

Comparative Parser Evaluation across Different Grammar Frameworks

Takuya Matsuzaki

Tsujii lab.

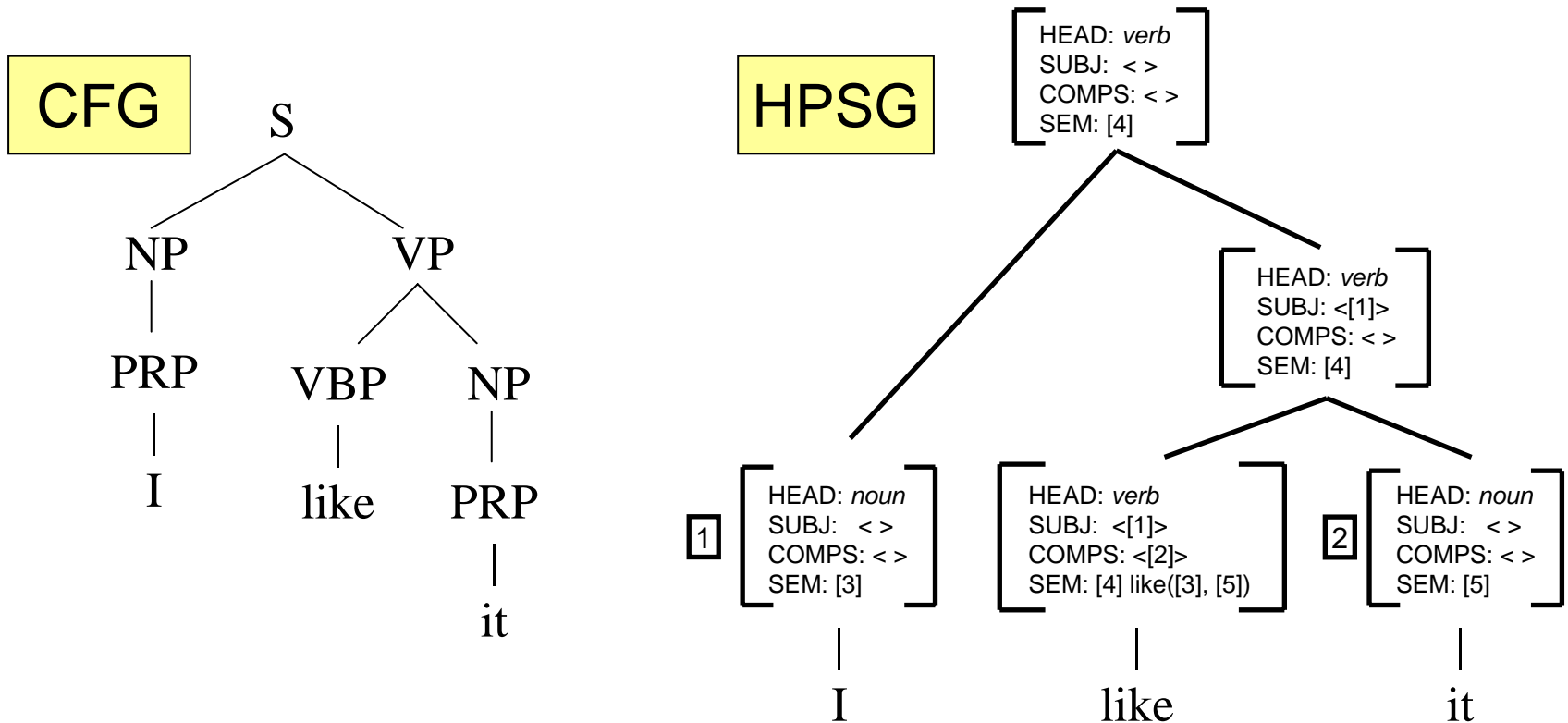
University of Tokyo

Motivation

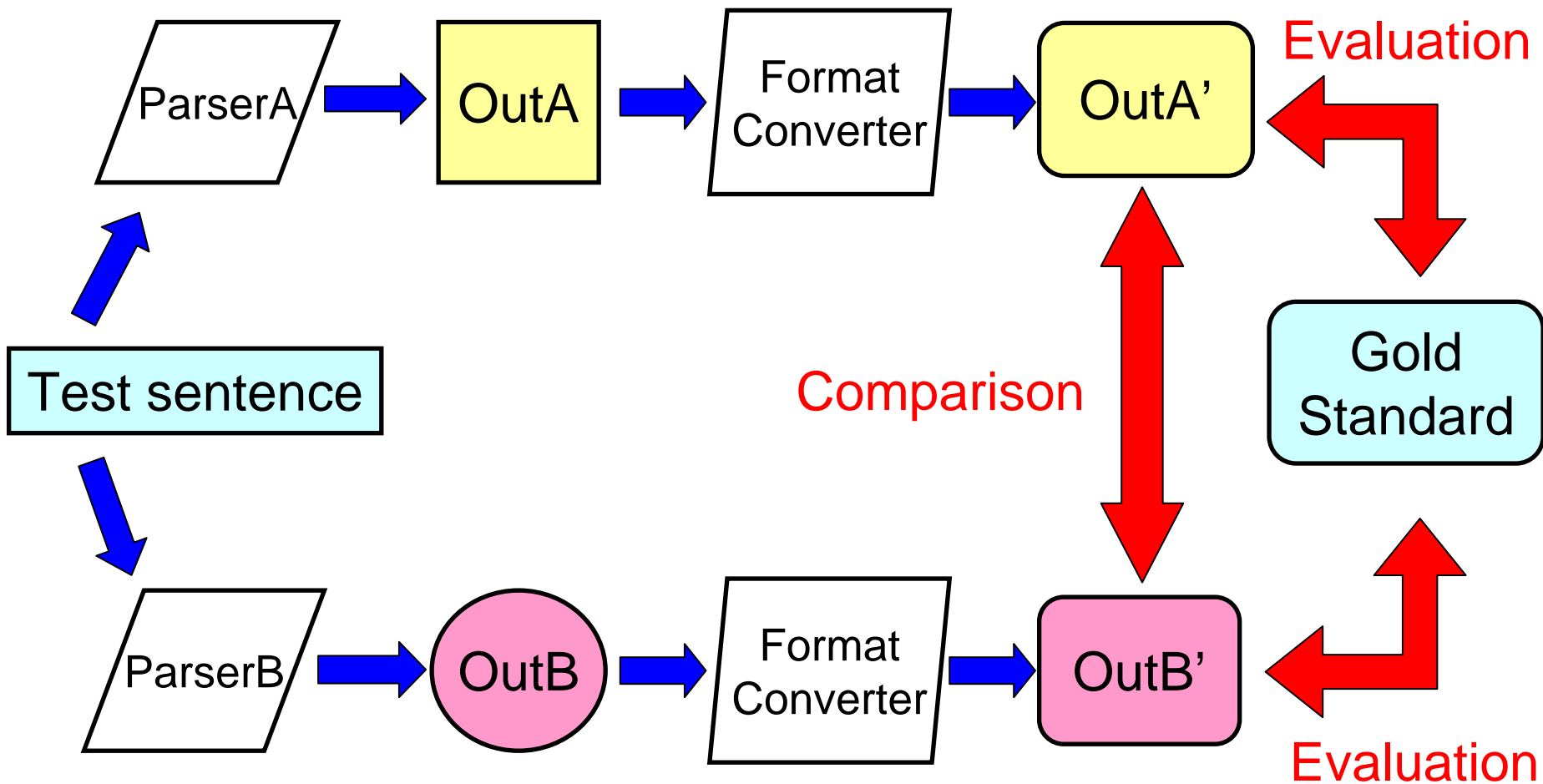
- Comparison among parsers based on different grammar frameworks:
 - HPSG parser vs. PCFG parser
 - HPSG parser vs. CCG parser
- Why it is important:
 - Comparison of linguistic theories from the viewpoint of parsing
 - Import/export parsing techniques across different grammar frameworks
 - “Which parser should I use for my NLP application?”

Problem

- Different representational devices are used in different grammar frameworks
 - The parsing results are not directly comparable



Comparative evaluation based on 'a common format'



Proposal

- Shallow CFG analysis as the common format
 - A simple conversion algorithm exists
 - Highly accurate conversion is possible
 - Meaningful differences among the parsers' performances are still observable

Outline

- Motivation
- Shallow CFG analysis as the common format
- Tree conversion algorithm
- Experiments
- Conclusion

Outline

- Motivation
- **Shallow CFG analysis as the common format**
- Tree conversion algorithm
- Experiments
- Conclusion

Candidates for the ‘common-format’

- ‘Deep’ representations
 - Predicate-argument structures
 - Grammatical relations (Carroll et al., 1998)

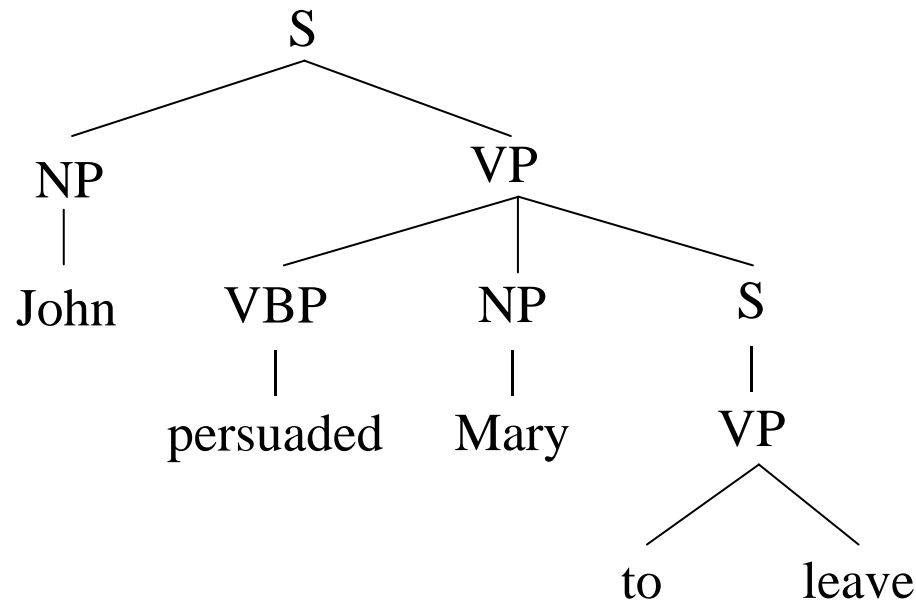
→ More informative, but the conversion is not necessarily easy

- ‘Shallow’ representations
 - CFG trees
 - Unlabeled bi-lexical dependencies

→ Less informative, but the conversion is easy

Shallow CFG analysis

- (simplified) PennTreebank-style analysis
 - No empty nodes, co-indexing and function-tags
 - Grammatical roles and long-distance dependencies are not directly represented



Shallow CFG analysis as the common-format

- Pro:
 - Highly accurate conversion is possible:
 - Accurate conversion is necessary to compare parsers at similar level of performance
 - Directly comparable to the output of the well-studied PCFG parsers (e.g., Charniak/Collins parser)
- Con:
 - Deep analysis results are only evaluated indirectly

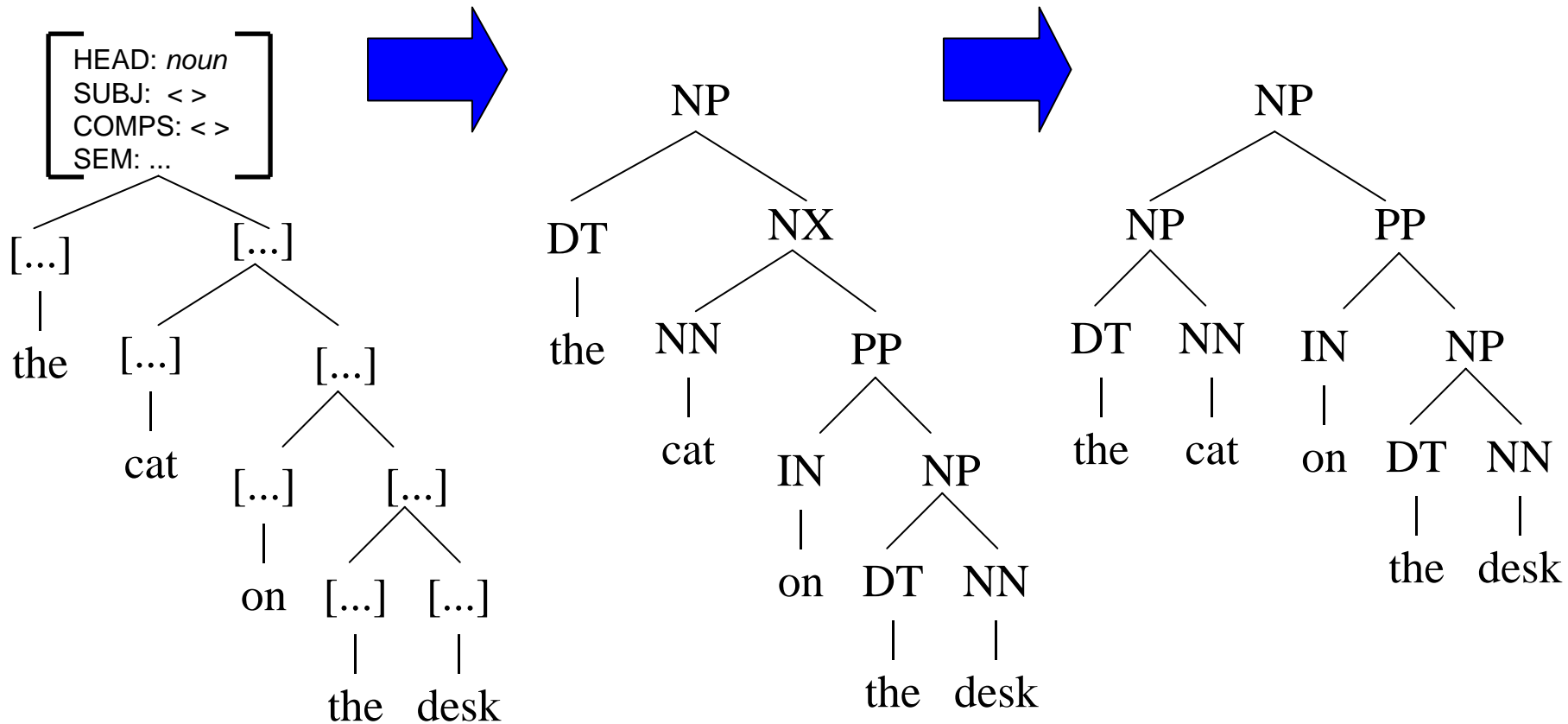
Plan of the talk

- Motivation
- Shallow CFG tree as the common format
- **Tree conversion algorithm**
- Experiments
- Conclusion

HPSG-to-CFG conversion

FSs → atomic labels

Tree transformation
based on SSG



Stochastic synchronous tree substitution grammar (SSTSG)

- Probabilistic model on pairs of trees:

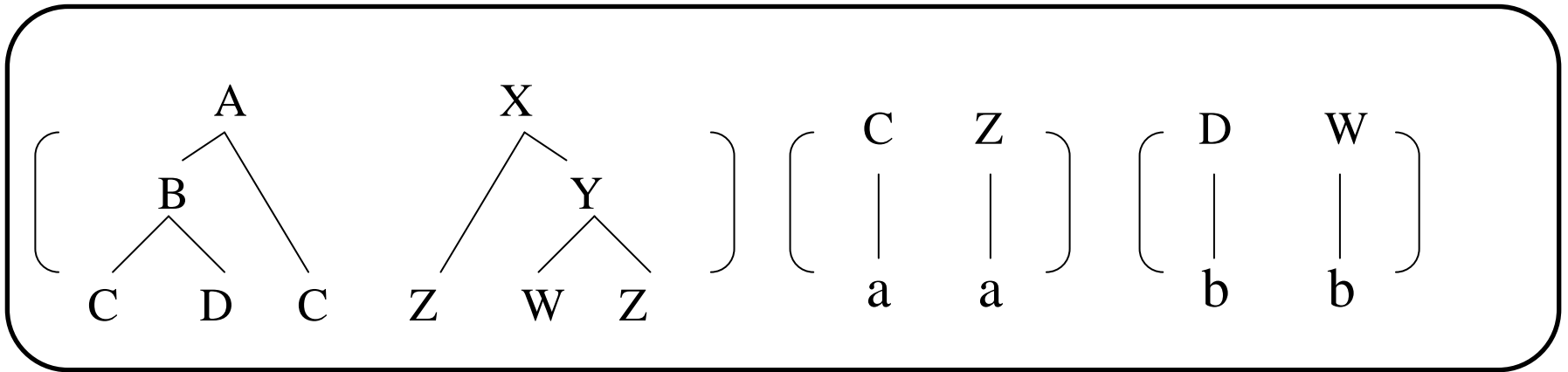
$$P(T, S)$$

- Transformation of tree $T \Leftrightarrow$ the max-prob opponent of T :

$$\text{Transform}(T) = \operatorname{argmax} P(T, S)$$

Example: Synchronous tree substitution grammar

- Synchronous productions:



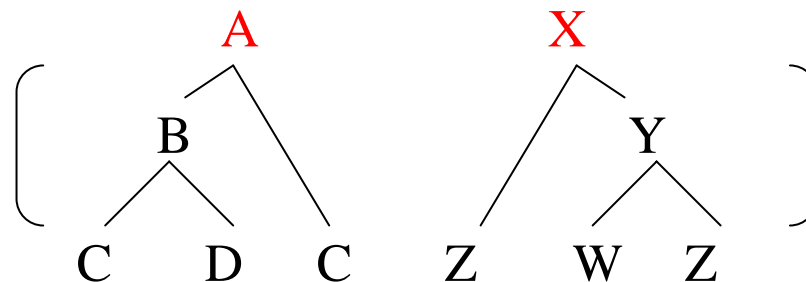
- Initial symbol pair: $\{ A \quad X \}$

Synchronous derivation

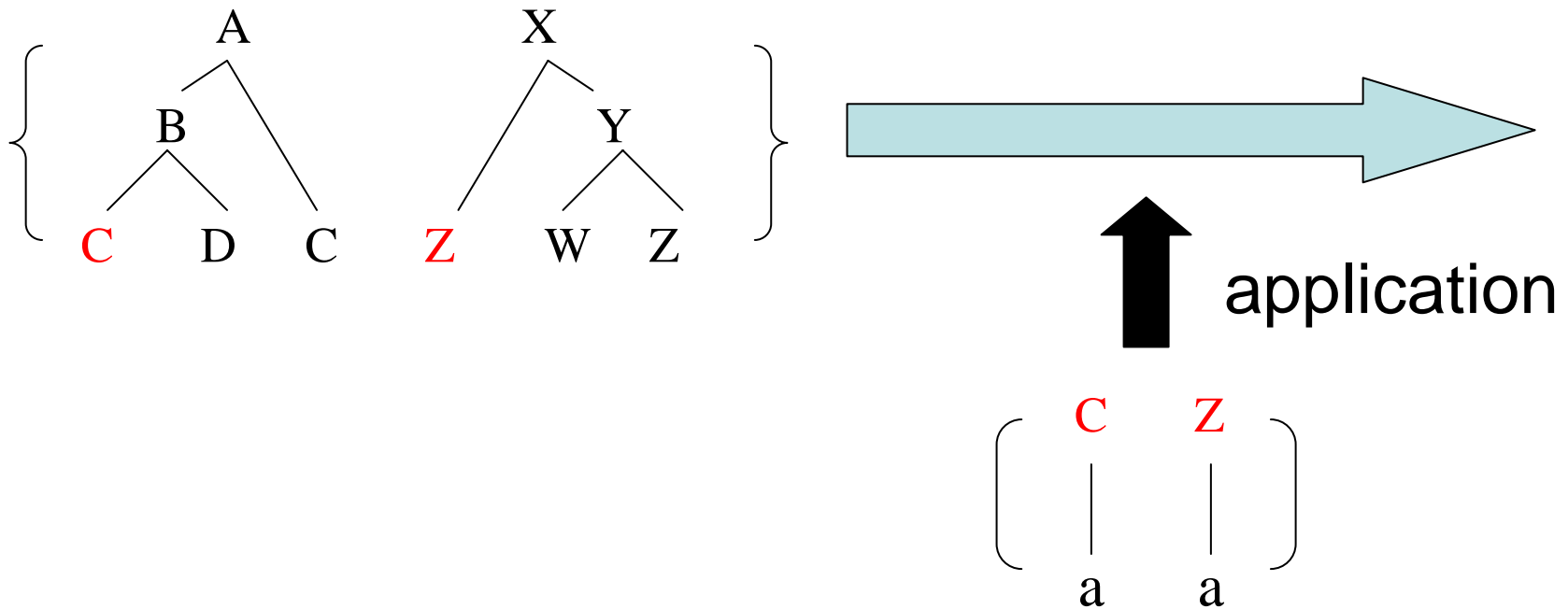
Init. pair: $\{ A \quad X \}$



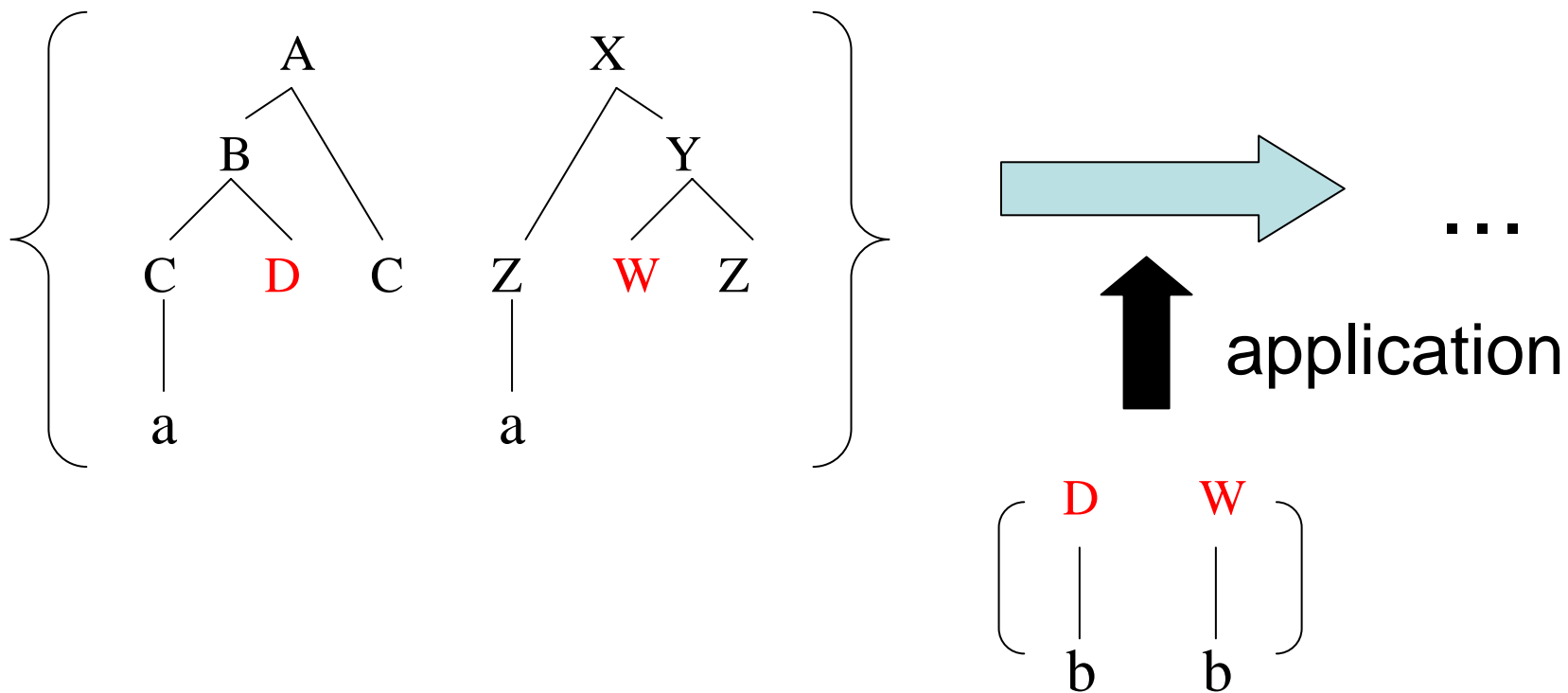
application



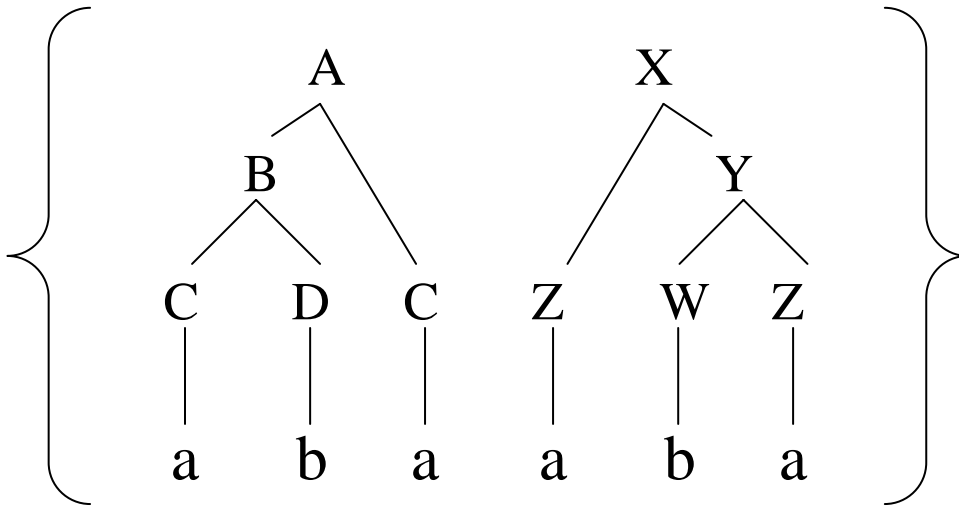
Synchronous derivation



Synchronous derivation



Synchronous derivation



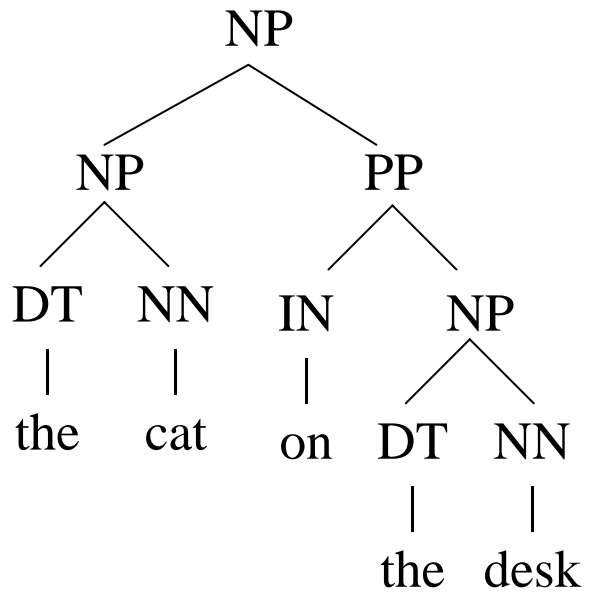
Inference

- We use ‘viterbi-approximation’:
 $\text{transform}(T) = \text{argmax } P(T, S) \doteq \text{argmax } P_d(T, S)$
 - An efficient algorithm exists (Eisner, 2003)
- $P_d(T, S)$: Probability of a **derivation** of (T, S) :
 $P_d(T, S) = \prod$ (probability of production)
- $P(T, S)$: Probability of a tree pair (T, S) :
 $P(T, S) = \sum P_d(T, S)$
 - The summation is for all derivations of (T, S)
 - Exponentially many derivations for (T, S) in general

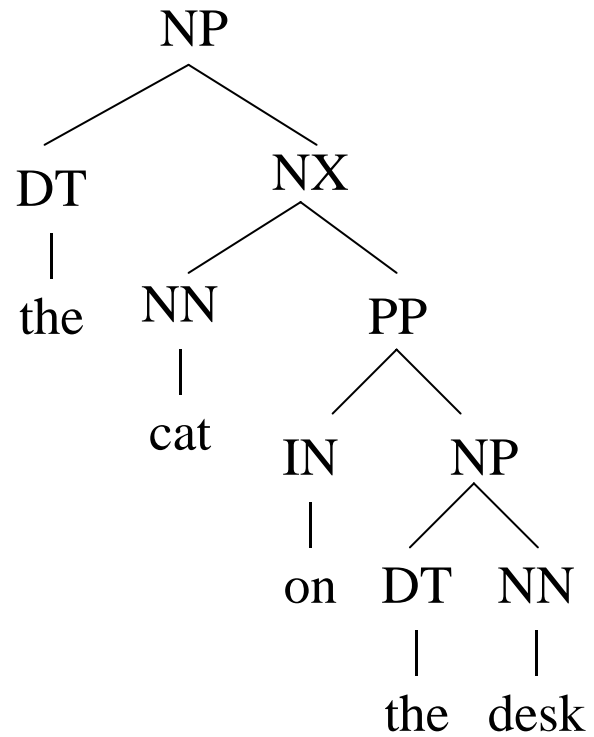
SSTSG acquisition

- Source: a parallel treebank
 - A set of pairs of trees:
{ (HPSG-tree, PTB-CFG-tree) }
- Acquisition algorithm:
 1. Find pairs of nodes on common spans
 2. Split the tree pair at the common span nodes
 3. Estimate the production probabilities

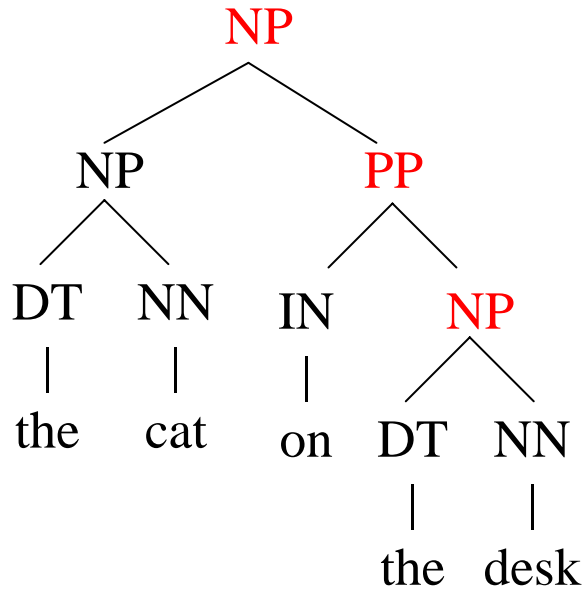
CFG tree



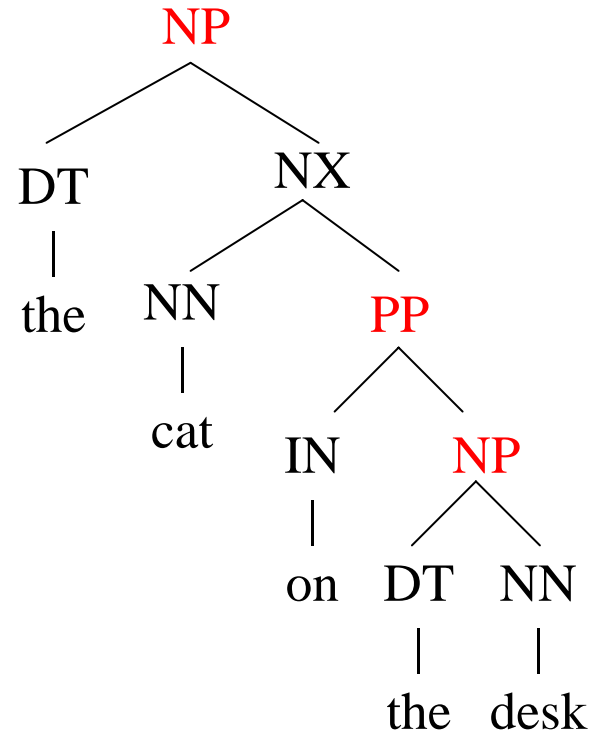
HPSG tree



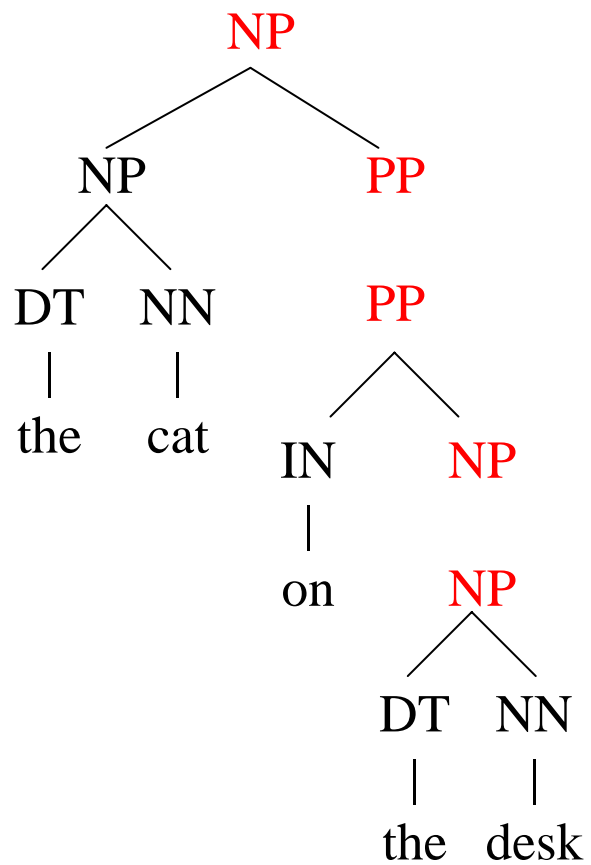
CFG tree



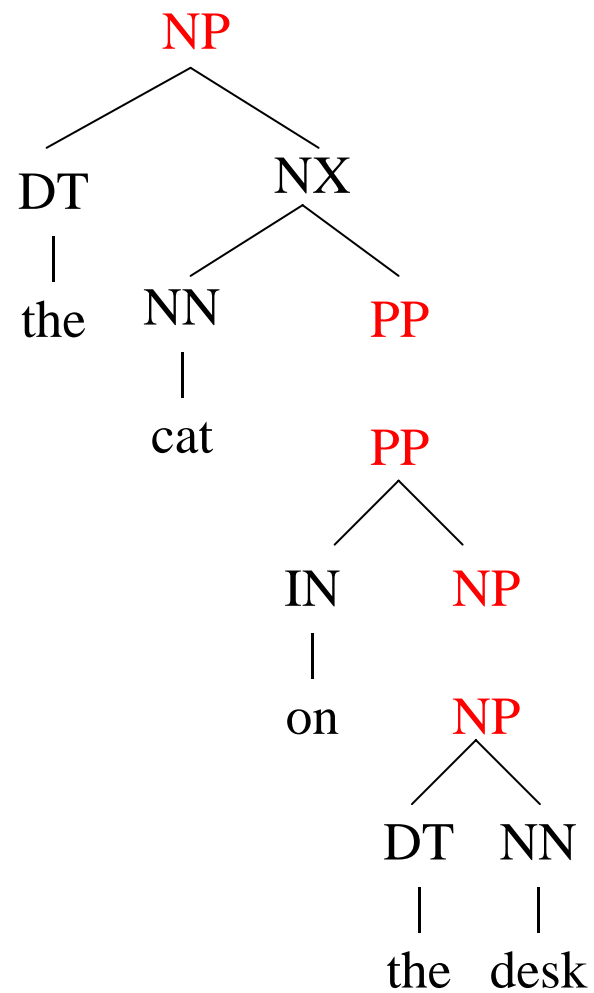
HPSG tree

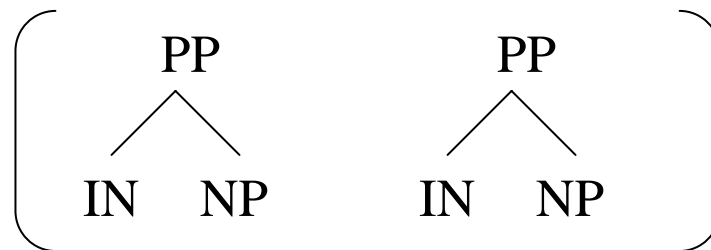
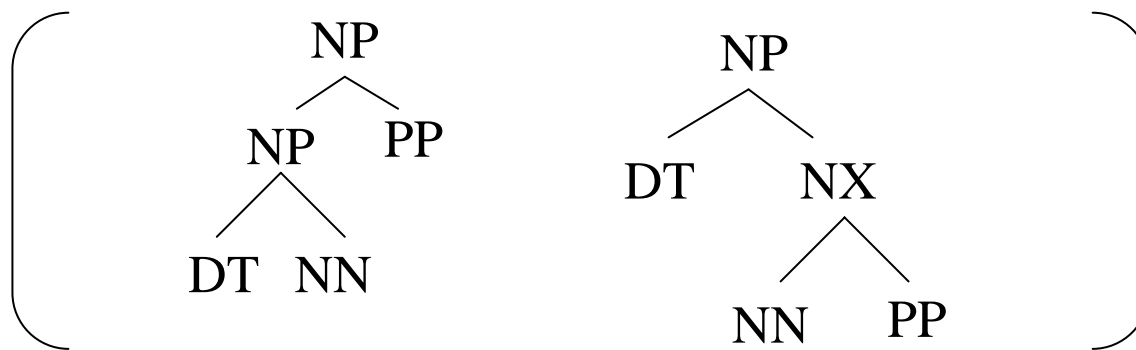


CFG tree



HPSG tree





Estimation of production probabilities

$$\begin{aligned}
 & P \left(\left(\begin{array}{c} \text{NP} \\ \swarrow \quad \searrow \\ \text{NP} \quad \text{PP} \\ \swarrow \quad \searrow \\ \text{DT} \quad \text{NN} \end{array} \quad \begin{array}{c} \text{NP} \\ \swarrow \quad \searrow \\ \text{DT} \quad \text{NX} \\ \swarrow \quad \searrow \\ \text{NN} \quad \text{PP} \end{array} \right) \right) \\
 &= \frac{\# \left(\left(\begin{array}{c} \text{NP} \\ \swarrow \quad \searrow \\ \text{NP} \quad \text{PP} \\ \swarrow \quad \searrow \\ \text{DT} \quad \text{NN} \end{array} \quad \begin{array}{c} \text{NP} \\ \swarrow \quad \searrow \\ \text{DT} \quad \text{NX} \\ \swarrow \quad \searrow \\ \text{NN} \quad \text{PP} \end{array} \right) \right)}{\# \left(\left(\begin{array}{c} \text{NP} \\ \triangle \\ * \end{array} \quad \begin{array}{c} \text{NP} \\ \triangle \\ * \end{array} \right) \right)}
 \end{aligned}$$

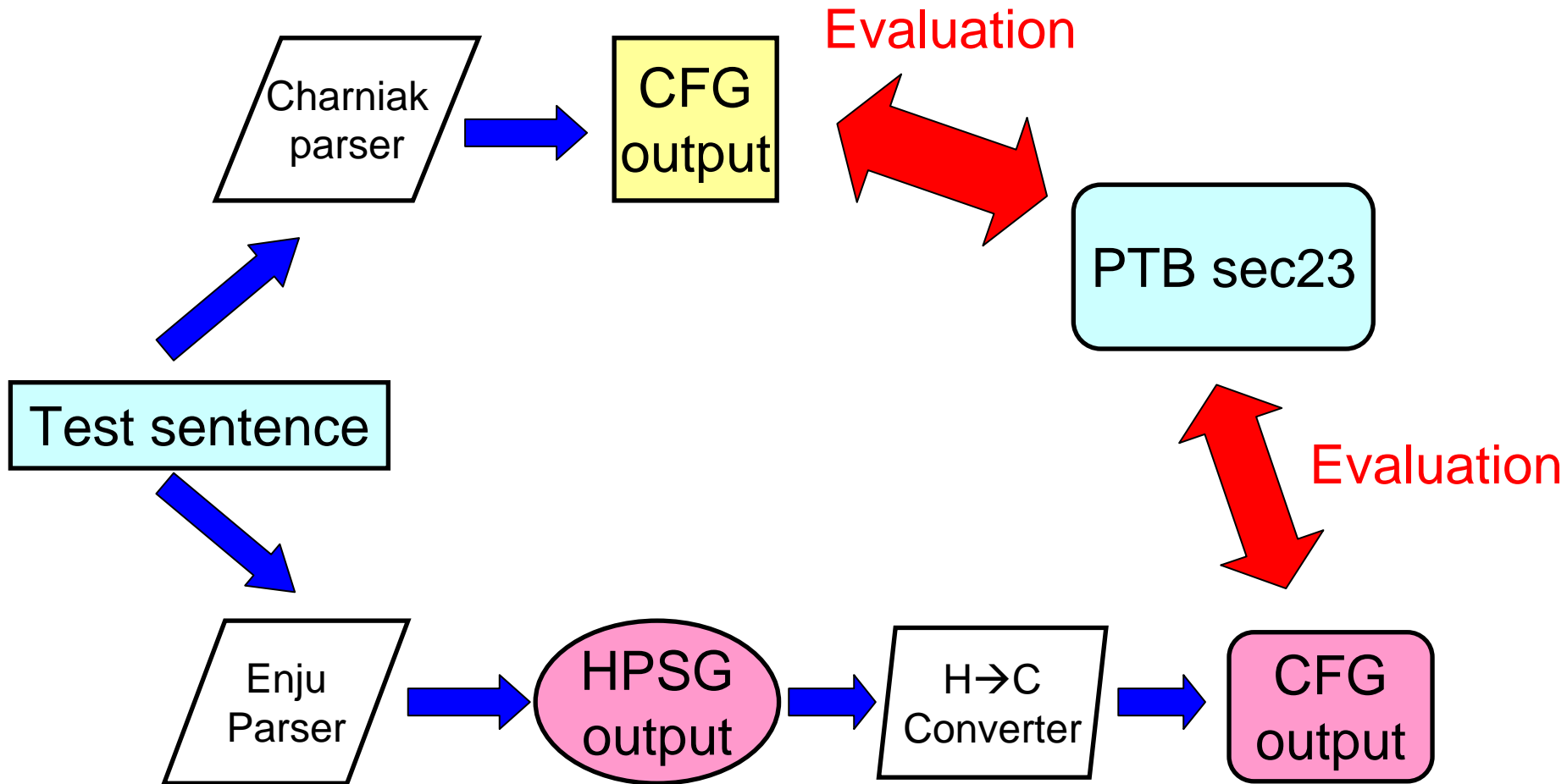
Outline

- Motivation
- Shallow CFG tree as the common format
- Tree conversion algorithm
- **Experiments**
- Conclusion

Experiment setting

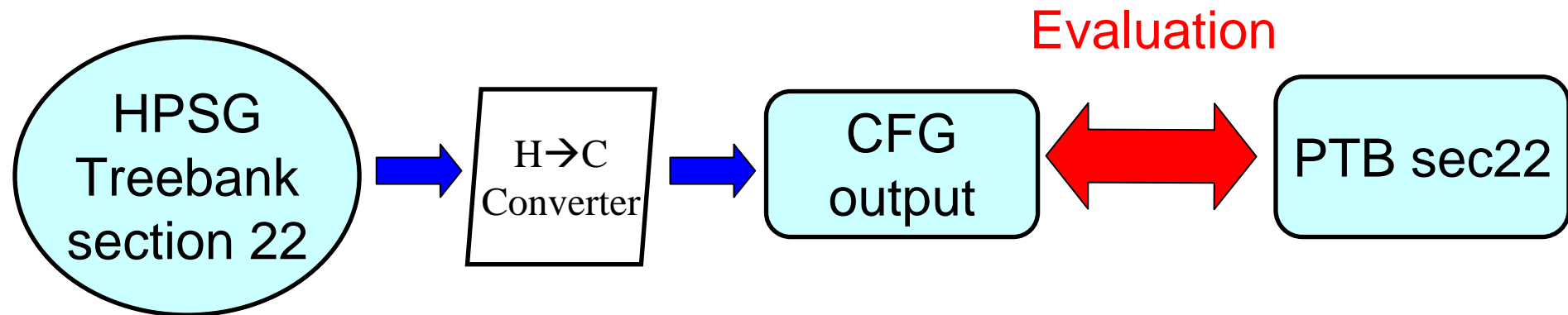
- Parsers
 - Enju parser (HPSG) [Ninomiya et al., 2006]
 - Charniak's parser (PTB-CFG) [Charniak, 2000]
 - Charniak & Johnson's parser/reranker (PTB-CFG) [Charniak and Johnson, 2005]
- Training data for the parsers and the converter
 - PennTreebank section 02-21
 - Enju HPSG treebank section 02-21 [Miyao, 2004]
 - Converted from PTB sec02-21 using hand-crafted transformation rules

Experiment 1 : HPSG \rightarrow CFG conv.



Experiment1 :

Evaluation of the converter



Result1 : HPSG \rightarrow CFG conv.

	LP (%)	LR (%)	F1 (%)	Dep (%)
H \rightarrow C converter	98.41	98.08	98.24	98.76

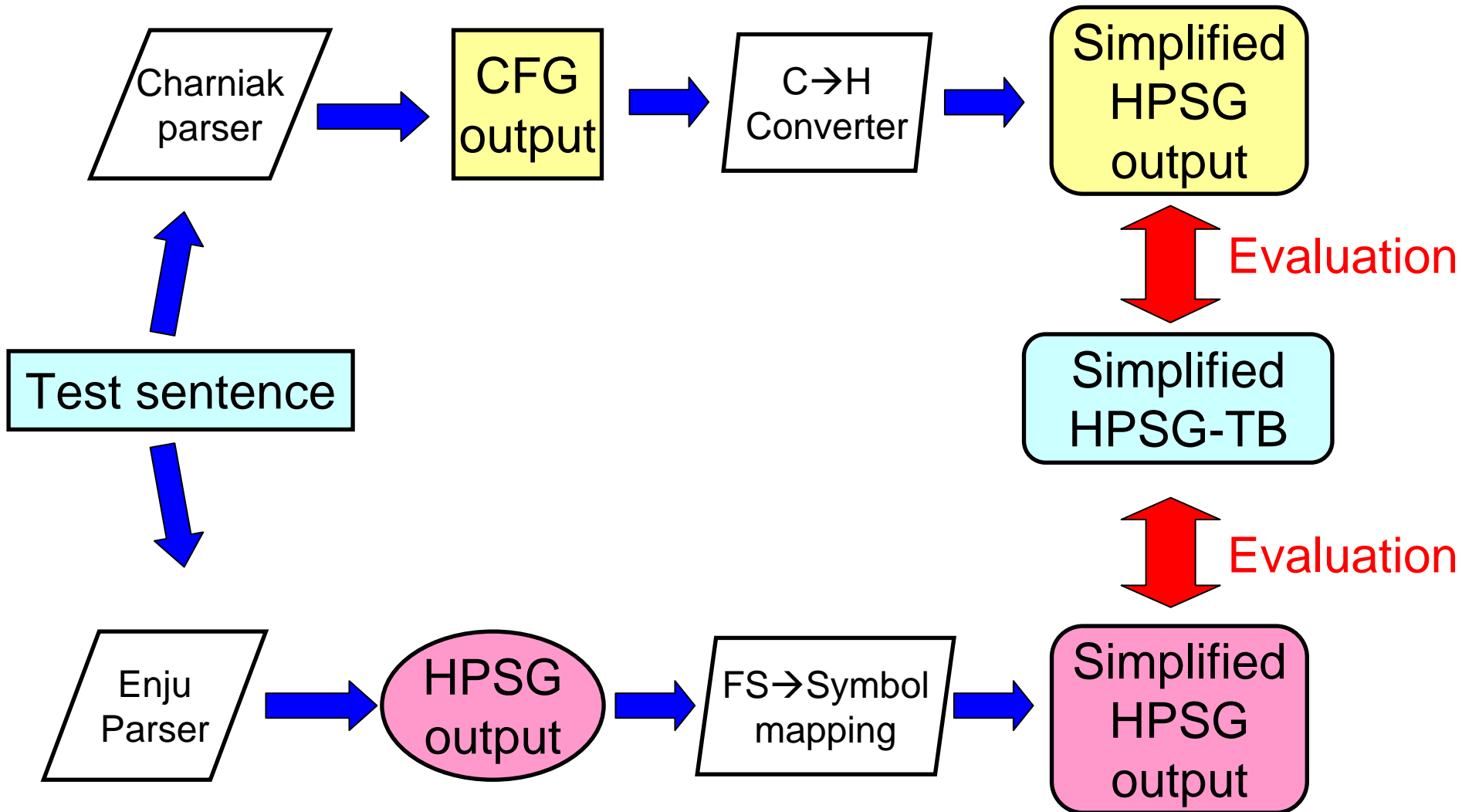
Parser	LP (%)	LR (%)	F1 (%)	Dep (%)
Enju + H \rightarrow C conv.	87.18	86.47	86.82	90.77
Charniak	89.49	88.78	89.13	92.50
C & J	91.79	90.88	91.33	93.66

- Dep: accuracy of head-dependent relations obtained by using Collins' head find rules

Experiment 2: CFG \rightarrow HPSG conv.

- Common format: Simplified HPSG tree
 - Feature structures are mapped to atomic symbols (e.g., S, SX, NP, NX, etc.)

Experiment2: CFG \rightarrow HPSG conv.



Result: CFG \rightarrow HPSG conv.

	LP (%)	LR (%)	F1 (%)
C \rightarrow H converter	97.54	97.45	97.49

Parser	LP (%)	LR (%)	F1 (%)
Enju	90.79	90.30	90.54
Charniak + Conv.	90.07	89.97	90.02
C & J + Conv.	91.44	91.31	91.37

Experiment 3:

Comparative error analysis

- Output format: bi-lexical dependencies obtained by applying Collins' head finding rules on (converted) PTB-CFG trees
- McNemar's test: significance of the difference in the number of errors
- Typing of errors: a tuple of
 - POS of dependent
 - POS of wrong head
 - POS of correct head
- Ex. PP-attachment error: (Prep, Verb, Noun)

McNemar's test on the # of errors

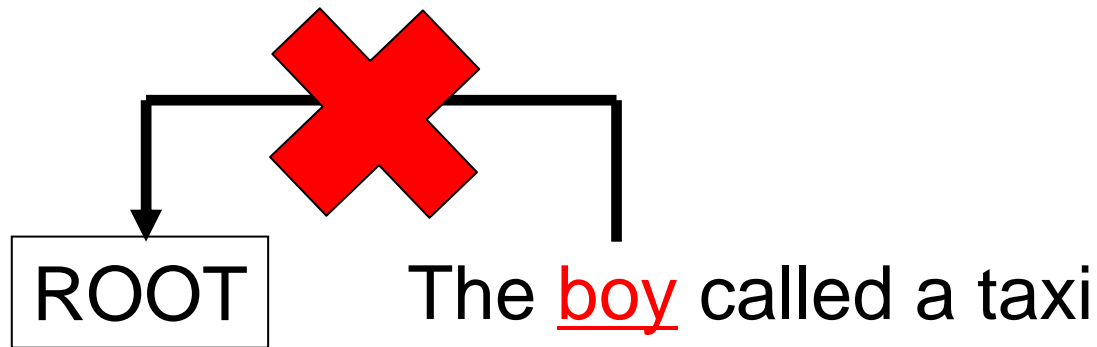
Dep POS	Head POS		#Error Enju	#Error Charniak	p-value
	Correct	Wrong			
VB	ROOT	NN	20	1	8.57e-5
NN	VB	NN	94	55	1.34e-4
NN	VB	ROOT	14	0	5.12e-4
CD	CD	IN	0	11	2.57e-3
CD	NN	CD	12	2	5.55e-3
VB	VB	NN	26	9	8.83e-3
DT	NN	VB	15	4	1.59e-2

McNemar's test on the # of errors

Dep POS	Head POS		#Error Enju	#Error Charniak	p-value
	Correct	Wrong			
VB	ROOT	NN	20	1	8.57e-5
NN	VB	NN	94	55	1.34e-4
NN	VB	ROOT	14	0	5.12e-4
CD	CD	IN	0	11	2.57e-3
CD	NN	CD	12	2	5.55e-3
VB	VB	NN	26	9	8.83e-3
DT	NN	VB	15	4	1.59e-2

Cause of the root identification errors

- Most of the root errors were caused by POS-tagging errors



Cause of the root identification errors

- Most of the root errors were caused by POS-tagging errors

The/DT boy/NN **called/VBN** a/DT taxi/NN

Cause of the root identification errors

- Most of the root errors were caused by POS-tagging errors

The/DT boy/NN **called/VBN** a/DT taxi/NN

- Different strategies for the POS-tagging
 - Charniak's parser: integrated with the parser
 - Enju parser: pre-processing before the parsing phase

Outline

- Motivation
- Shallow CFG tree as the common format
- Tree conversion algorithm
- Experiments
- **Conclusion**

Conclusion

- Comparative evaluation of parsers using shallow CFG analysis as the 'common-format'
- Accurate conversion algorithm based on a synchronous grammar
- Meaningful difference in the parsers' performances is still observable in such a shallow representation

Cross-bracketing transformation

Proposal:

Shallow CFG analysis as the common format

- ‘Deep part’ of the analysis (e.g., long-distance dependencies) is not represented, but
- Highly accurate conversion is possible

Proposal

- CFG trees as the common-format
 - A simple conversion algorithm exists
 - Based on synchronous grammars
 - Highly accurate conversion is possible
 - e.g., 98% F1-score for HPSG-to-CFG conversion
 - Meaningful differences among the parsers' performances are still observable