# Binarization for Machine Translation

Liang Huang 黃 亮
University of Pennsylvania


Joint work with Hao Zhang 張 浩 (Rochester),
Dan Gildea (Rochester), and Kevin Knight (USC/ISI)

Hong Kong University of Science and Technology, July 30, 2007

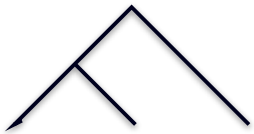
# Overview of Binarization

| parsing | machine translation |
| --- | --- |
| CFG | Synchronous CFG |
| $\mathbf{S} \rightarrow \mathbf{NP\ VP\ PP}$ | $\mathbf{S} \rightarrow \begin{matrix}\mathbf{NP\ PP\ VP} \\ \mathbf{NP\ VP\ PP}\end{matrix}$ |

# Overview of Binarization

## parsing

CFG



$S \rightarrow NP\ VP\ PP$

## machine translation

Synchronous CFG

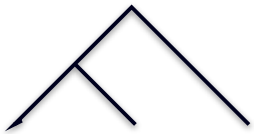
$$S \rightarrow \begin{matrix} NP & PP & VP \\ NP & VP & PP \end{matrix}$$

for cubic time parsing

# Overview of Binarization

## parsing

CFG => CNF



$S \rightarrow NP\ VP\ PP$

## machine translation

Synchronous CFG

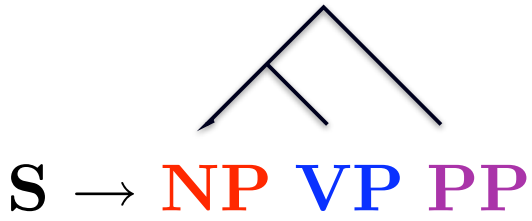
$S \rightarrow \begin{matrix} NP\ PP\ VP \\ NP\ VP\ PP \end{matrix}$

for cubic time parsing

# Overview of Binarization

## parsing

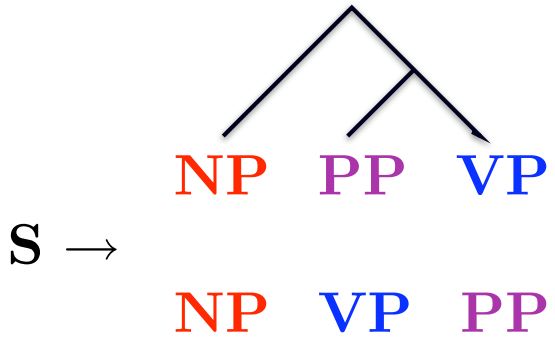
CFG => CNF




for cubic time parsing

## machine translation

Synchronous CFG

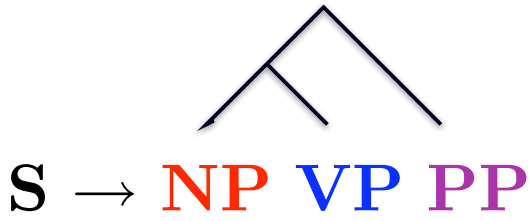



for polynomial time decoding

# Overview of Binarization

## parsing

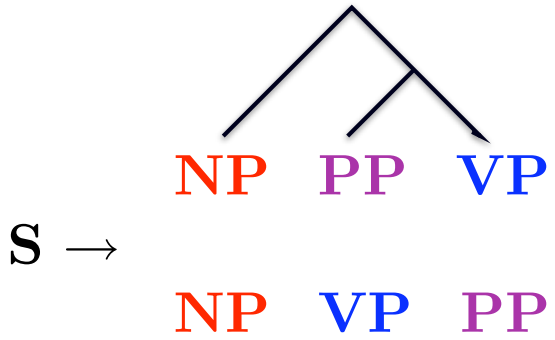
CFG => CNF




for cubic time parsing

## machine translation

Synchronous CFG => ITG

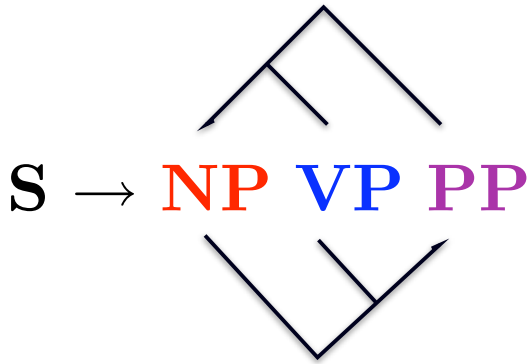



for polynomial time decoding

# Overview of Binarization

## parsing

CFG => CNF

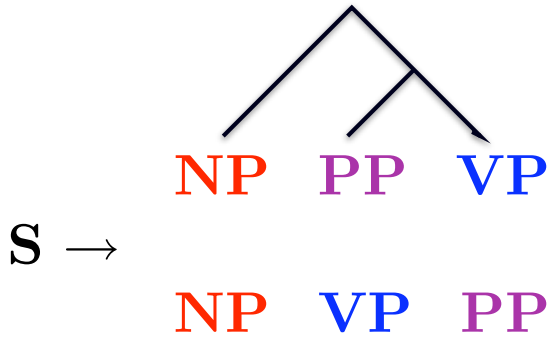



for cubic time parsing

different schemes
work equally well

## machine translation

Synchronous CFG => ITG

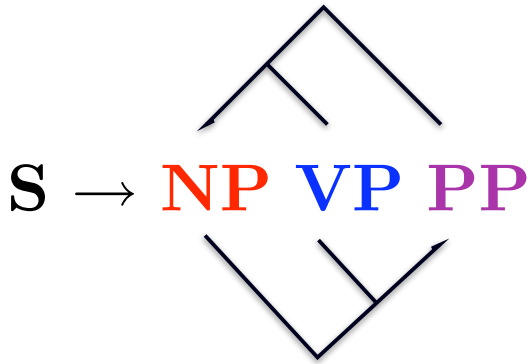



for polynomial time decoding

# Overview of Binarization

## parsing

CFG => CNF

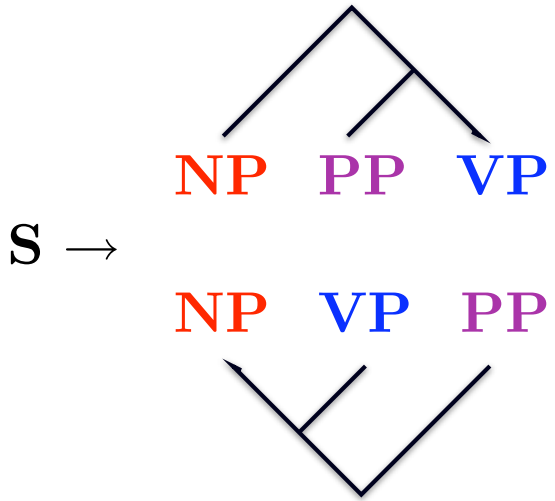



for cubic time parsing

different schemes
work equally well

## machine translation

Synchronous CFG => ITG




for polynomial time decoding

different schemes
work very differently

# Outline

- Overview: Syntax-based Machine Translation
  - Synchronous Grammars
  - Translation as Parsing
  - ISI Syntax-based System
- The Problem of Binarization
- Our Solution: Synchronous Binarization
  - Linear-time Binarization Algorithm
- Experiments
- Application: Testing the ITG Hypothesis

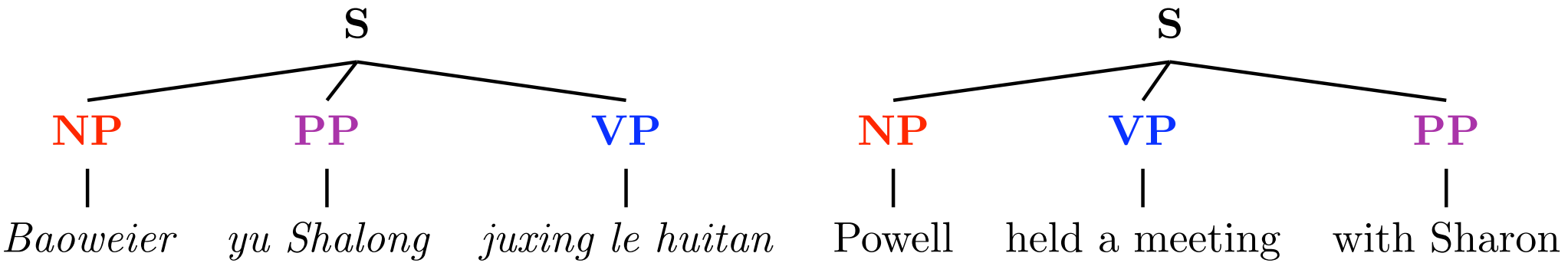
# Outline

- Overview: Syntax-based Machine Translation
  - Synchronous Grammars
  - Translation as Parsing
  - ISI Syntax-based System
- The Problem of Binarization
- Our Solution: Synchronous Binarization
  - Linear-time Binarization Algorithm
- Experiments
- Application: Testing the ITG Hypothesis

# Synchronous CFGs

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

| | | | |
|---|---|---|---|
| **S** | → | **NP**$^{(1)}$ **PP**$^{(2)}$ **VP**$^{(3)}$, | **NP**$^{(1)}$ **VP**$^{(3)}$ **PP**$^{(2)}$ |
| **NP** | → | *Baoweier*, | Powell |
| **PP** | → | *yu Shalong*, | with Sharon |
| **VP** | → | *juxing le huitan*, | held a meeting |

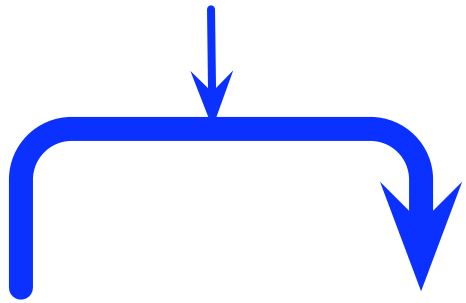



(Aho/Ullman, 1972) (Wu, 1997) (Yamada/Knight, 2001) (Chiang, 2005) (Satta/Peserico, 2005)

# Translation as Parsing

- Many problems in syntax-based MT reduce to parsing
  - here we consider decoding as monolingual parsing

| | | | |
|---|---|---|---|
| $\mathbf{S}$ | $\rightarrow$ | $\mathbf{NP}^{(1)}\ \mathbf{PP}^{(2)}\ \mathbf{VP}^{(3)}$, | $\mathbf{NP}^{(1)}\ \mathbf{VP}^{(3)}\ \mathbf{PP}^{(2)}$ |
| $\mathbf{NP}$ | $\rightarrow$ | *Baoweier*, | Powell |
| $\mathbf{PP}$ | $\rightarrow$ | *yu Shalong*, | with Sharon |
| $\mathbf{VP}$ | $\rightarrow$ | *juxing le huitan*, | held a meeting |



*Baoweier yu Shalong juxing le huitan* Powell held a meeting with Sharon

(Wu, 1996) (Melamed, 2004) (Chiang, 2005) (Satta and Peserico, 2005)

# Translation as Parsing

- Many problems in syntax-based MT reduce to parsing
  - here we consider decoding as monolingual parsing

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{NP}^{(1)}\ \mathbf{PP}^{(2)}\ \mathbf{VP}^{(3)}, \\ \mathbf{NP} &\rightarrow \textit{Baoweier}, \\ \mathbf{PP} &\rightarrow \textit{yu Shalong}, \\ \mathbf{VP} &\rightarrow \textit{juxing le huitan}, \end{aligned}$$

*Baoweier yu Shalong juxing le huitan*

1 2 3 4 5 6 7

(Wu, 1996) (Melamed, 2004) (Chiang, 2005) (Satta and Peserico, 2005)

# Translation as Parsing

- Many problems in syntax-based MT reduce to parsing
  - here we consider decoding as monolingual parsing

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{NP}^{(1)}\ \mathbf{PP}^{(2)}\ \mathbf{VP}^{(3)}, \\ \mathbf{NP} &\rightarrow \textit{Baoweier}, \\ \mathbf{PP} &\rightarrow \textit{yu Shalong}, \\ \mathbf{VP} &\rightarrow \textit{juxing le huitan}, \end{aligned}$$


(Wu, 1996) (Melamed, 2004) (Chiang, 2005) (Satta and Peserico, 2005)

# Translation as Parsing

- Many problems in syntax-based MT reduce to parsing
  - here we consider decoding as monolingual parsing

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{NP}^{(1)}\ \mathbf{PP}^{(2)}\ \mathbf{VP}^{(3)}, \\ \mathbf{NP} &\rightarrow \textit{Baoweier}, \\ \mathbf{PP} &\rightarrow \textit{yu Shalong}, \\ \mathbf{VP} &\rightarrow \textit{juxing le huitan}, \end{aligned}$$


(Wu, 1996) (Melamed, 2004) (Chiang, 2005) (Satta and Peserico, 2005)

# Translation as Parsing

- Many problems in syntax-based MT reduce to parsing
  - here we consider decoding as monolingual parsing

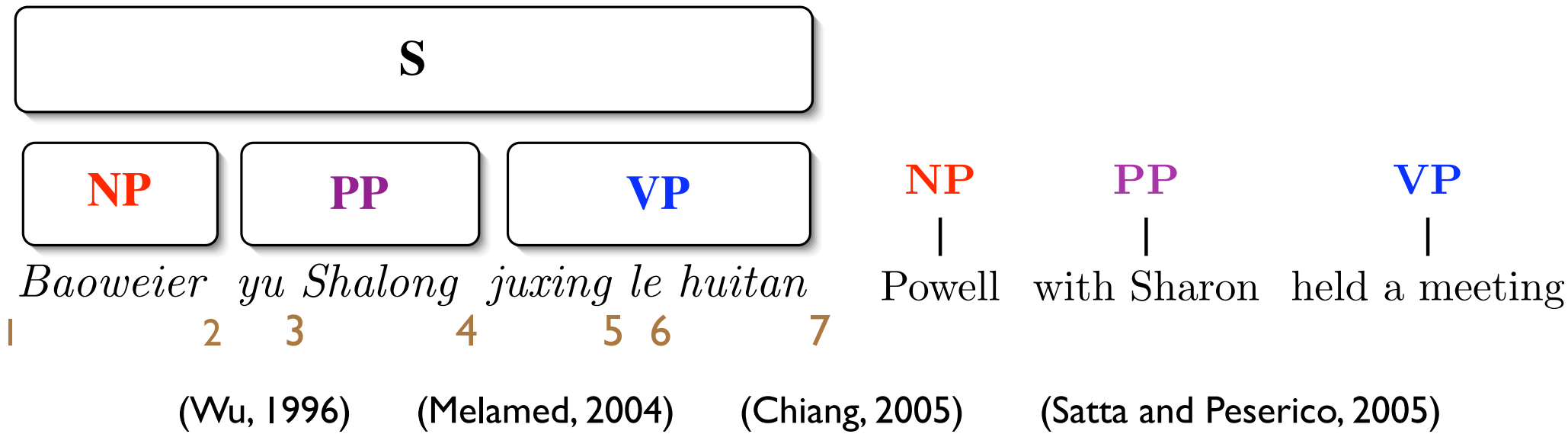
$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{NP}^{(1)}\ \mathbf{PP}^{(2)}\ \mathbf{VP}^{(3)}, \\ \mathbf{NP} &\rightarrow \textit{Baoweier}, \\ \mathbf{PP} &\rightarrow \textit{yu Shalong}, \\ \mathbf{VP} &\rightarrow \textit{juxing le huitan}, \end{aligned}$$


(Wu, 1996) (Melamed, 2004) (Chiang, 2005) (Satta and Peserico, 2005)

# Translation as Parsing

- Many problems in syntax-based MT reduce to parsing
  - here we consider decoding as monolingual parsing

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{NP}^{(1)}\ \mathbf{PP}^{(2)}\ \mathbf{VP}^{(3)}, \\ \mathbf{NP} &\rightarrow \textit{Baoweier}, \\ \mathbf{PP} &\rightarrow \textit{yu Shalong}, \\ \mathbf{VP} &\rightarrow \textit{juxing le huitan}, \end{aligned}$$

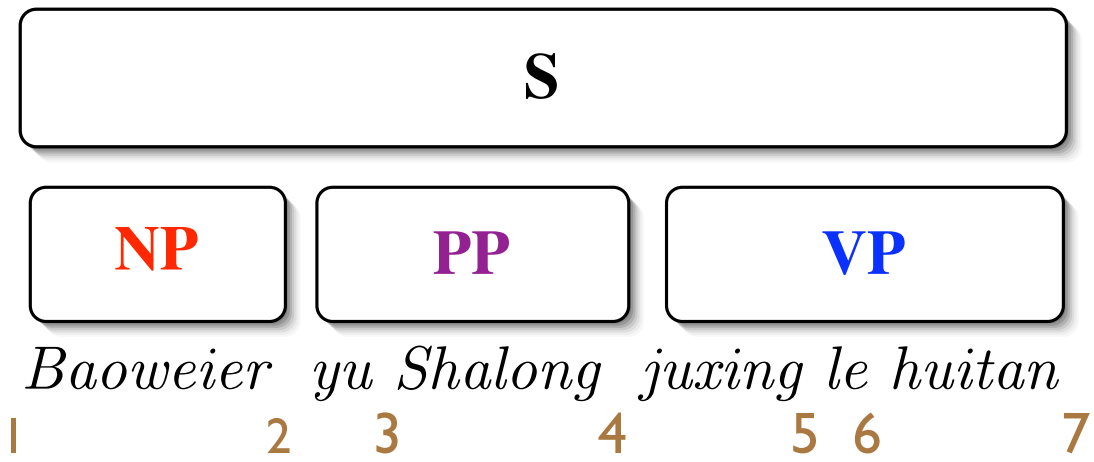


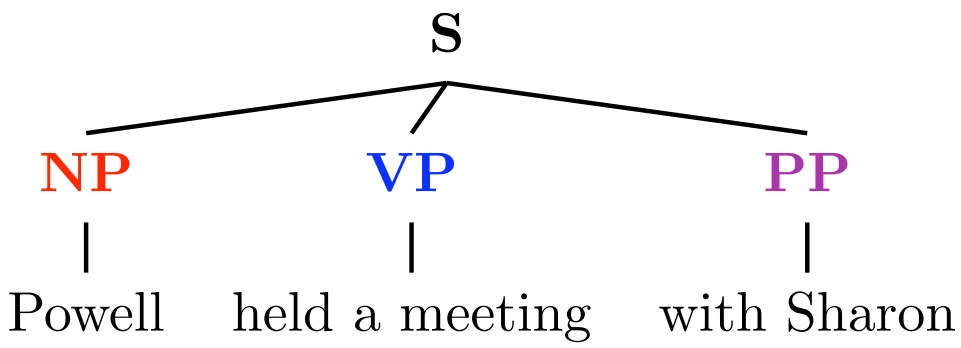

(Wu, 1996) (Melamed, 2004) (Chiang, 2005) (Satta and Peserico, 2005)

# Translation as Parsing

- Many problems in syntax-based MT reduce to parsing
  - here we consider decoding as monolingual parsing

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{NP}^{(1)}\ \mathbf{PP}^{(2)}\ \mathbf{VP}^{(3)}, \\ \mathbf{NP} &\rightarrow \textit{Baoweier}, \\ \mathbf{PP} &\rightarrow \textit{yu Shalong}, \\ \mathbf{VP} &\rightarrow \textit{juxing le huitan}, \end{aligned}$$



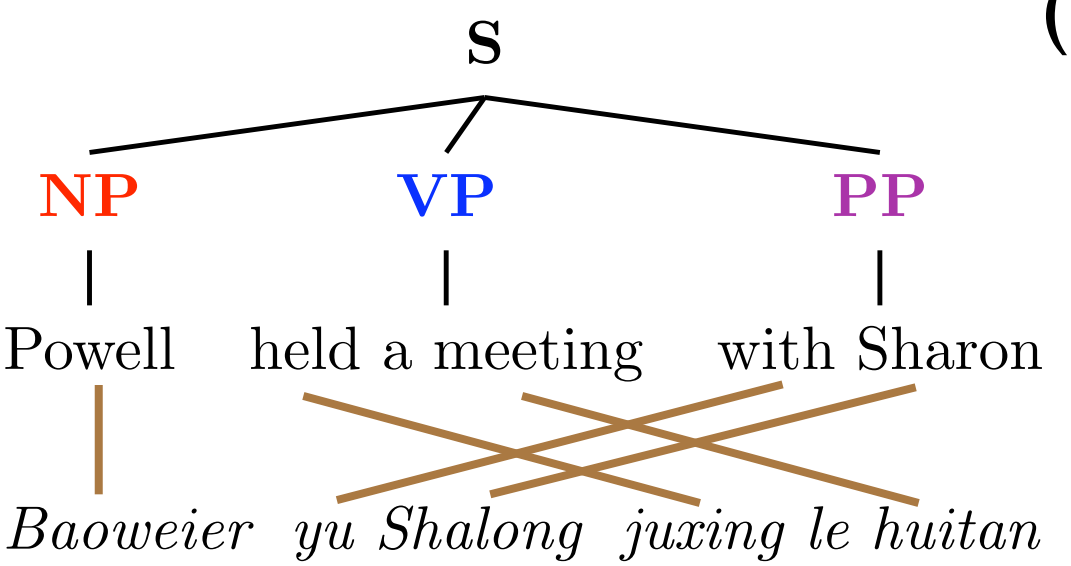




(Wu, 1996) (Melamed, 2004) (Chiang, 2005) (Satta and Peserico, 2005)

# Where are these rules from?

- Induce synchronous grammars from parallel corpora
  - ISI syntax-based system

(e-tree, f-string, alignment)



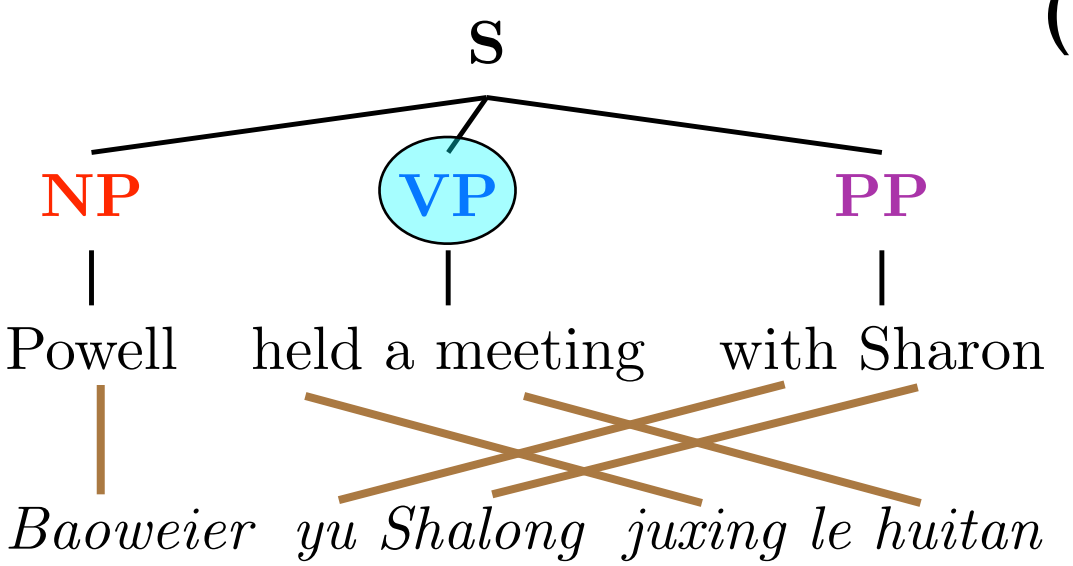

GHKM extraction
(Galley et al., 2004)

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

# Where are these rules from?

- Induce synchronous grammars from parallel corpora
  - ISI syntax-based system

(e-tree, f-string, alignment)



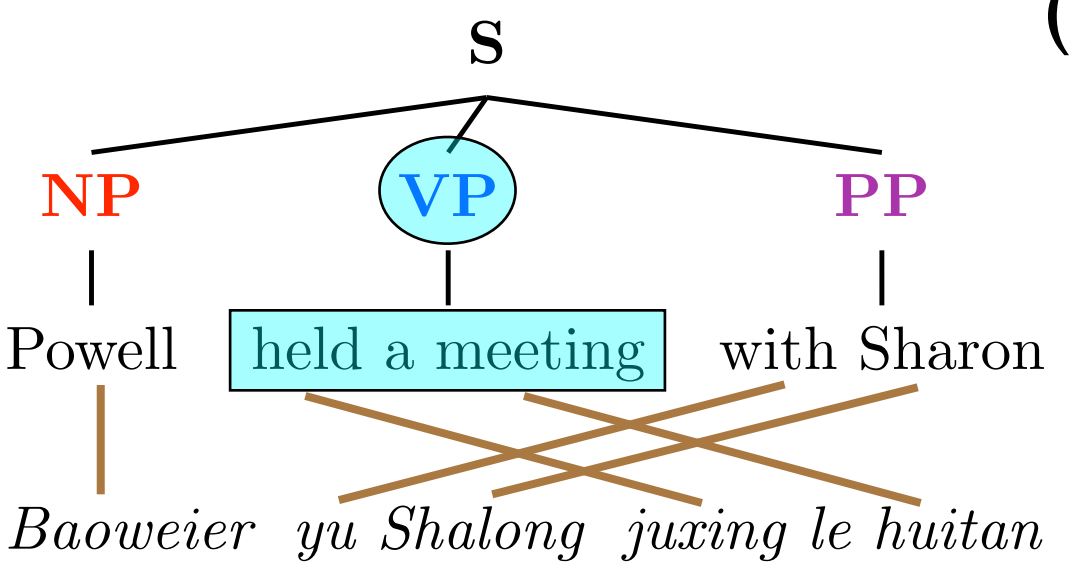

GHKM extraction
(Galley et al., 2004)

| | | | |
|---|---|---|---|
| S | → | $\mathbf{NP}^{(1)}\ \mathbf{PP}^{(2)}\ \mathbf{VP}^{(3)}$, | $\mathbf{NP}^{(1)}\ \mathbf{VP}^{(3)}\ \mathbf{PP}^{(2)}$ |
| NP | → | *Baoweier*, | Powell |
| PP | → | *yu Shalong*, | with Sharon |
| VP | → | *juxing le huitan*, | held a meeting |

# Where are these rules from?

- Induce synchronous grammars from parallel corpora
  - ISI syntax-based system

(e-tree, f-string, alignment)



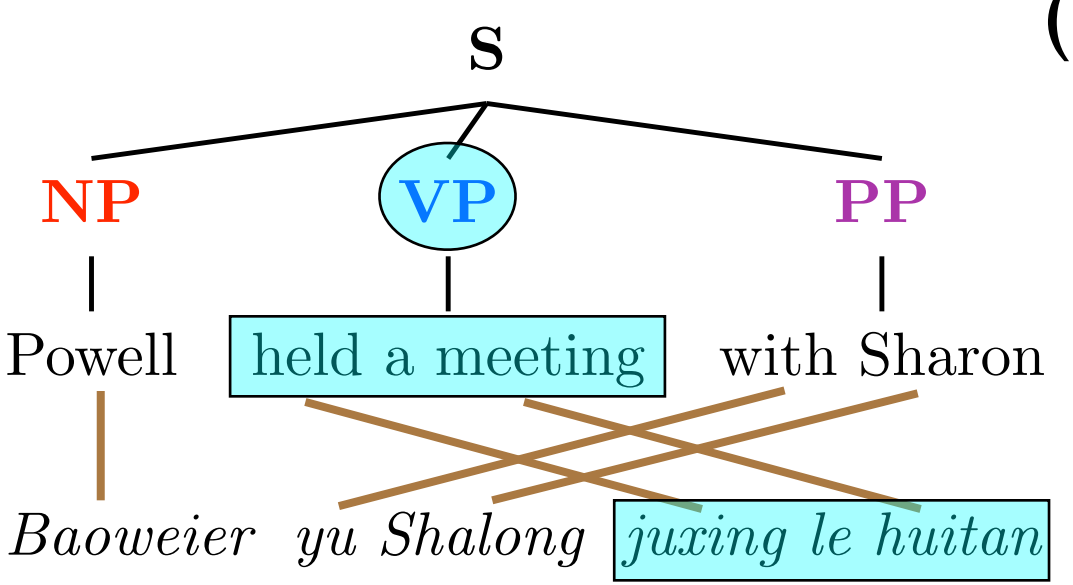

GHKM extraction
(Galley et al., 2004)

| | | | |
|---|---|---|---|
| S | → | $\mathbf{NP}^{(1)}\ \mathbf{PP}^{(2)}\ \mathbf{VP}^{(3)}$, | $\mathbf{NP}^{(1)}\ \mathbf{VP}^{(3)}\ \mathbf{PP}^{(2)}$ |
| NP | → | *Baoweier*, | Powell |
| PP | → | *yu Shalong*, | with Sharon |
| VP | → | *juxing le huitan*, | held a meeting |

# Where are these rules from?

- Induce synchronous grammars from parallel corpora
  - ISI syntax-based system

(e-tree, f-string, alignment)

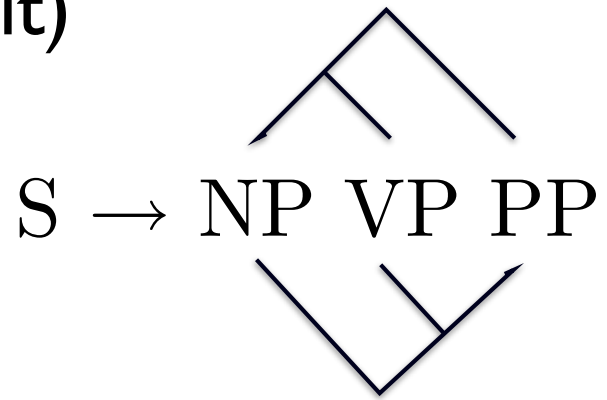


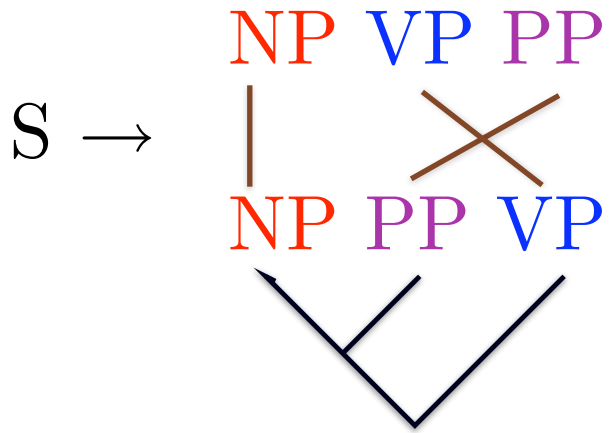

GHKM extraction
(Galley et al., 2004)

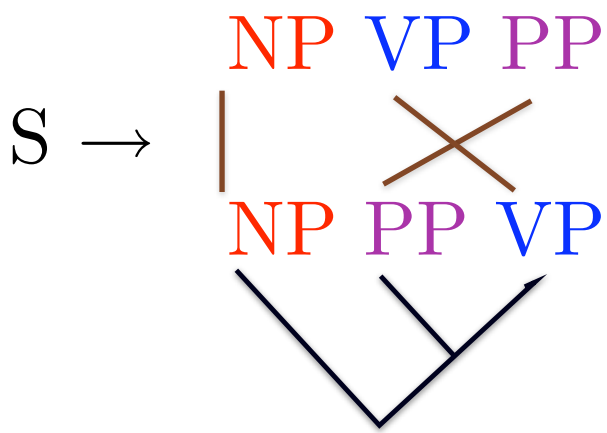
| | | | |
|---|---|---|---|
| S | → | $\mathbf{NP}^{(1)}\ \mathbf{PP}^{(2)}\ \mathbf{VP}^{(3)}$, | $\mathbf{NP}^{(1)}\ \mathbf{VP}^{(3)}\ \mathbf{PP}^{(2)}$ |
| NP | → | *Baoweier*, | Powell |
| PP | → | *yu Shalong*, | with Sharon |
| VP | → | *juxing le huitan*, | held a meeting |

# CKY / binarization

- Efficient parsing requires binary-branching grammars
  - e.g., CKY (explicit) or Earley (implicit)
- Many ways to binarize
  - They all work fine



# Integrating Language Models

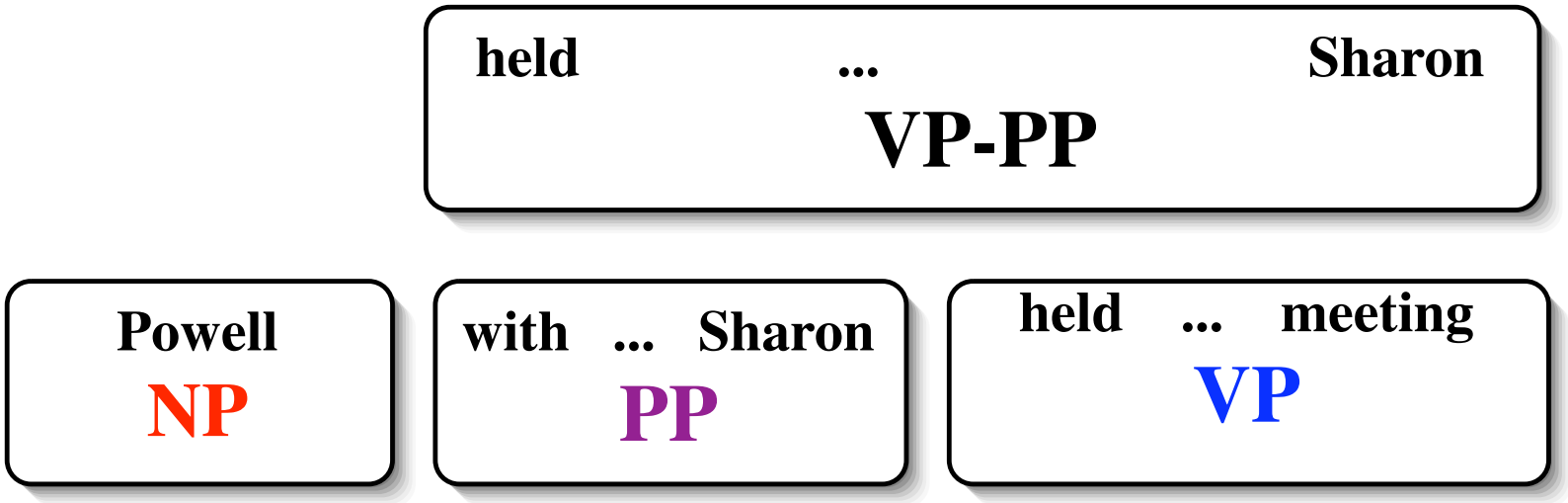
- dynamic programming algorithm
  - remembering "boundary words"
- different binarization has very different effect


*Baoweier yu Shalong juxing le huitan*

(Wu, 1996) (Chiang, 2005)

# Integrating Language Models

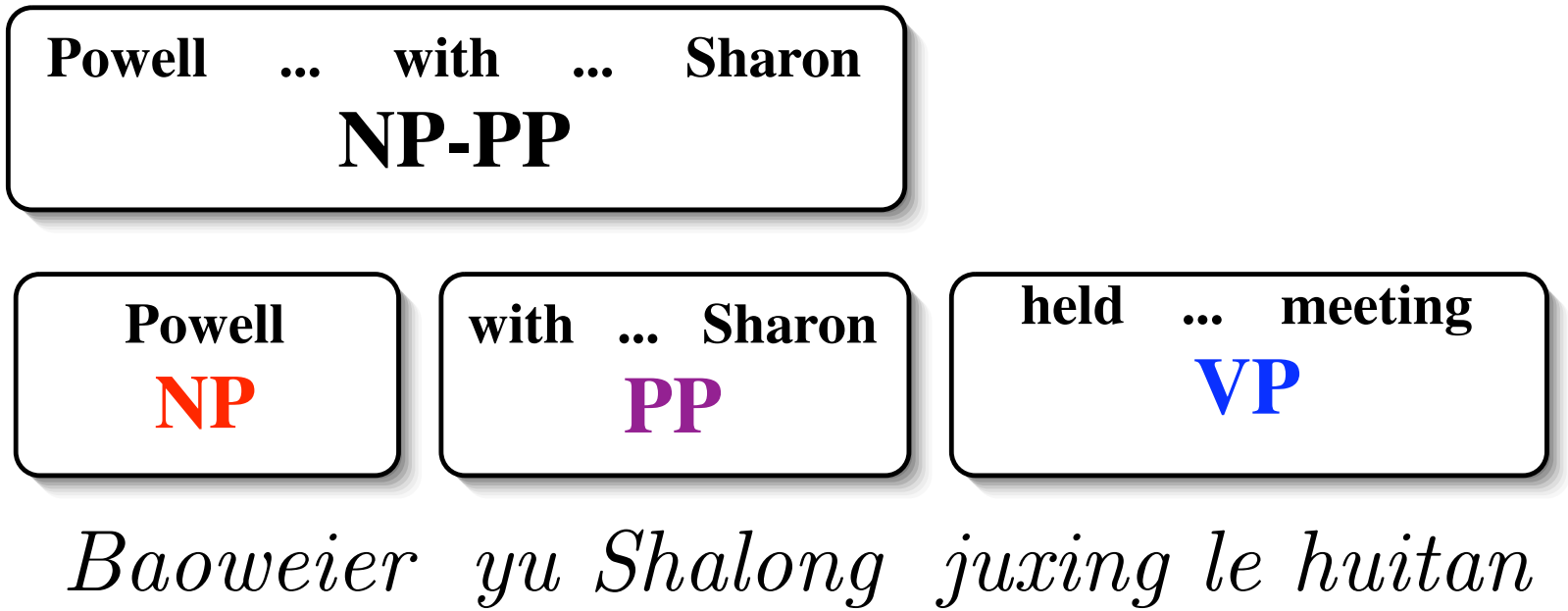
- dynamic programming algorithm
  - remembering "boundary words"
- different binarization has very different effect



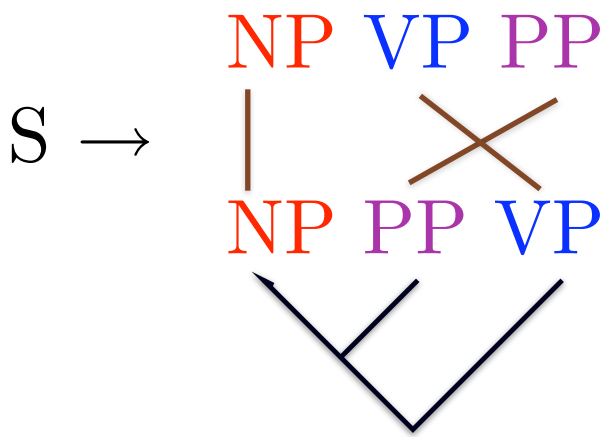

(Wu, 1996) (Chiang, 2005)

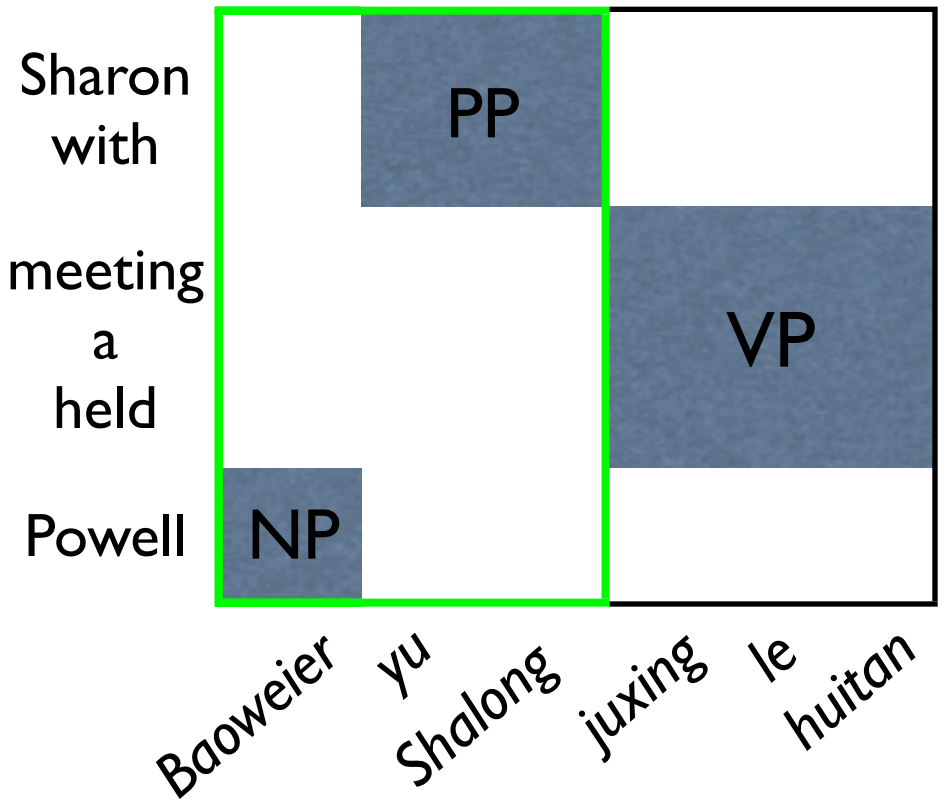
# Integrating Language Models

- dynamic programming algorithm
  - remembering "boundary words"
- different binarization has very different effect

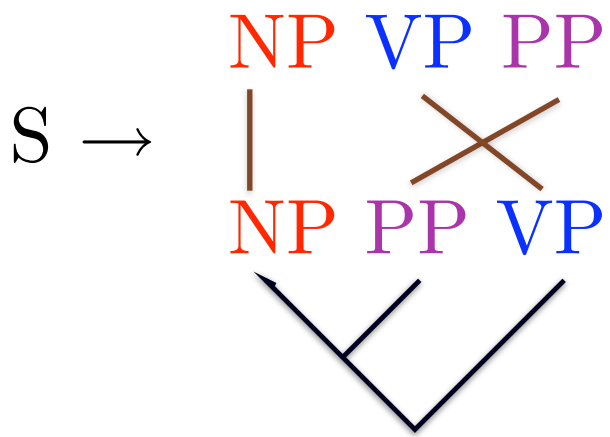


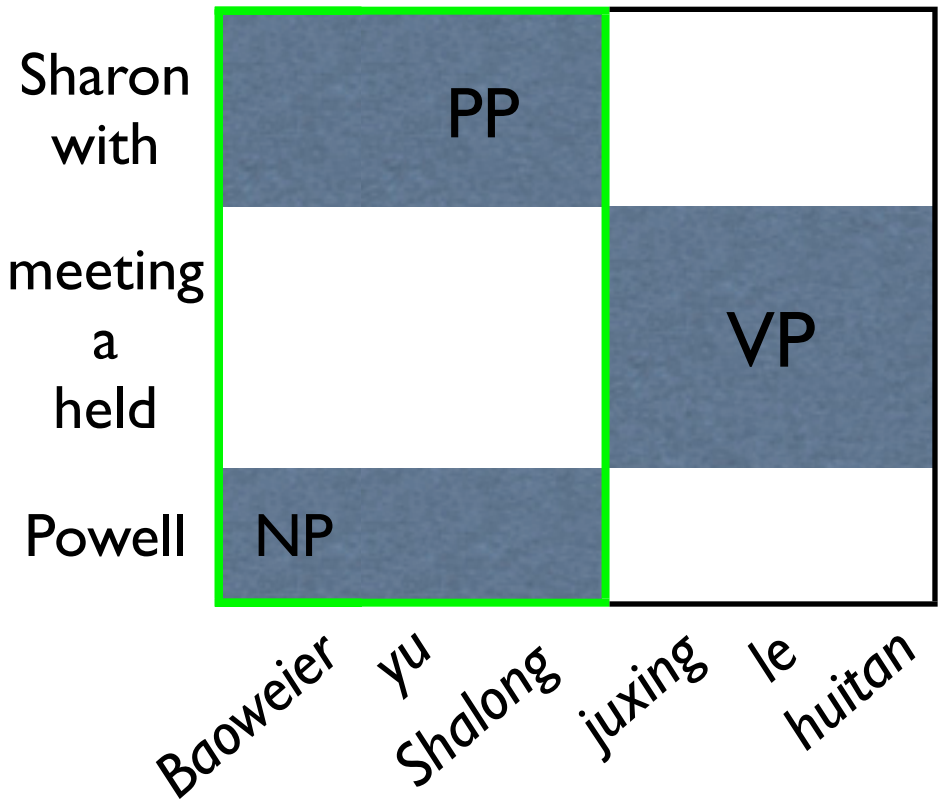
# Binarization in 2D



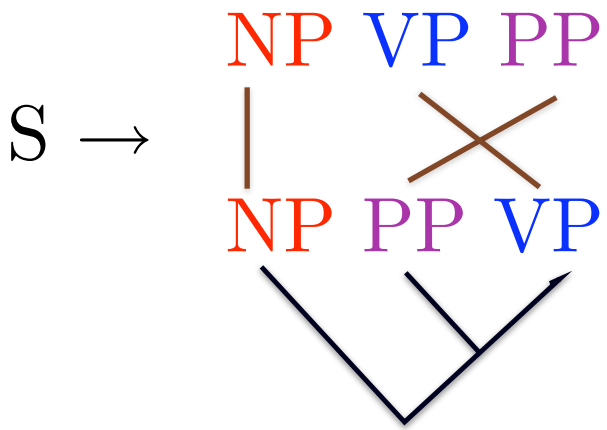



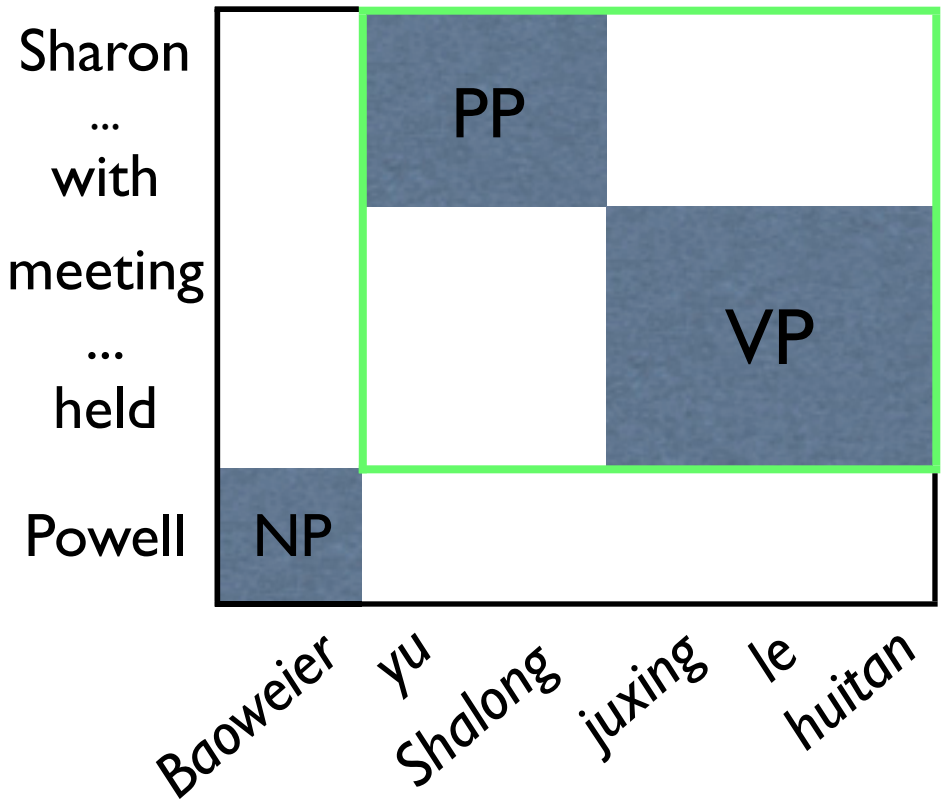
# Binarization in 2D



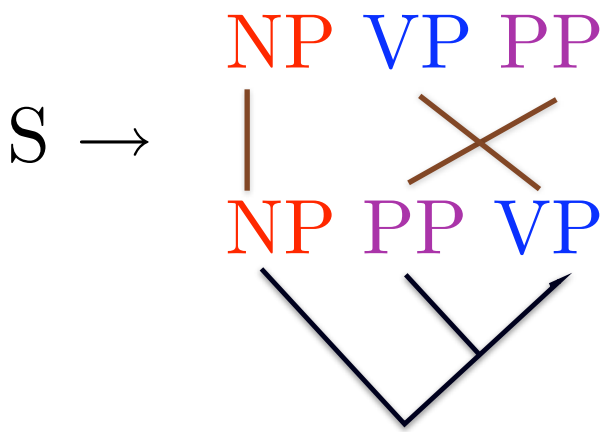



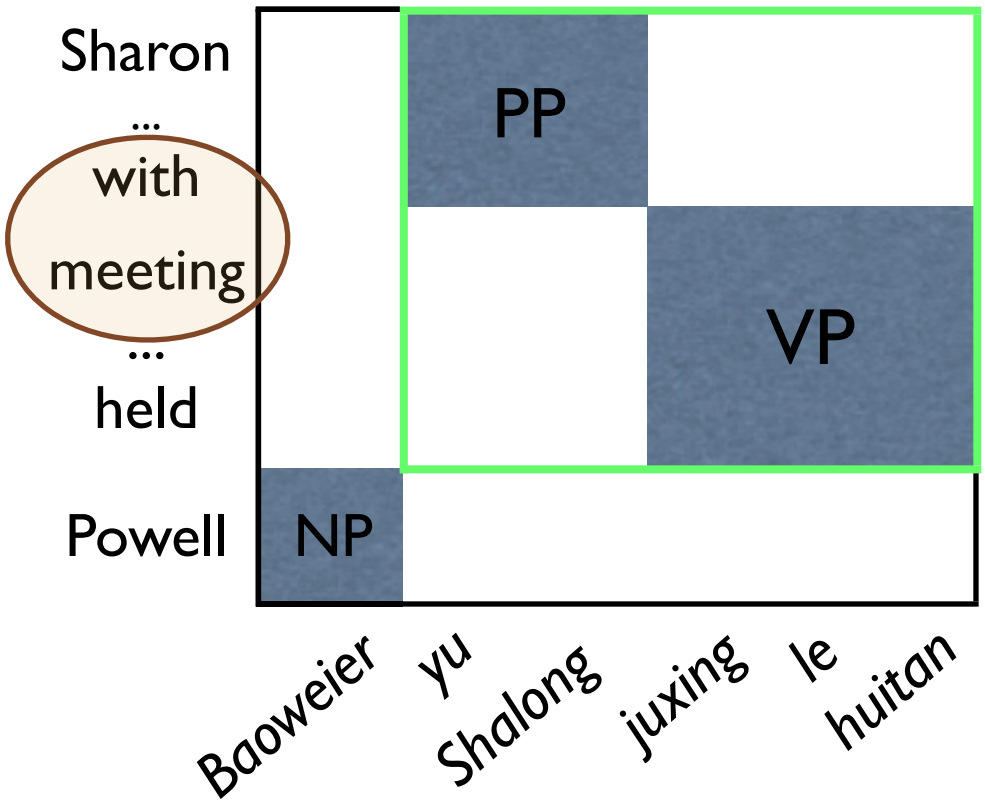
# Binarization in 2D



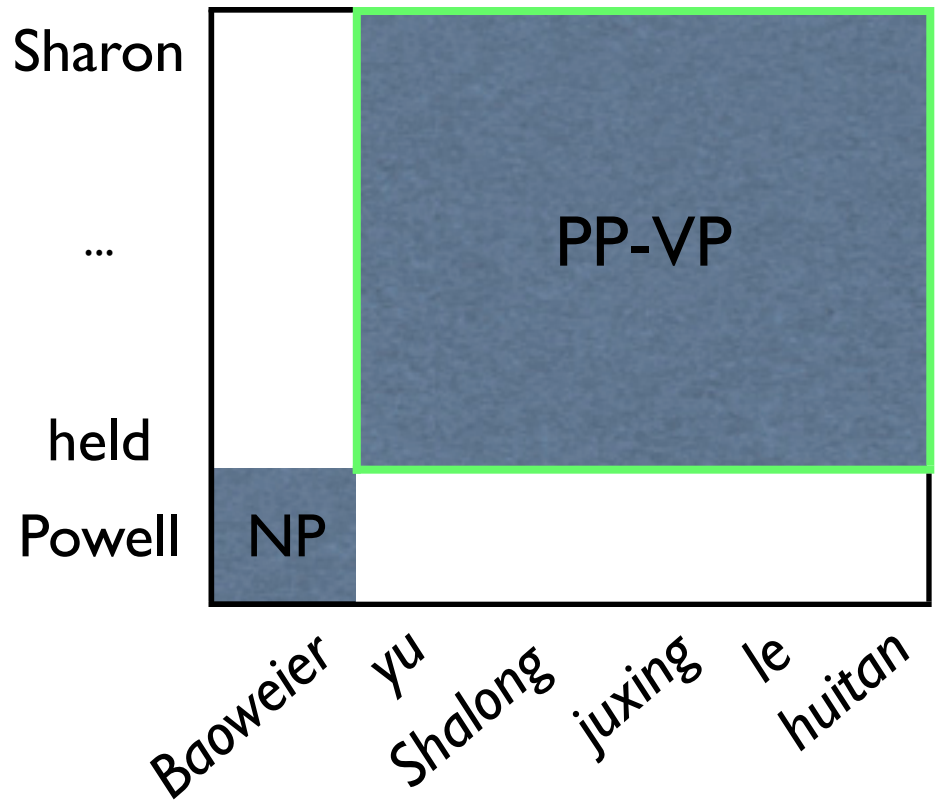



# Binarization in 2D



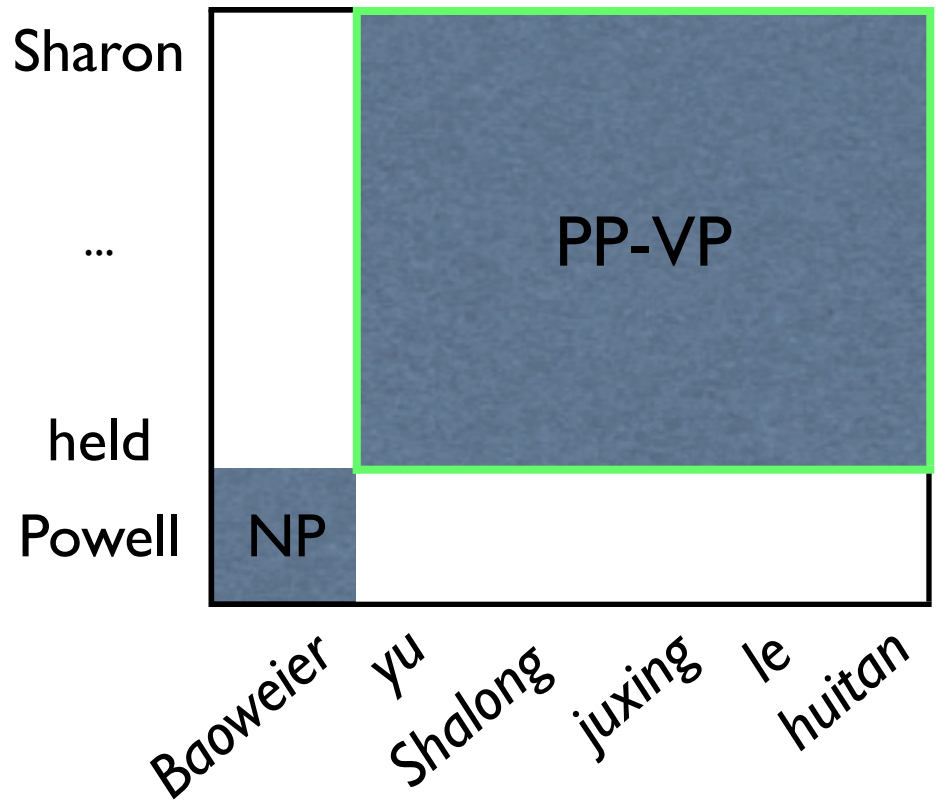



# Binarization in 2D



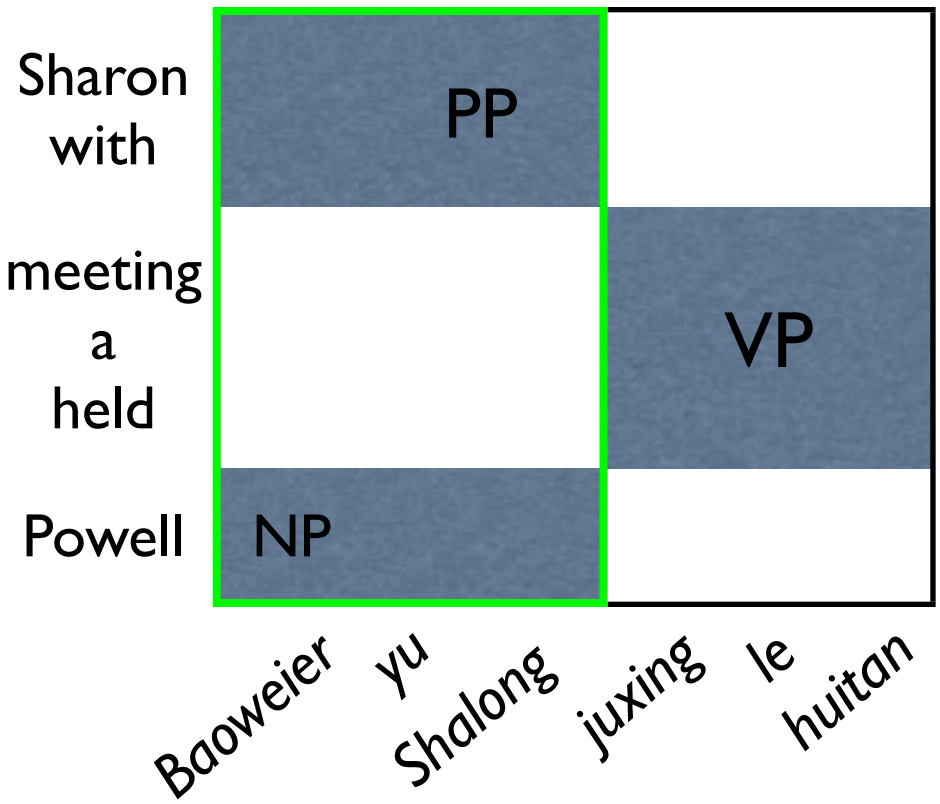
# Binarization in 2D

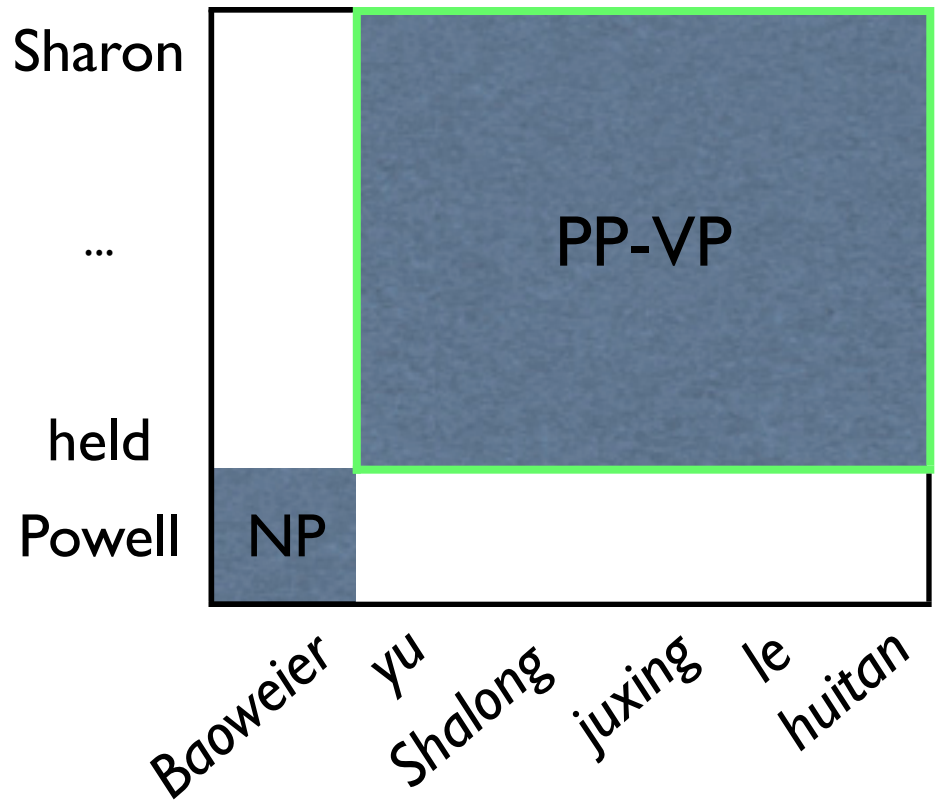
- good binarization: can score n-gram combo costs ASAP; only storing two (sets of) boundary words



# Binarization in 2D

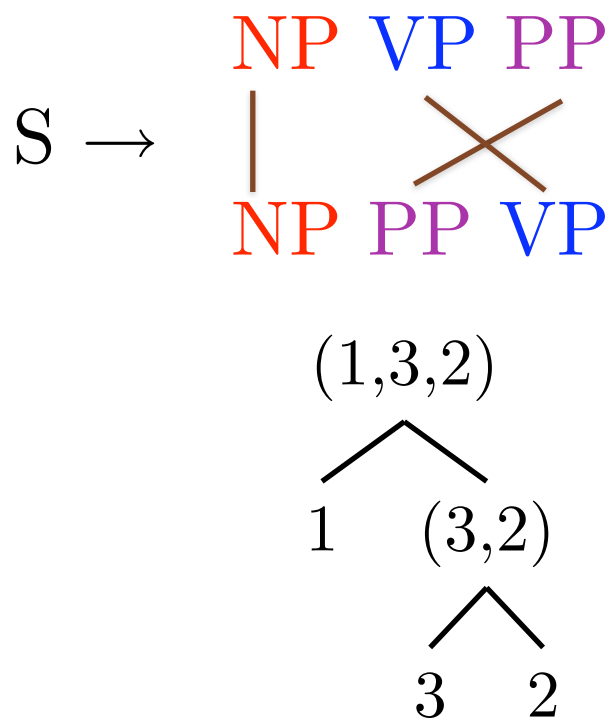
- good binarization: can score n-gram combo costs ASAP; only storing two (sets of) boundary words
- bad binarization: have to wait till all nonterminals of a rule have been recognized; storing all LM words

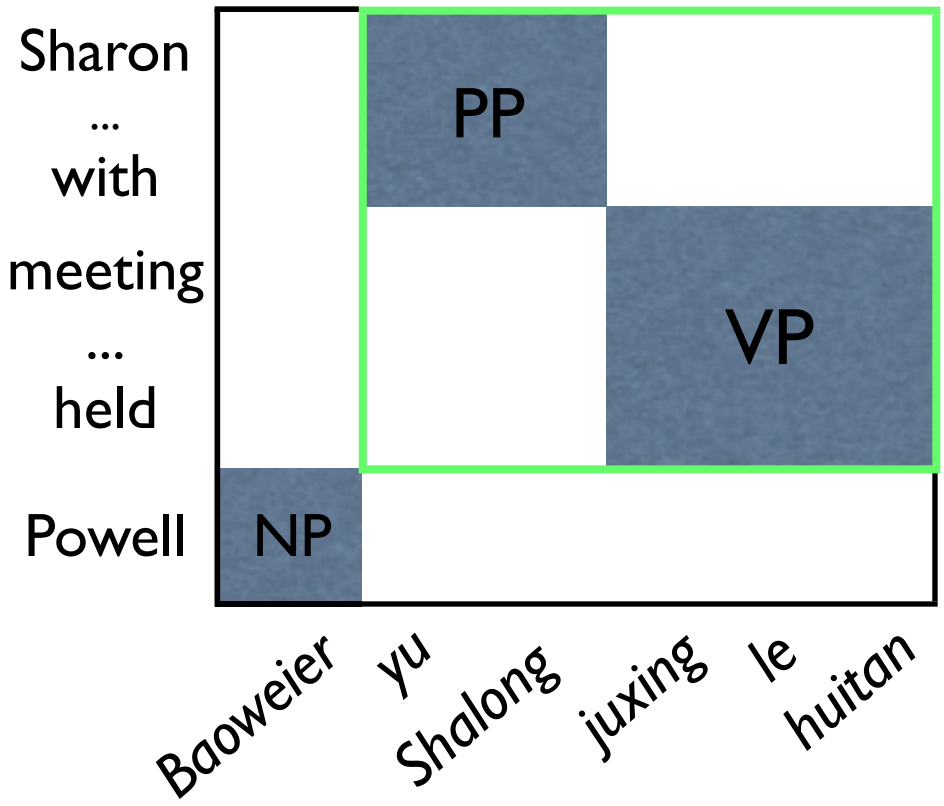


# Synchronous Binarization

- converts an SCFG into binary-branching
- intuition: should have contiguous spans on both sides
- binarizing the permutation of nonterminals



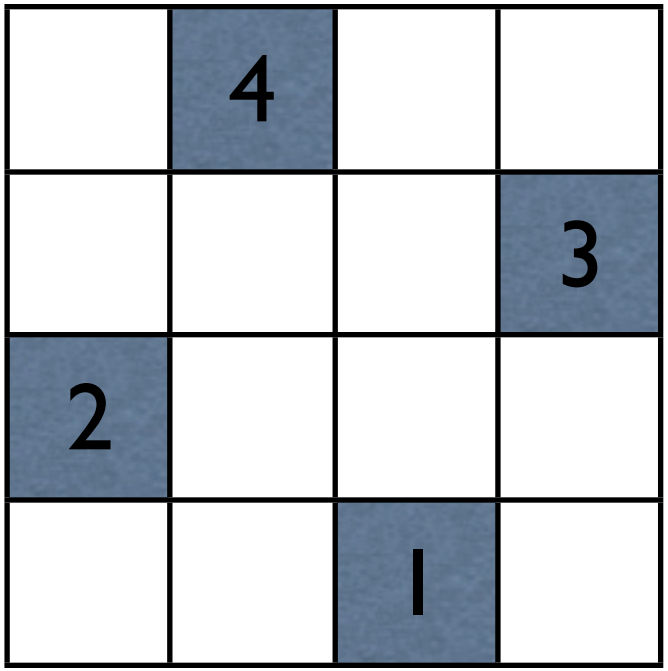
# Is every rule binarizable?

# Is every rule binarizable?

- for n < 4, yes

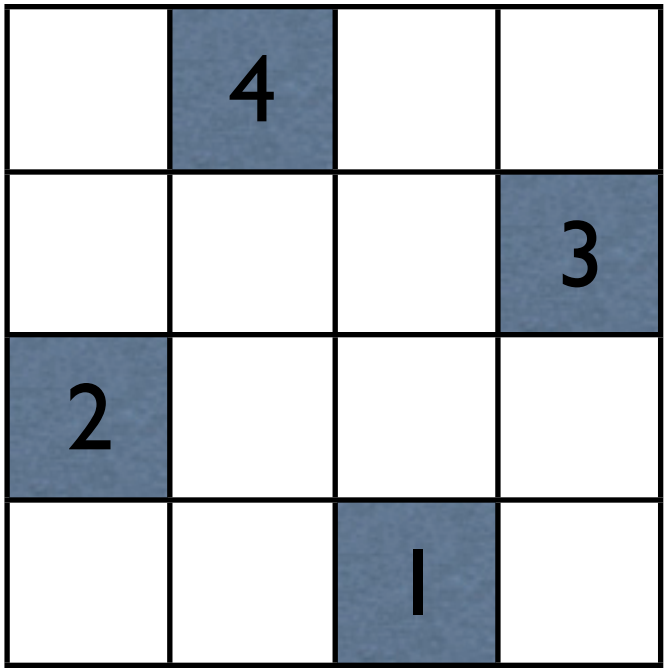
# Is every rule binarizable?

- for n < 4, yes
- for n >= 4, many permutations non-binarizable



# Is every rule binarizable?

- for n < 4, yes
- for n >= 4, many permutations non-binarizable
- Q: are these bad cases linguistically motivated?



# Is every rule binarizable?

- for n < 4, yes
- for n >= 4, many permutations non-binarizable
- Q: are these bad cases linguistically motivated?
  - ITG hypothesis: no

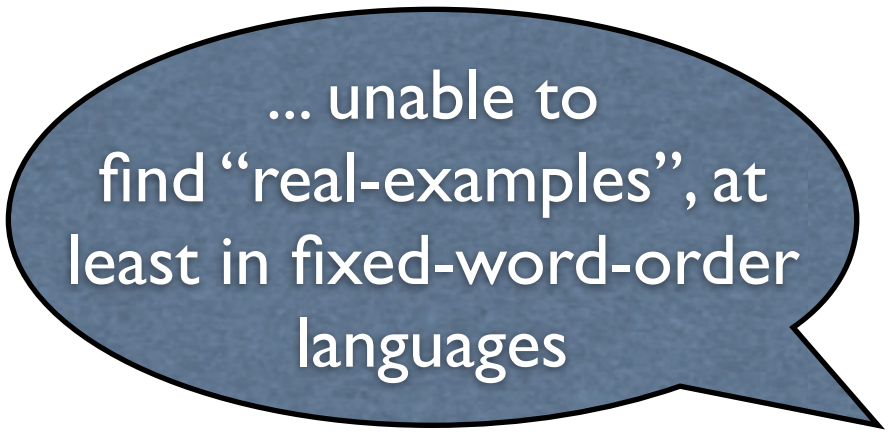
... unable to find "real-examples", at least in fixed-word-order languages

Wu (1997)

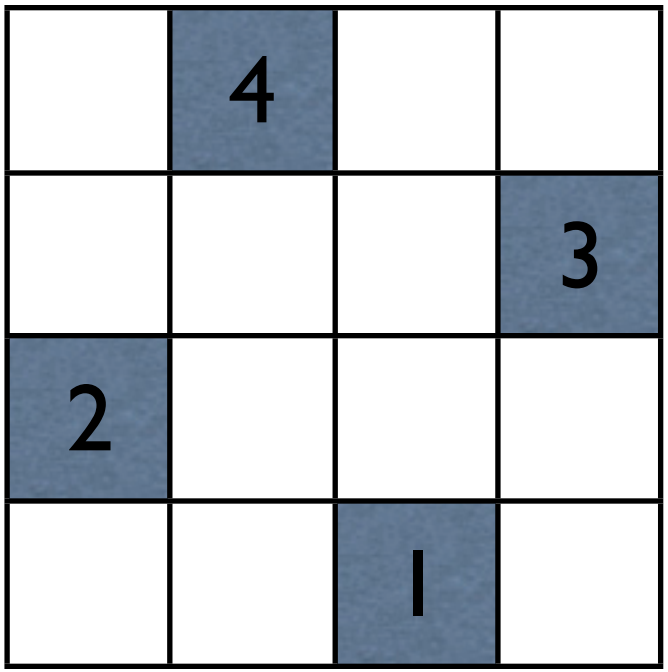
| | | | |
|---|---|---|---|
| | 4 | | |
| | | | 3 |
| 2 | | | |
| | | 1 | |

# Is every rule binarizable?

- for n < 4, yes
- for n >= 4, many permutations non-binarizable
- Q: are these bad cases linguistically motivated?
  - ITG hypothesis: no
  - we will answer it empirically

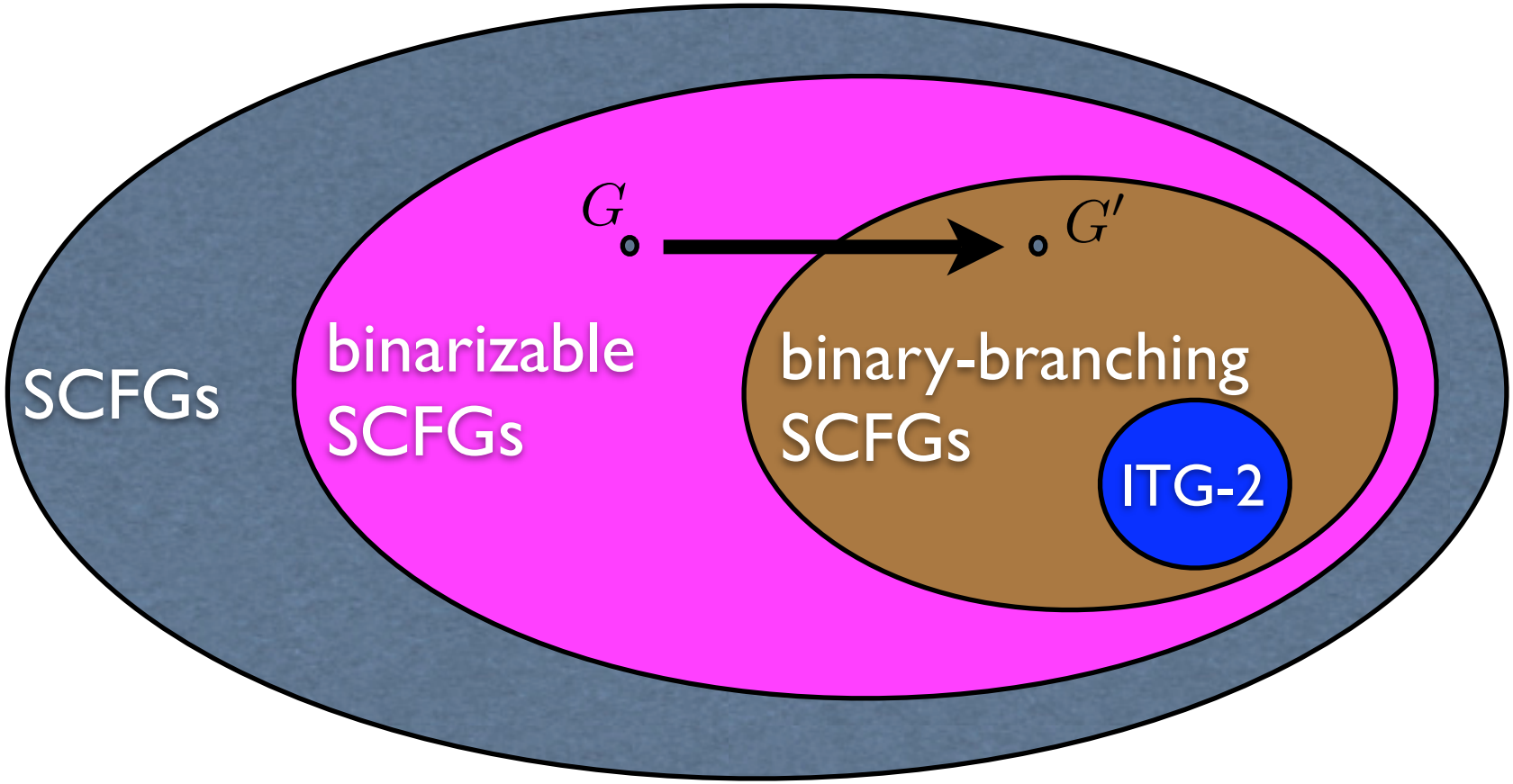



Wu (1997)



# The Big Picture

- synchronous binarization
  - *n*-ary SCFG *G* => binary-branching *G'*, if possible



# How to find good binarizations?

# Naïve Algorithm

# Naïve Algorithm

- Try all binary bracketings
  - finds all possible binarizations
  - exponential complexity (Catalan number)

# Naïve Algorithm

- Try all binary bracketings
  - finds all possible binarizations
  - exponential complexity (Catalan number)

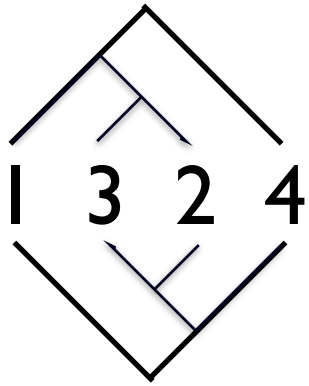
(1 ((3 2) 4))
(1 (3 (2 4)))
(((1 3) 2) 4)
...

# Naïve Algorithm

- Try all binary bracketings
  - finds all possible binarizations
  - exponential complexity (Catalan number)

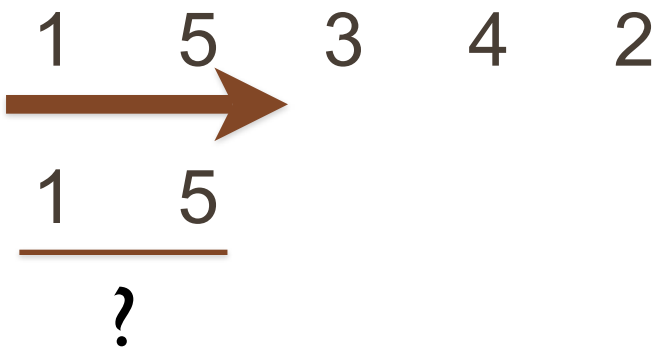
(1 ((3 2) 4))
(1 (3 (2 4)))
(((1 3) 2) 4)
...

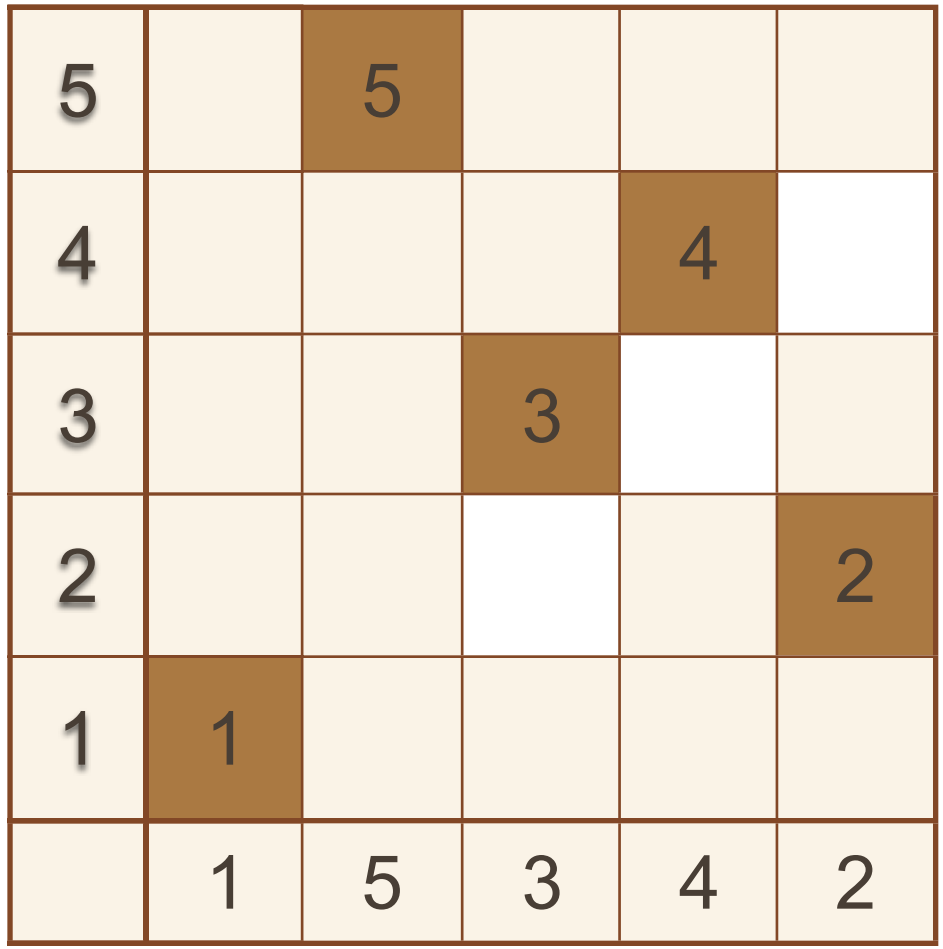
- However
  - we need a much faster algorithm!
  - one synchronous binarization is enough
  - we prefer unique solution for each permutation
    - sharing sub-binarizations



# Linear-time Algorithm

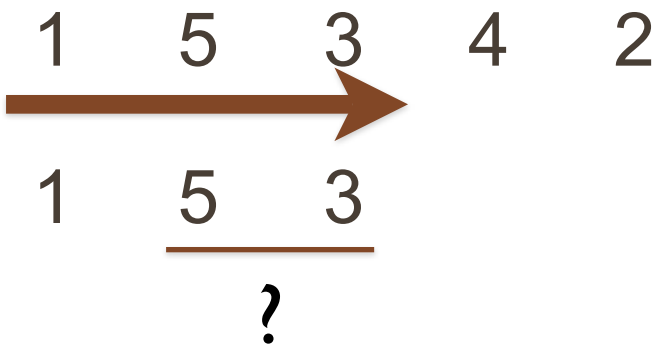
- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

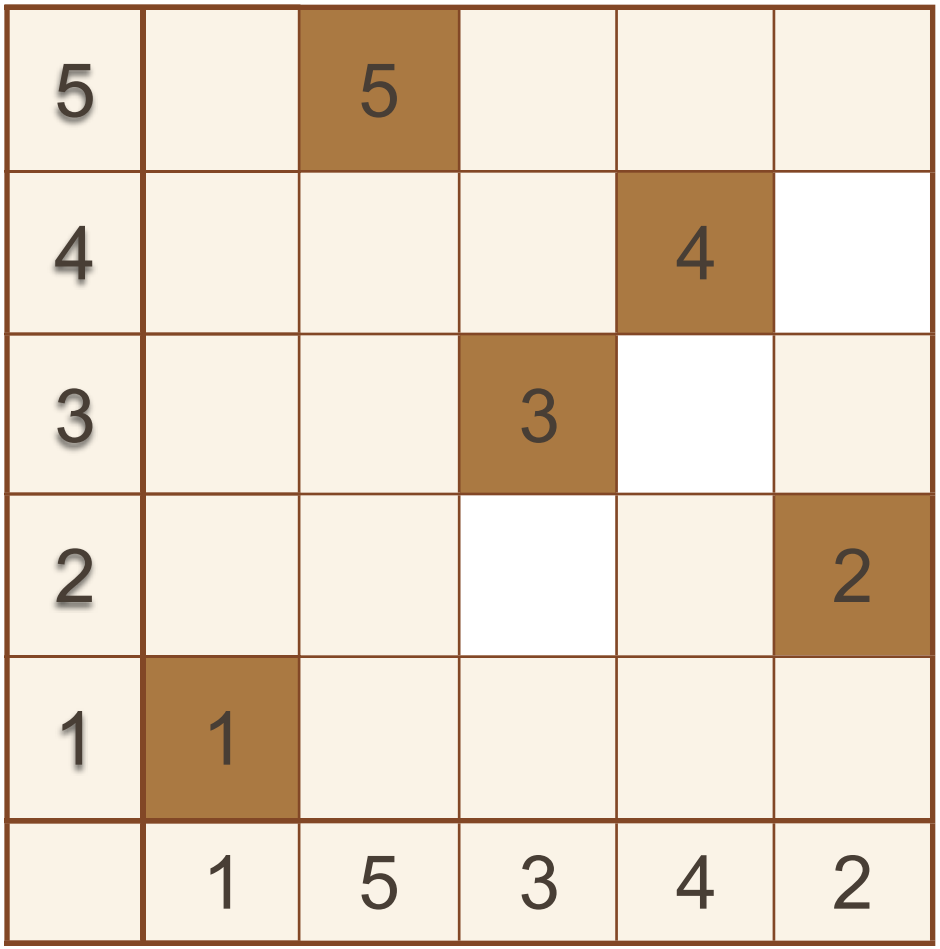


# Linear-time Algorithm

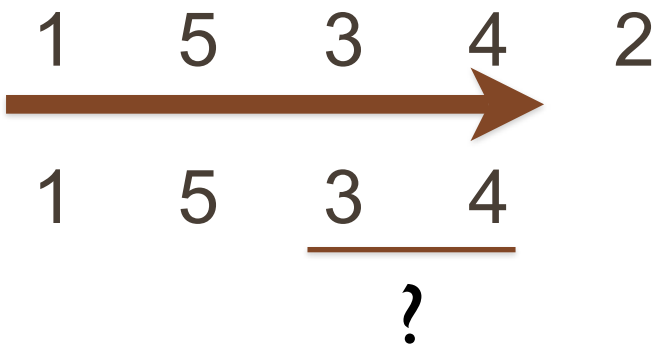
- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

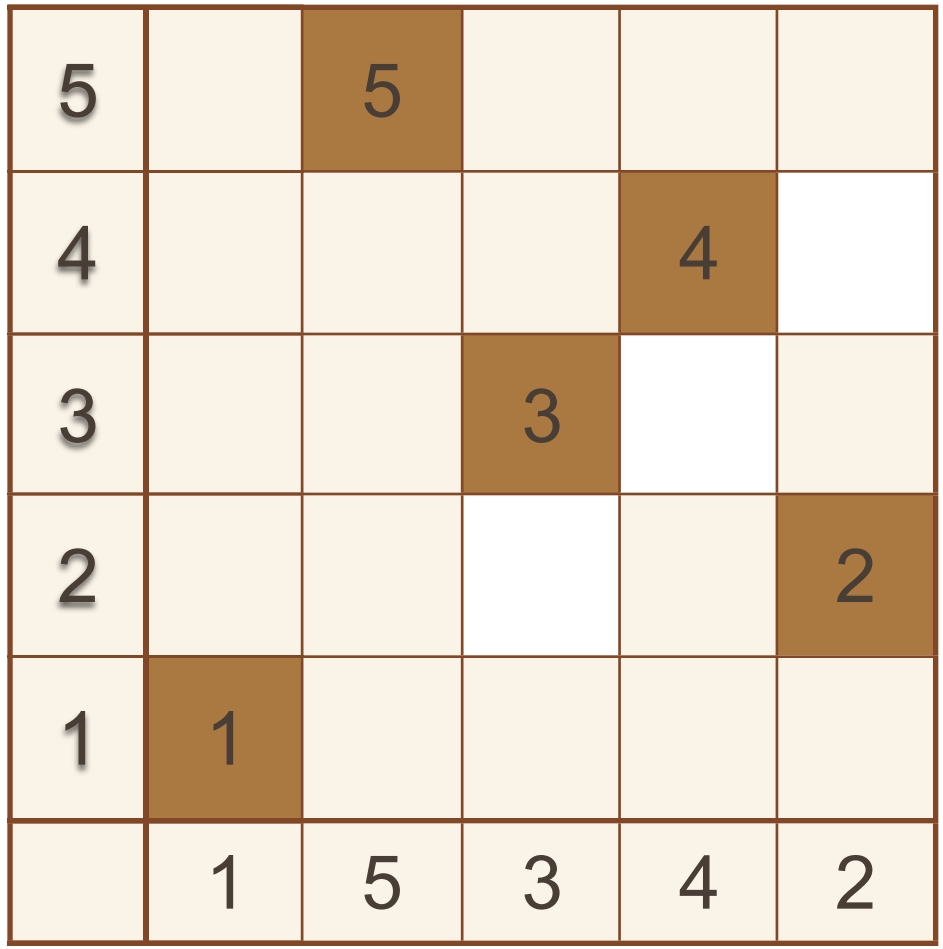


# Linear-time Algorithm

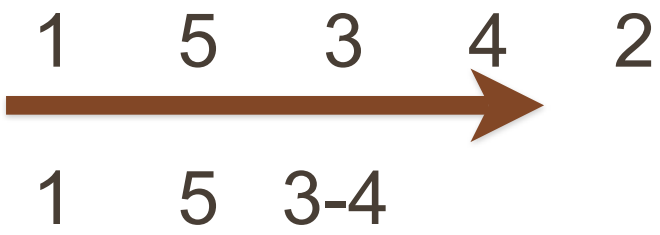
- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack



# Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

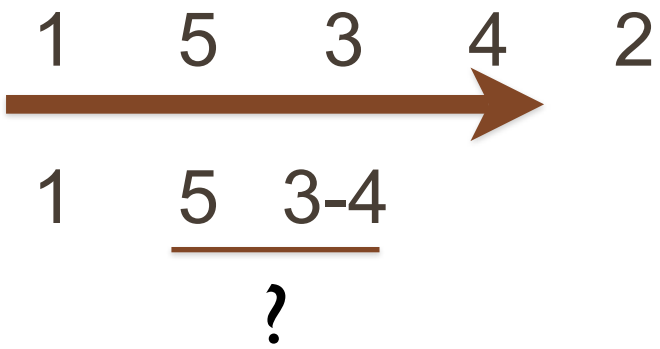



[3, 4]

| | | | | |
|---|---|---|---|---|
| 5 | | 5 | | |
| 4<br>\|<br>3 | | | 3-4 | |
| 2 | | | | 2 |
| 1 | 1 | | | |
| | 1 | 5 | 3-4 | 2 |

# Linear-time Algorithm

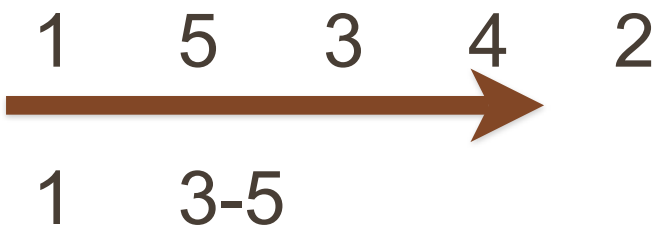
- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack



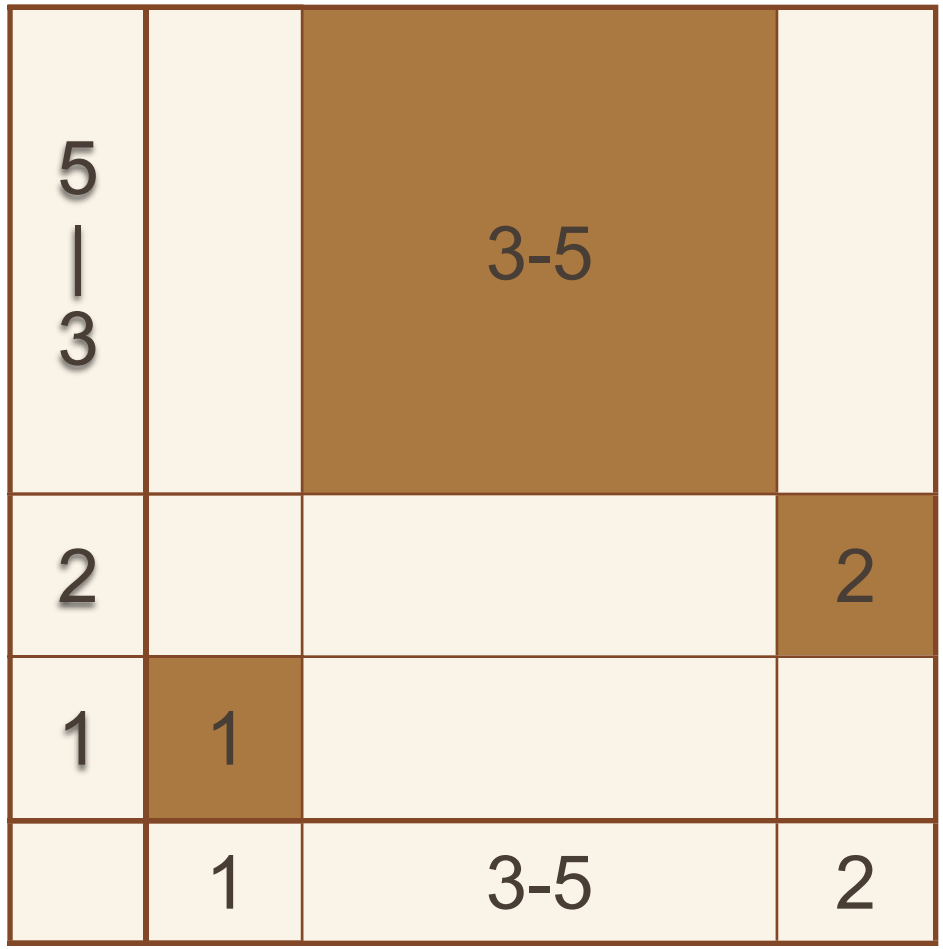

| | | | | |
|---|---|---|---|---|
| 5 | | 5 | | |
| 4<br>\|<br>3 | | | 3-4 | |
| 2 | | | | 2 |
| 1 | 1 | | | |
| | 1 | 5 | 3-4 | 2 |

# Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

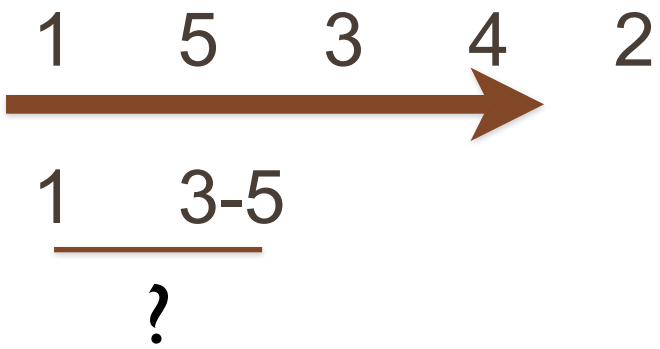


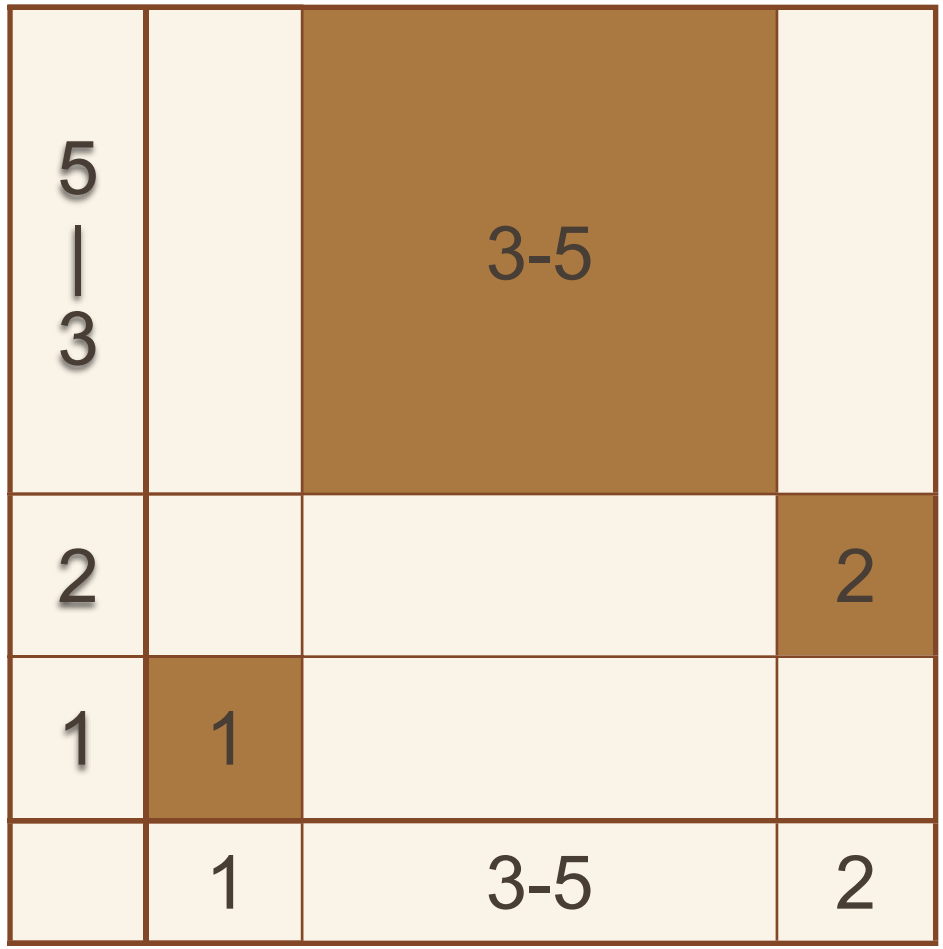

<5, [3, 4]>



# Linear-time Algorithm

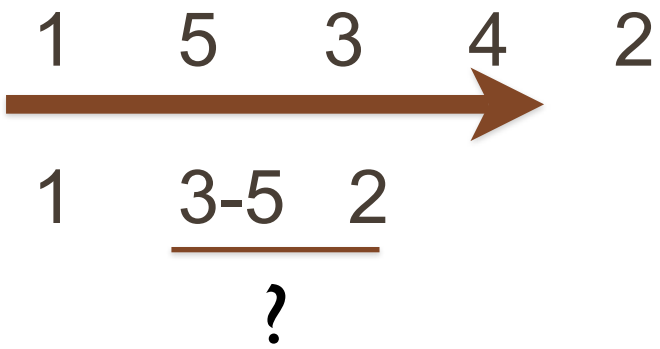
- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

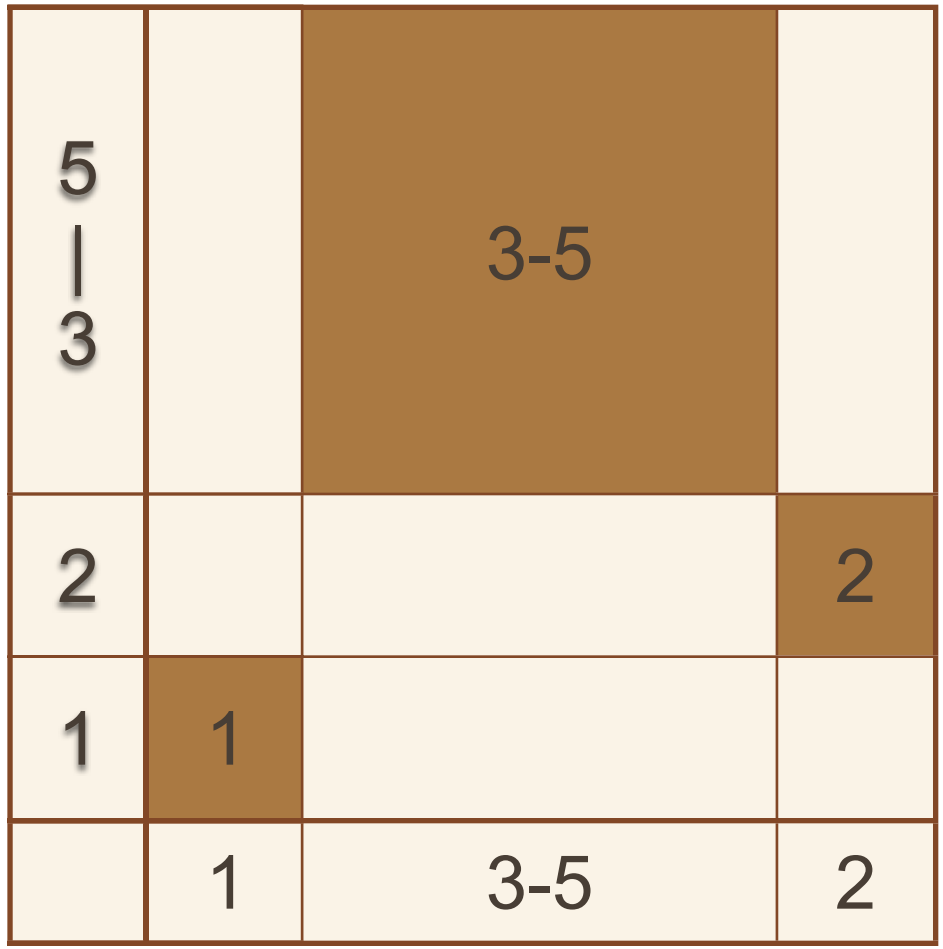


# Linear-time Algorithm

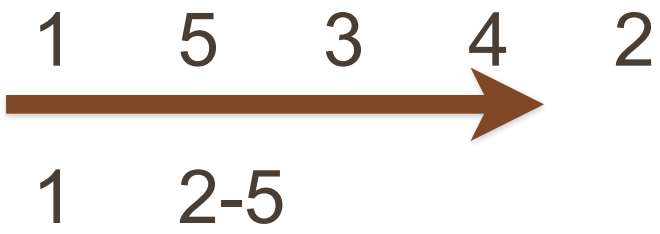
- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack



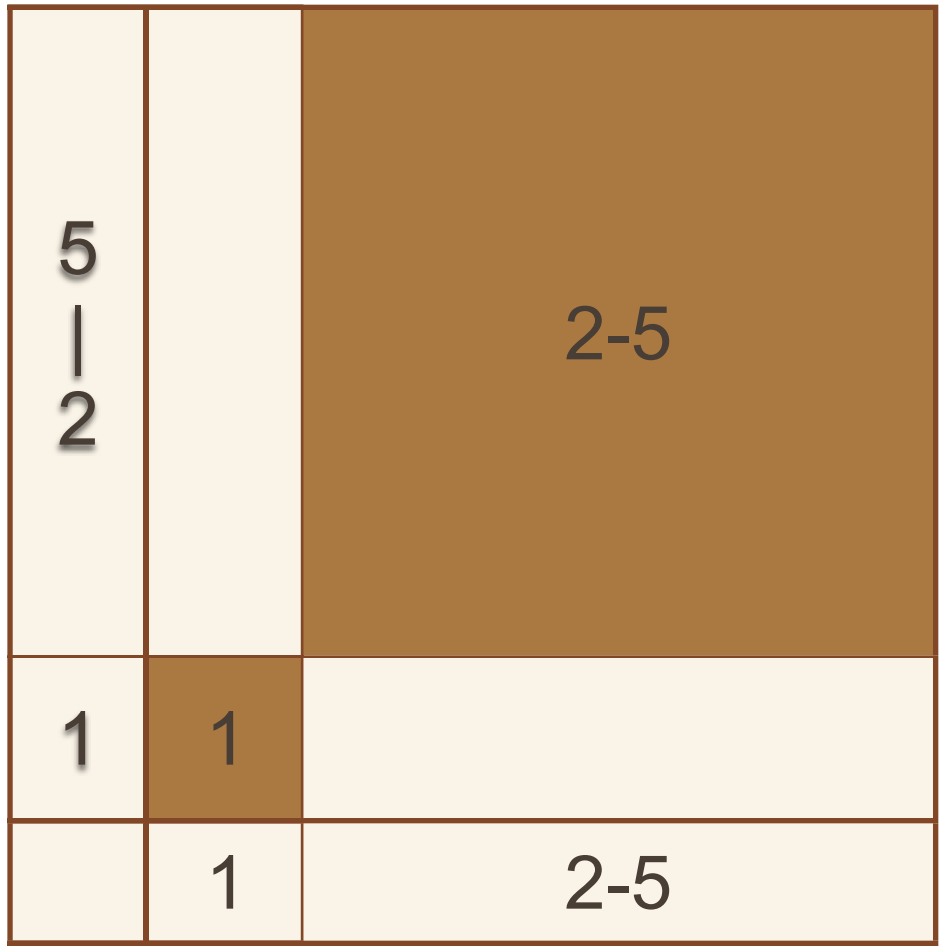



# Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

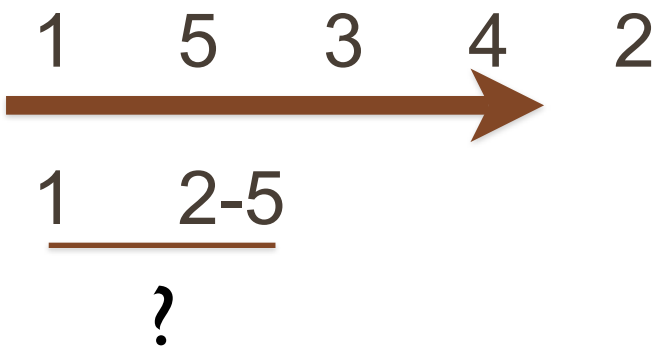


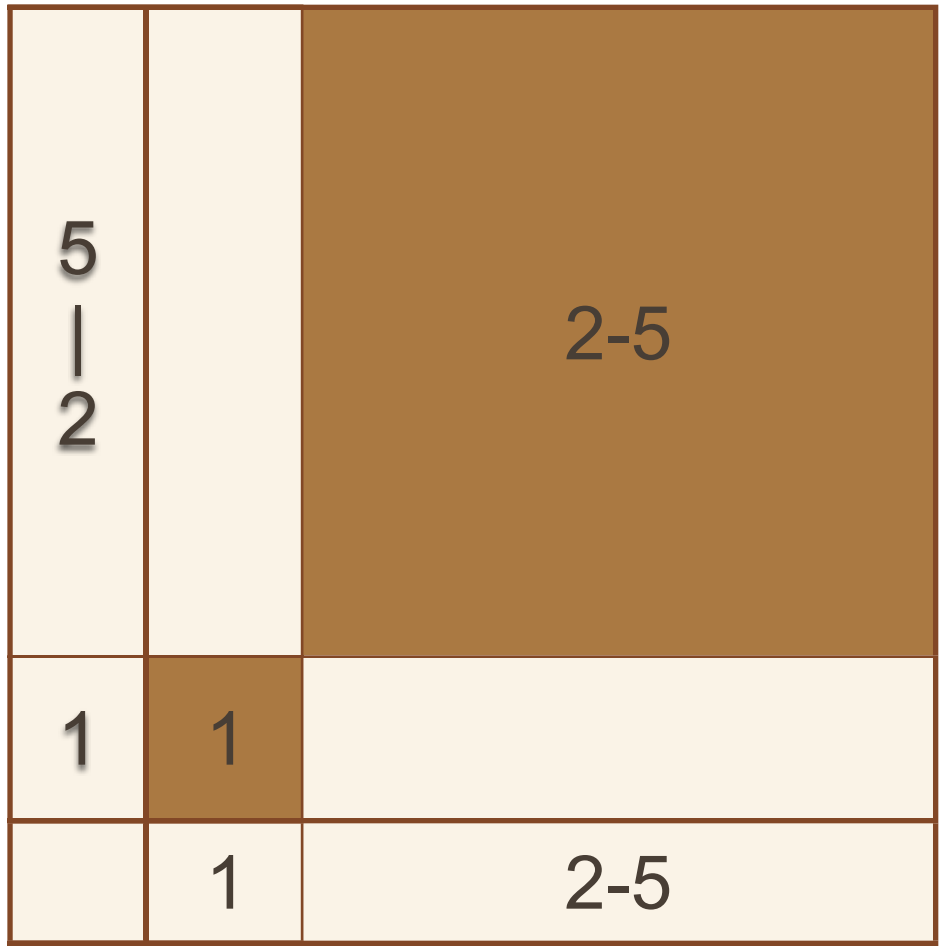

<<5, [3, 4]>, 2>



# Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

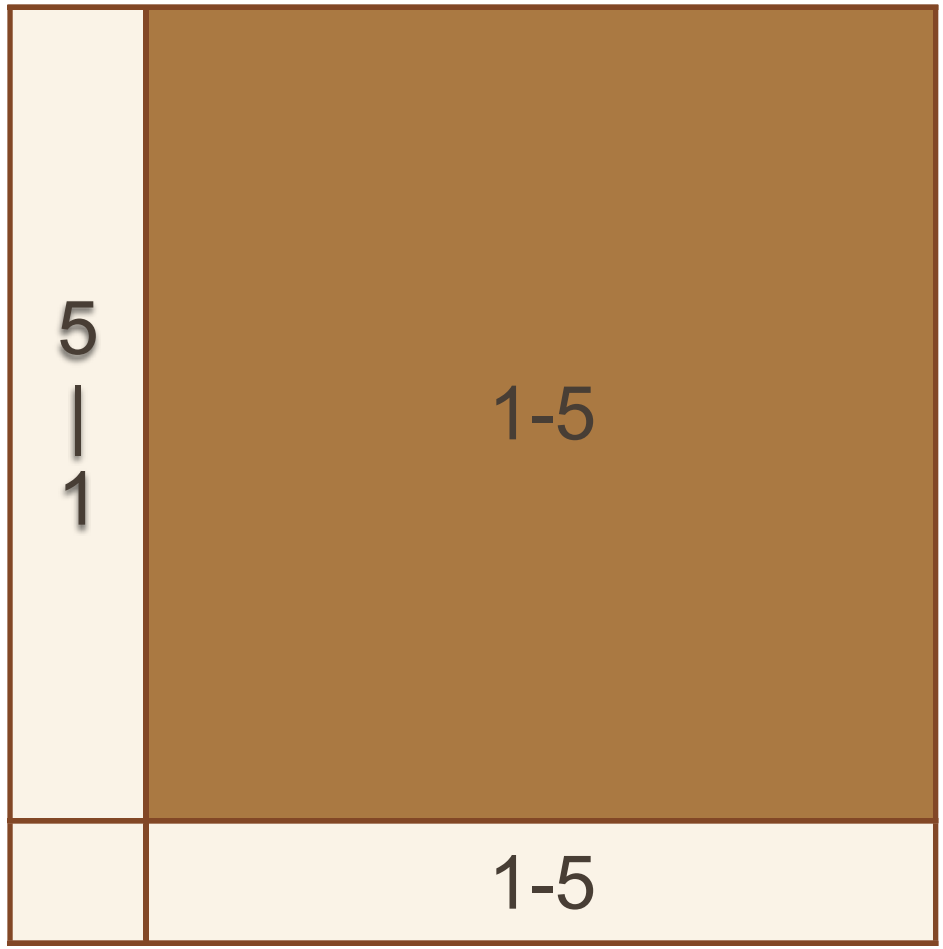


# Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack


[1, <<5, [3, 4]>, 2>]

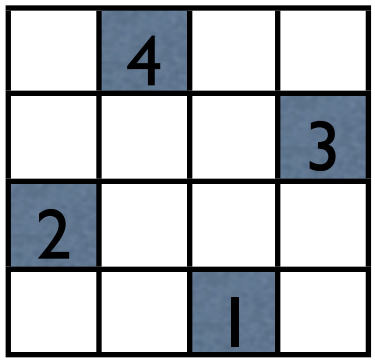


# Properties of the Algorithm

- runs in $O(n)$ time
  - $n$ shifts and at most $(n-1)$ reductions
- the permutation is synchronously binarizable
  - iff. it is reduced to exactly one element at the end
  - and in such case, the algorithm will return
    - the unique left-heavy binarization tree
    - corresponding to the non-ambiguous ITG (Wu, 1997)

# Properties of the Algorithm

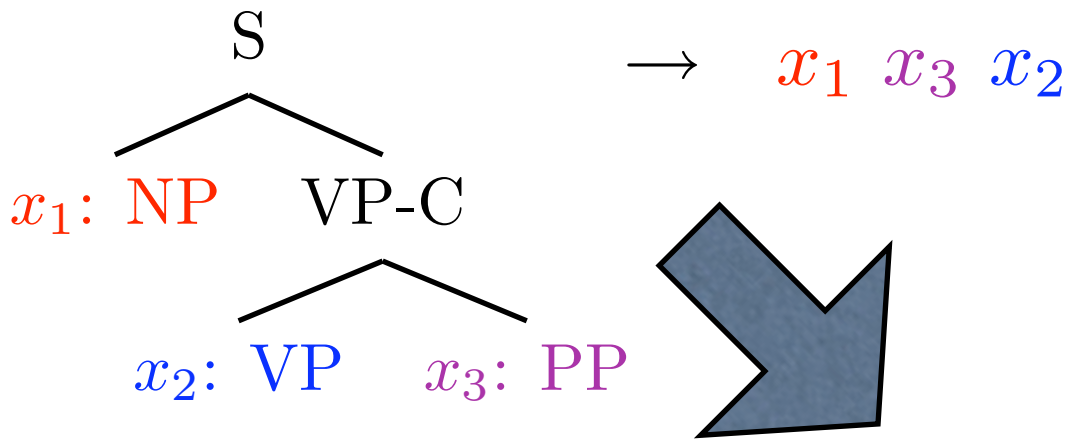
- runs in $O(n)$ time
  - $n$ shifts and at most $(n-1)$ reductions
- the permutation is synchronously binarizable
  - iff. it is reduced to exactly one element at the end
  - and in such case, the algorithm will return
    - the unique left-heavy binarization tree
    - corresponding to the non-ambiguous ITG (Wu, 1997)



# Experiments

# System

- ISI syntax-based MT system (Galley et al., 2004)
- based on tree-to-string rules
  - English-side subtrees and Chinese-side strings
    - when decoding, internally flattened to SCFGs



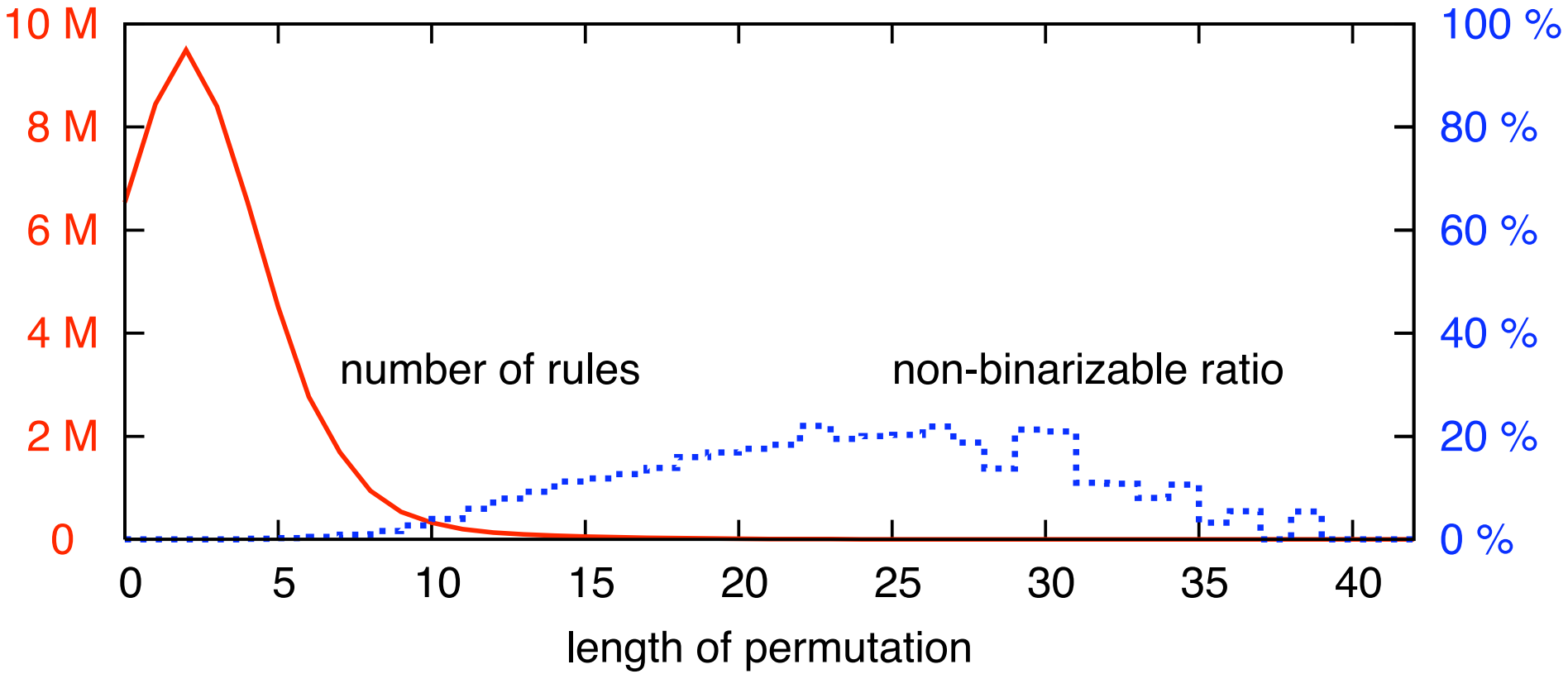

S $\rightarrow$ NP VP PP, NP PP VP

# Data

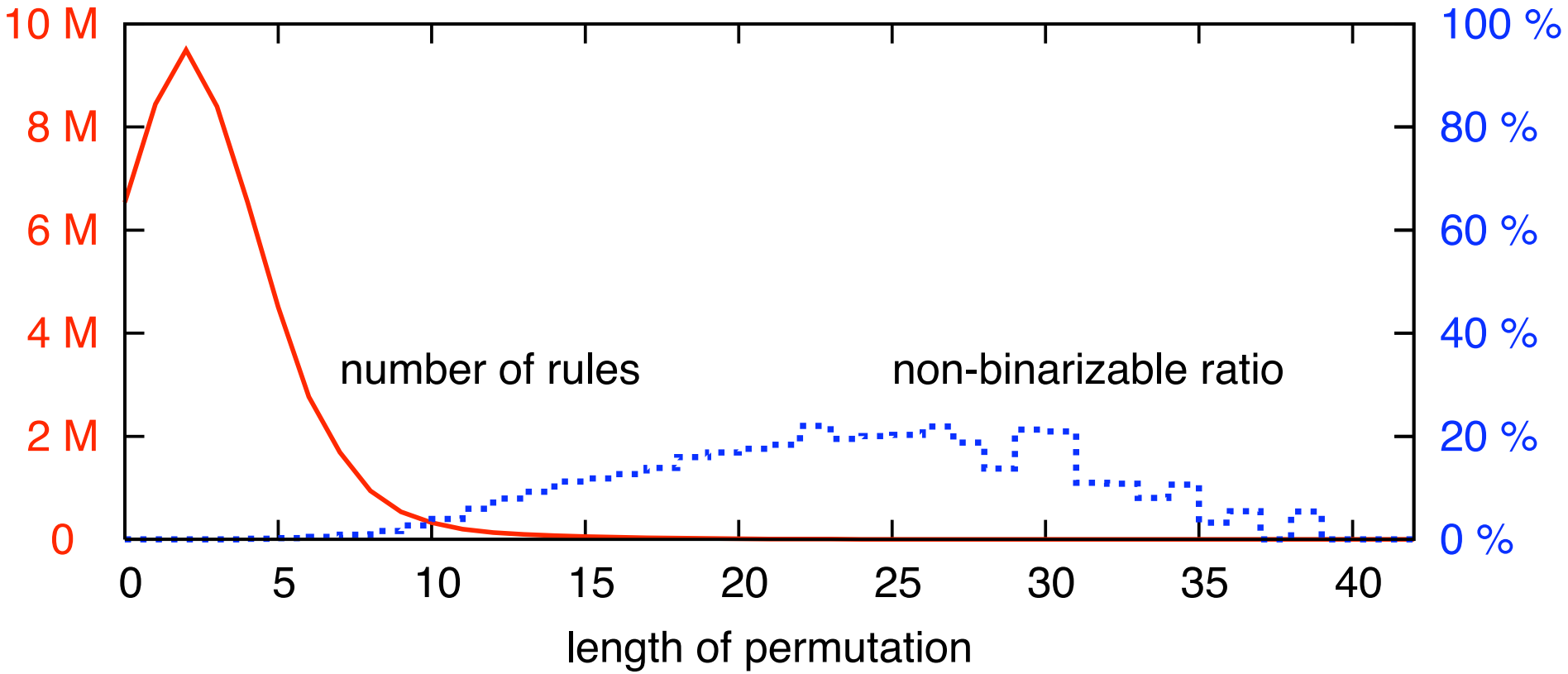
- training data
  - English/Chinese parallel corpus
    - 2.9 M sentence pairs (50 M English words)
  - word-aligned by GIZA++ (symmetrized by "union")
  - English side parsed by a variant of Collins parser
  - 50.8 M tree-to-string rules extracted
- test data
  - 116 short sentences (<= 16 words) from NIST 2002

# Binarizability

# Binarizability

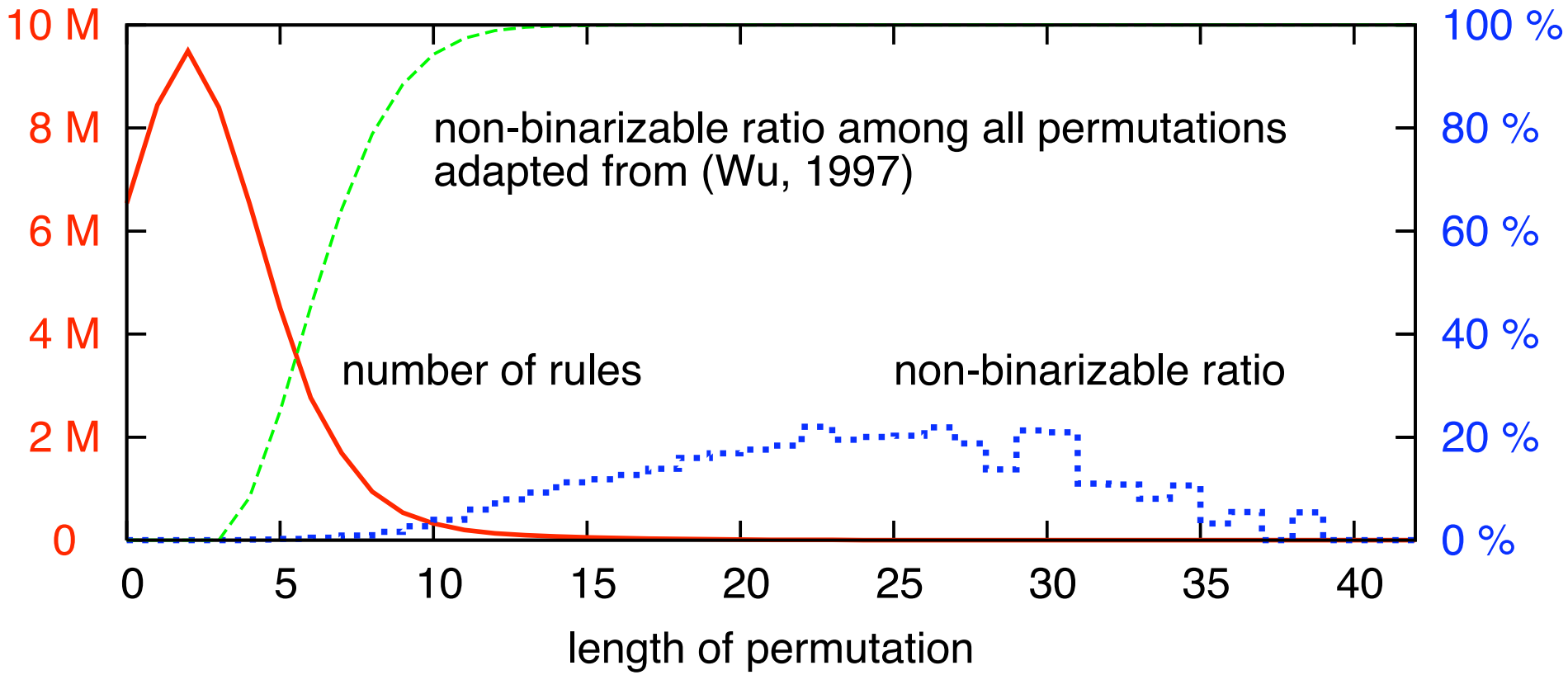


# Binarizability



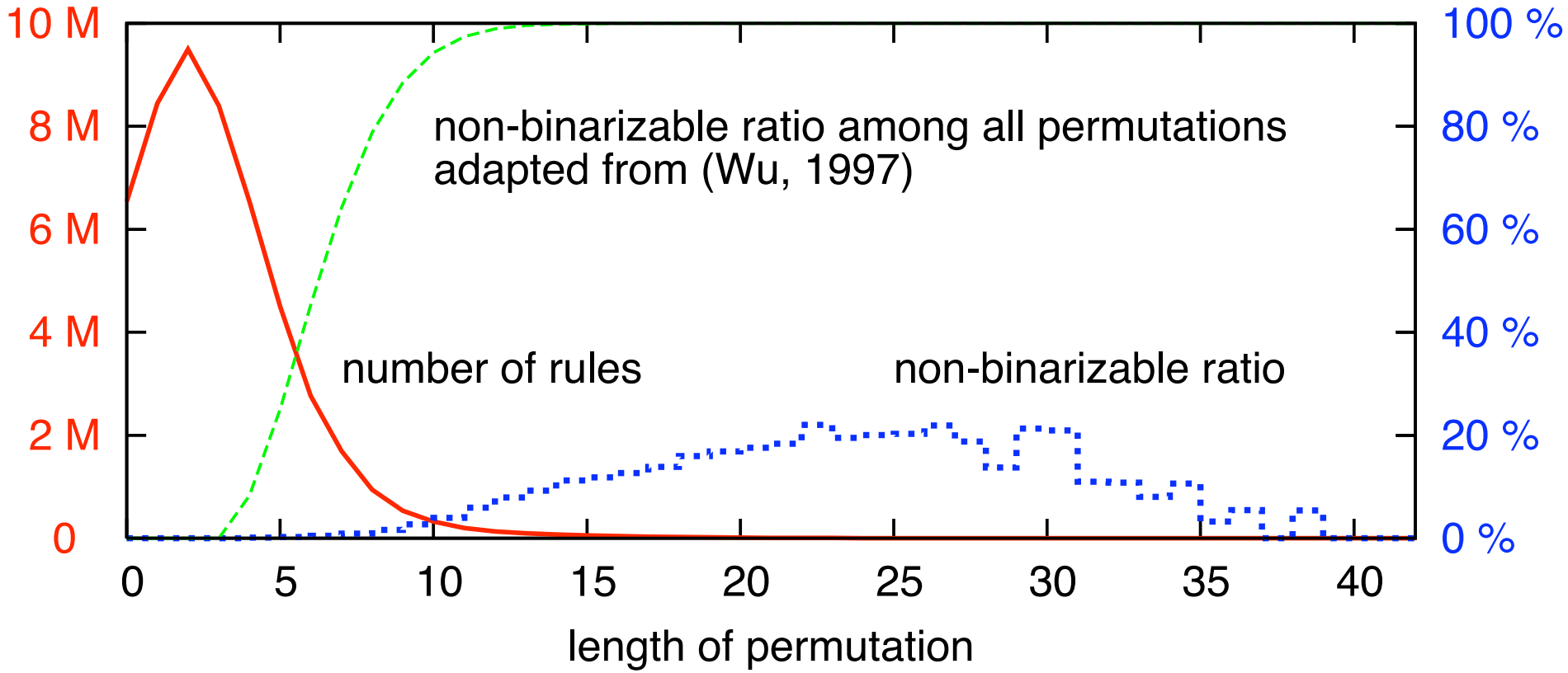

- Among 50.8 M rules, only 0.3 % are non-binarizable

# Binarizability




- Among 50.8 M rules, only 0.3 % are non-binarizable
  - although theoretically this ratio is much higher

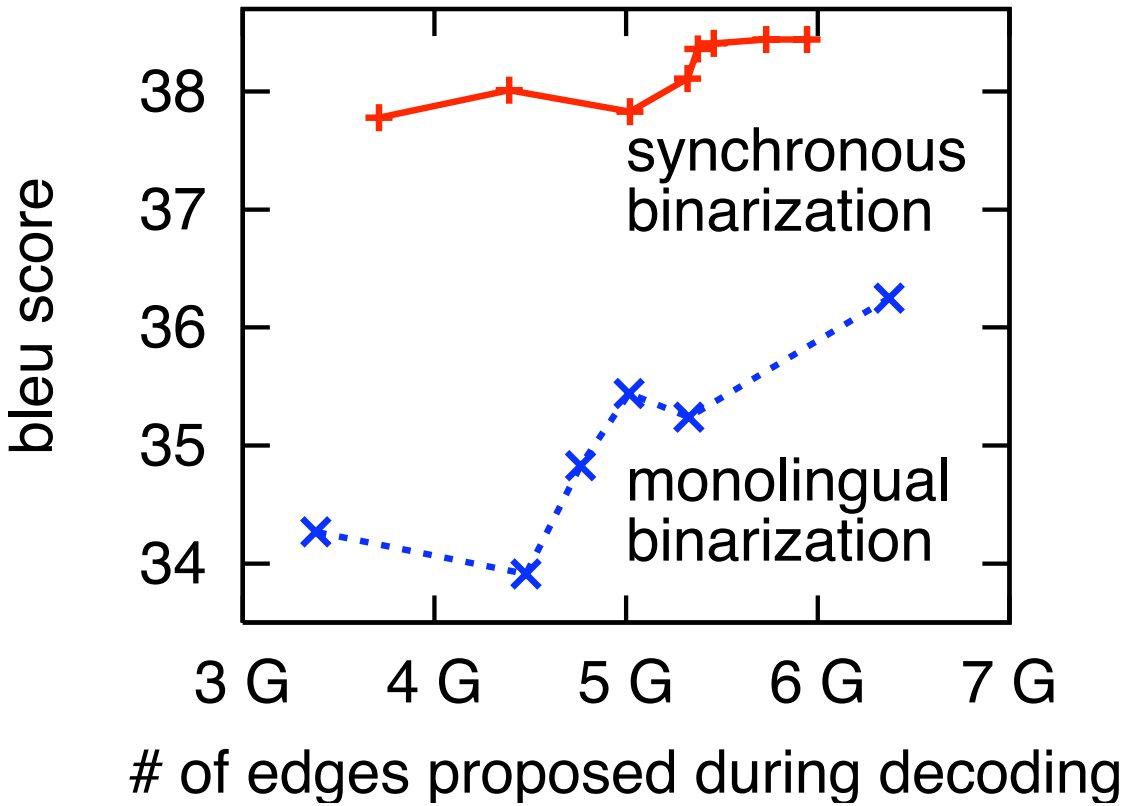
# Binarizability




- Among 50.8 M rules, only 0.3 % are non-binarizable
  - although theoretically this ratio is much higher
- identical decoding results without these non-bin rules

# Better Decoding

- decoding is based on the CKY algorithm
- BLEU scores at various beam settings
  - synchronous binarization consistently better




| binarization | BLEU |
|---|---|
| monolingual | 36.25 |
| synchronous | 38.44 |

(~ CPU time)

# Examples

# Examples

- the last step is for them to receive new job training .

# Examples

- the last step is for them to receive new job training .
- the final step is to receive new work training .

# Examples

- the last step is for them to receive new job training .
- the final step is to receive new work training .
- the final step is to receive new job training .

# Examples

- the last step is for them to receive new job training .
- the final step is to receive new work training .
- the final step is to receive new job training .

# Examples

- the last step is for them to receive new job training .
- the final step is to receive new work training .
- the final step is to receive new job training .

- turkey has agreed in principle to take over from britain the command of the peace - keeping troops in afghanistan .

# Examples

- the last step is for them to receive new job training .
- the final step is to receive new work training .
- the final step is to receive new job training .

- turkey has agreed in principle to take over from britain the command of the peace - keeping troops in afghanistan .
- turkey already agreed in principle that the rebels from the british took over the command of afghan peacekeeping force .

# Examples

- the last step is for them to receive new job training .
- the final step is to receive new work training .
- the final step is to receive new job training .

- turkey has agreed in principle to take over from britain the command of the peace - keeping troops in afghanistan .
- turkey already agreed in principle that the rebels from the british took over the command of afghan peacekeeping force .
- turkey has agreed in principle to take command of afghan peacekeeping force there from britain .

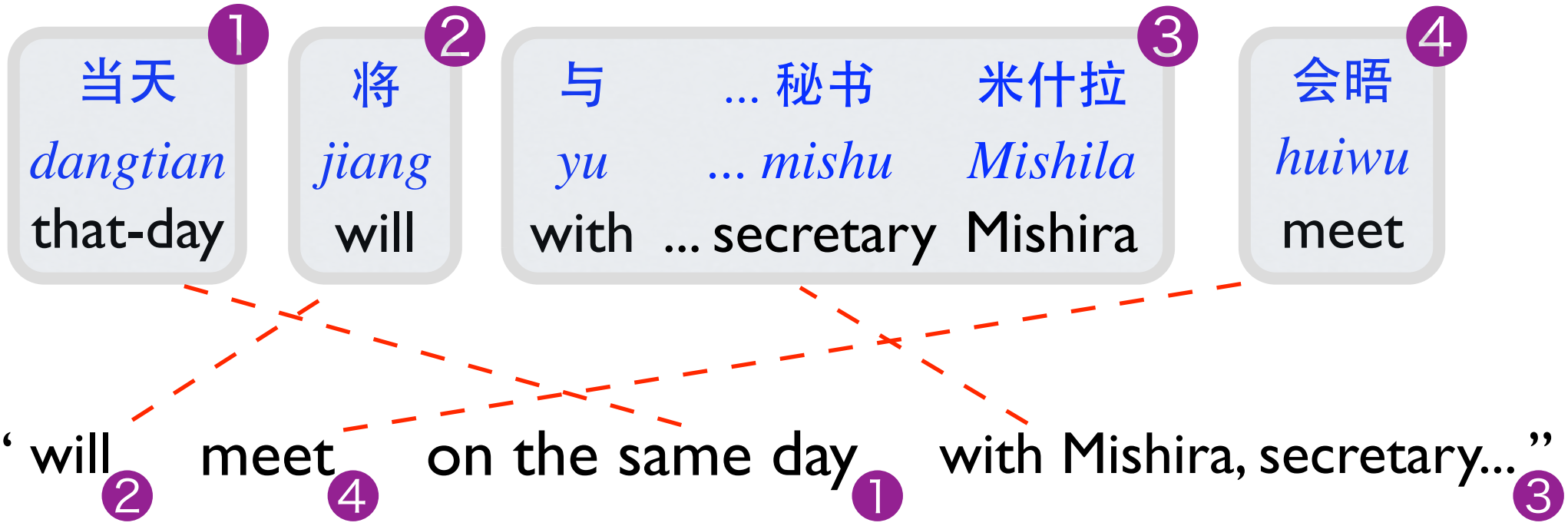
# Application: Testing the ITG Hypothesis

general approach:

1. obtain hand-aligned parallel text
2. parse the English-side
3. do GHKM-style rule extraction
4. verify non-binarizable examples

# Chinese-English

- 935 hand-aligned sentences from (Liu, Liu, Lin, 2005)
  - non-binarizability: 0.3%

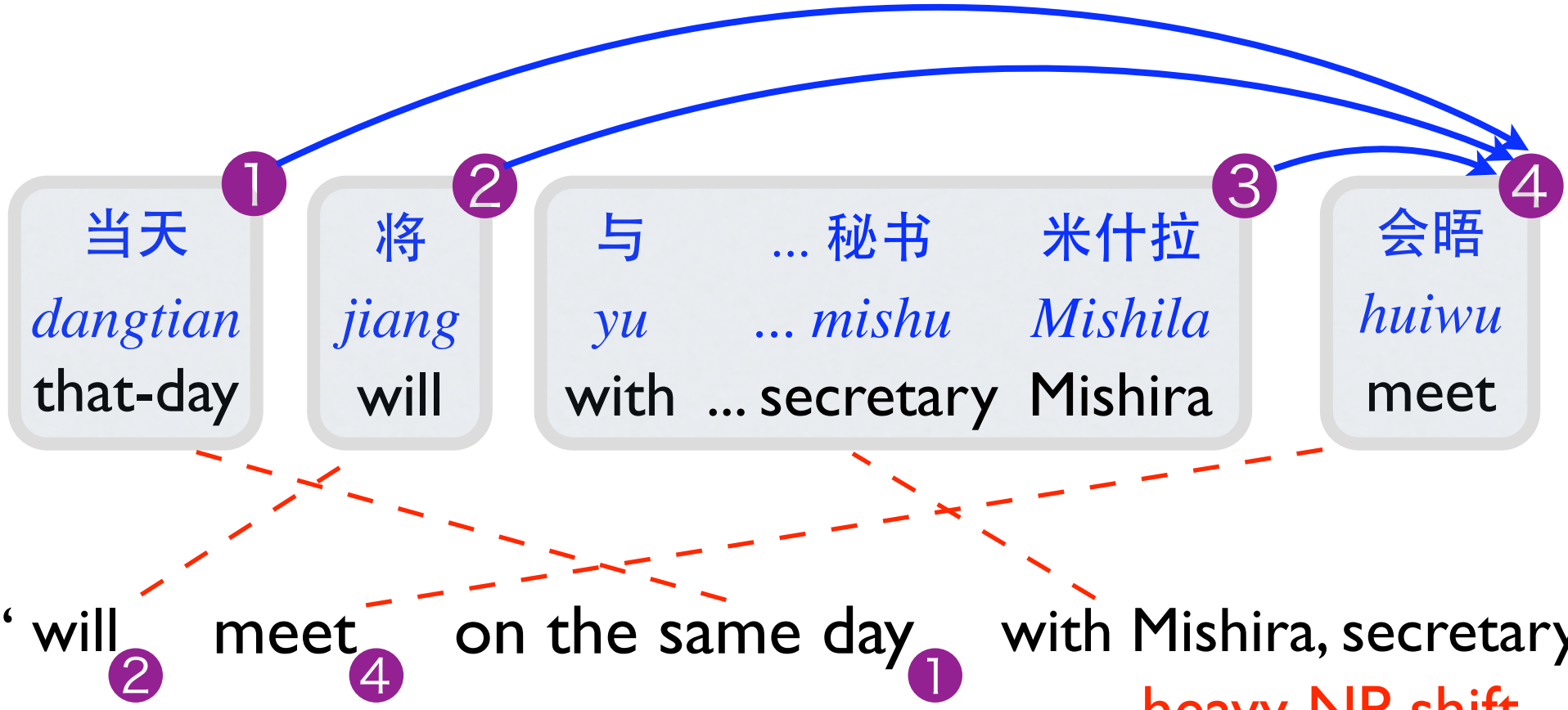


# Chinese-English

- 935 hand-aligned sentences from (Liu, Liu, Lin, 2005)
  - non-binarizability: 0.3%

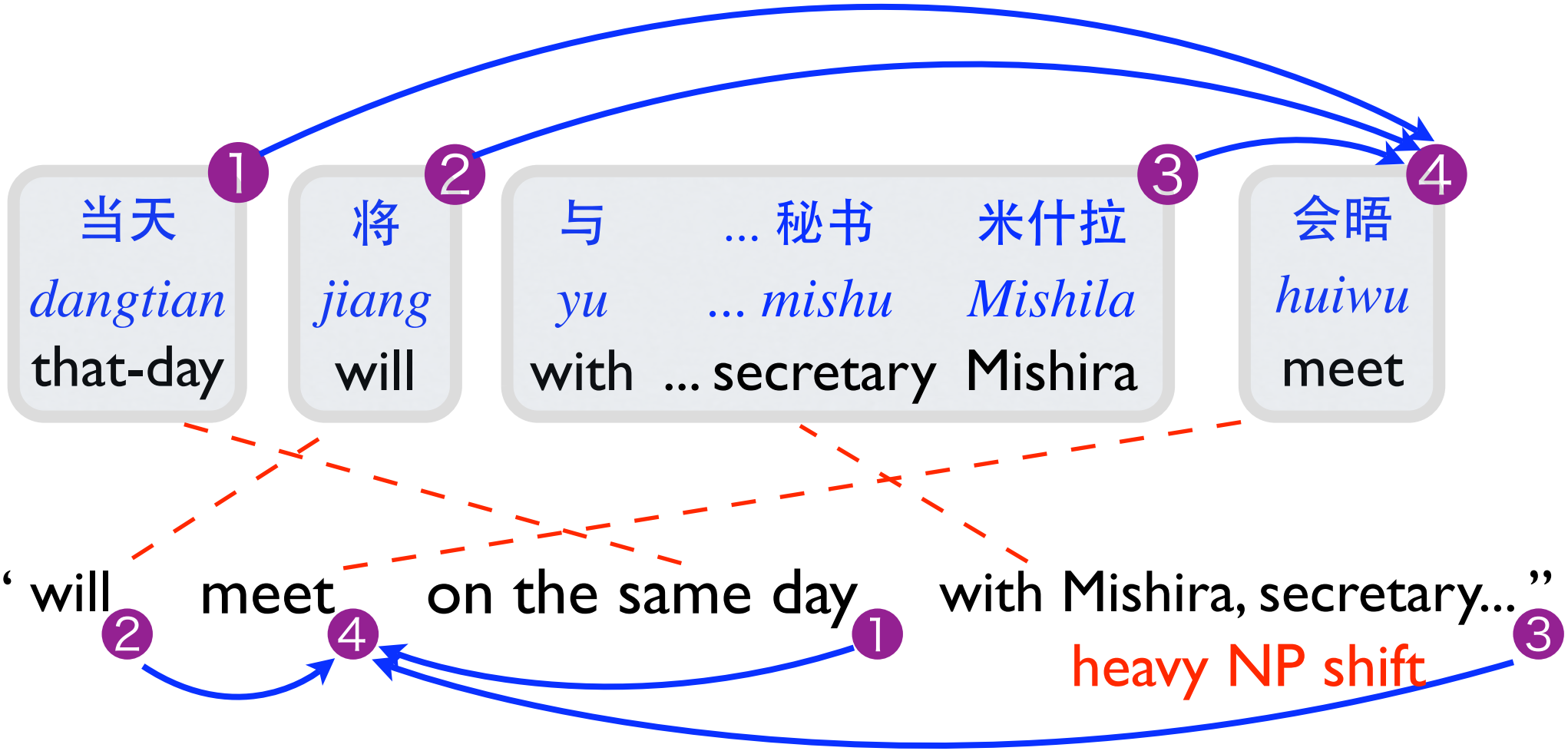
# Chinese-English

- 935 hand-aligned sentences from (Liu, Liu, Lin, 2005)
  - non-binarizability: 0.3%



# Chinese-English

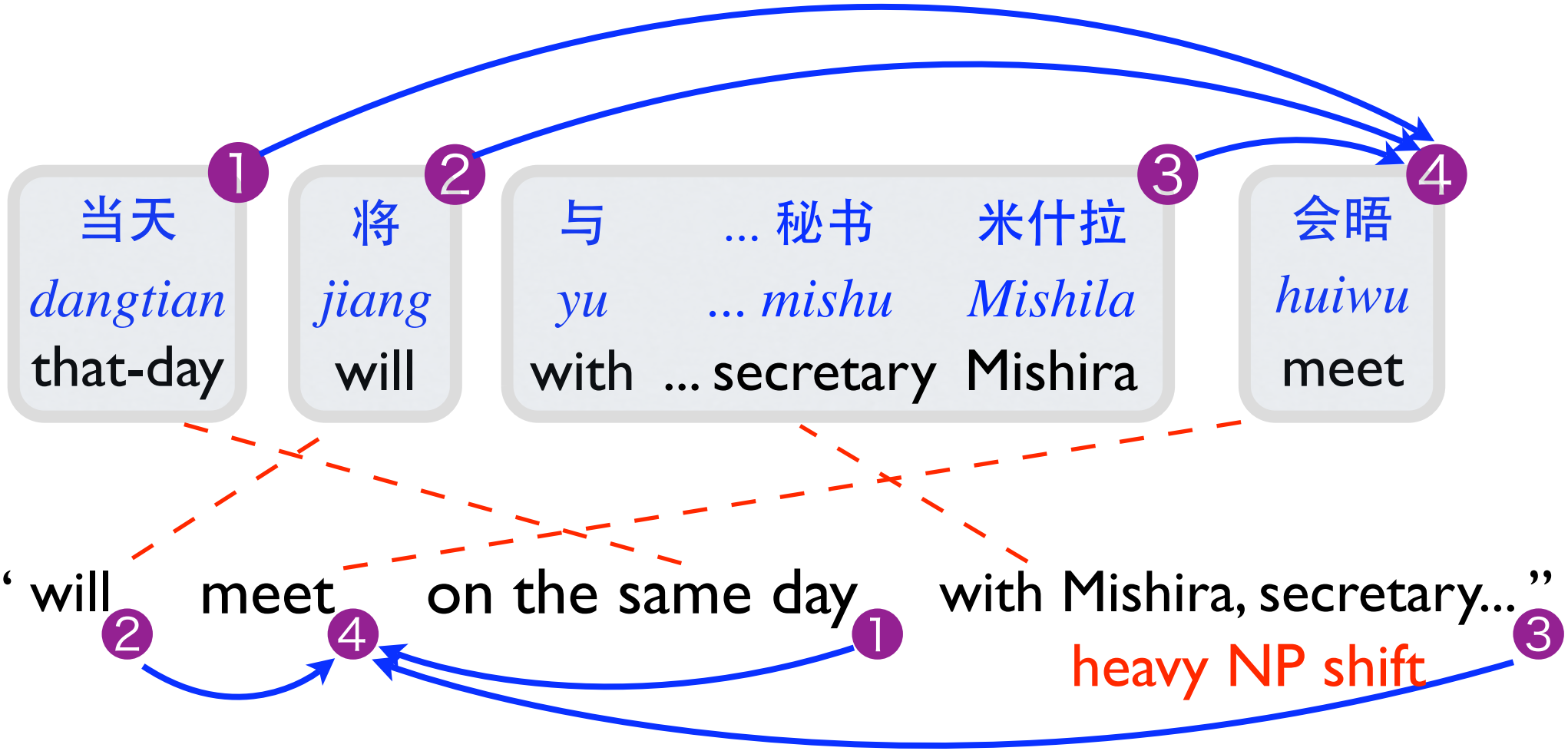
- 935 hand-aligned sentences from (Liu, Liu, Lin, 2005)
  - non-binarizability: 0.3%



# Chinese-English

- 935 hand-aligned sentences from (Liu, Liu, Lin, 2005)
  - non-binarizability: 0.3%

isomorphic dep-trees

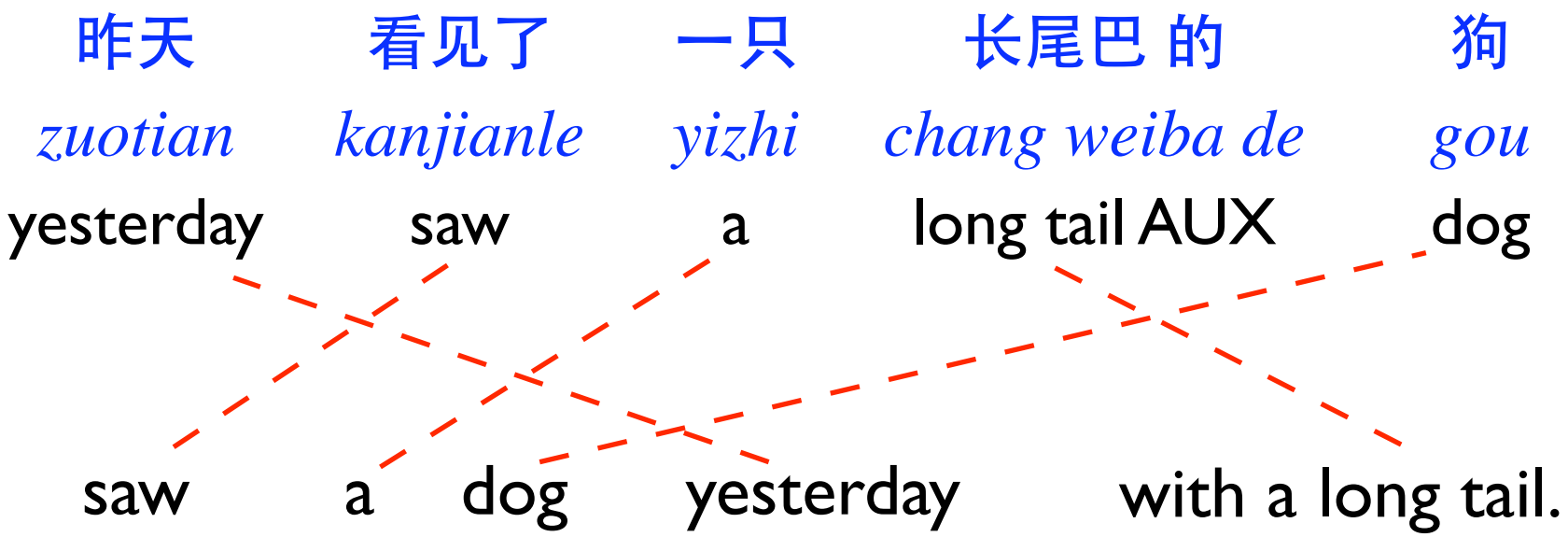


# Chinese-English

- example 2: slightly non-literal translation
  - non-isomorphic dependency trees (Eisner, 2003)

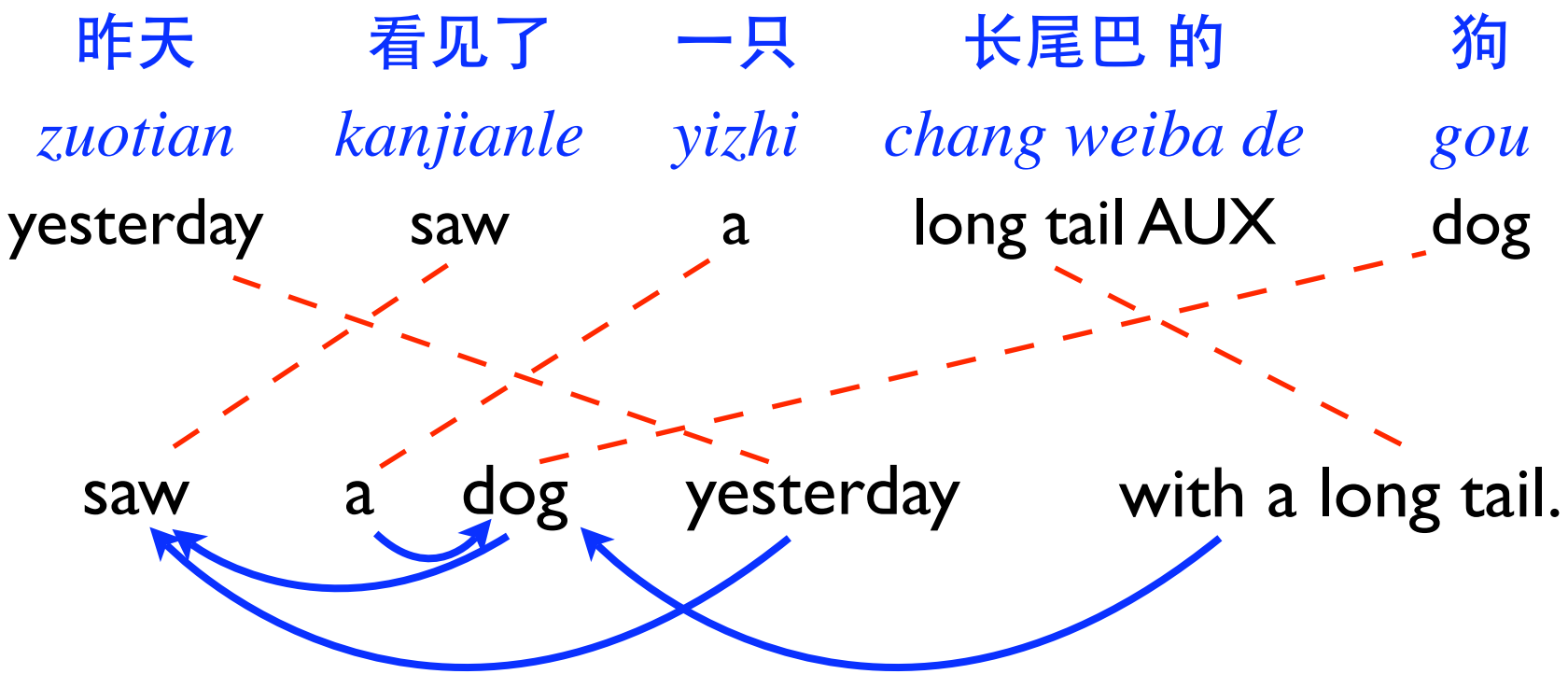
# Side note: Non-Projectivity

- synthetic example (Chris Quirk, p.c.) with a non-projective English side
- clearly non-binarizable, but even the English side alone is beyond CFG! => this pair is beyond SCFG



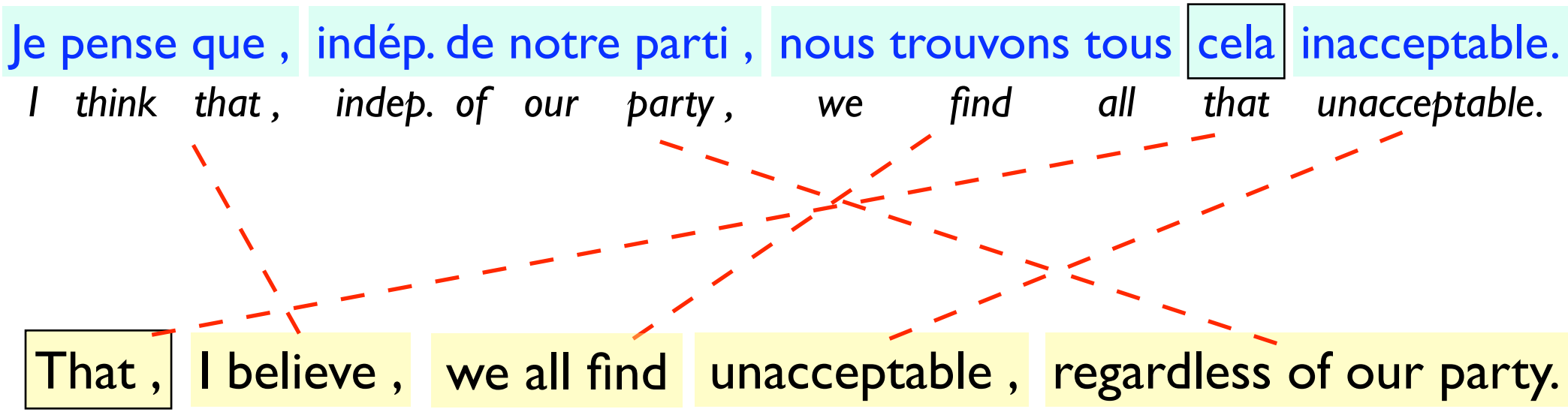
# Side note: Non-Projectivity

- synthetic example (Chris Quirk, p.c.) with a non-projective English side
- clearly non-binarizable, but even the English side alone is beyond CFG! => this pair is beyond SCFG



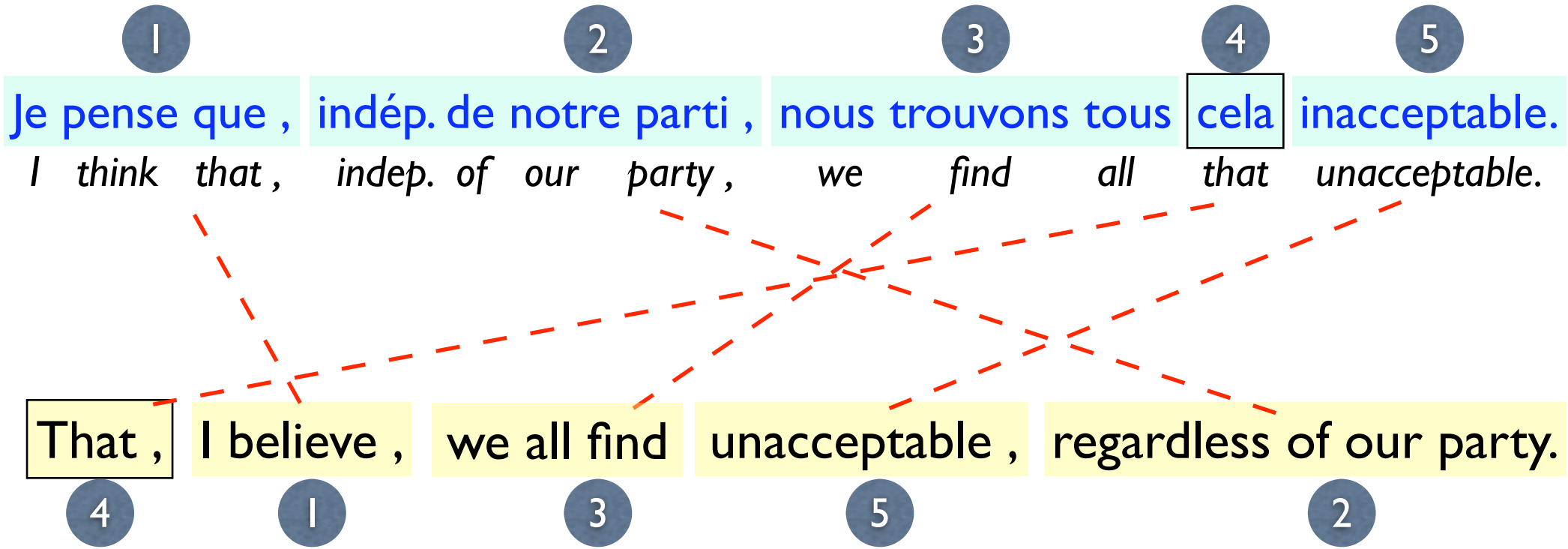
# French-English

- 447 sentences from NAACL 03 workshop (Mihalcea and Pederson, 2003)
- 2659 rules, 2 non-binarizable (0.1%)
- example 1: topicalization



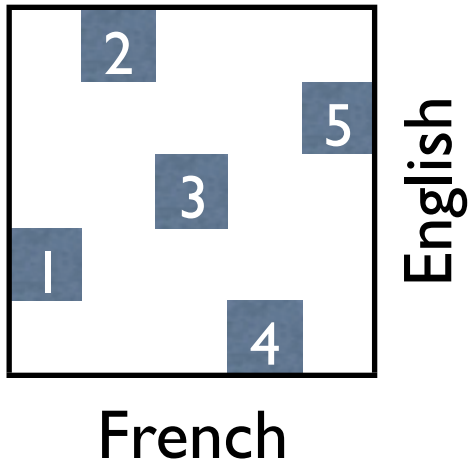
# French-English

- 447 sentences from NAACL 03 workshop (Mihalcea and Pederson, 2003)
- 2659 rules, 2 non-binarizable (0.1%)
- example 1: topicalization



# French-English

- 447 sentences from NAACL 03 workshop (Mihalcea and Pederson, 2003)
- 2659 rules, 2 non-binarizable (0.1%)
- example 1: topicalization

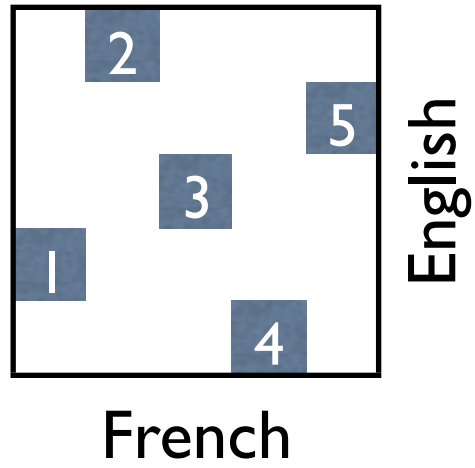



| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Je pense que , | indép. de notre parti , | nous trouvons tous | cela | inacceptable. |
| *I think that ,* | *indep. of our party ,* | *we find all* | *that* | *unacceptable.* |

| That , | I believe , | we all find | unacceptable , | regardless of our party. |
|---|---|---|---|---|
| 4 | 1 | 3 | 5 | 2 |

# French-English

- example 2: movement of adverbials




1 2 3 4 5

Le entreprise , dans le secteur privé , fut rebaptisée en 1934 ...

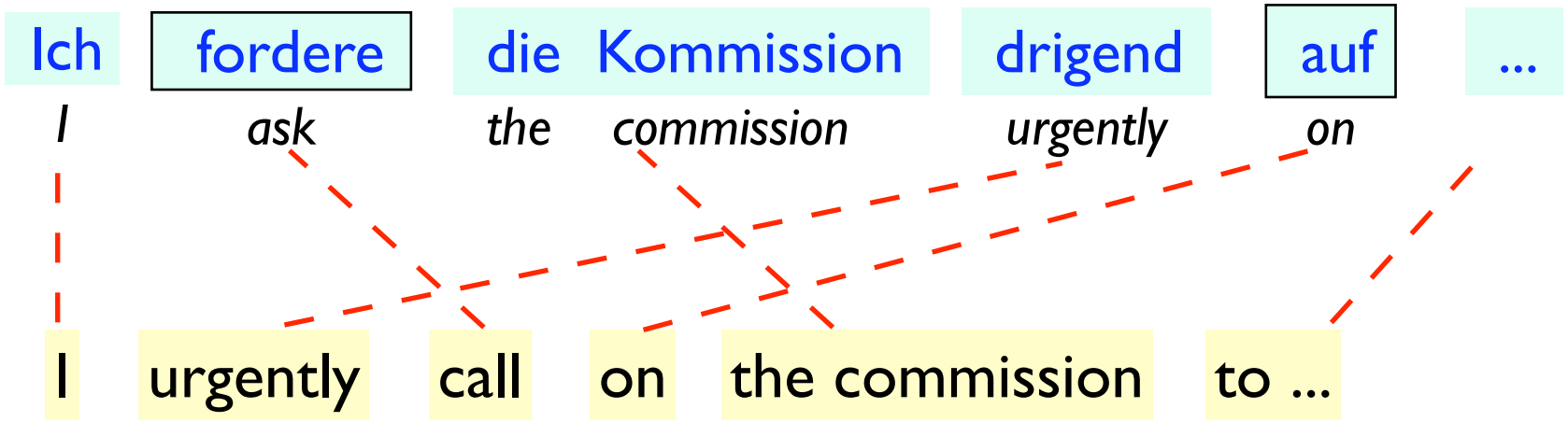
NP

In 1934 , the company was renamed ... , still under private ownership.

4 1 3 5 2

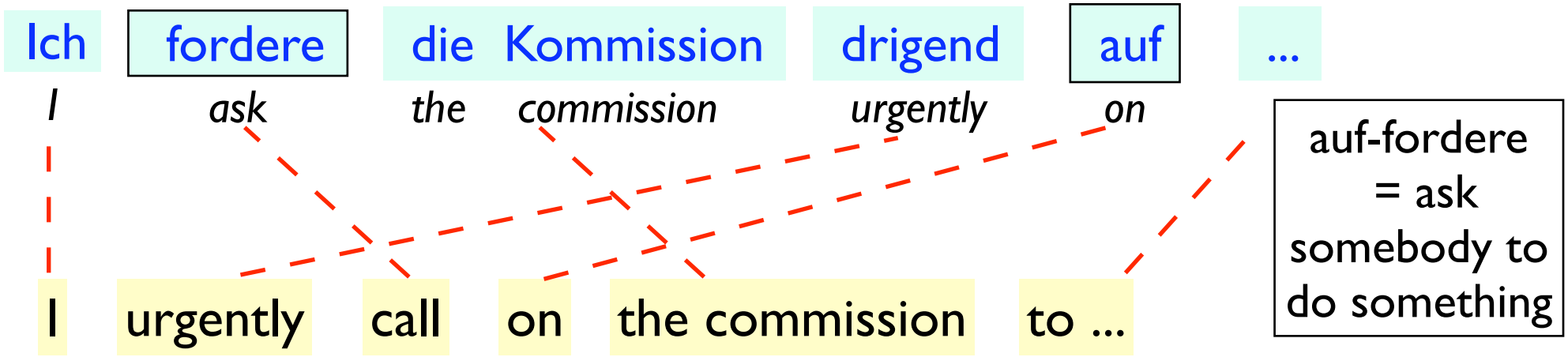
# German-English

- 220 sentences in Europarl, annotated by a German native speaker (Callison-Burch, p.c.)
- 2328 rules, 13 non-binarizable (0.6%)
- many long rules due to German word order
  - V-2, V-final, extraposition, separable prefixes, ...



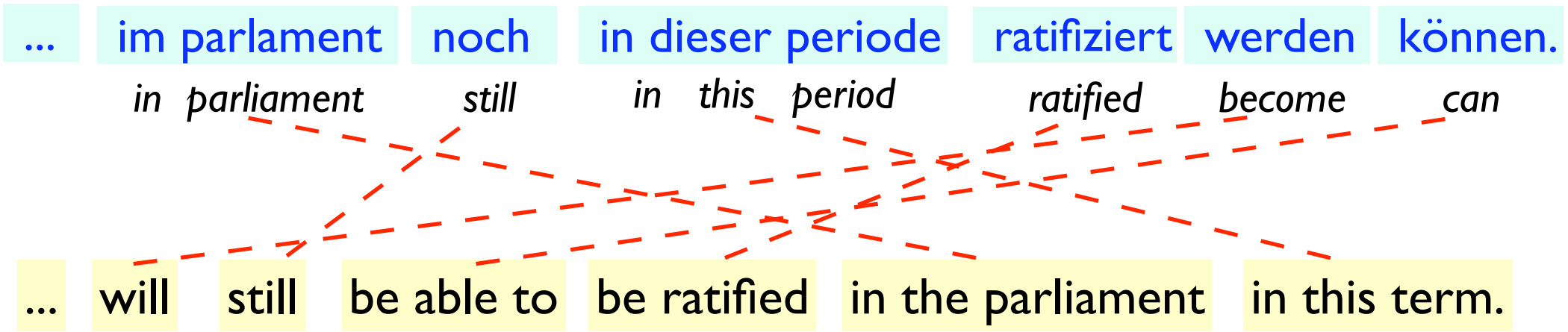
# German-English

- 220 sentences in Europarl, annotated by a German native speaker (Callison-Burch, p.c.)
- 2328 rules, 13 non-binarizable (0.6%)
- many long rules due to German word order
  - V-2, V-final, extraposition, separable prefixes, ...



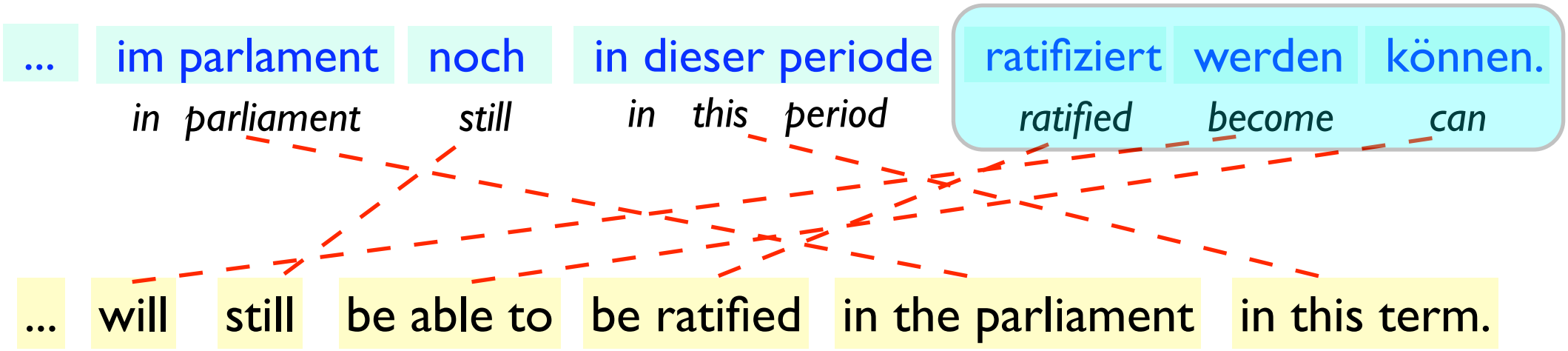
# German-English

- example 2: verb-final within an embedded clause



# German-English

- example 2: verb-final within an embedded clause



# Summary of Testings

- The ITG hypothesis is largely correct, even between a free word-order language (German) and a fixed word-order language (English).
- but there do exist linguistically interesting examples beyond ITG, even between fixed word-order languages.

| | # of sentences | # of rules | non-bin ratio | major reasons for non-bin |
|---|---|---|---|---|
| Chinese | 935 | 13713 | 0.3% | adverbials, heavy NP shift, ... |
| French | 447 | 2659 | 0.1% | adverbials, topicalization |
| German | 220 | 2328 | 0.6% | V2, V-final, extraposition, ... |

# Summary of Testings

- The ITG hypothesis is largely correct, even between a free word-order language (German) and a fixed word-order language (English).
- but there do exist linguistically interesting examples beyond ITG, even between fixed word-order languages.

| | # of sentences | # of rules | non-bin ratio | major reasons for non-bin |
|---|---|---|---|---|
| Chinese | 935 | 13713 | 0.3% | adverbials, heavy NP shift, ... |
| French | 447 | 2659 | 0.1% | adverbials, topicalization |
| German | 220 | 2328 | 0.6% | V2, V-final, extraposition, ... |

*how about German-Czech?* *Japanese-English?*

# Conclusion and Future Work

# Conclusion and Future Work

- Theoretical side
  - a theory on binarization for synchronous grammars
    - theorems and proofs in the paper
  - an efficient algorithm for binarizing permutations

# Conclusion and Future Work

- Theoretical side
  - a theory on binarization for synchronous grammars
    - theorems and proofs in the paper
  - an efficient algorithm for binarizing permutations
- Practical side
  - syntactic re-orderings are largely binarizable
    - but there do exist valid non-binarizable re-orderings
  - binarized grammars improve translation quality

# Conclusion and Future Work

- Theoretical side
  - a theory on binarization for synchronous grammars
    - theorems and proofs in the paper
  - an efficient algorithm for binarizing permutations
- Practical side
  - syntactic re-orderings are largely binarizable
    - but there do exist valid non-binarizable re-orderings
  - binarized grammars improve translation quality
- Future Work: binarizing other formalisms like STAGs

# Thank you!
# 谢谢!

Questions?
Comments?

# Thank you!
# 谢谢!

Questions?
Comments?