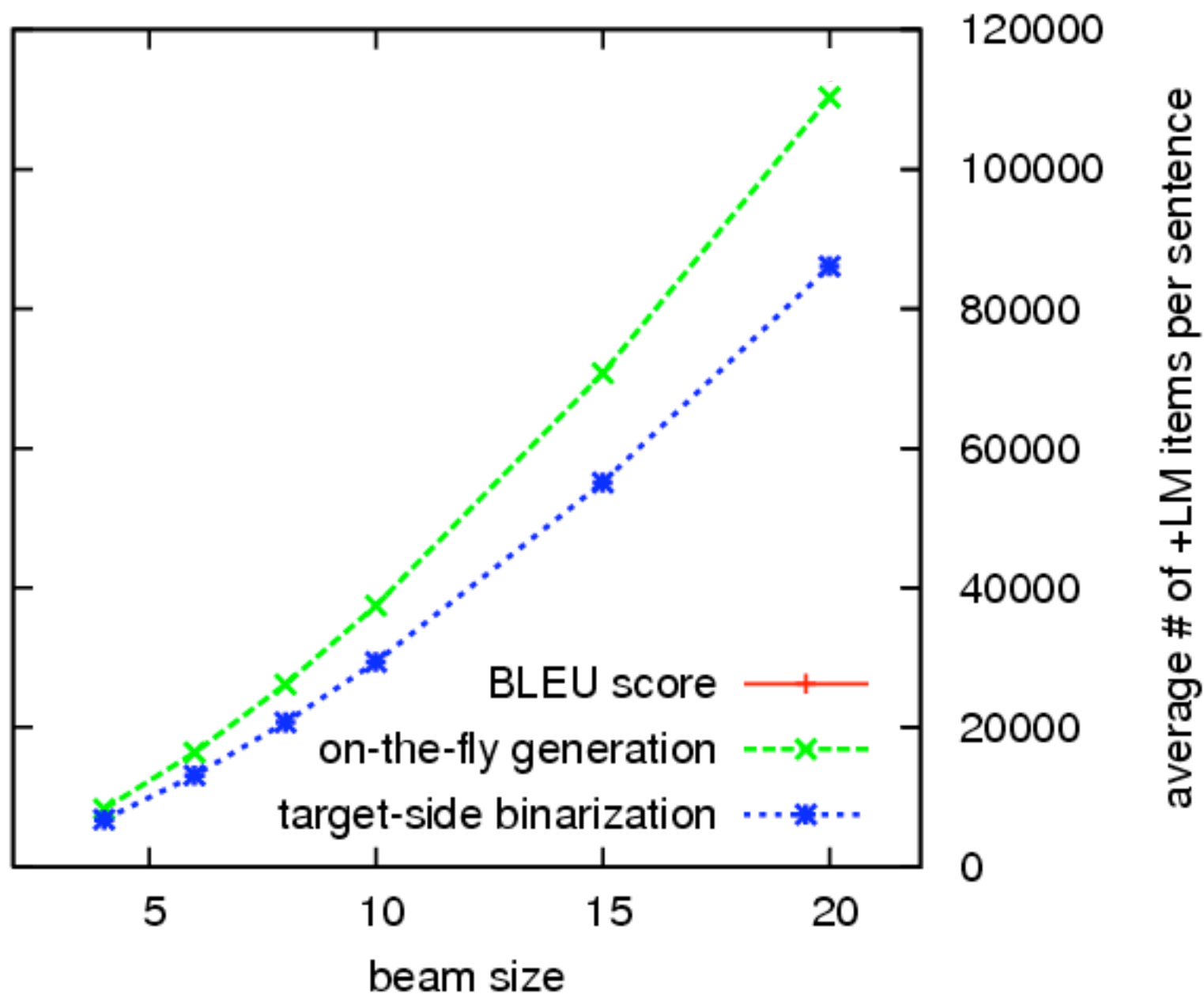
# Data

- trained on a parallel corpus of 28.3 M words on the English-side
- tested on 140 sentences (9-36 words)
- English-side parsed by a variant of Collins parser
- GIZA++ alignment
- extracted 24.7 M rules using (Galley et al., 2004)
- slightly better than Pharaoh (Koehn, 2004) in BLEU
- saved forests for these 140 sentences
  - non-binarizable ratio: 0.25% cf. 0.3% in (Zhang et al., 06)

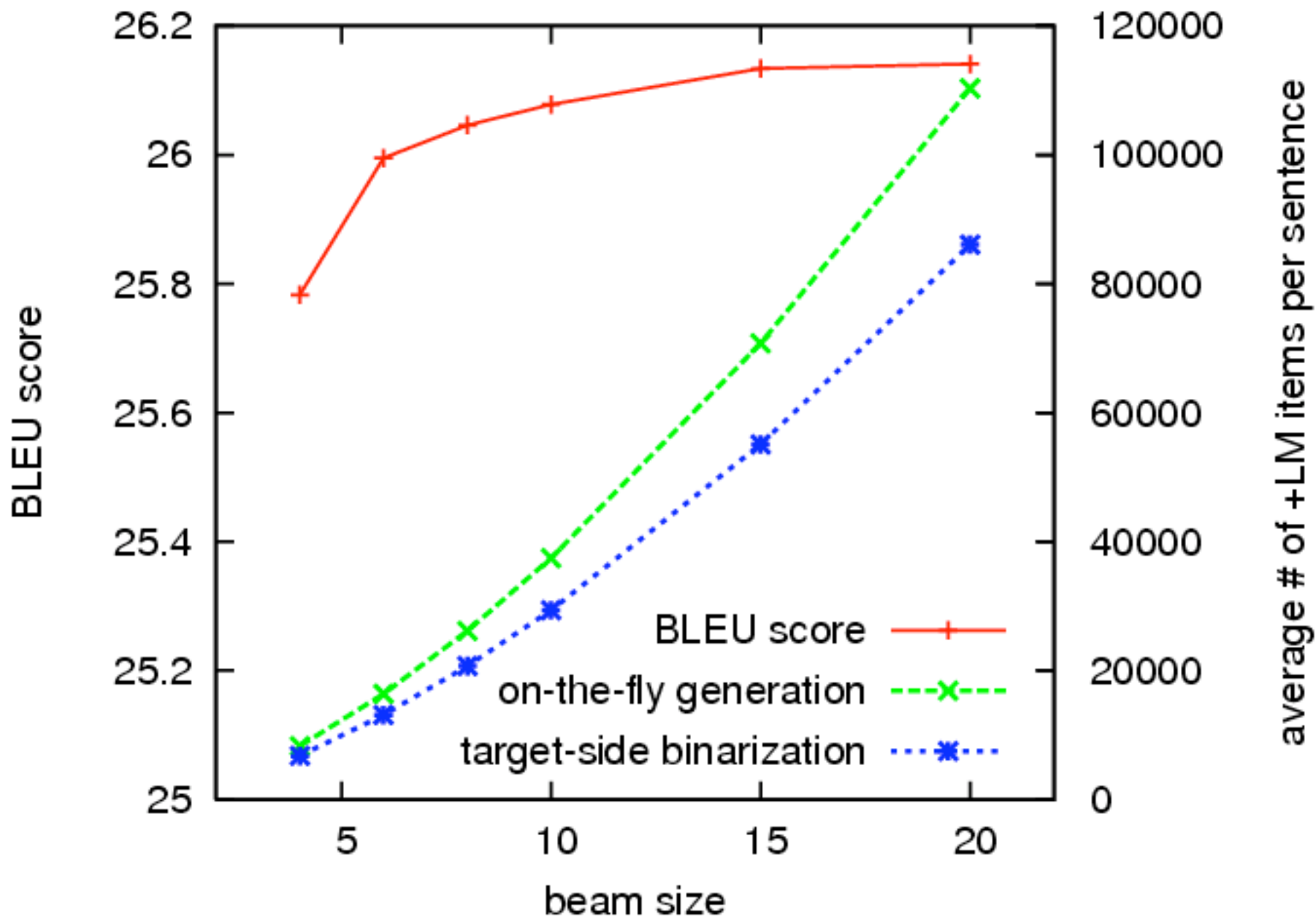
# Forest Size



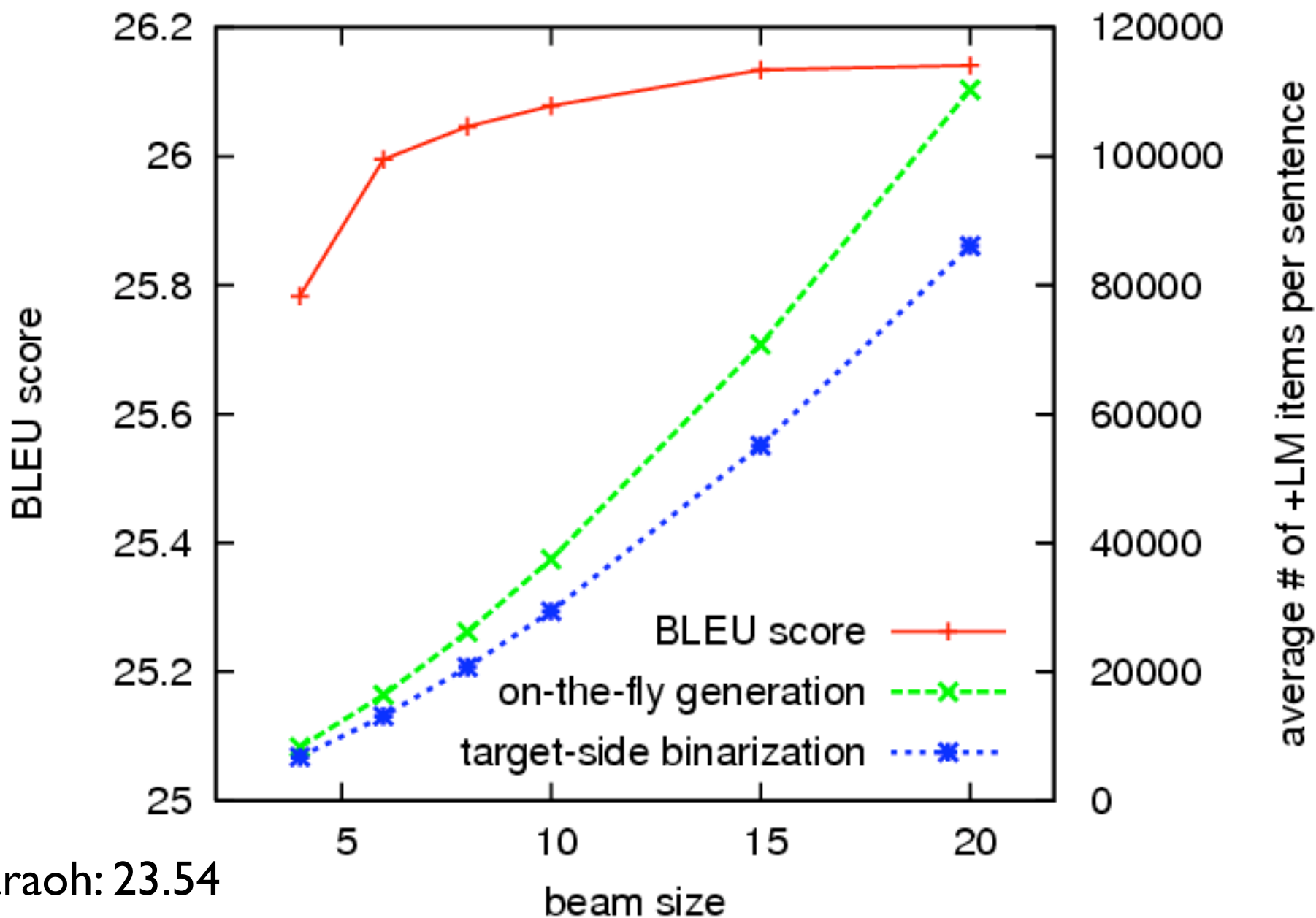
# Decoding



# Decoding

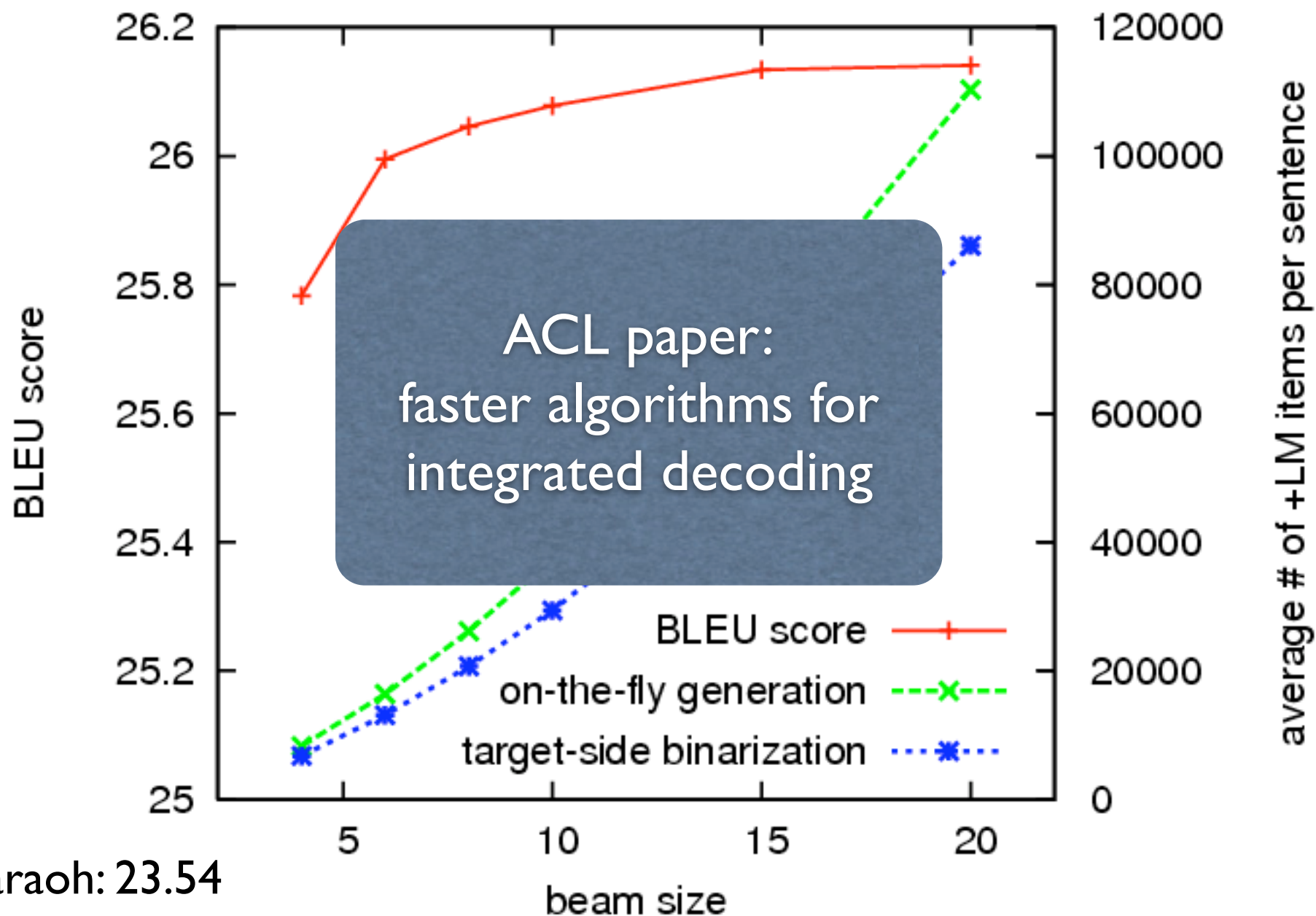


# Decoding



Pharaoh: 23.54

# Decoding



Pharaoh: 23.54

# Conclusion

- a simpler alternative binarization scheme
- theoretical comparison of the three binarization schemes in two popular types of systems
  - string-input: synchronous; tree-input: target-side
- empirical evidence on a tree-to-string system
- tree-input (“syntax-directed”) is a promising direction
  - linear-time decoding
  - easy binarization
  - decoupled parsing and transduction grammars

# Thanks!

