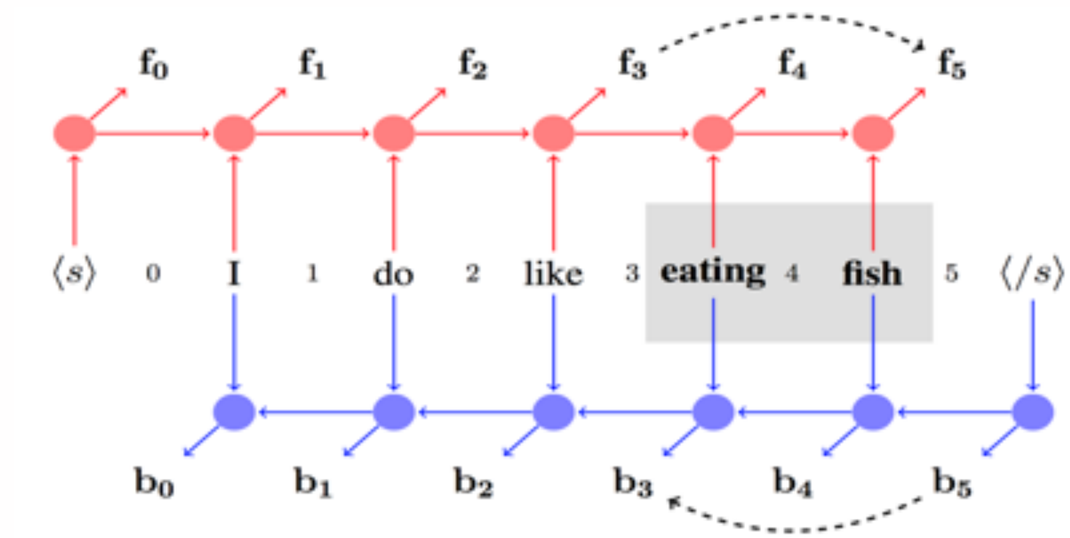
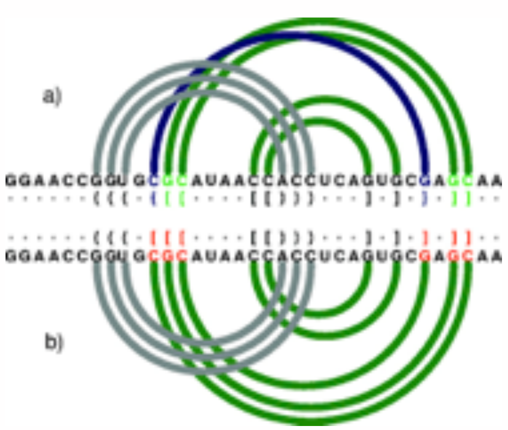
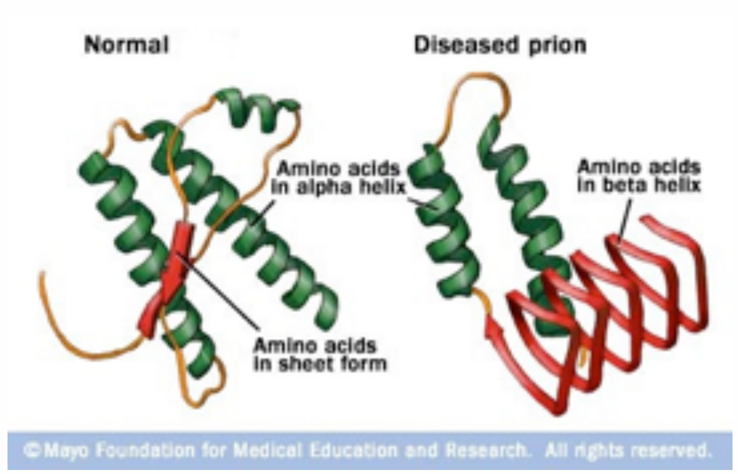
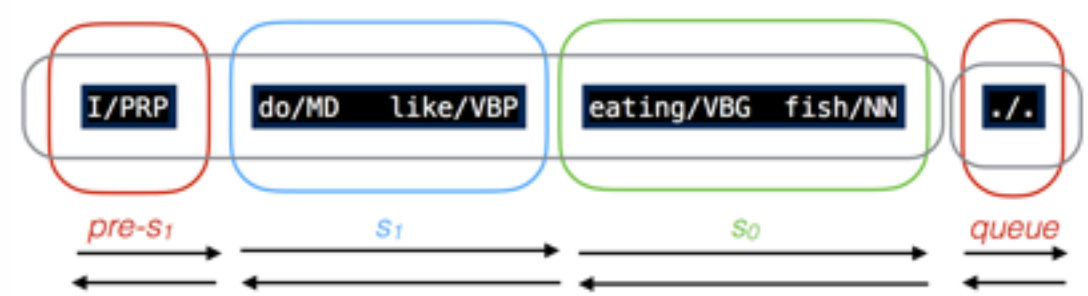
Marrying Dynamic Programming with Recurrent Neural Networks

I eat sushi with tuna from Japan







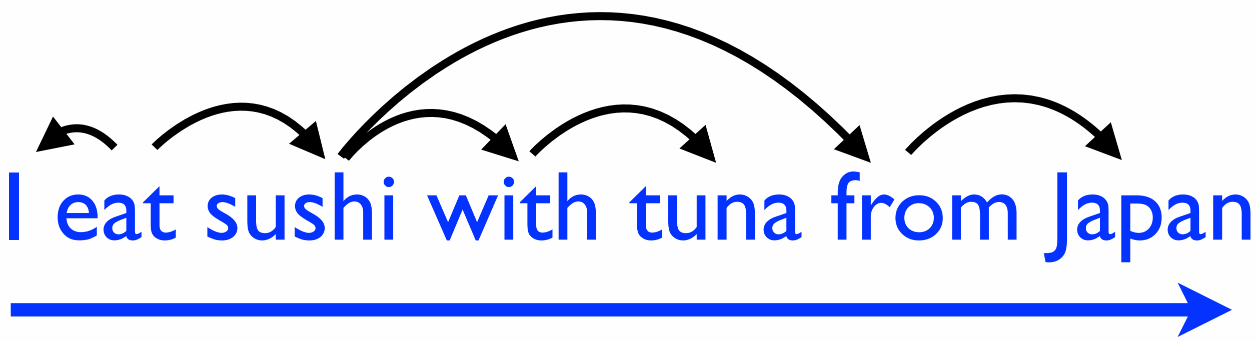


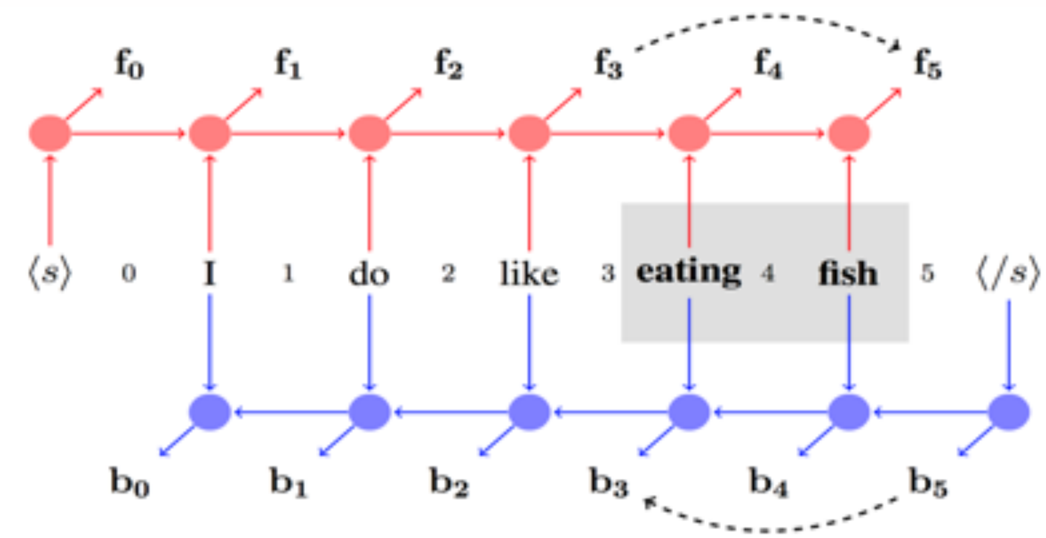
Liang Huang

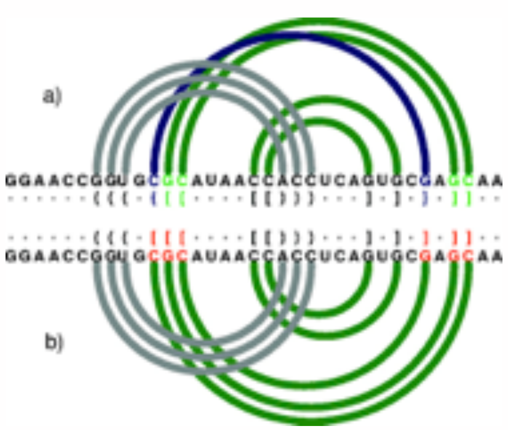
Oregon State University

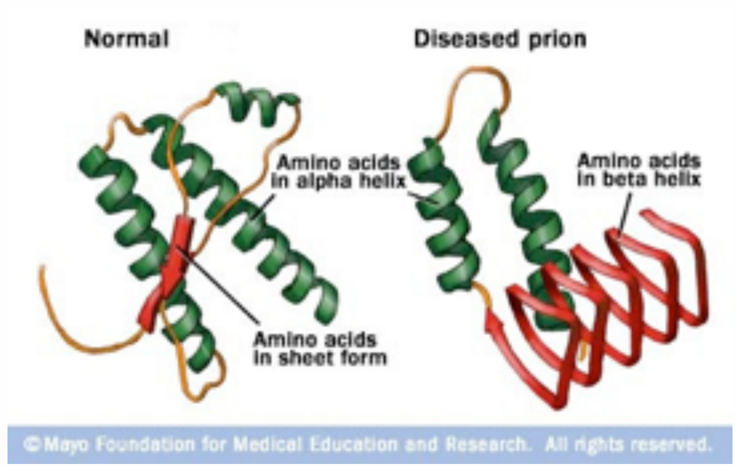
Structured Prediction Workshop, EMNLP 2017, Copenhagen, Denmark

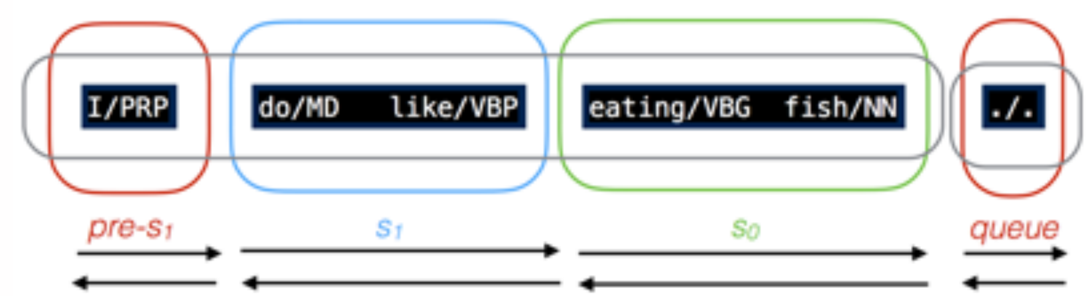
Marrying Dynamic Programming with Recurrent Neural Networks









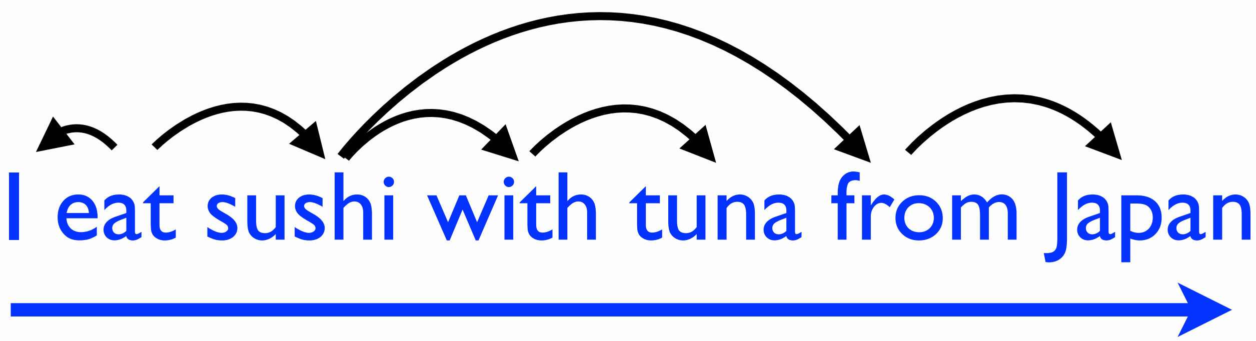


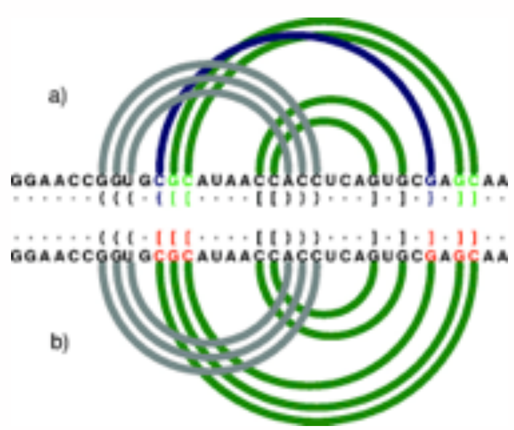
Liang Huang

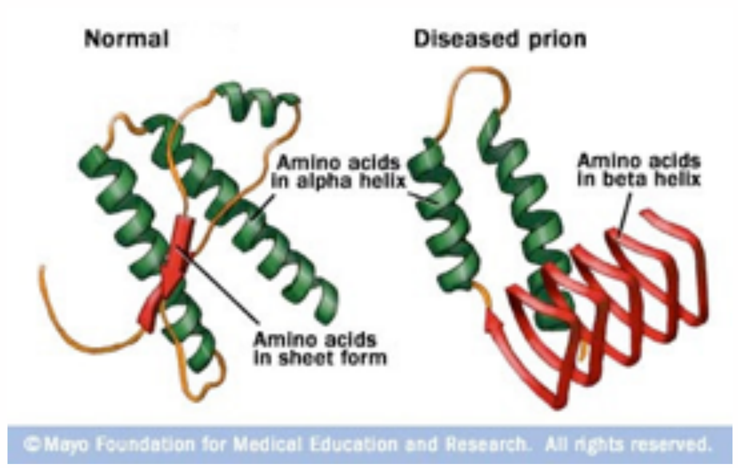
Oregon State University

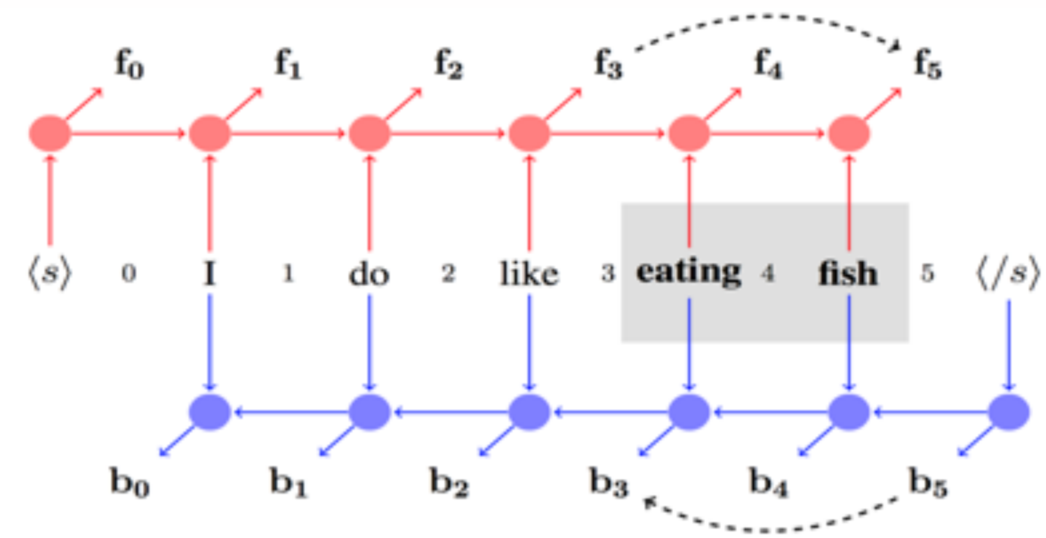
Structured Prediction Workshop, EMNLP 2017, Copenhagen, Denmark

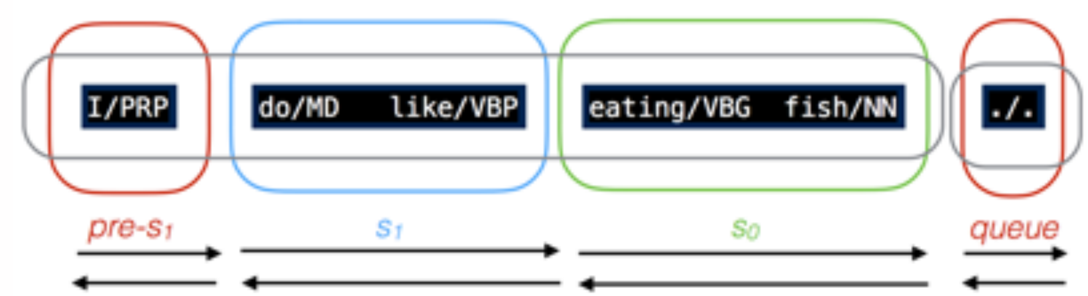
Marrying Dynamic Programming with Recurrent Neural Networks

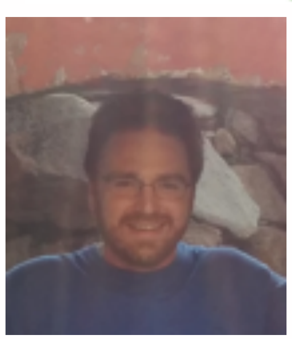











James Cross

Liang Huang

Oregon State University

Structured Prediction Workshop, EMNLP 2017, Copenhagen, Denmark

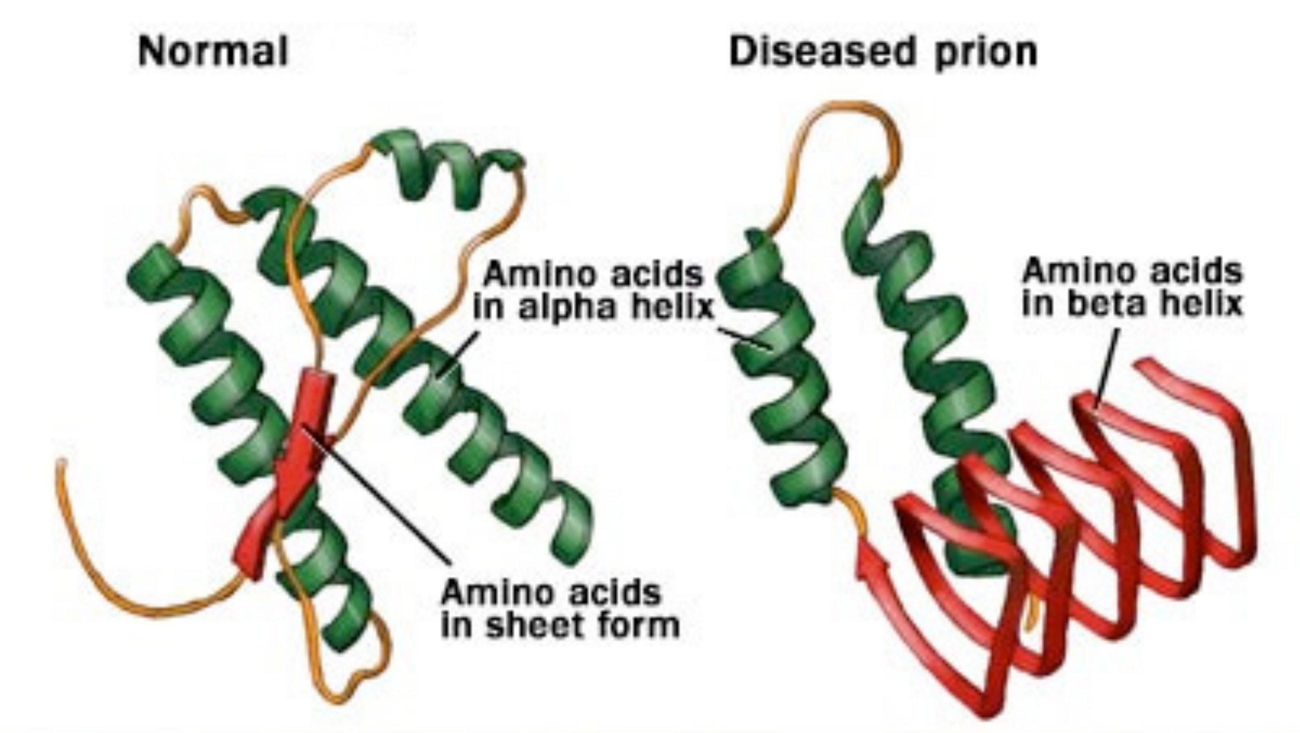
Structured Prediction is Hard!

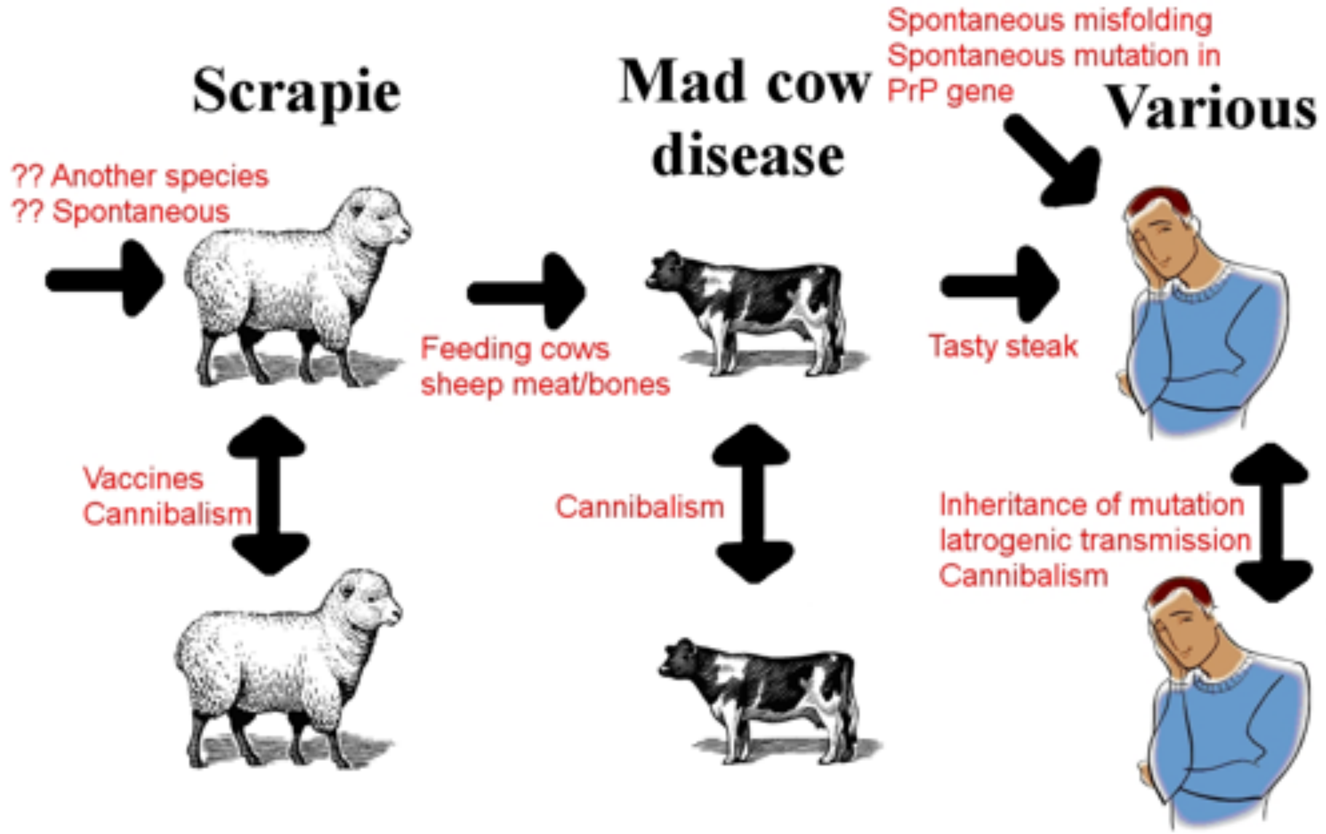
Not Easy for Humans Either...

(structural ambiguity :-P)

Not Even Easy for Nature!

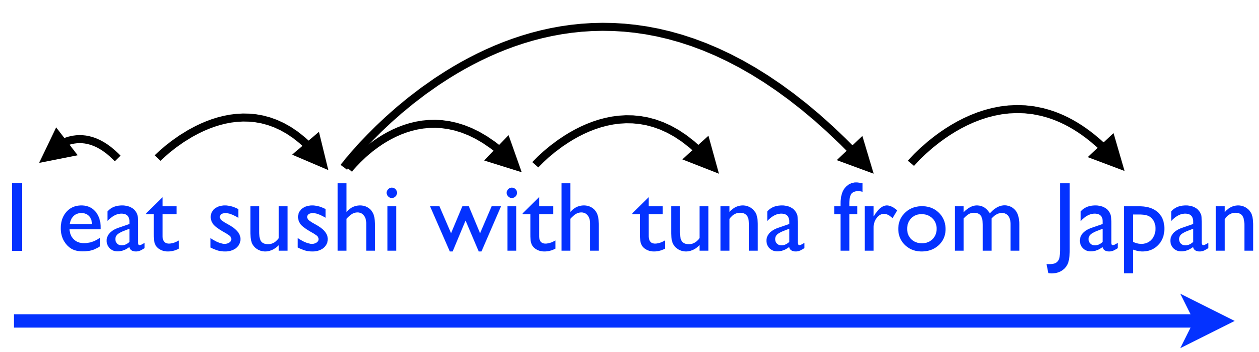
- prion: "misfolded protein"
  - structural ambiguity for the same amino-acid sequence
  - similar to different interpretations under different contexts
  - causes mad-cow diseases etc.

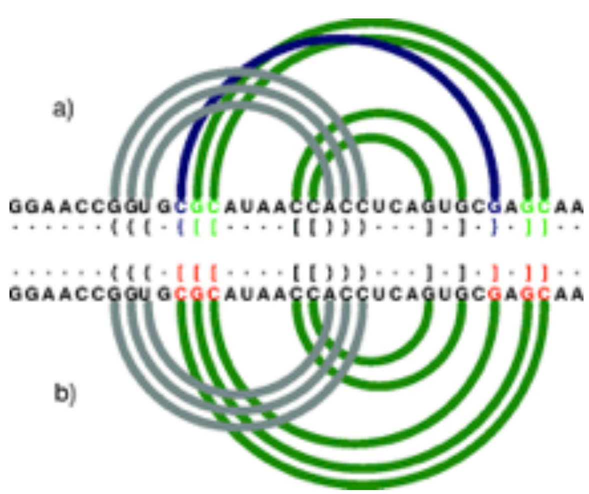


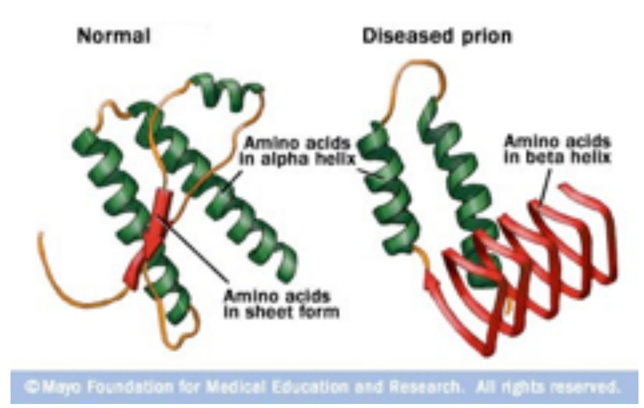


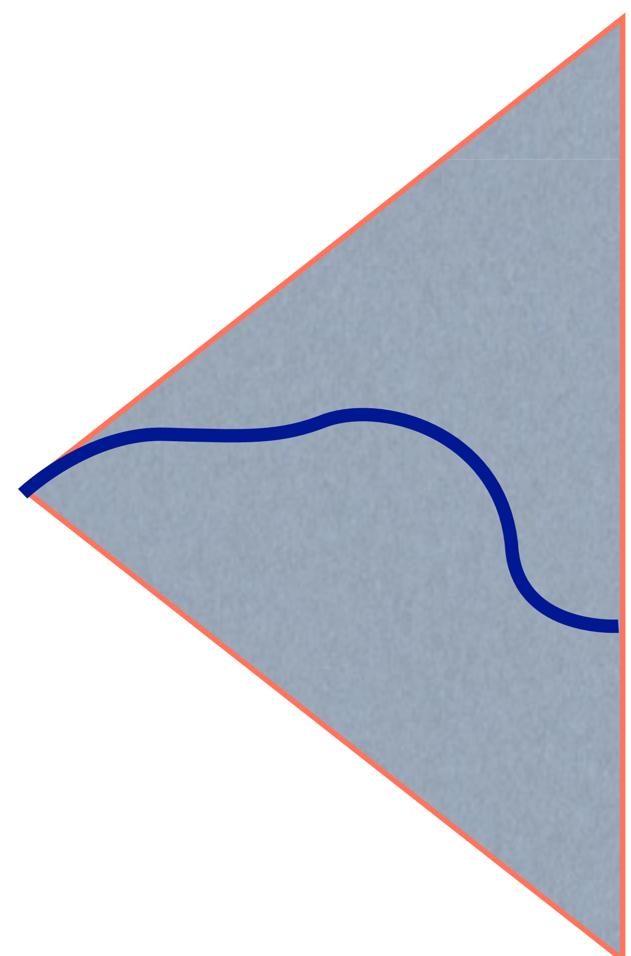
Case Study: Parsing and Folding

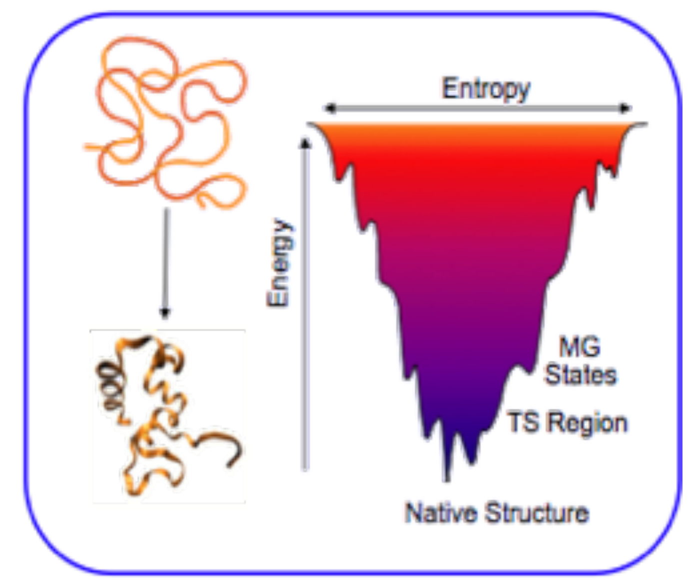
- both problems have exponentially large search space
  - both can be modeled by grammars (context-free & above)
- question 1: how to search for the highest-scoring structure?
- question 2: how to make gold structure score the highest?





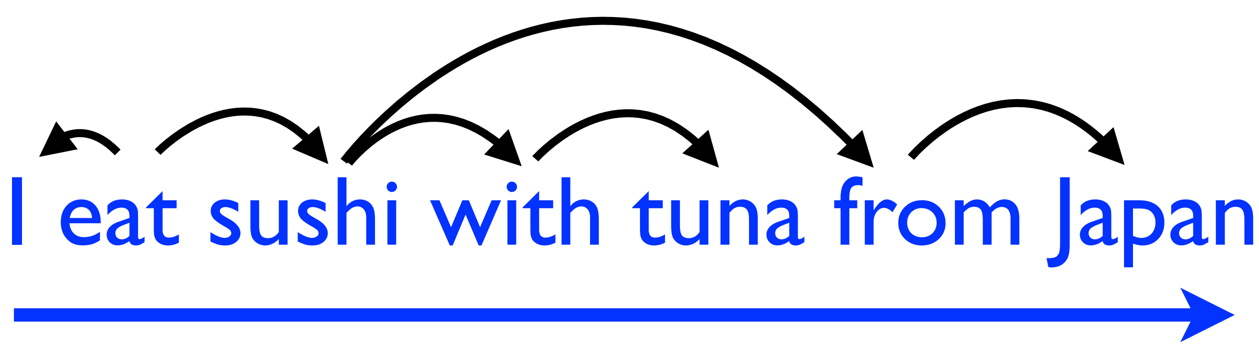


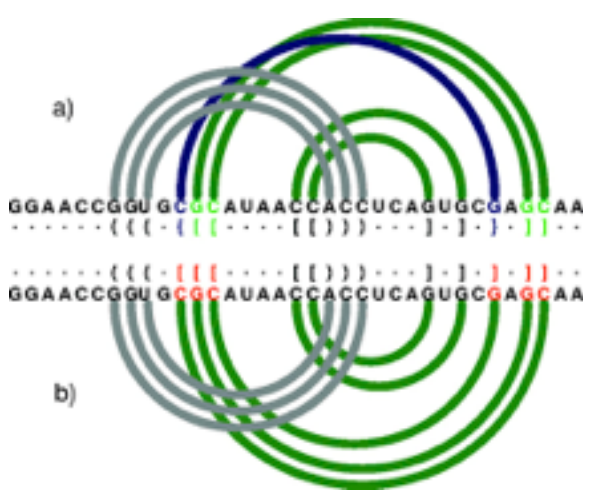


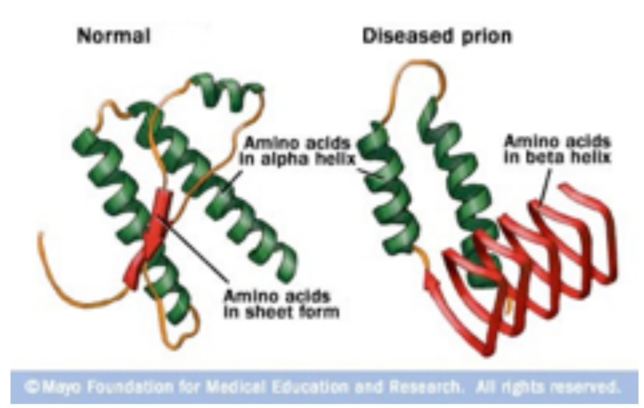


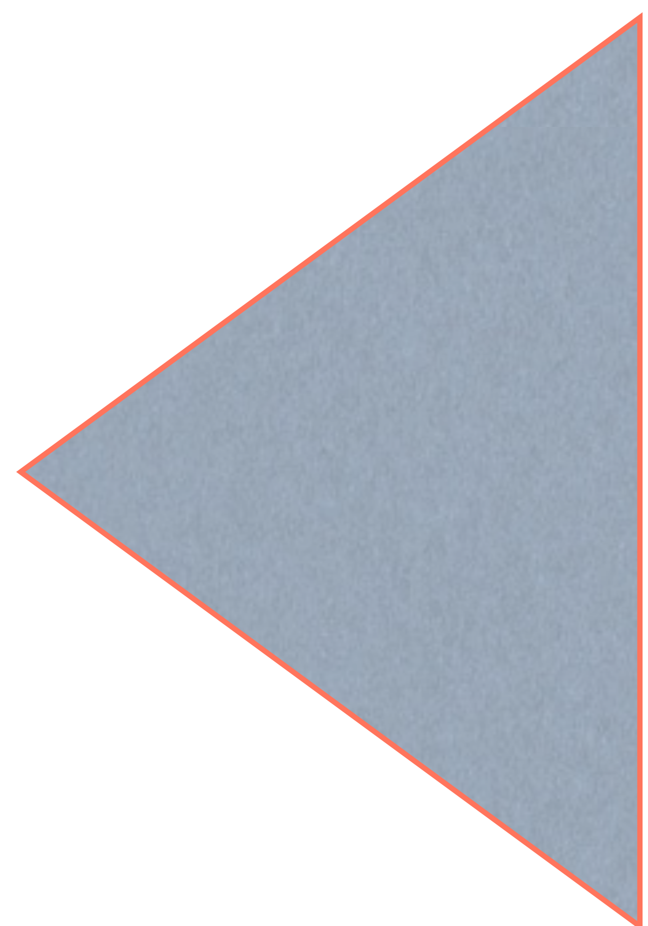
Solutions to Search and Learning

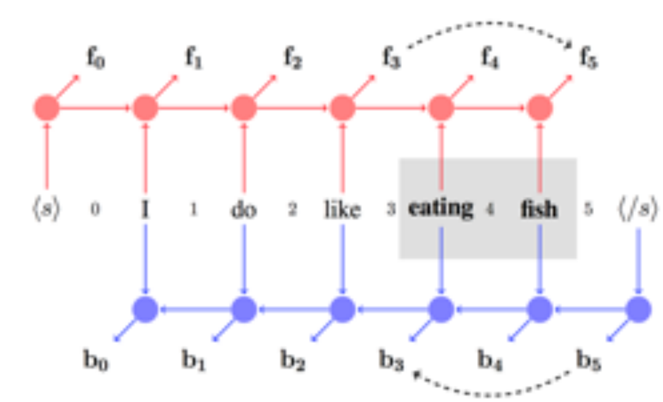
- question 1: how to search for the highest-scoring structure?
  - answer: dynamic programming to factor search space
- question 2: how to make gold structure score the highest?
  - answer: neural nets to automate feature engineering
- But do DP and neural nets like each other??





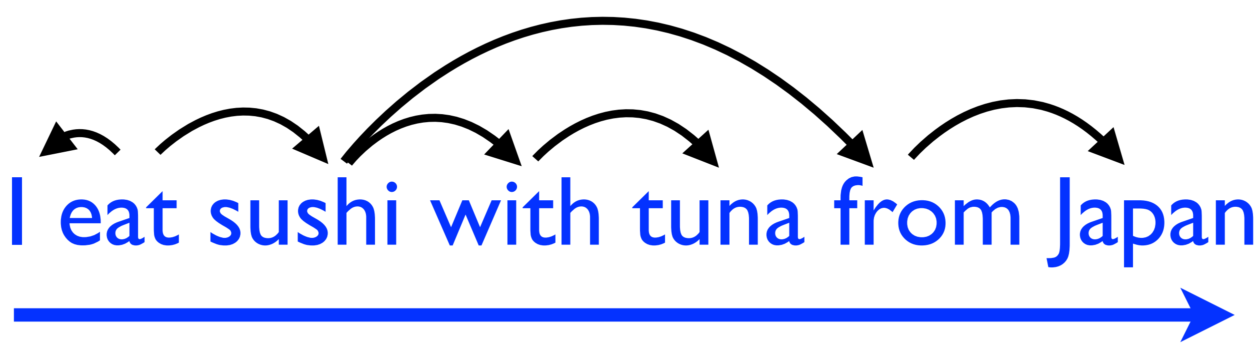


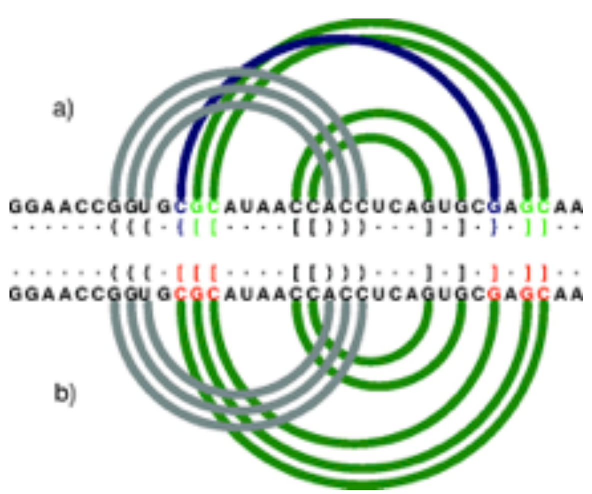


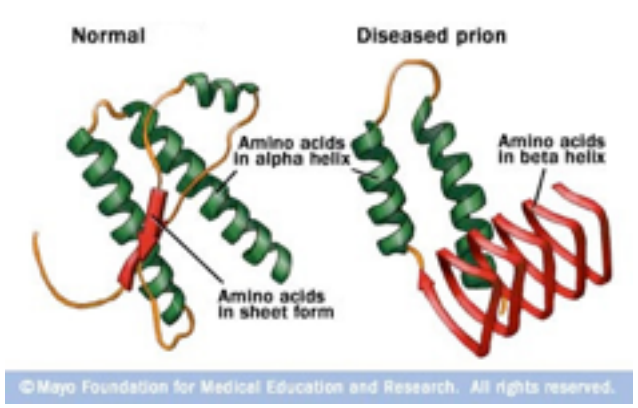


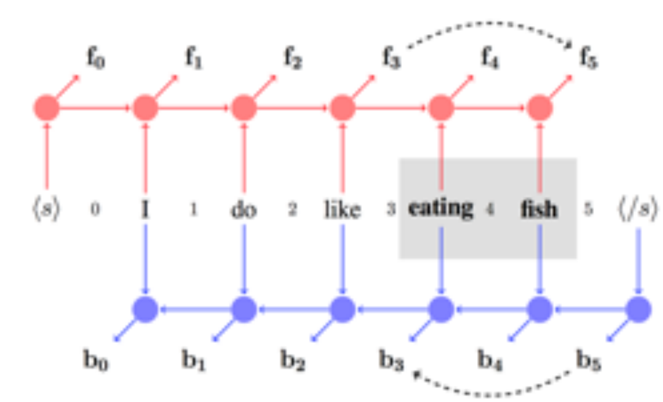
Solutions to Search and Learning

- question 1: how to search for the highest-scoring structure?
  - answer: dynamic programming to factor search space
- question 2: how to make gold structure score the highest?
  - answer: neural nets to automate feature engineering
- But do DP and neural nets like each other??





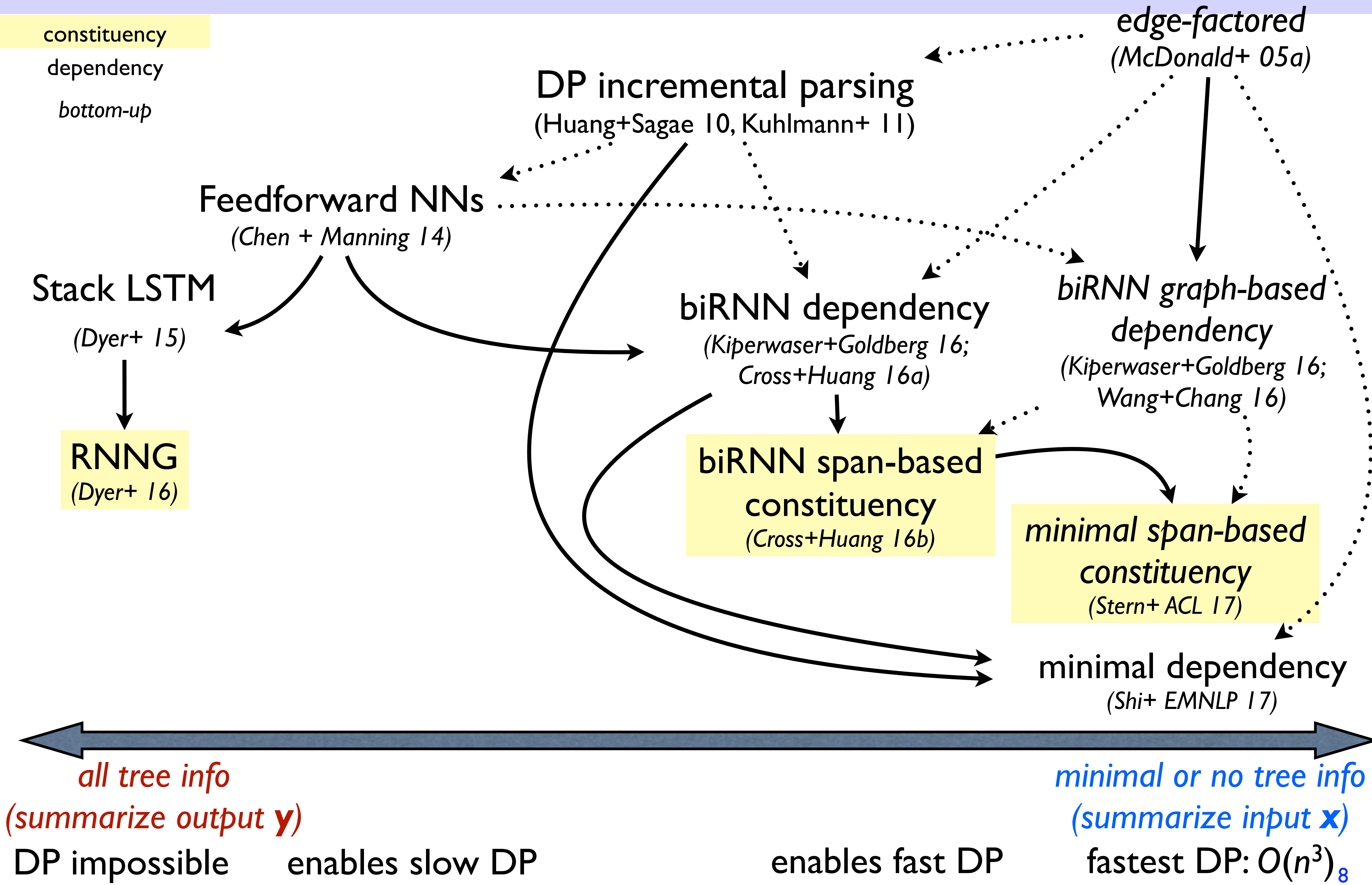




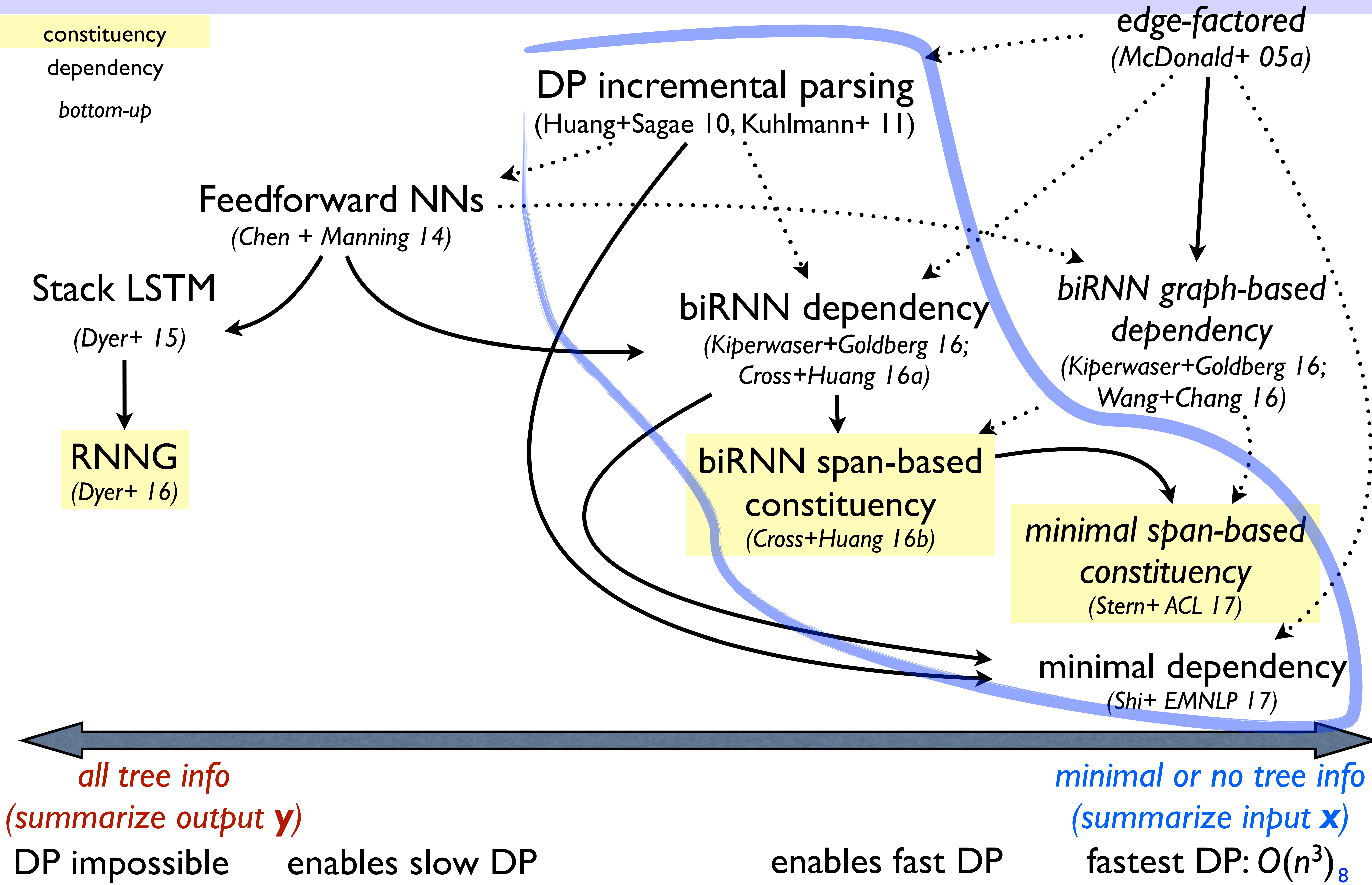
In this talk...

- Background
- Dynamic Programming for Incremental Parsing
- Features: from sparse to neural to recurrent neural nets
- Bidirectional RNNs: minimal features; no tree structures!
    - dependency parsing (Kiperwaser+Goldberg, 2016, Cross+Huang, 2016a)
    - span-based constituency parsing (Cross+Huang, 2016b)
- Marrying DP & RNNs *(mostly not my work!)*
    - transition-based dependency parsing (Shi et al, EMNLP 2017)
    - minimal span-based constituency parsing (Stern et al, ACL 2017)

Spectrum: Neural Incremental Parsing



Spectrum: Neural Incremental Parsing



Incremental Parsing with Dynamic Programming

(Huang & Sagae, ACL 2010*; Kuhlmann et al., ACL 2011; Mi & Huang, ACL 2015)

* *best paper nominee*

Incremental Parsing with Dynamic Programming

(Huang & Sagae, ACL 2010*; Kuhlmann et al., ACL 2011; Mi & Huang, ACL 2015)

* *best paper nominee*

Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

action	stack	queue

Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

	action	stack	queue
0	-	$\boxed{}$	I eat sushi ...

Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

	action	stack	queue
0	-		I eat sushi ...
1	shift	I	eat sushi with ...

Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

	action	stack	queue
0	-		I eat sushi ...
1	shift	I	eat sushi with ...
2	shift	I eat	sushi with tuna ...

Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

	action	stack	queue
0	-		I eat sushi ...
1	shift	I	eat sushi with ...
2	shift	I eat	sushi with tuna ...
3	l-reduce	eat I	sushi with tuna ...

Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

	action	stack	queue
0	-		I eat sushi ...
1	shift	I	eat sushi with ...
2	shift	I eat	sushi with tuna ...
3	l-reduce	eat I	sushi with tuna ...
4	shift	eat sushi I	with tuna from ...

Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

	action	stack	queue
0	-		I eat sushi ...
1	shift	I	eat sushi with ...
2	shift	I eat	sushi with tuna ...
3	l-reduce	eat I	sushi with tuna ...
4	shift	eat sushi I	with tuna from ...
5a	r-reduce	eat I sushi	with tuna from ...

Incremental Parsing (Shift-Reduce)

I eat sushi with tuna from Japan in a restaurant

	action	stack	queue
0	-		I eat sushi ...
1	shift	I	eat sushi with ...
2	shift	I eat	sushi with tuna ...
3	l-reduce	eat I	sushi with tuna ...
4	shift	eat sushi I	with tuna from ...
5a	r-reduce	eat I sushi	with tuna from ...
5b	shift	eat sushi with I	tuna from Japan ...

Incremental Parsing (Shift-Reduce)

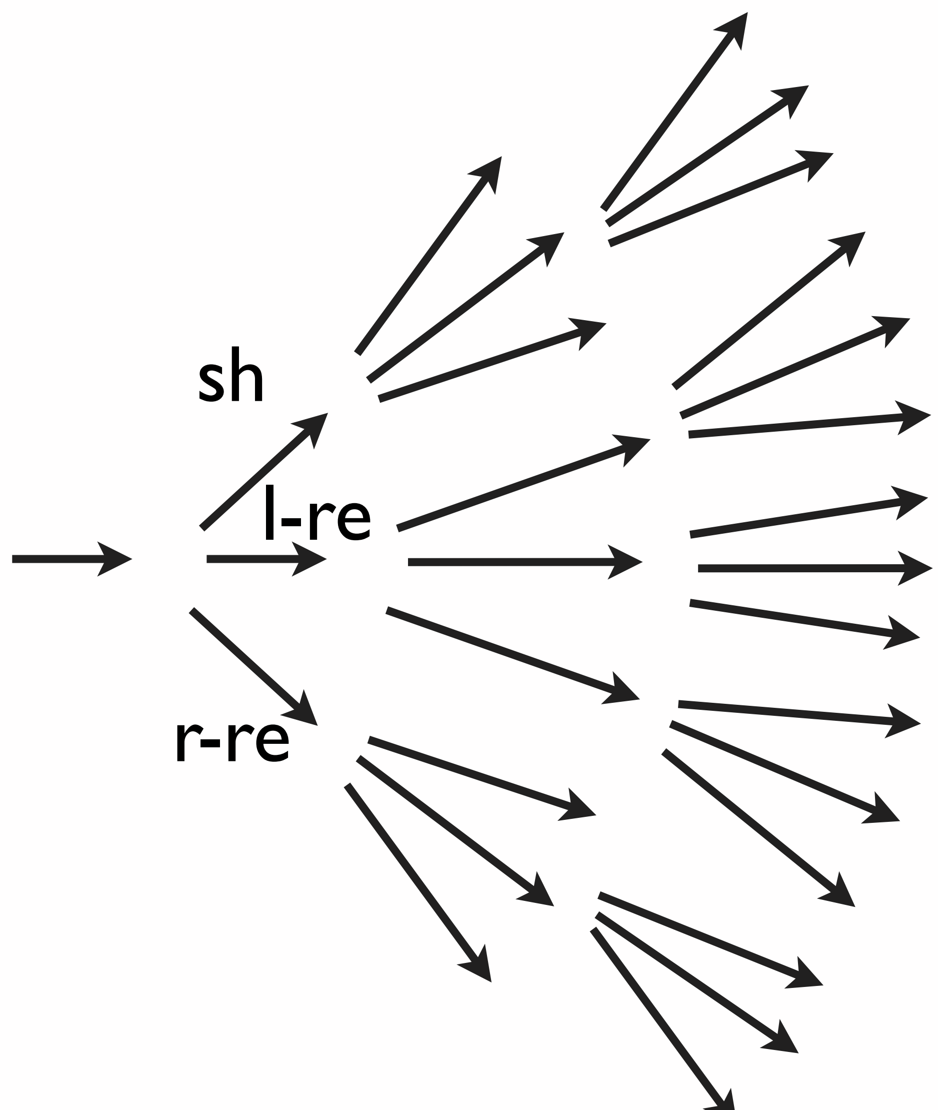
I eat sushi with tuna from Japan in a restaurant

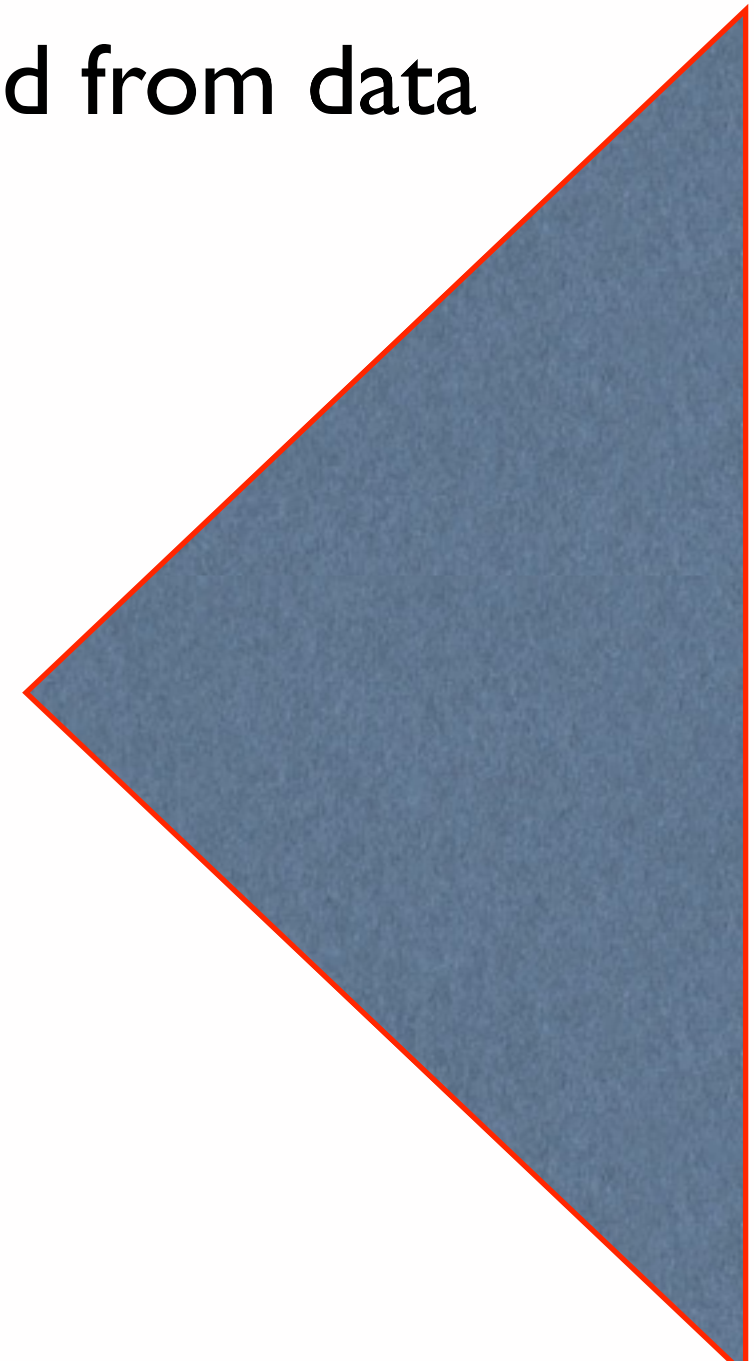
	action	stack	queue
0	-		I eat sushi ...
1	shift	I	eat sushi with ...
2	shift	I eat	sushi with tuna ...
3	l-reduce	eat	sushi with tuna ...
4	shift	eat sushi	with tuna from ...
5a	r-reduce	eat sushi	with tuna from ...
5b	shift	eat sushi with	tuna from Japan ...

shift-reduce conflict

Greedy Search

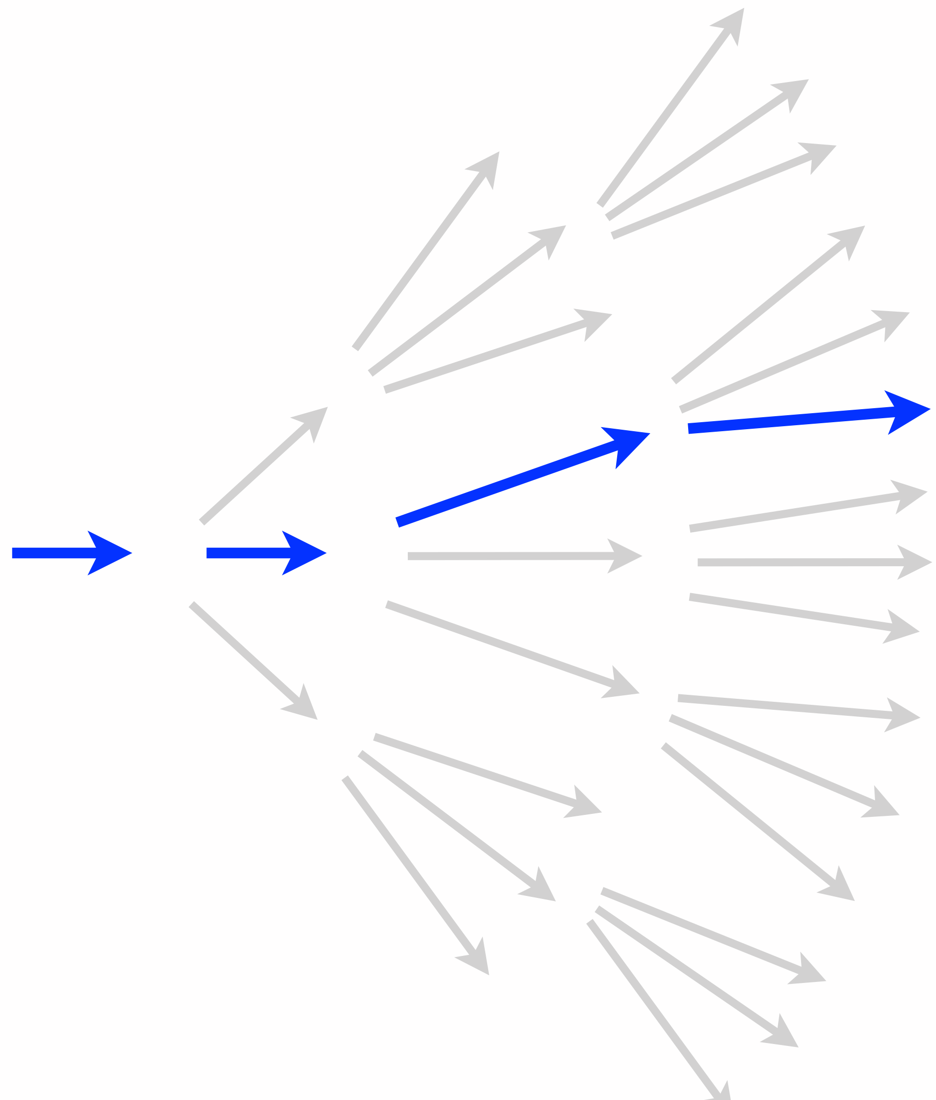
- each state => three new states (shift, l-reduce, r-reduce)
- greedy search: always pick the best next state
  - "best" is defined by a score learned from data

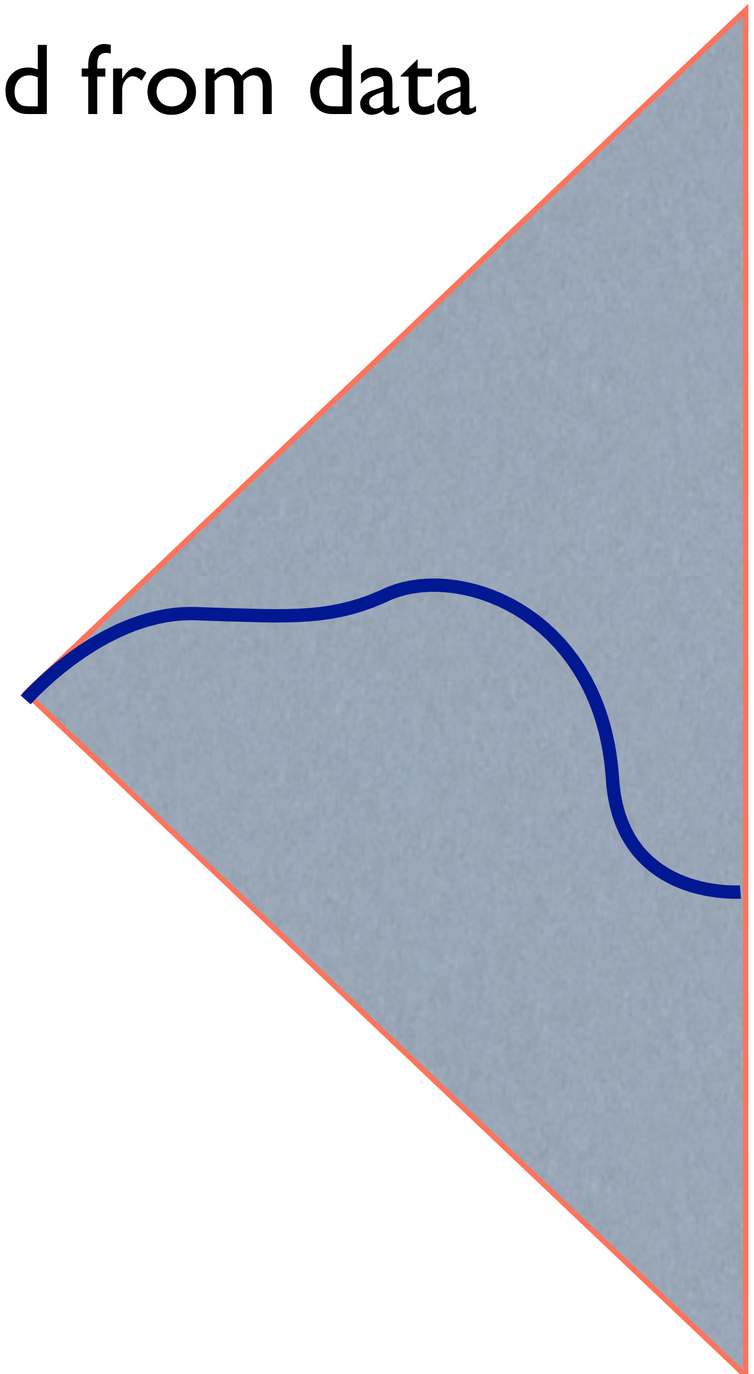




Greedy Search

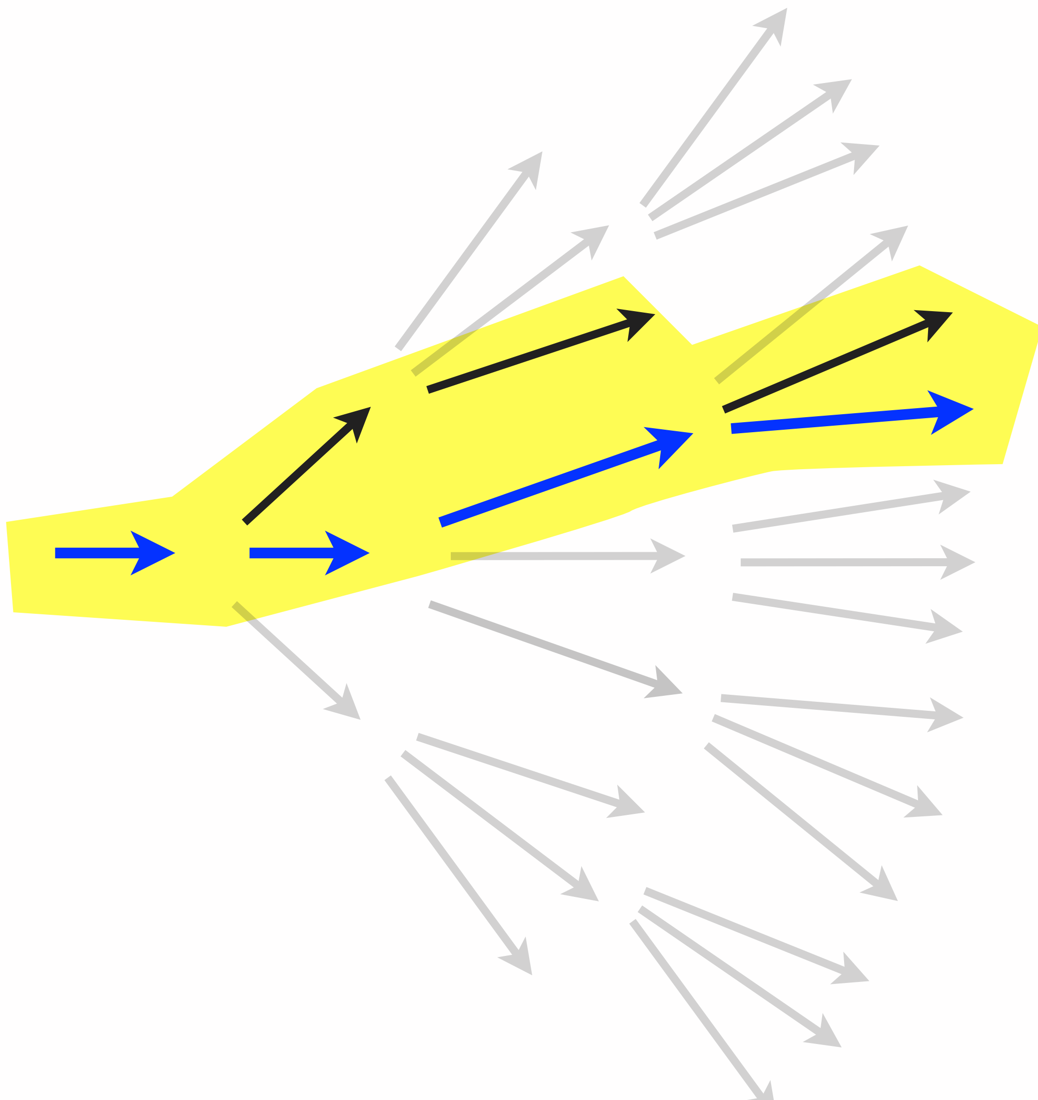
- each state => three new states (shift, l-reduce, r-reduce)
- greedy search: always pick the best next state
  - "best" is defined by a score learned from data

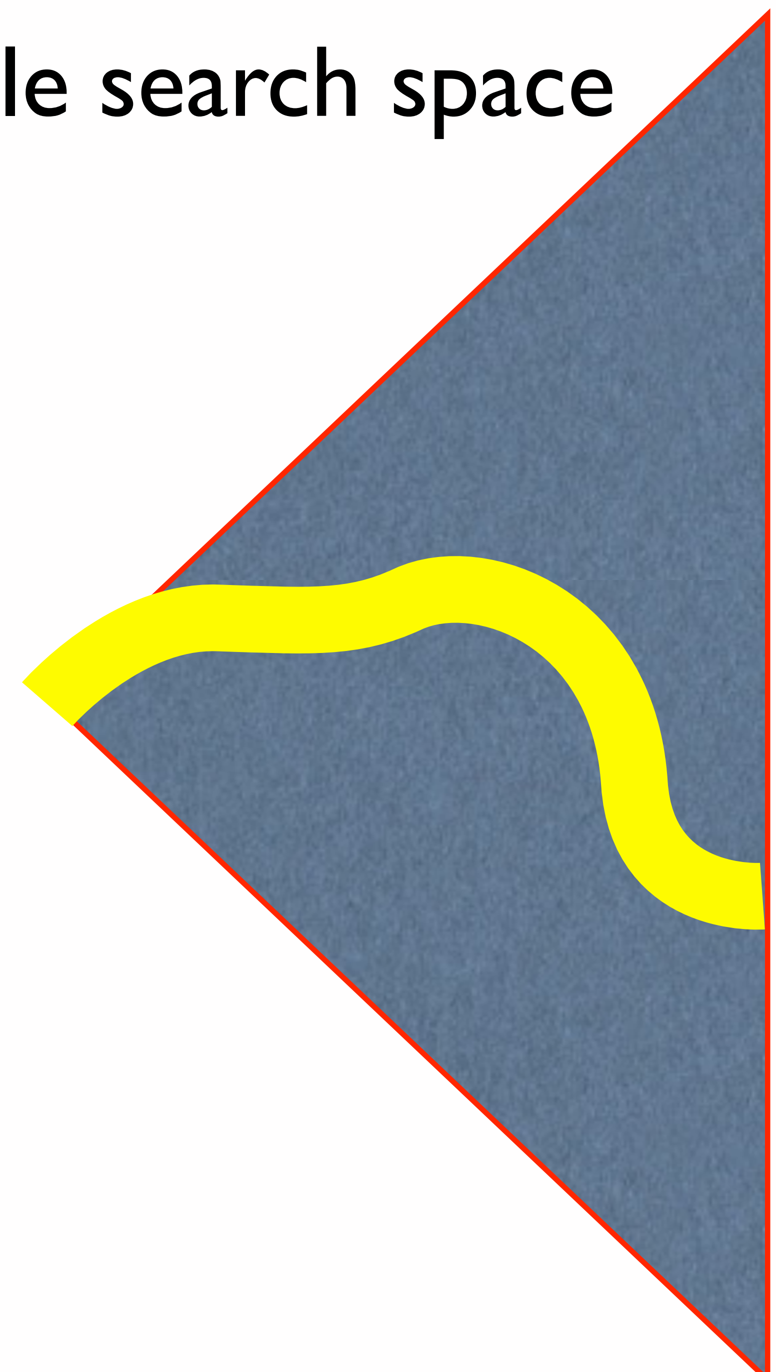




Beam Search

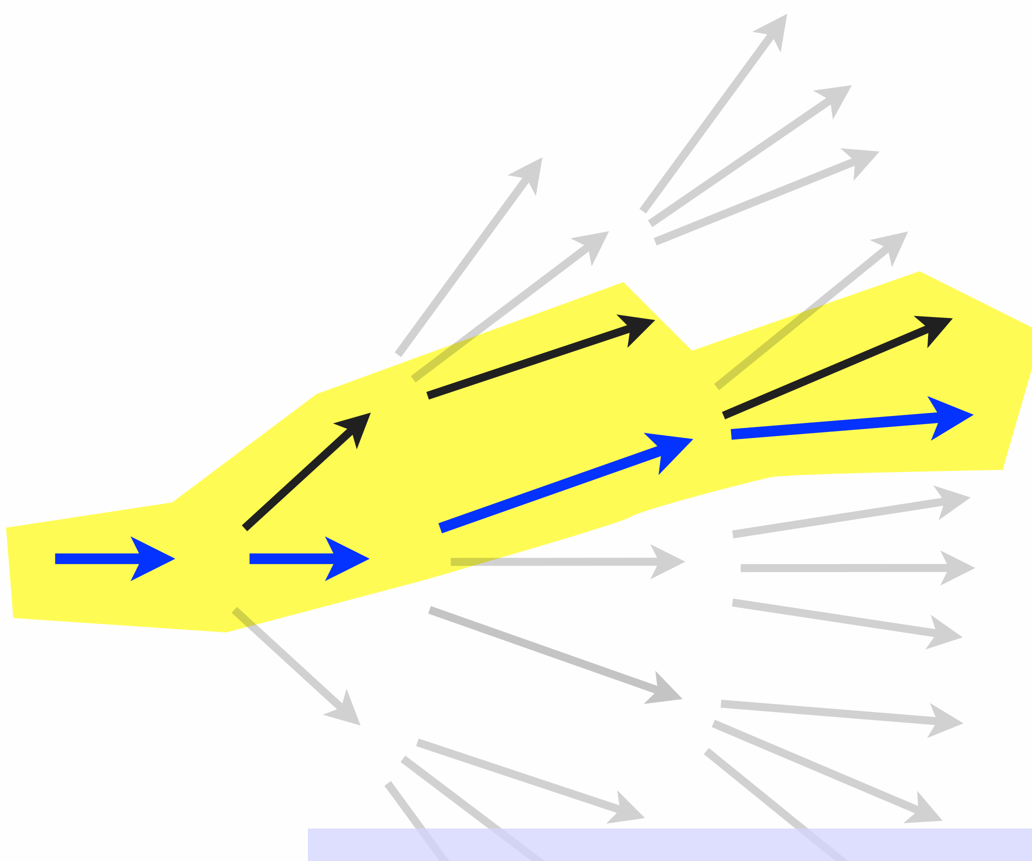
- each state => three new states (shift, l-reduce, r-reduce)
- beam search: always keep top-*b* states
  - still just a tiny fraction of the whole search space

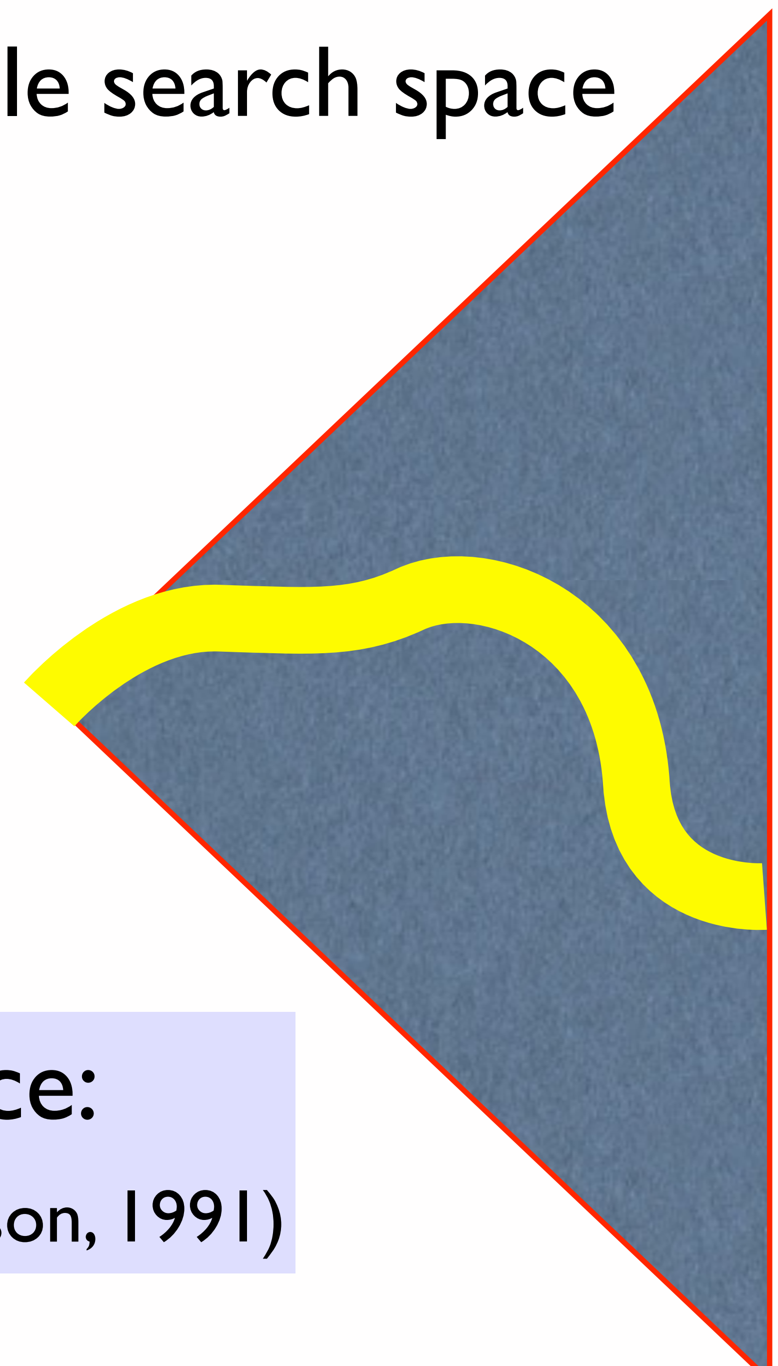




Beam Search

- each state => three new states (shift, l-reduce, r-reduce)
- beam search: always keep top-*b* states
  - still just a tiny fraction of the whole search space

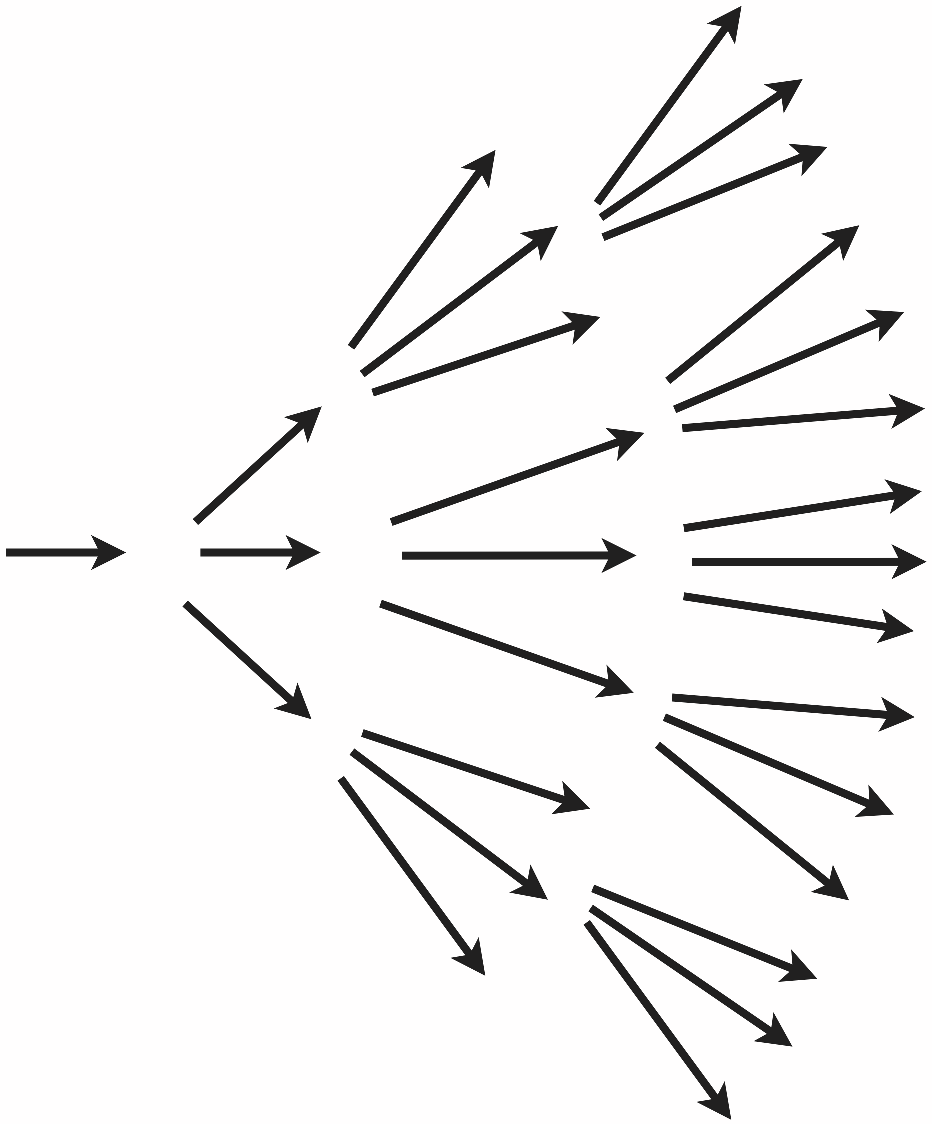


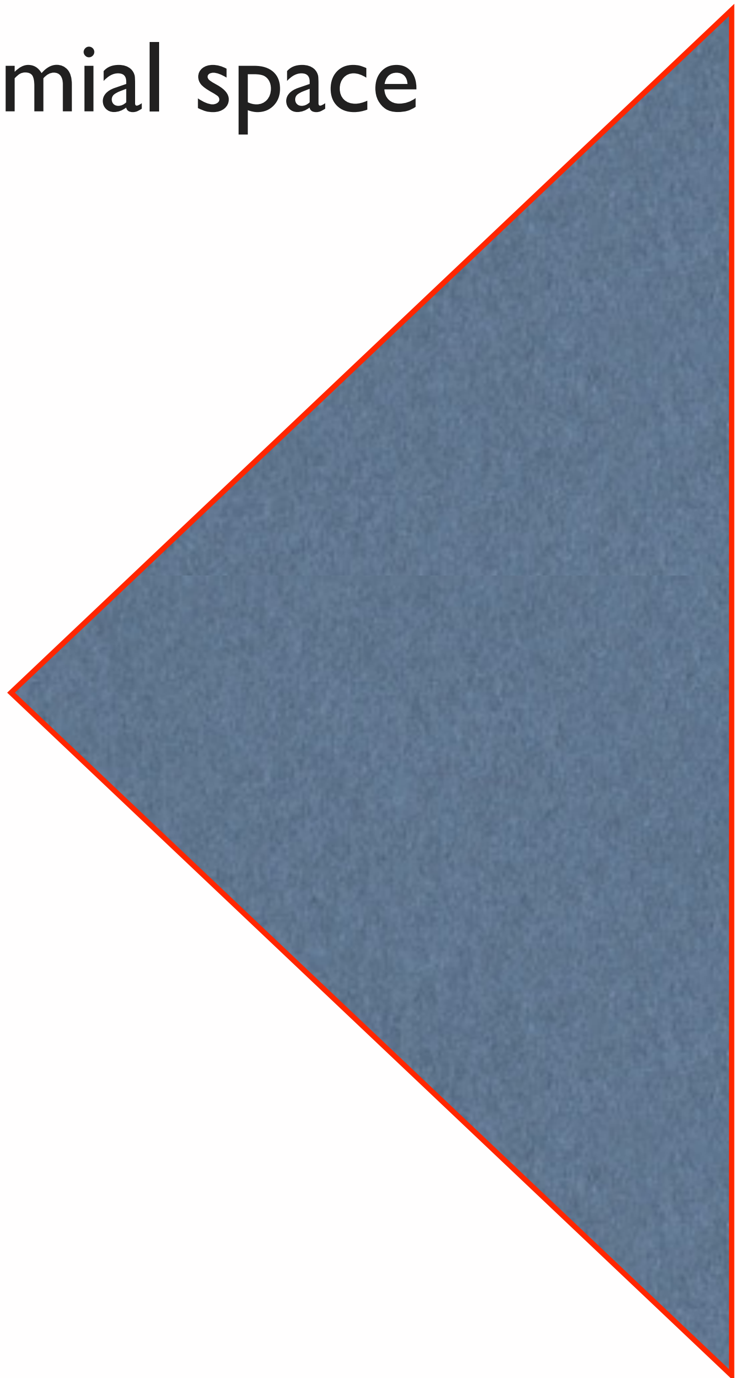


psycholinguistic evidence:
parallelism (Fodor et al, 1974; Gibson, 1991)

Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space

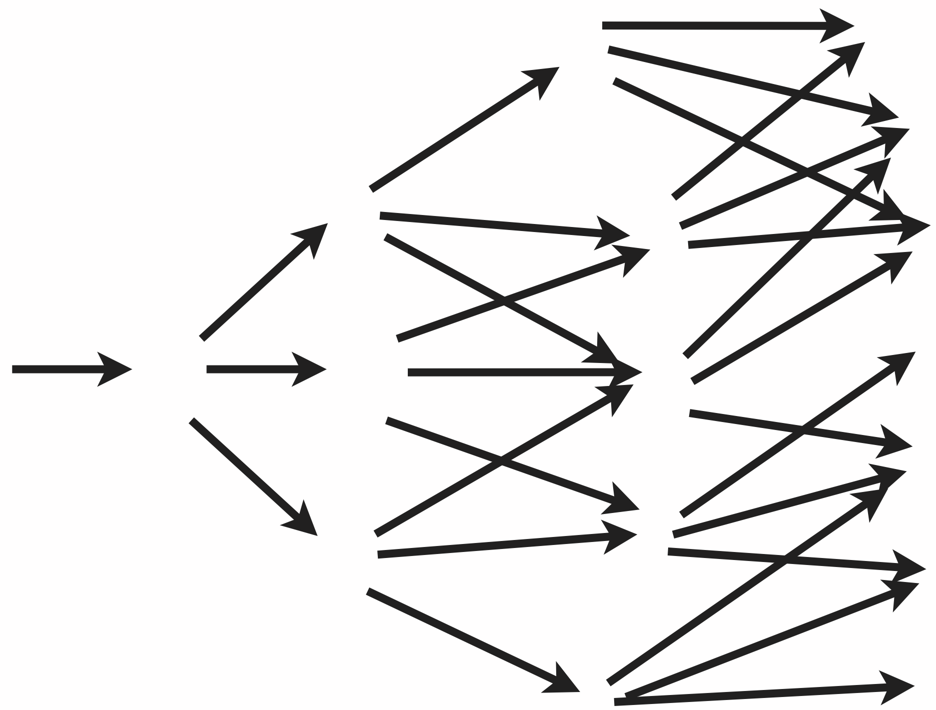


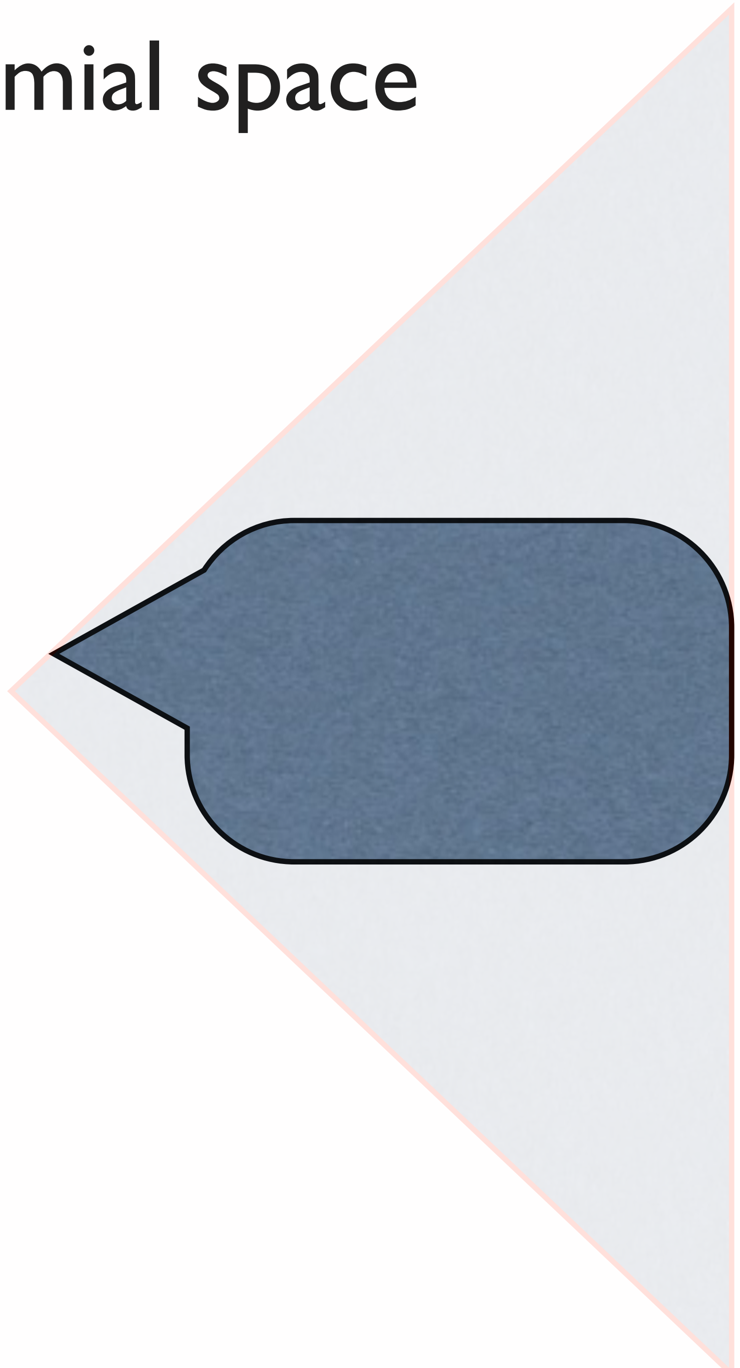


(Huang and Sagae, 2010)

Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space

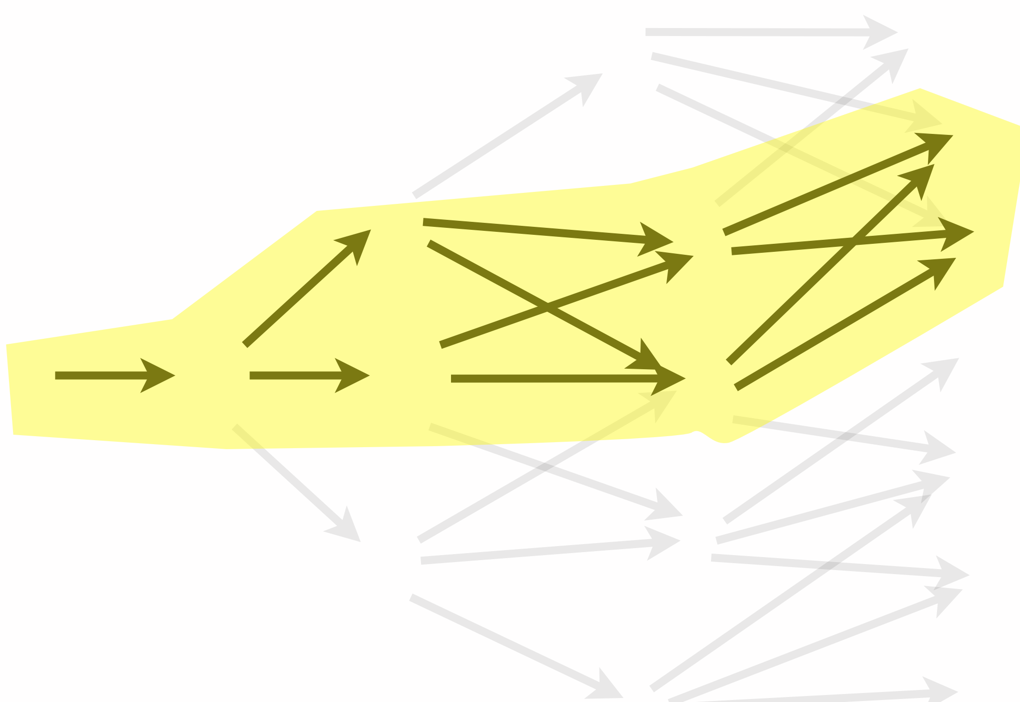


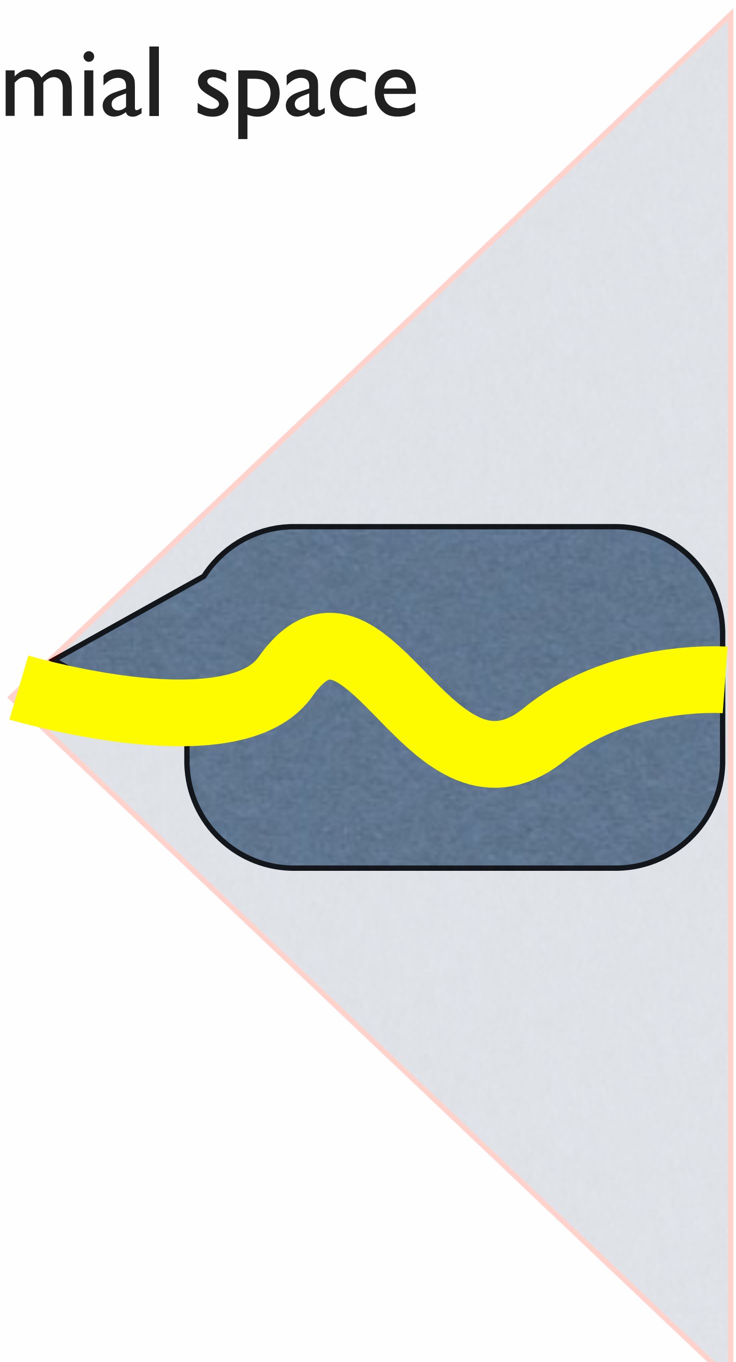


(Huang and Sagae, 2010)

Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space

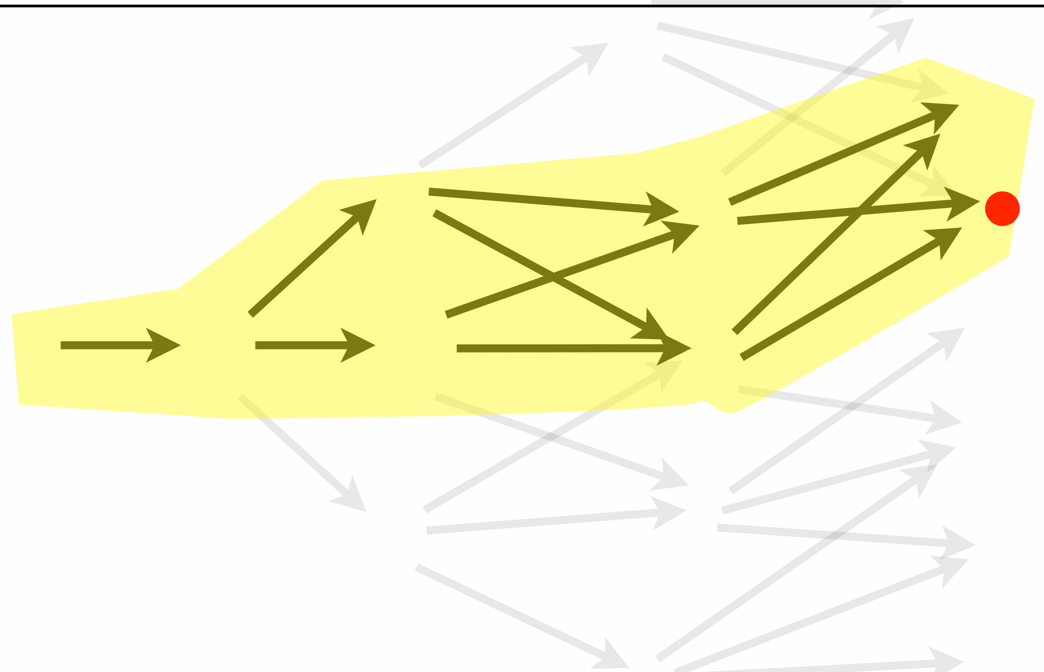




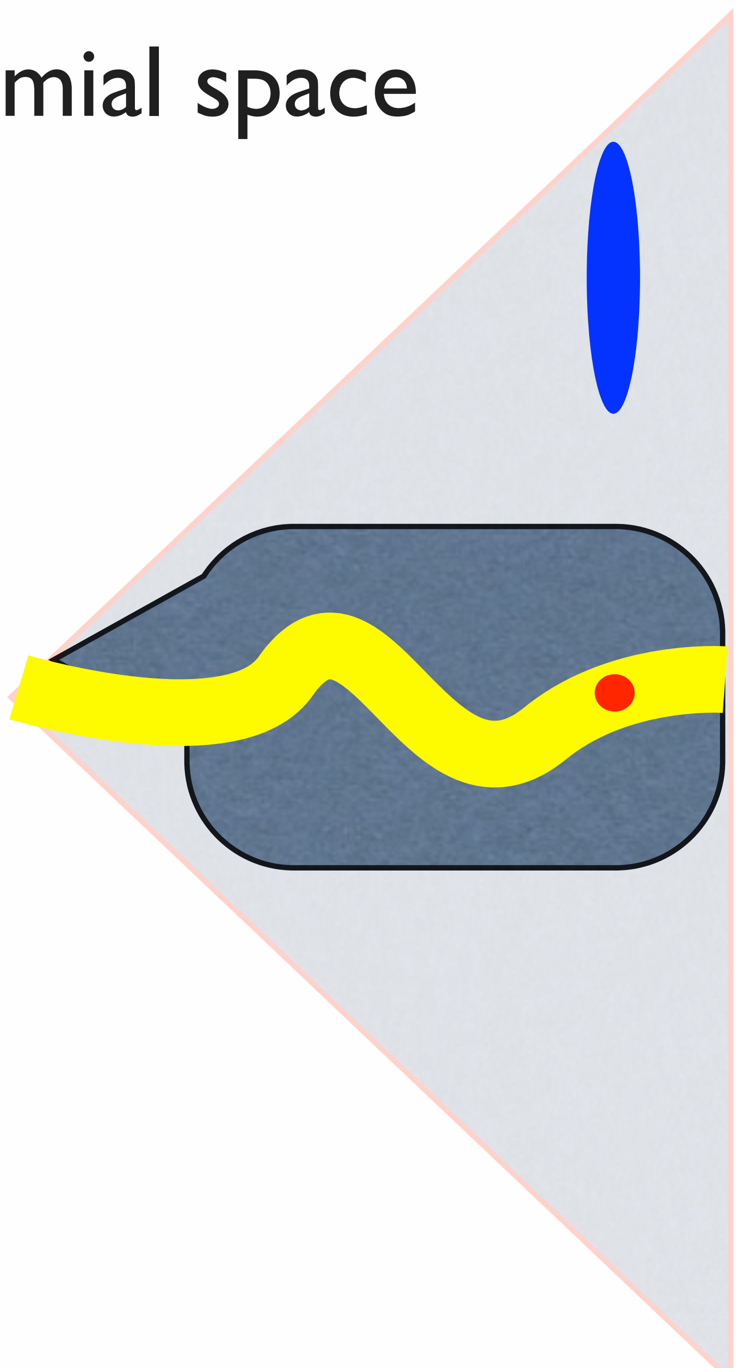
Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space

each DP state corresponds to
exponentially many non-DP states



graph-structured stack
(Tomita, 1986)

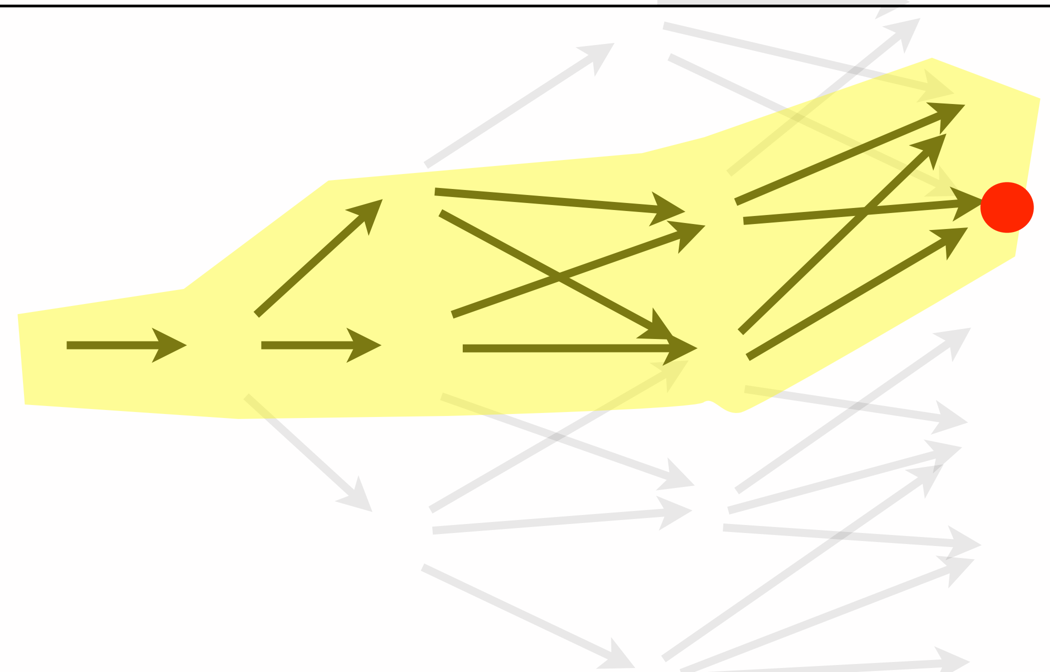


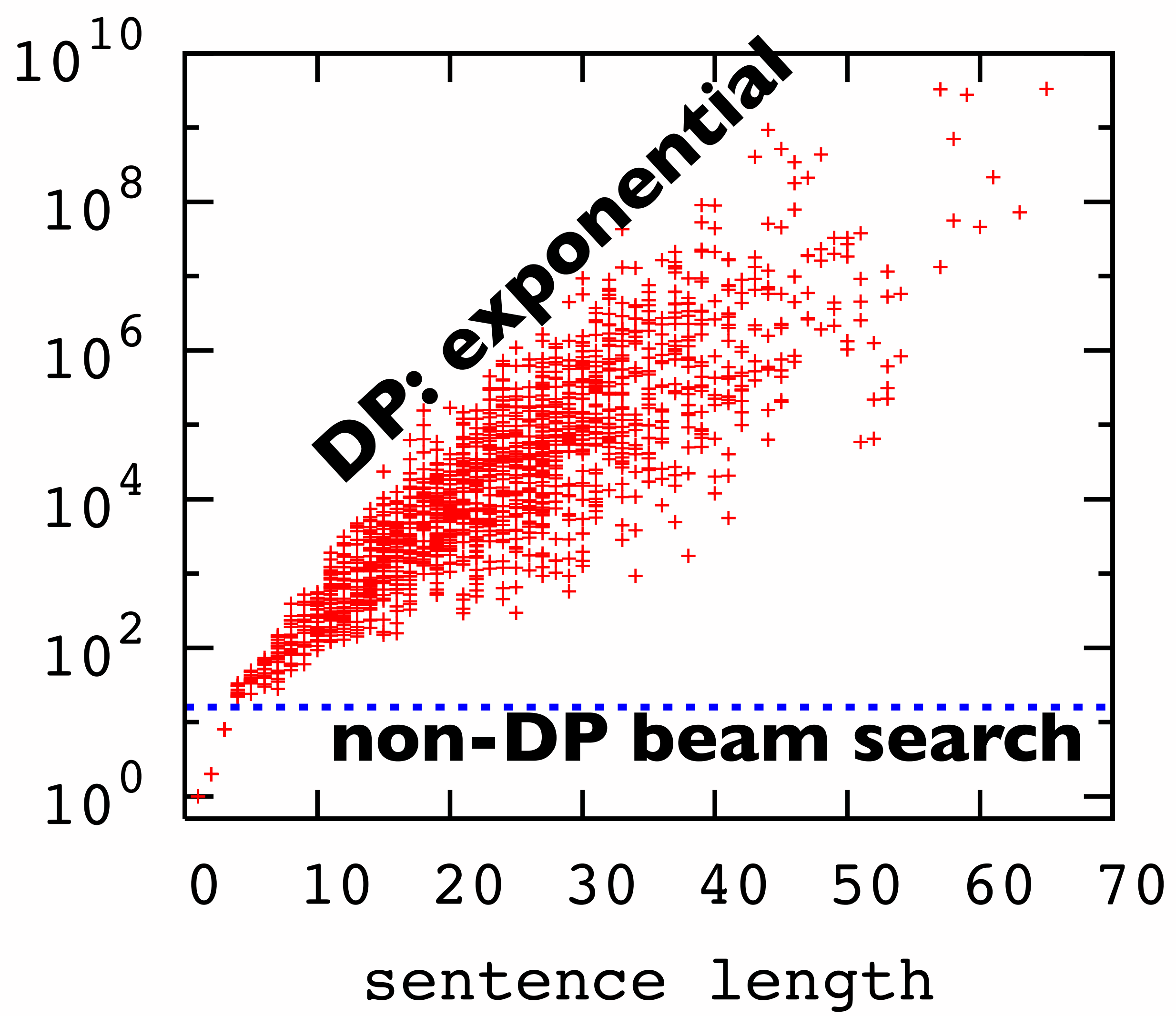
(Huang and Sagae, 2010)

Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space

each DP state corresponds to exponentially many non-DP states



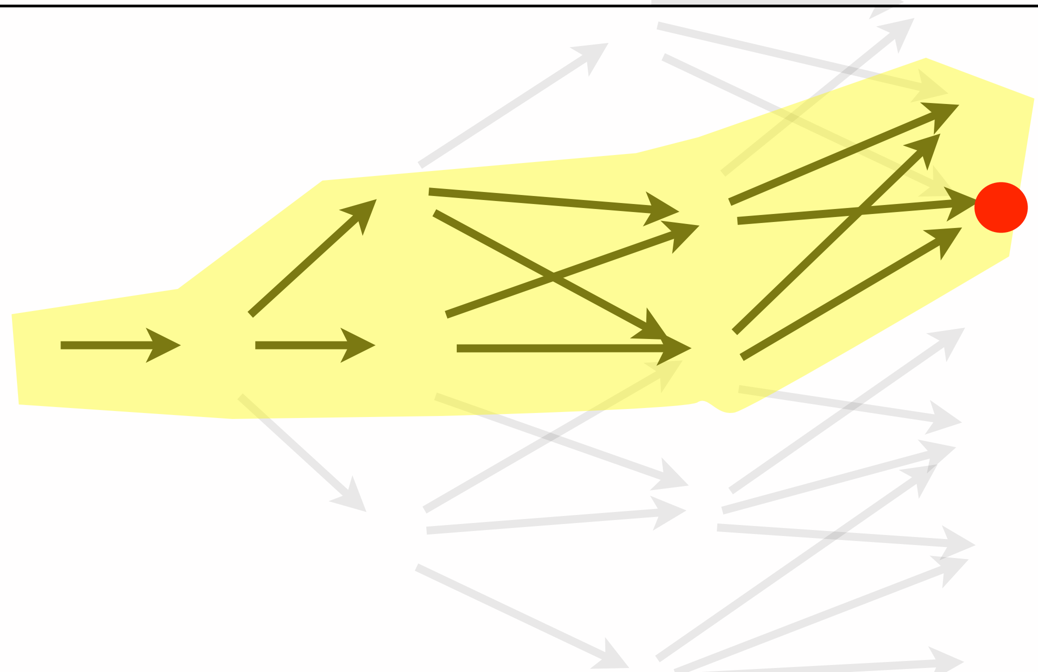


(Huang and Sagae, 2010)

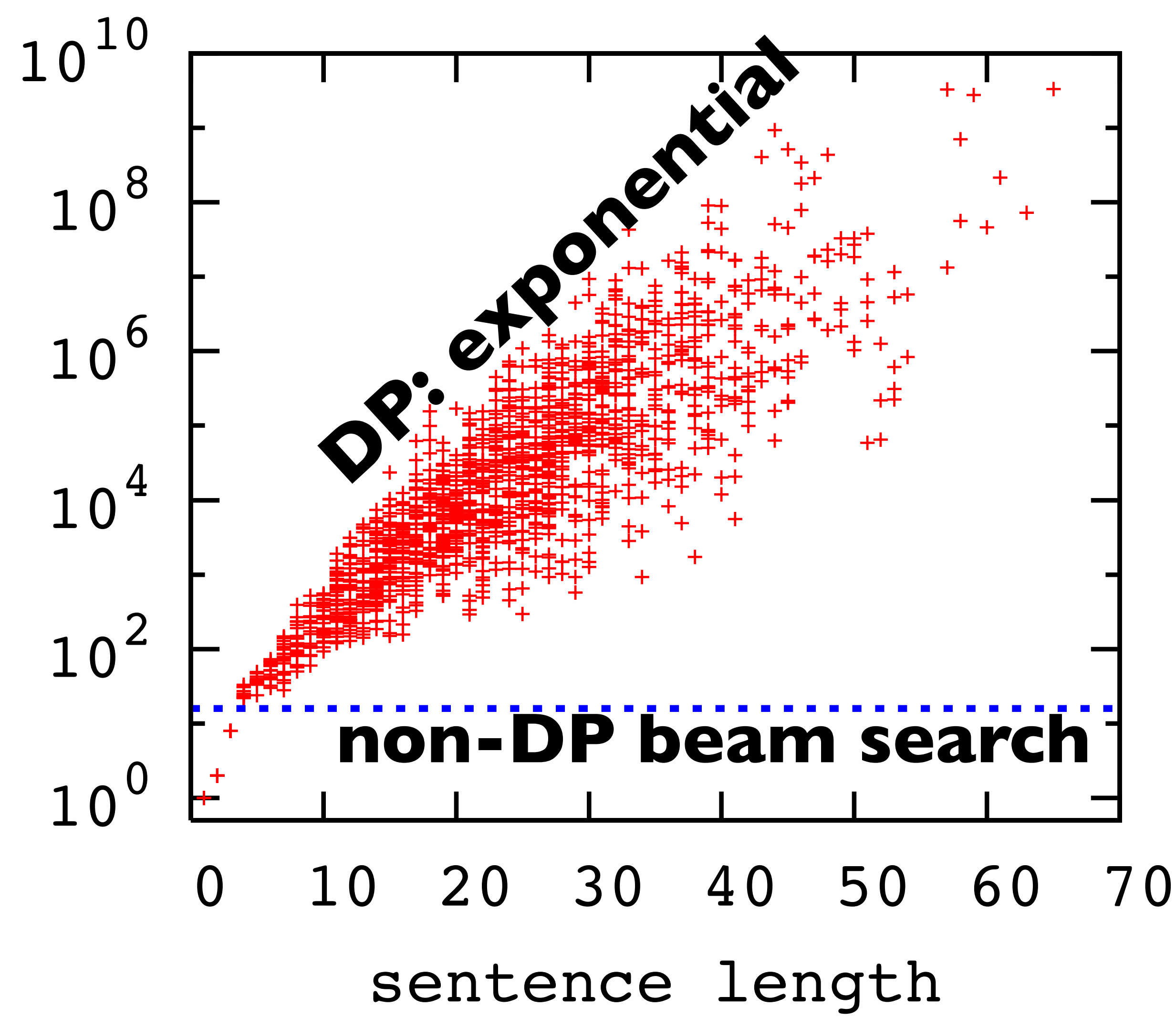
Dynamic Programming

- each state => three new states (shift, l-reduce, r-reduce)
- key idea of DP: share common subproblems
  - merge equivalent states => polynomial space

each DP state corresponds to exponentially many non-DP states



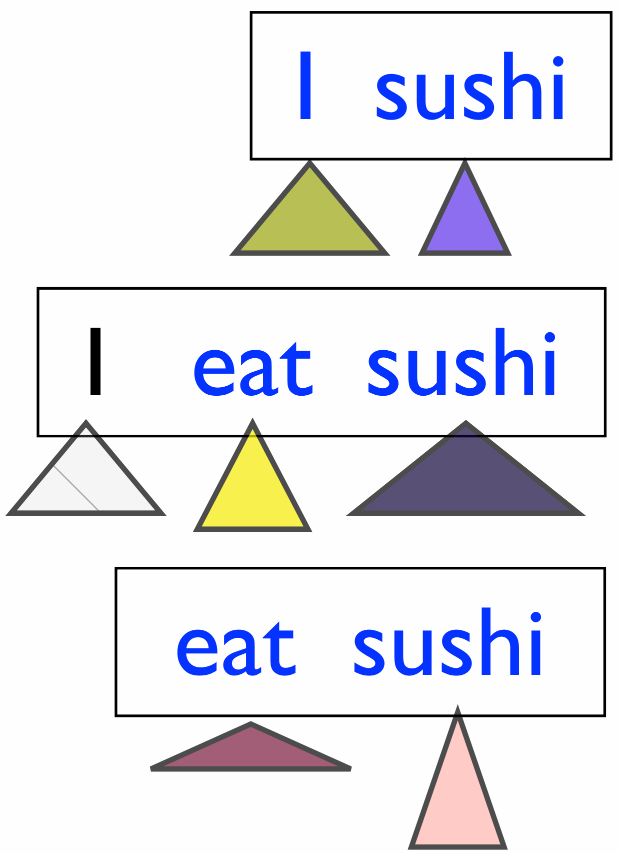
graph-structured stack
(Tomita, 1986)



(Huang and Sagae, 2010)

Merging (Ambiguity Packing)

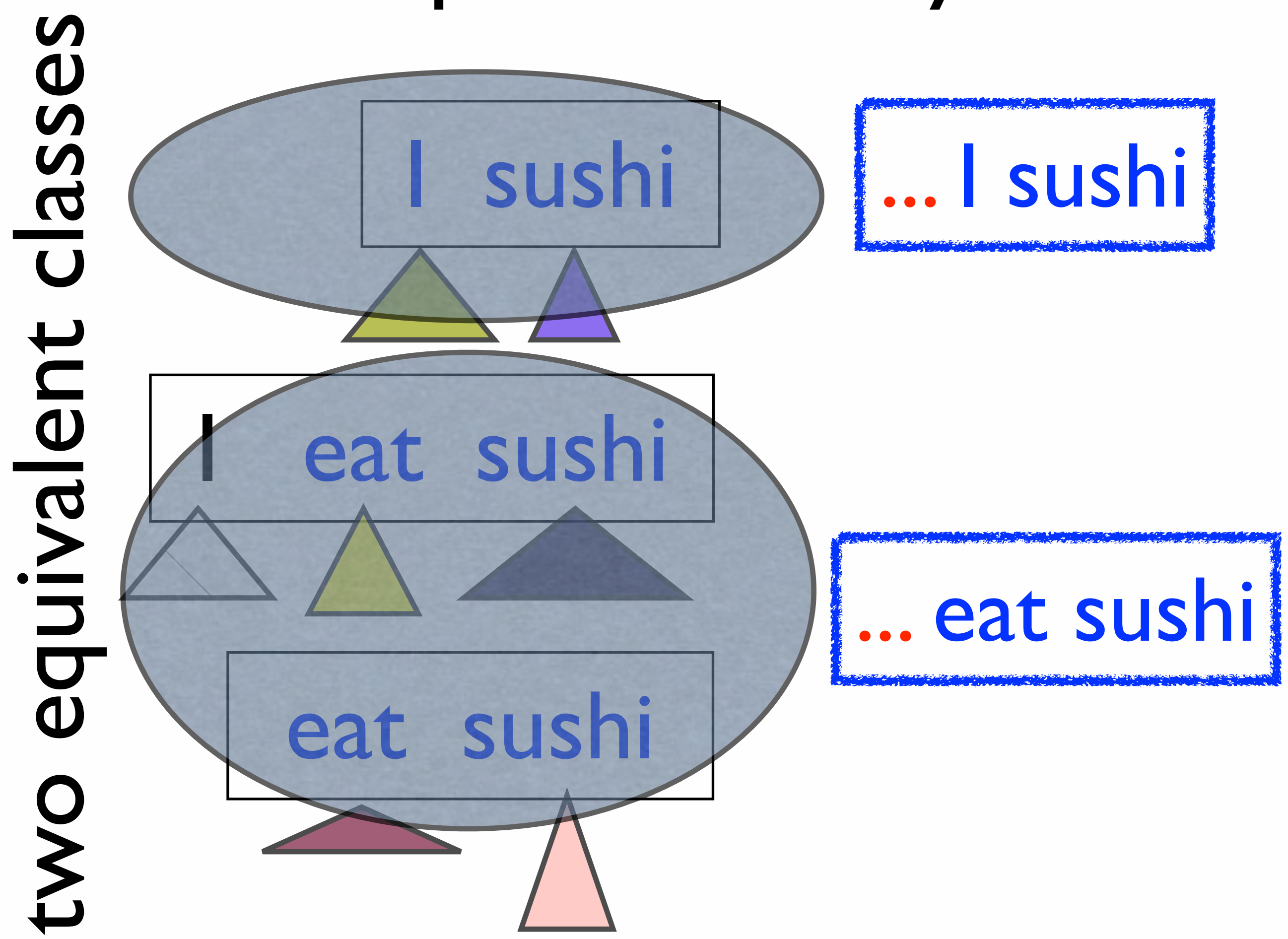
- two states are equivalent if they agree on features
  - because same features guarantee same cost
  - example: if we only care about the last 2 words on stack



(Huang and Sagae, 2010)

Merging (Ambiguity Packing)

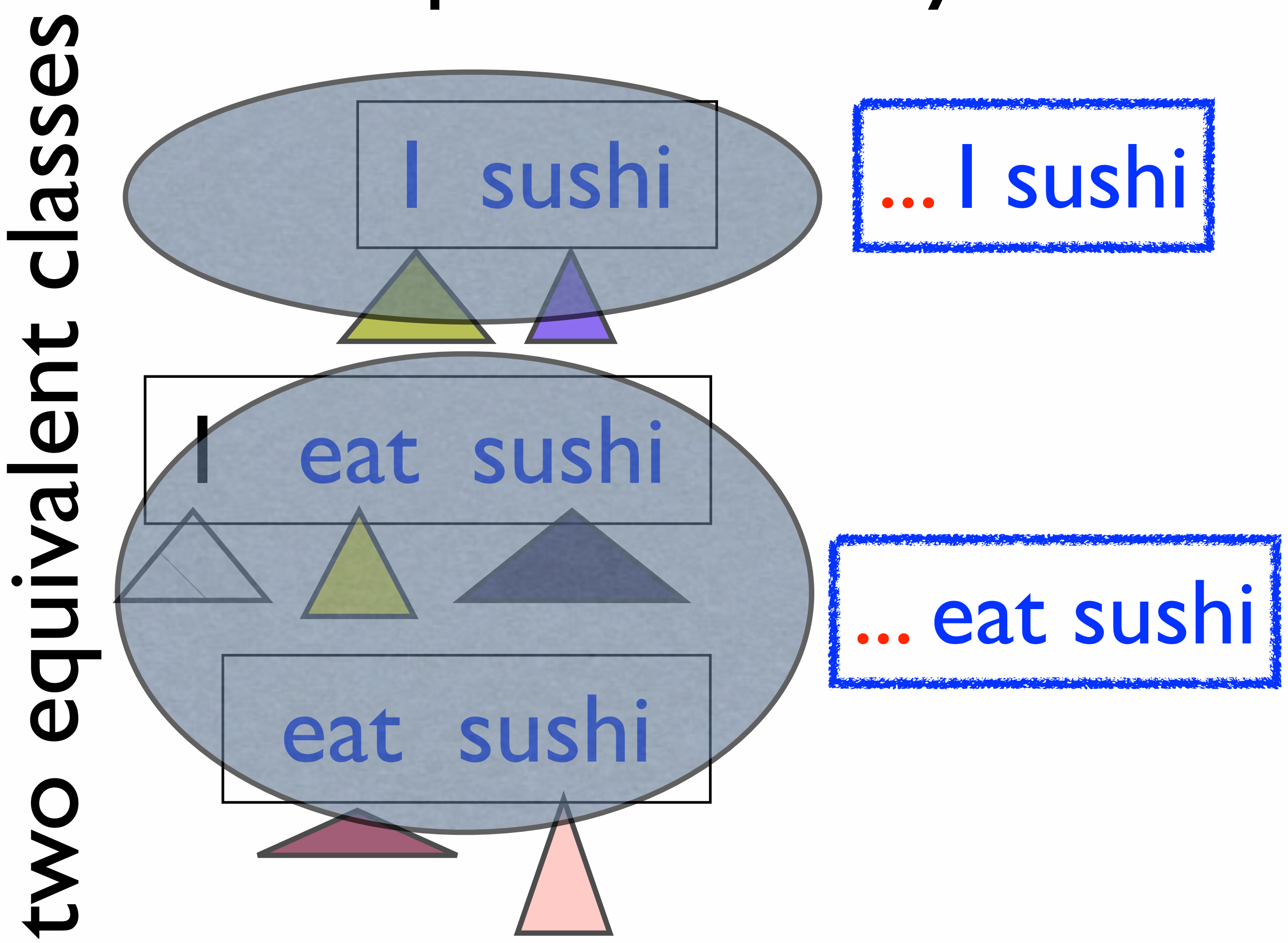
- two states are equivalent if they agree on features
  - because same features guarantee same cost
  - example: if we only care about the last 2 words on stack



(Huang and Sagae, 2010)

Merging (Ambiguity Packing)

- two states are equivalent if they agree on features
  - because same features guarantee same cost
  - example: if we only care about the last 2 words on stack



psycholinguistic evidence (eye-tracking experiments):

delayed disambiguation

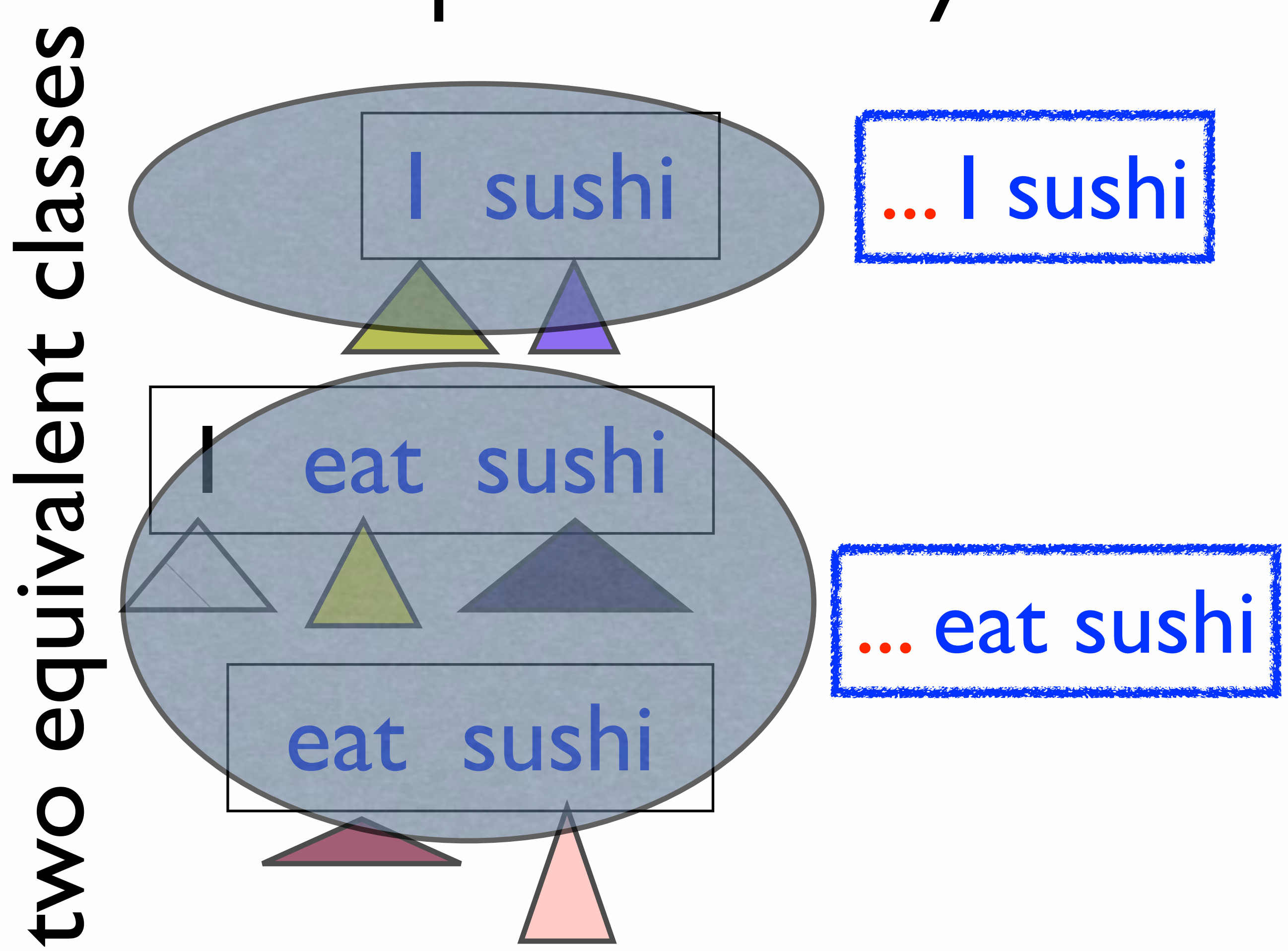
John and Mary had 2 papers
John and Mary had 2 papers

Frazier and Rayner (1990), Frazier (1999)

(Huang and Sagae, 2010)

Merging (Ambiguity Packing)

- two states are equivalent if they agree on features
  - because same features guarantee same cost
  - example: if we only care about the last 2 words on stack



psycholinguistic evidence (eye-tracking experiments):

delayed disambiguation

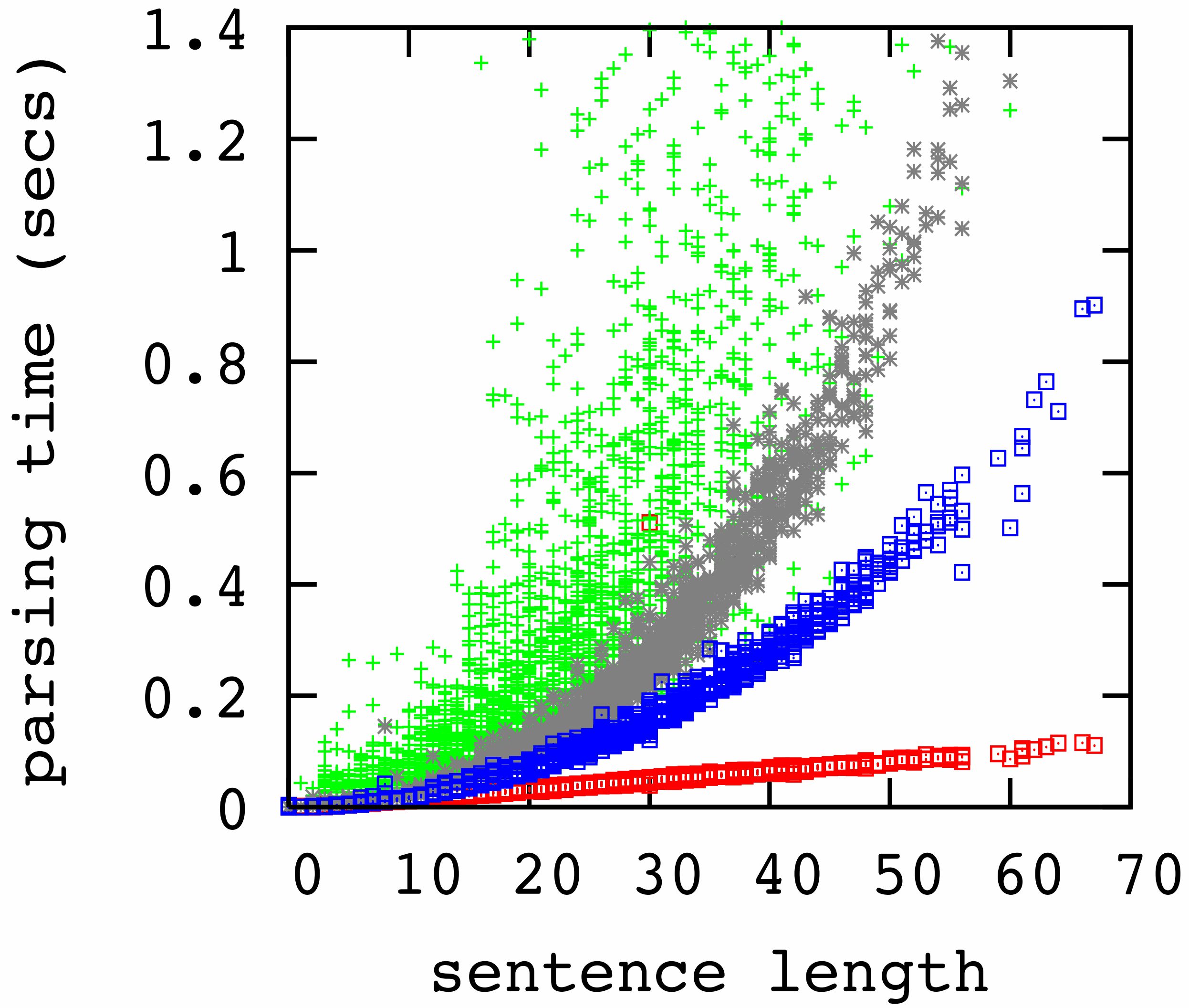
John and Mary had 2 papers each
John and Mary had 2 papers together

Frazier and Rayner (1990), Frazier (1999)

(Huang and Sagae, 2010)

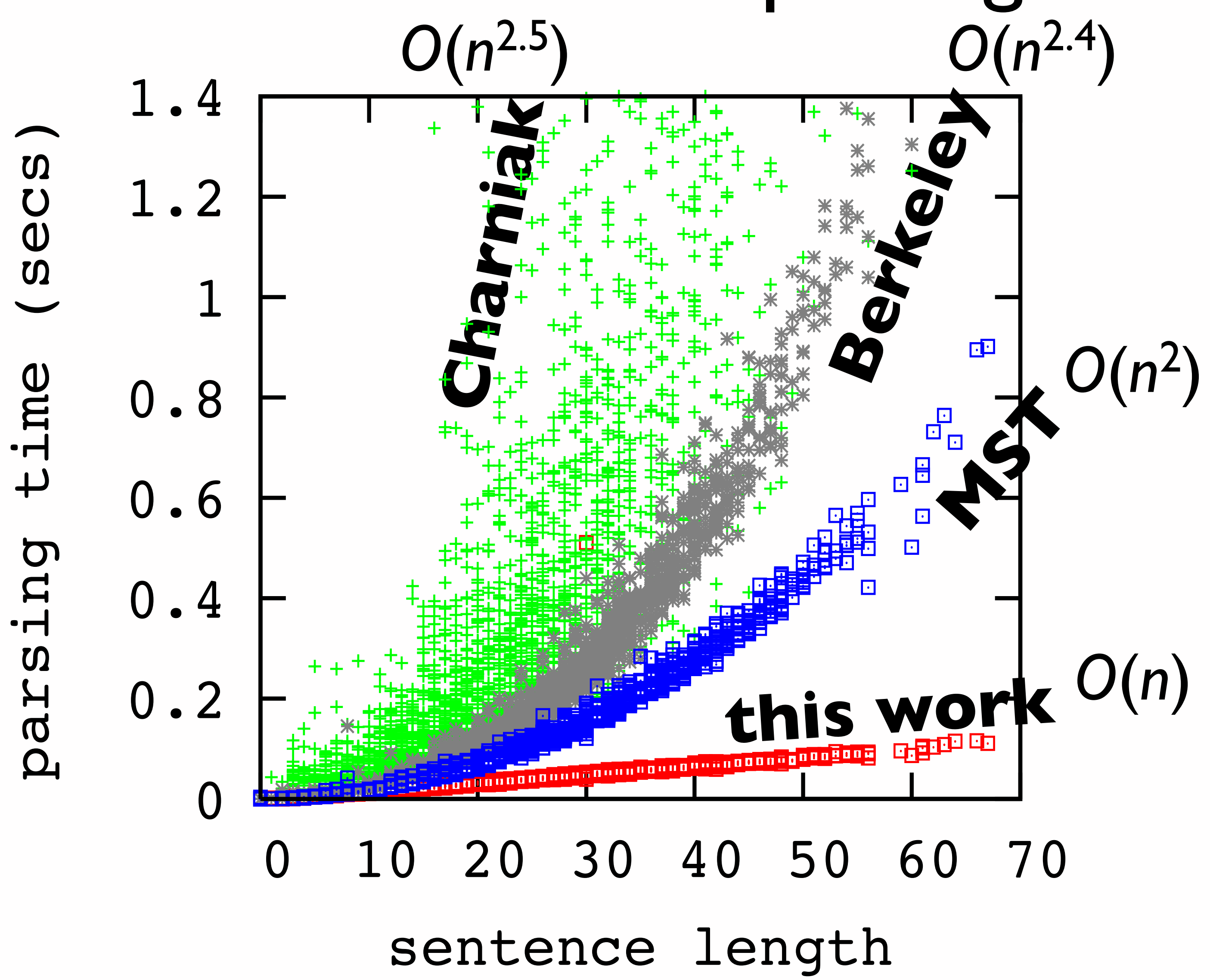
Result: linear-time, DP, and accurate!

- very fast linear-time dynamic programming parser
- explores *exponentially* many trees (and outputs forest)
- state-of-the-art parsing accuracy on English & Chinese



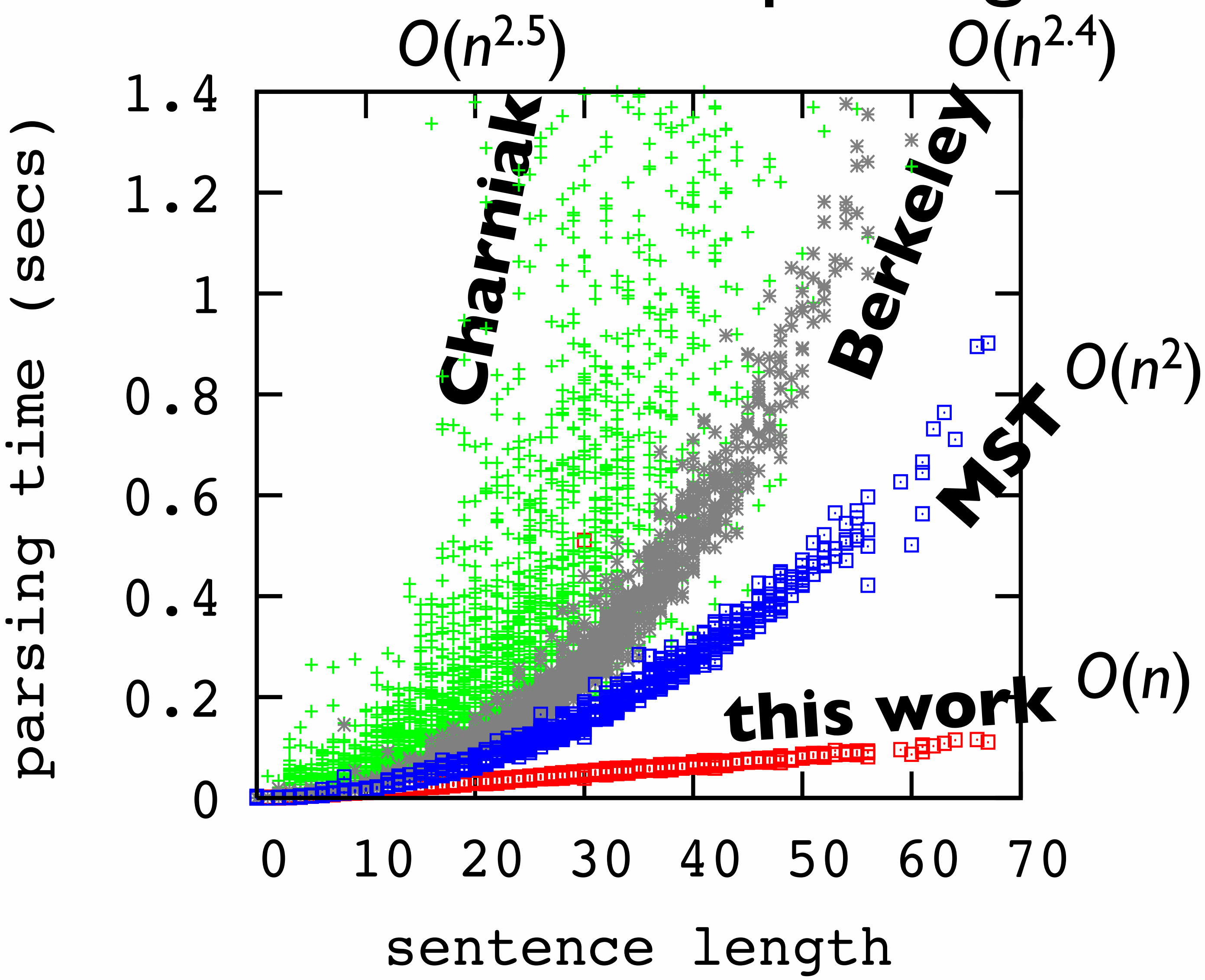
Result: linear-time, DP, and accurate!

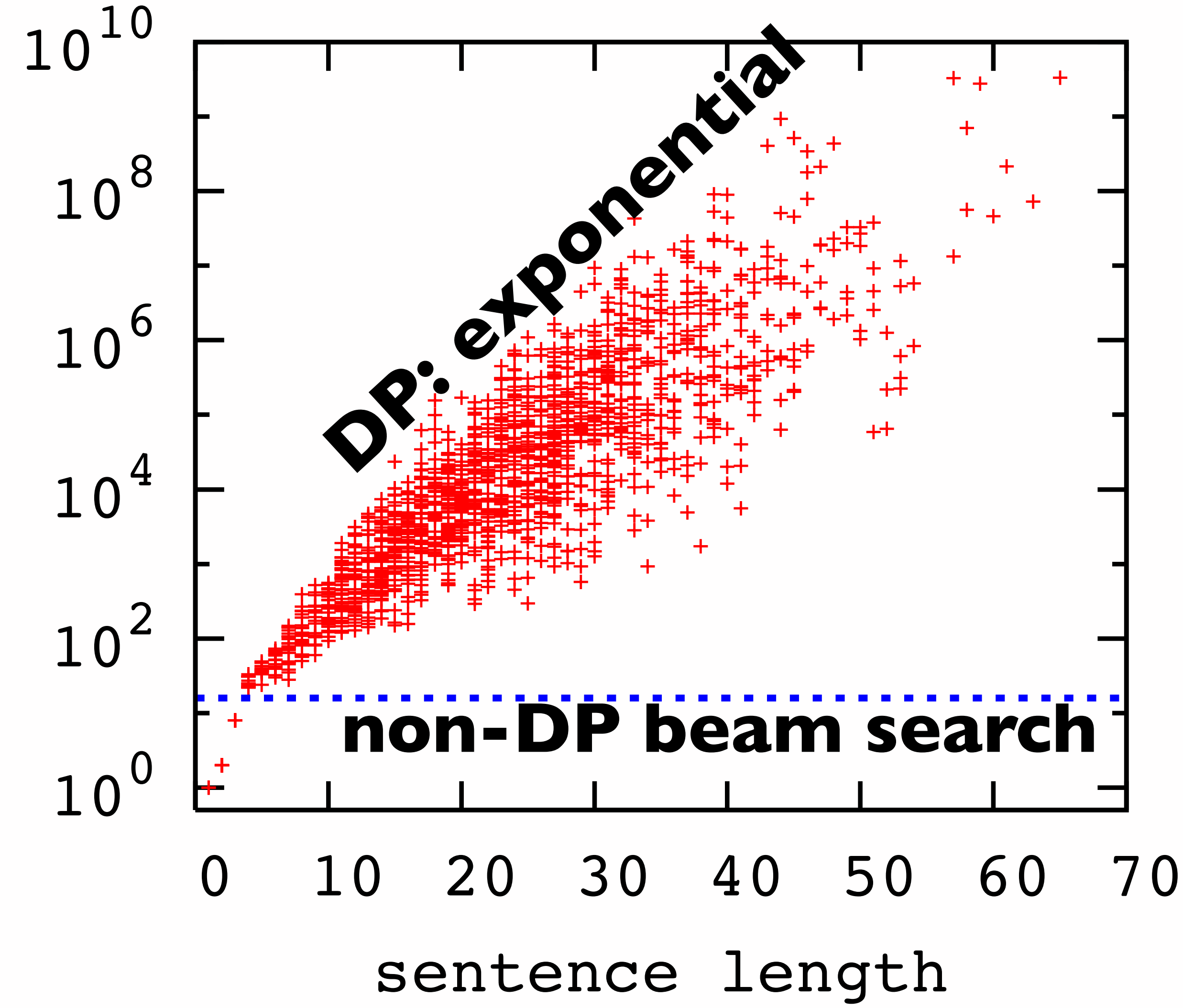
- very fast linear-time dynamic programming parser
- explores *exponentially* many trees (and outputs forest)
- state-of-the-art parsing accuracy on English & Chinese



Result: linear-time, DP, and accurate!

- very fast linear-time dynamic programming parser
- explores *exponentially* many trees (and outputs forest)
- state-of-the-art parsing accuracy on English & Chinese

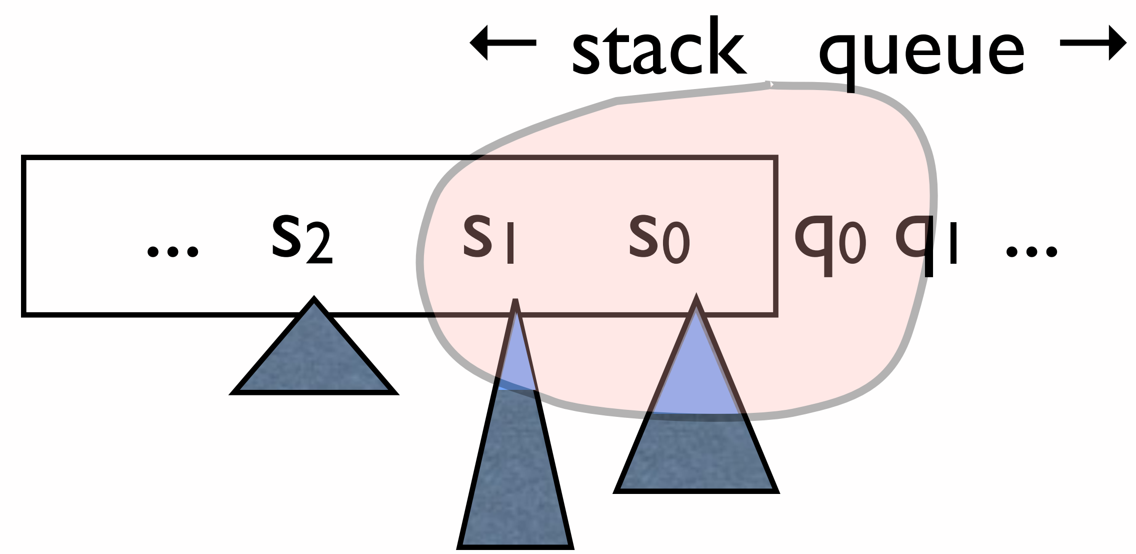




In this talk...

- Background
- Dynamic Programming for Incremental Parsing
- Features: from sparse to neural to recurrent neural nets
- Bidirectional RNNs: minimal features; no tree structures!
  - dependency parsing (Kiperwaser+Goldberg, 2016, Cross+Huang, 2016a)
  - span-based constituency parsing (Cross+Huang, 2016b)
- Marrying DP & RNNs *(mostly not my work!)*
  - minimal span-based constituency parsing (Stern et al, ACL 2017)
  - transition-based dependency parsing (Shi et al, EMNLP 2017)

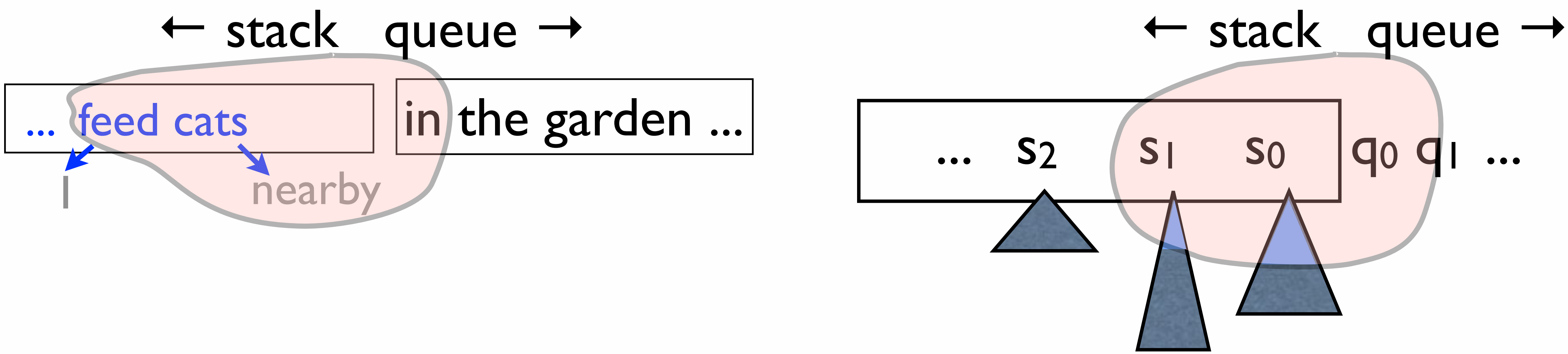
Sparse Features



- score each action using features **f** and weights **w**
  - features are drawn from a *local window*
    - abstraction (or signature) of a state -- this inspires DP!
  - weights trained by structured perceptron (Collins 02)

(Huang+Sagae, 2010)

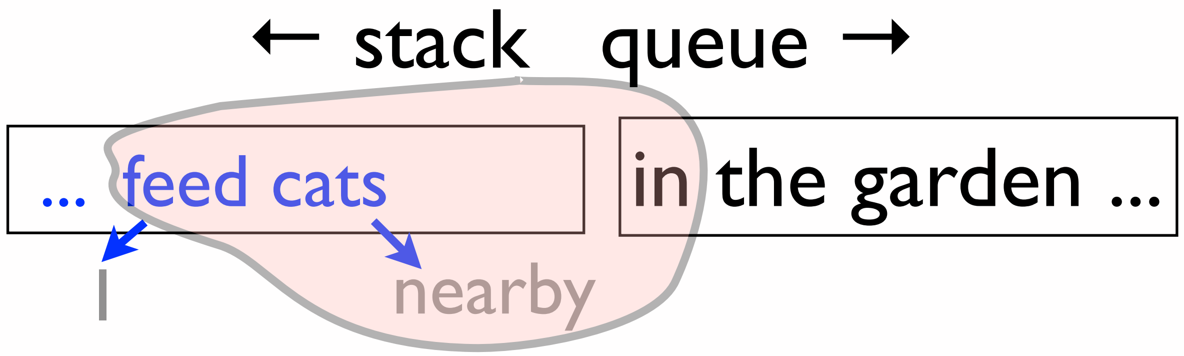
Sparse Features

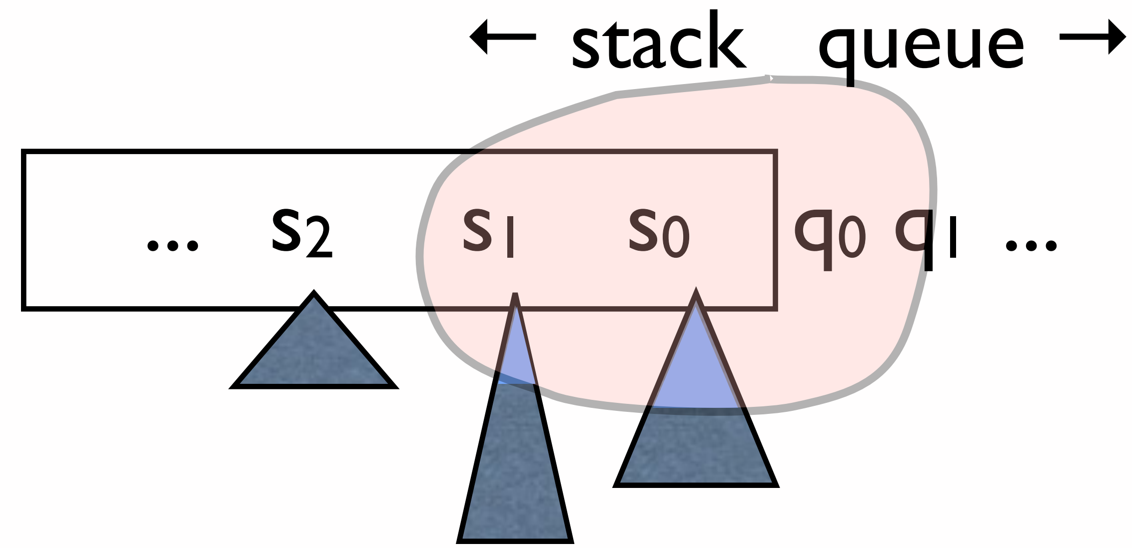


- score each action using features **f** and weights **w**
  - features are drawn from a *local window*
    - abstraction (or signature) of a state -- this inspires DP!
  - weights trained by structured perceptron (Collins 02)

(Huang+Sagae, 2010)

Sparse Features





features:

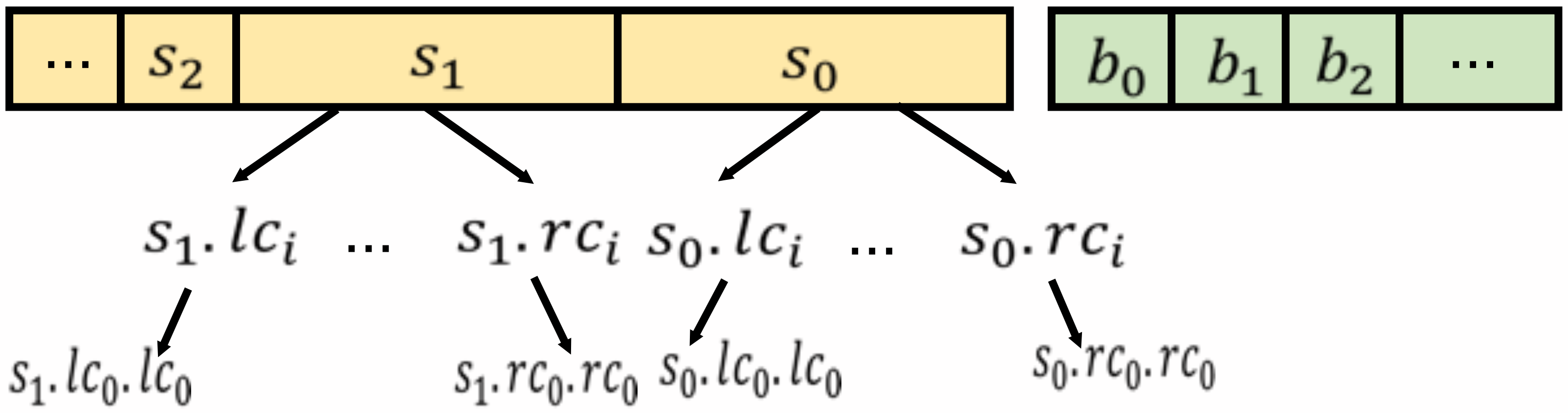
($s_0.w$, $s_0.rc$, q_0, ...) = (cats, nearby, in, ...)

- score each action using features **f** and weights **w**
  - features are drawn from a *local window*
    - abstraction (or signature) of a state -- this inspires DP!
  - weights trained by structured perceptron (Collins 02)

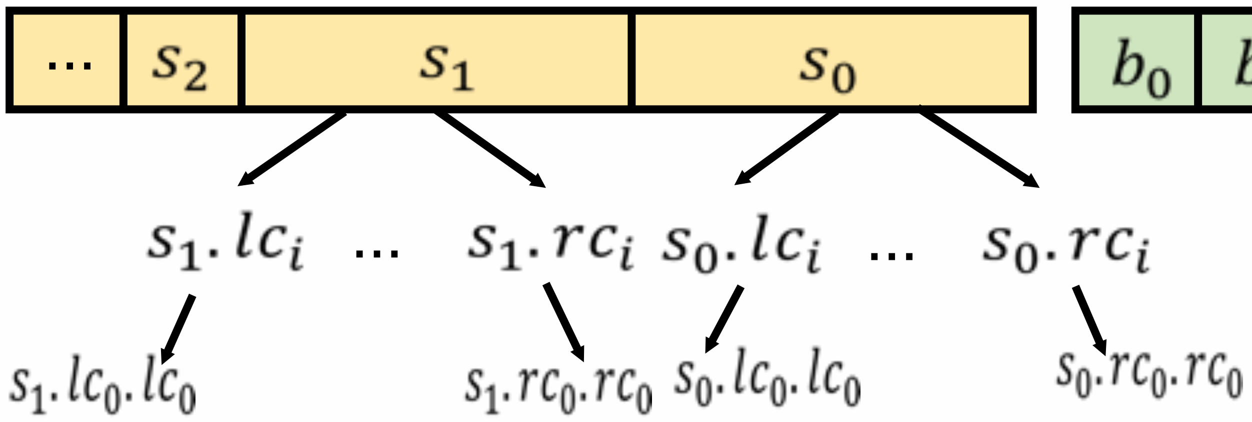
(Huang+Sagae, 2010)

From Sparse to Neural to RNN

(Chen+Manning 2014)



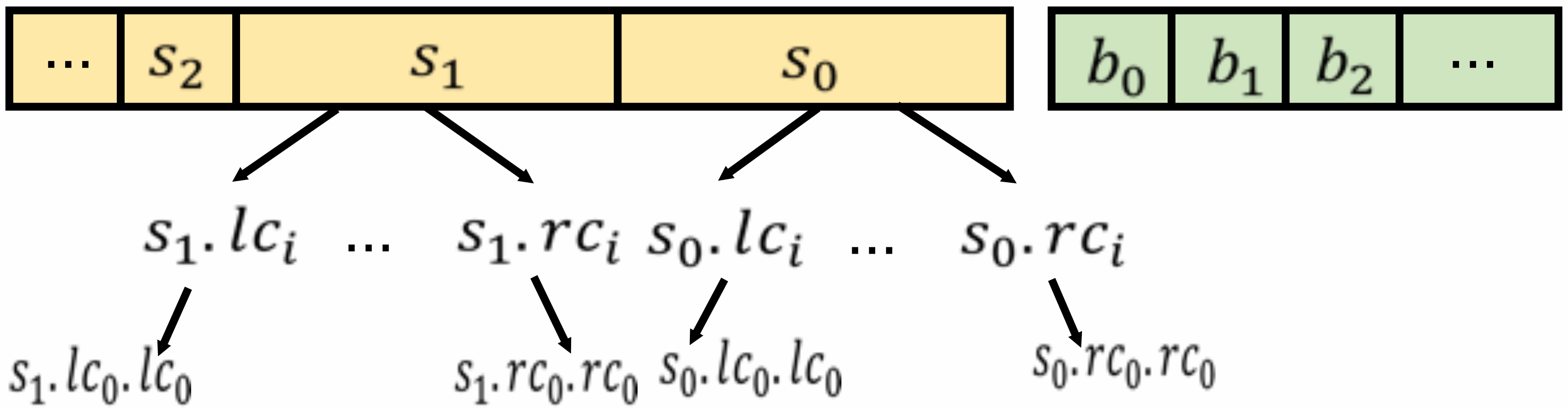
From Sparse to Neural to RNN



(Chen+Manning 2014)

- neural nets can automate feature engineering :-)
- but early neural work (e.g., Chen+Manning 14) still use lots of manually designed atomic features on the stack

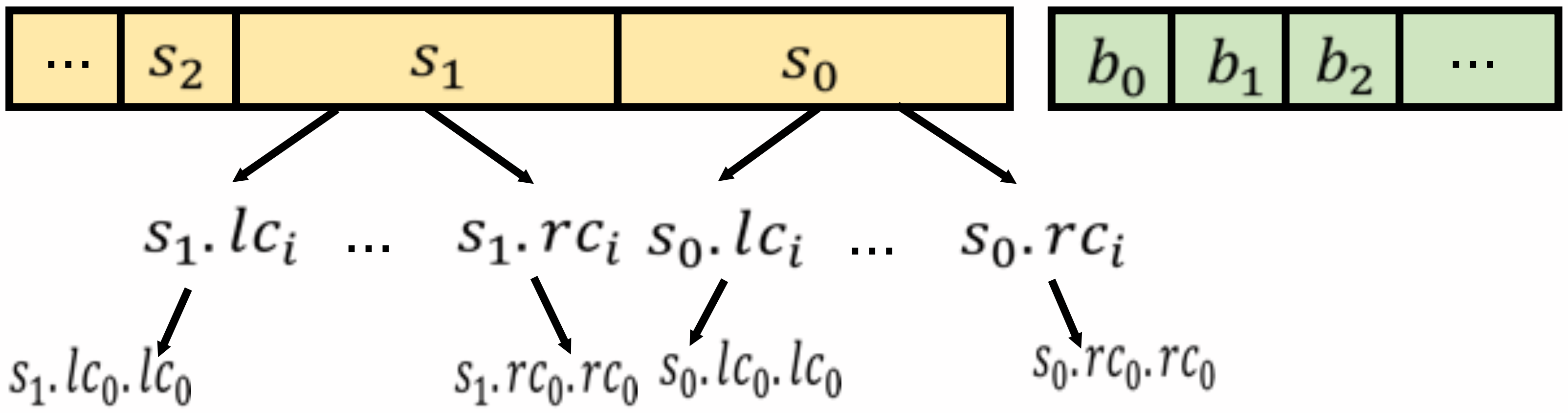
From Sparse to Neural to RNN



(Chen+Manning 2014)

- neural nets can automate feature engineering :-)
- but early neural work (e.g., Chen+Manning 14) still use lots of manually designed atomic features on the stack
- can we automate even more?
  - option 1: summarize the whole stack (part of **y**) using RNNs => stack LSTM / RNNG (Dyer+ 15, 16)
  - option 2: summarize the whole input (**x**) using RNNs => biLSTM dependency parsing (Kiperwaser+Goldberg 16, Cross+Huang 16a) biLSTM constituency parsing (Cross+Huang 16b)

From Sparse to Neural to RNN



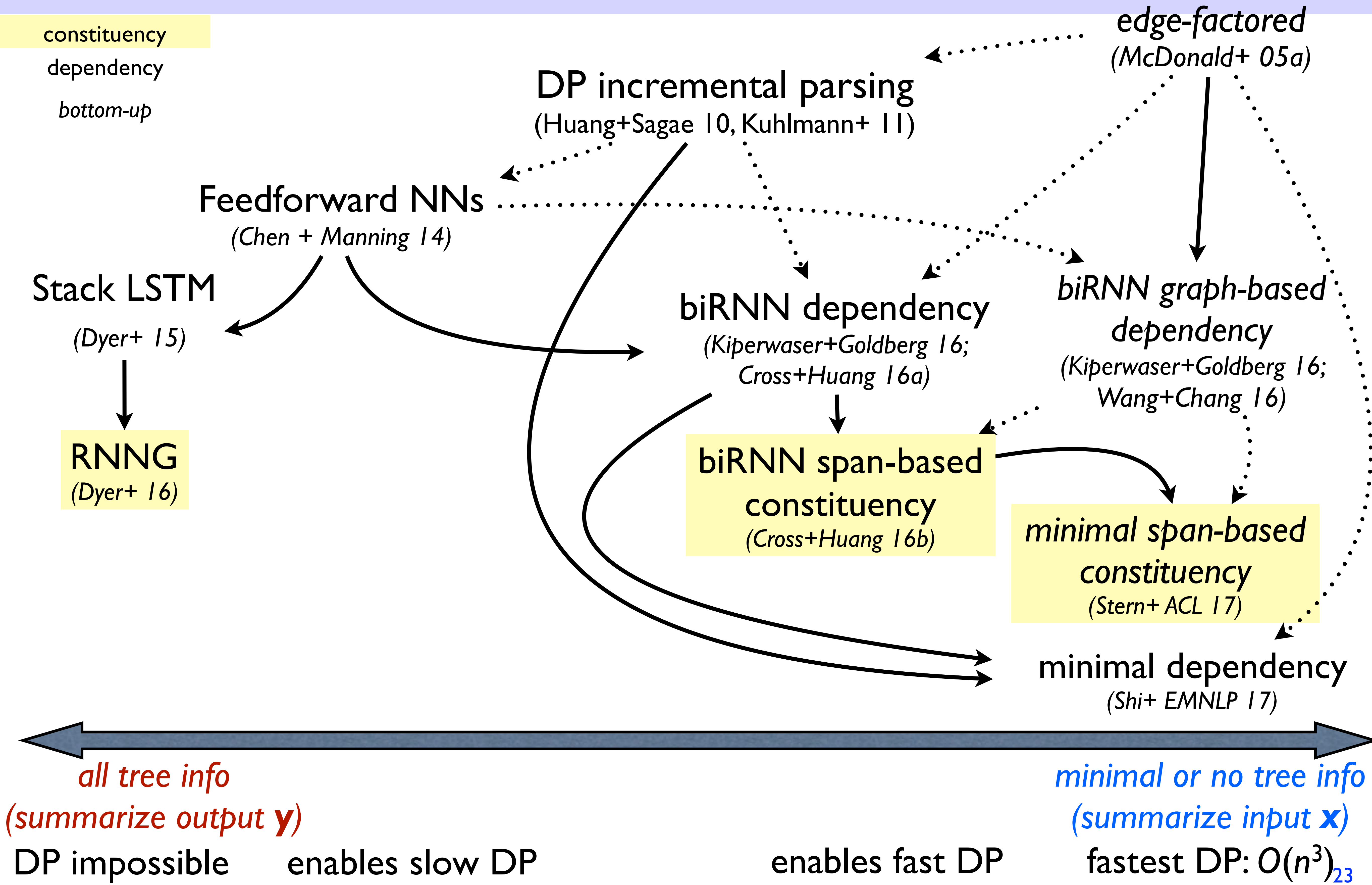
(Chen+Manning 2014)

- neural nets can automate feature engineering :-)
- but early neural work (e.g., Chen+Manning 14) still use lots of manually designed atomic features on the stack
- can we automate even more?
  - option 1: summarize the whole stack (part of **y**) using RNNs => stack LSTM / RNNG (Dyer+ 15, 16)

    rules out DP! :(
  - option 2: summarize the whole input (**x**) using RNNs => biLSTM dependency parsing (Kiperwaser+Goldberg 16, Cross+Huang 16a) biLSTM constituency parsing (Cross+Huang 16b)

    enables DP! :)

Spectrum: Neural Incremental Parsing

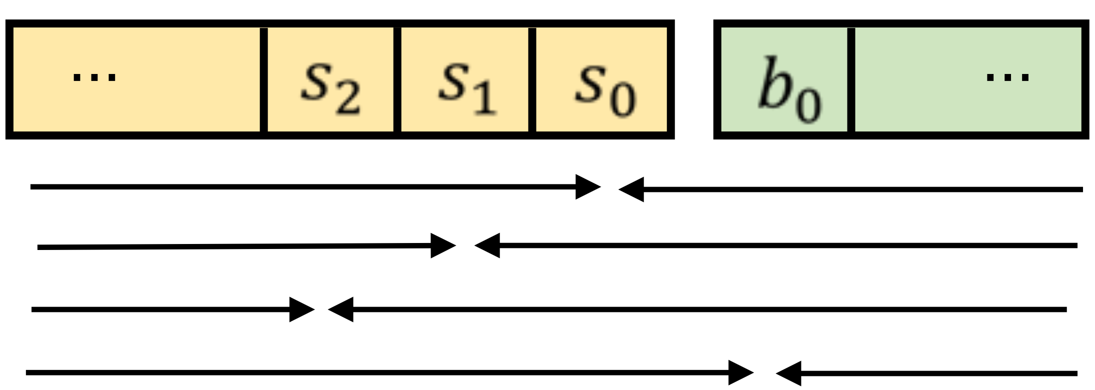


In this talk...

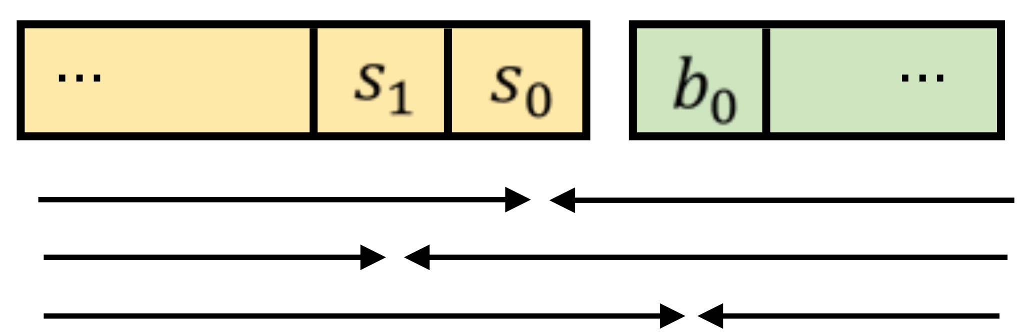
- Background
- Dynamic Programming for Incremental Parsing
- Interlude: NN Features: from feedforward to recurrent
- Bidirectional RNNs: minimal features; no tree structures!
  - dependency parsing (Kiperwaser+Goldberg, 2016, Cross+Huang, 2016a)
  - span-based constituency parsing (Cross+Huang, 2016b)
- Marrying DP & RNNs *(mostly not my work!)*
  - minimal span-based constituency parsing (Stern et al, ACL 2017)
  - transition-based dependency parsing (Shi et al, EMNLP 2017)

biRNN for Dependency Parsing

- several parallel efforts in 2016 used biLSTM features
  - Kiperwaser+Goldberg 2016: four positional feats; arc-eager
  - Cross+Huang ACL 2016: three positional feats; arc-standard
  - Wang+Chang 2016: two positional feats; graph-based
- all inspired by sparse edge-factored model (McDonald+05)
  - use positions to summarize the input **x**, not the output **y**!
  - => $O(n^3)$ DP, e.g. graph-based, but also incremental!



(Kiperwaser and Goldberg 2016)

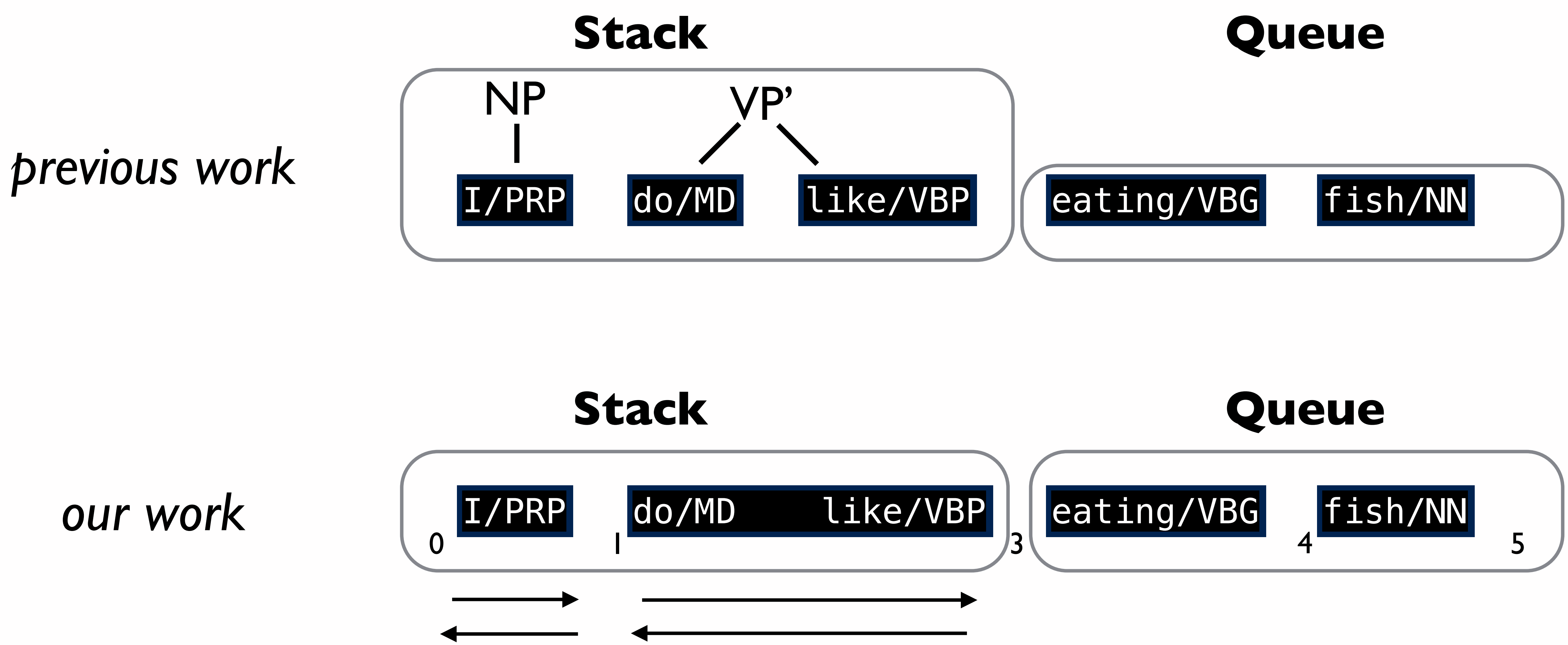


(Cross and Huang, ACL 2016)

these developments lead to state-of-the-art in dependency parsing

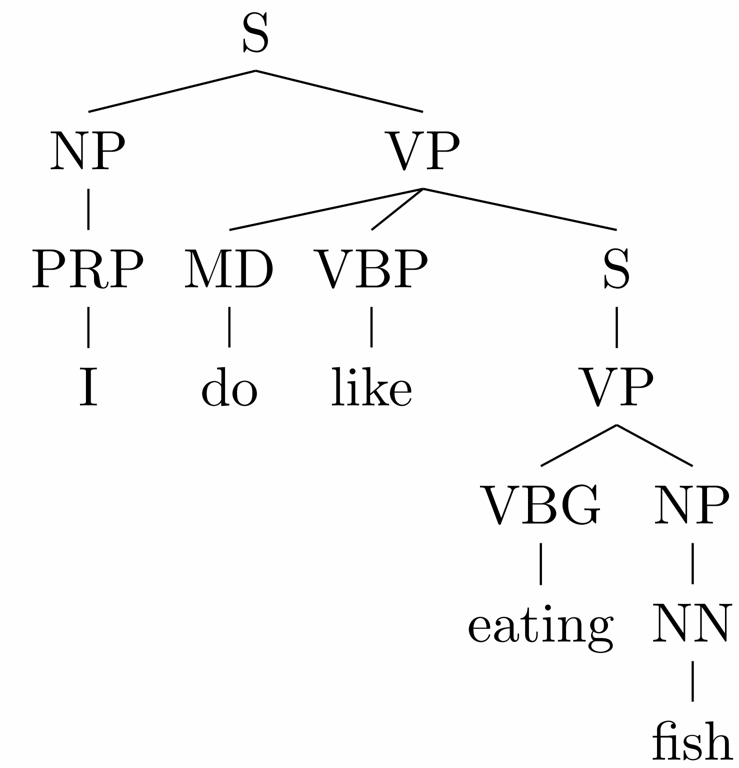
Span-Based Constituency Parsing

- previous work uses tree structures on stack
- we simplify to operate directly on sentence spans
- simple-to-implement linear-time parsing



(Cross and Huang, EMNLP 2016)

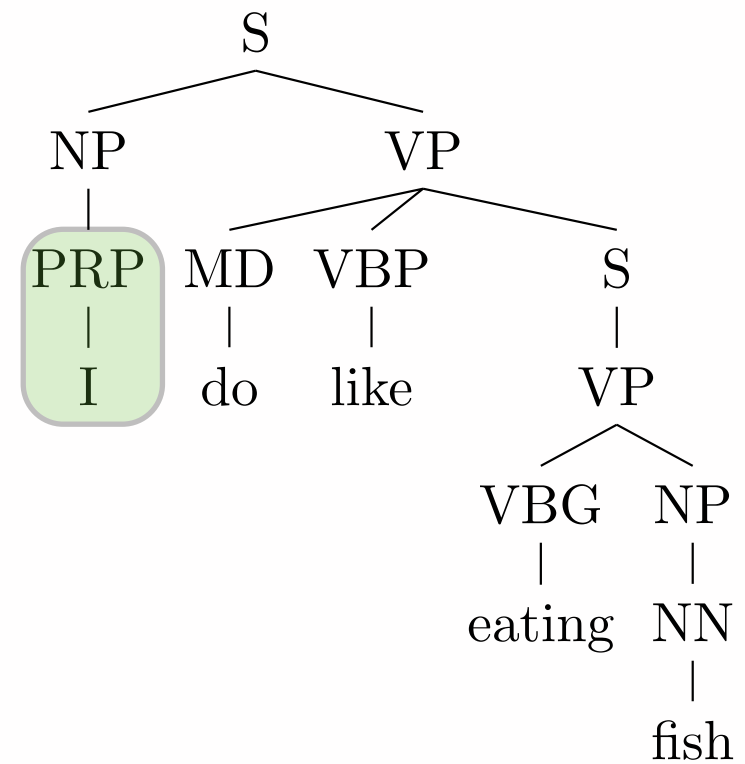
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

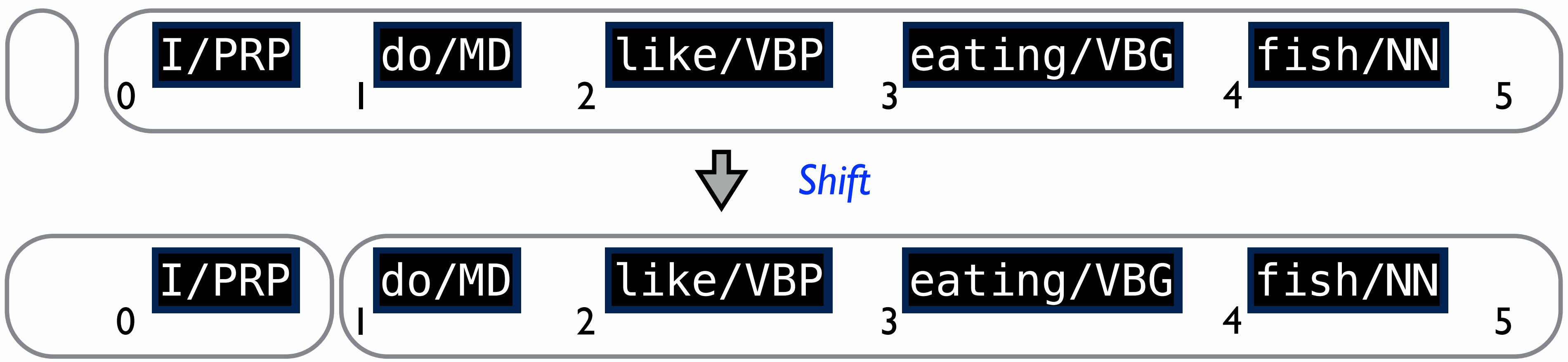


current brackets t = {}

(Cross and Huang, EMNLP 2016)

Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

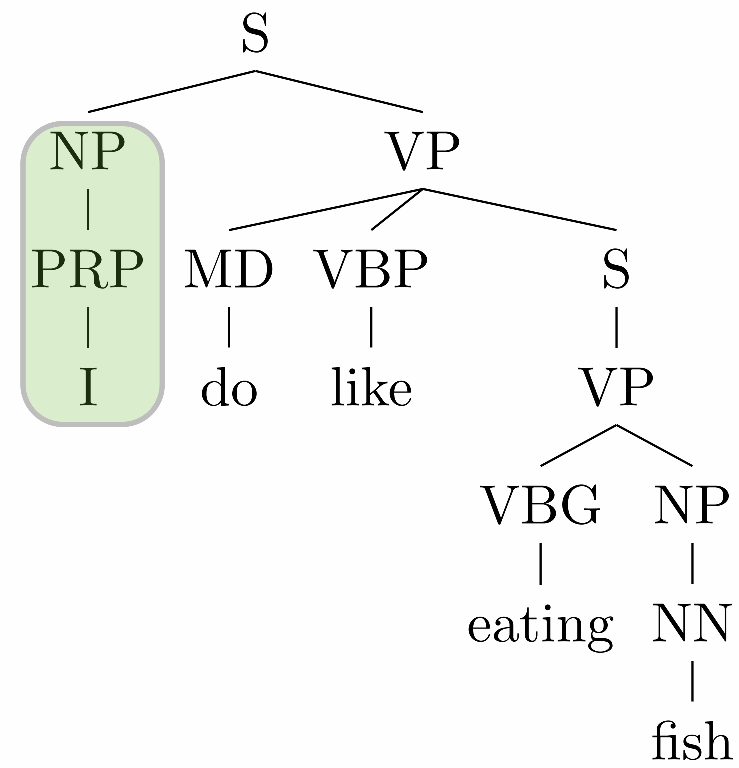


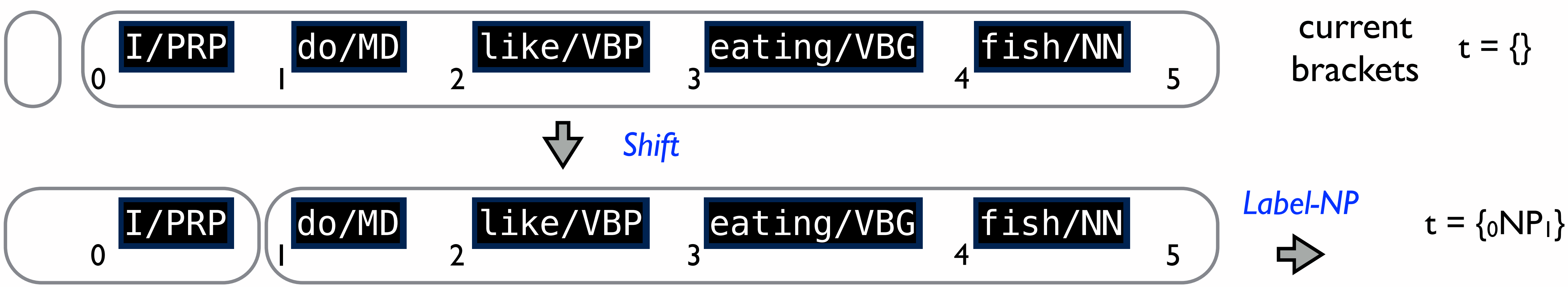


current brackets t = {}

(Cross and Huang, EMNLP 2016)

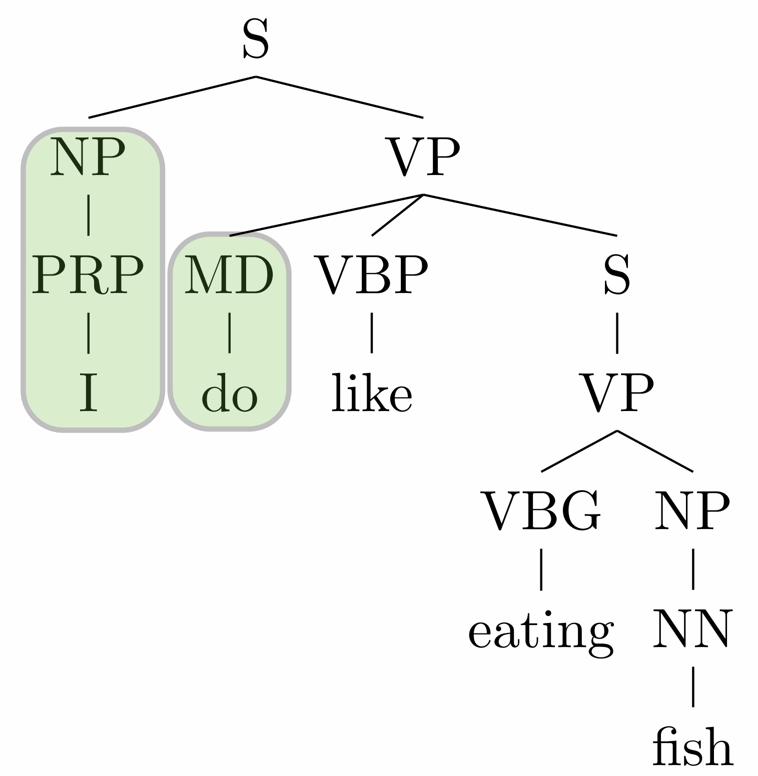
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

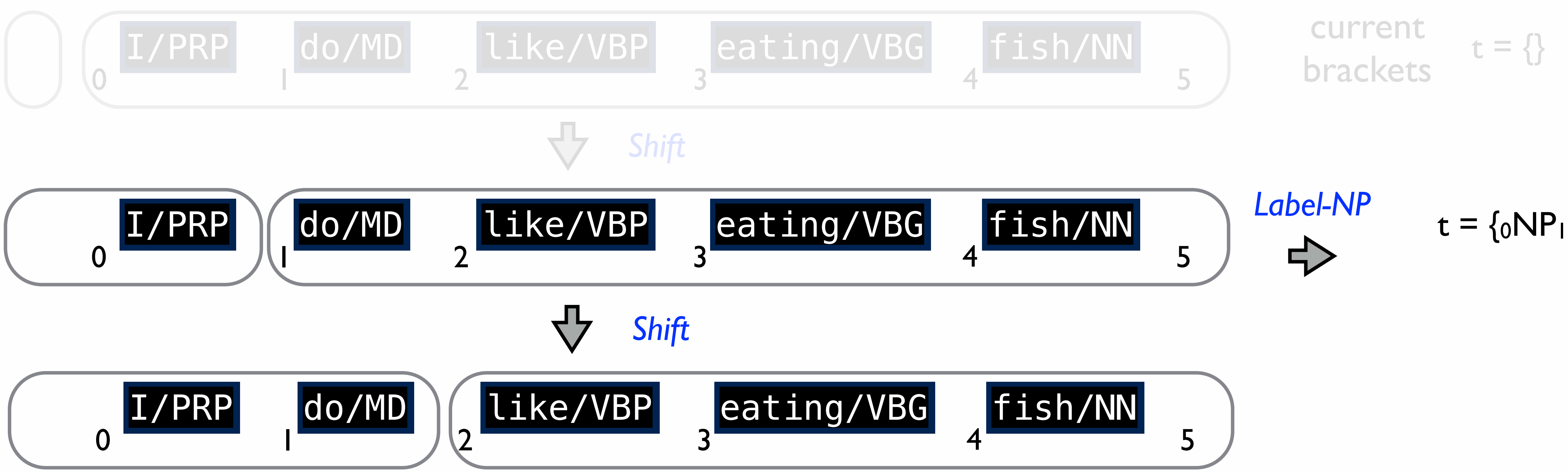




(Cross and Huang, EMNLP 2016)

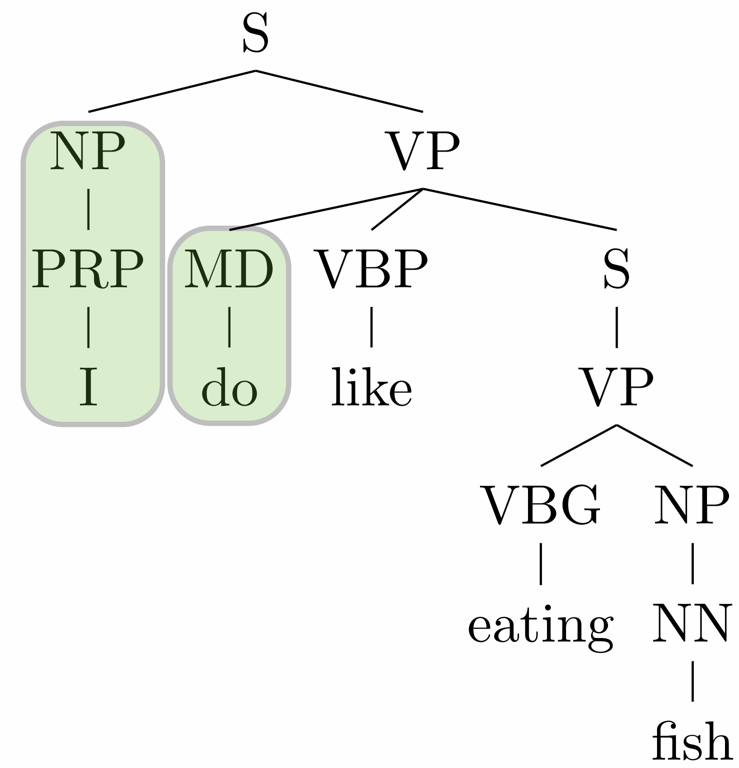
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

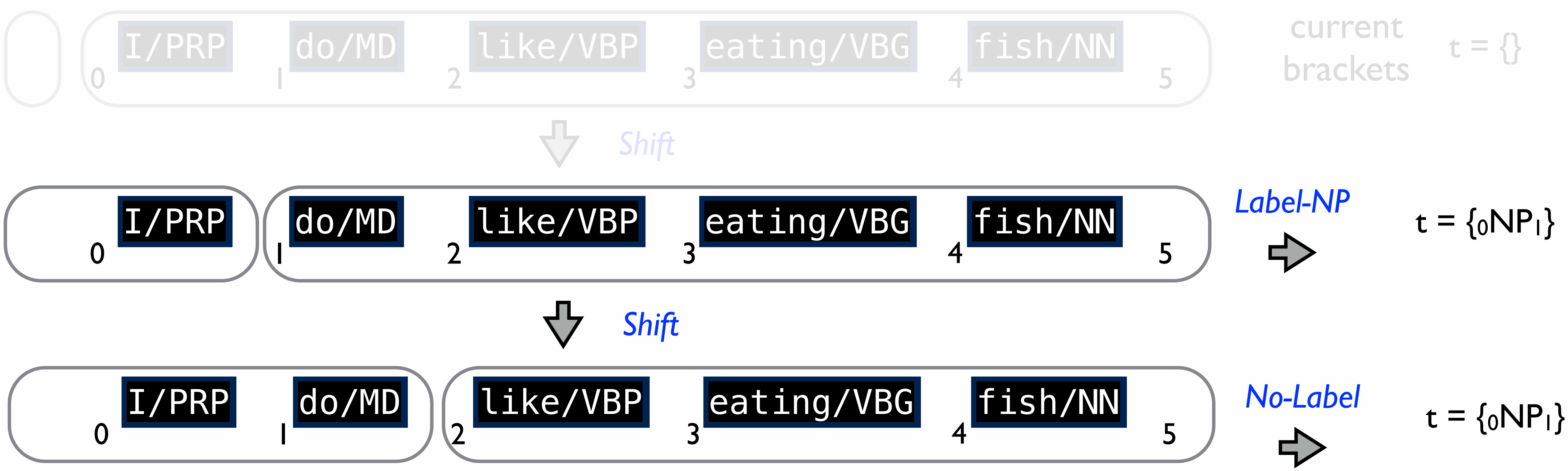




(Cross and Huang, EMNLP 2016)

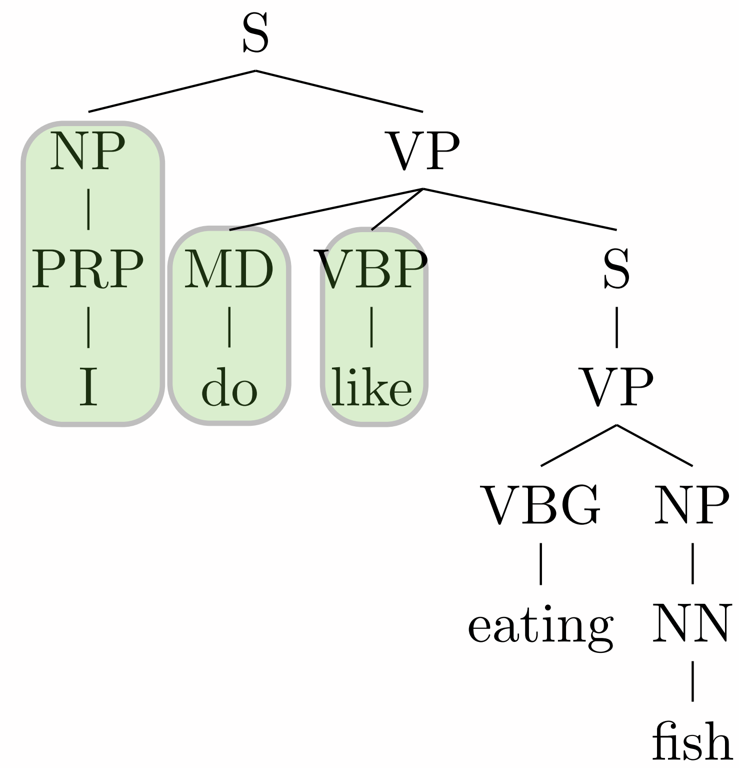
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

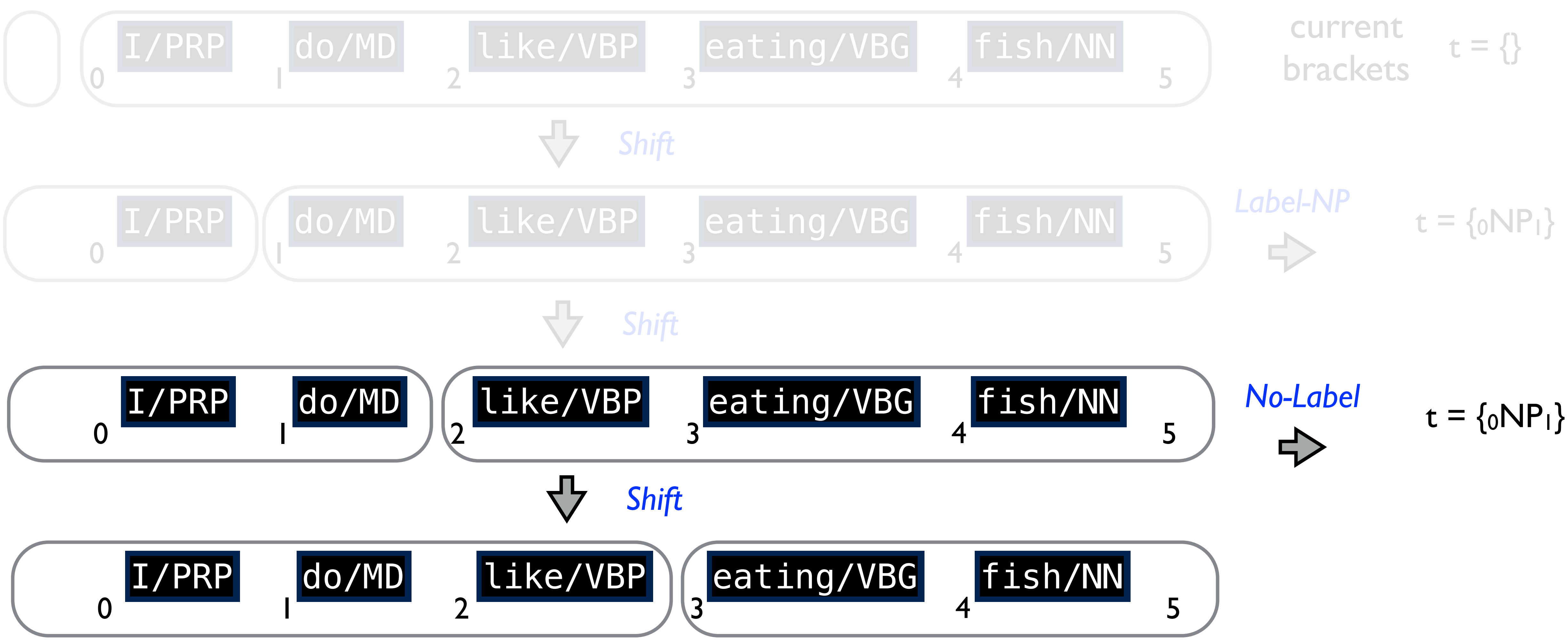




(Cross and Huang, EMNLP 2016)

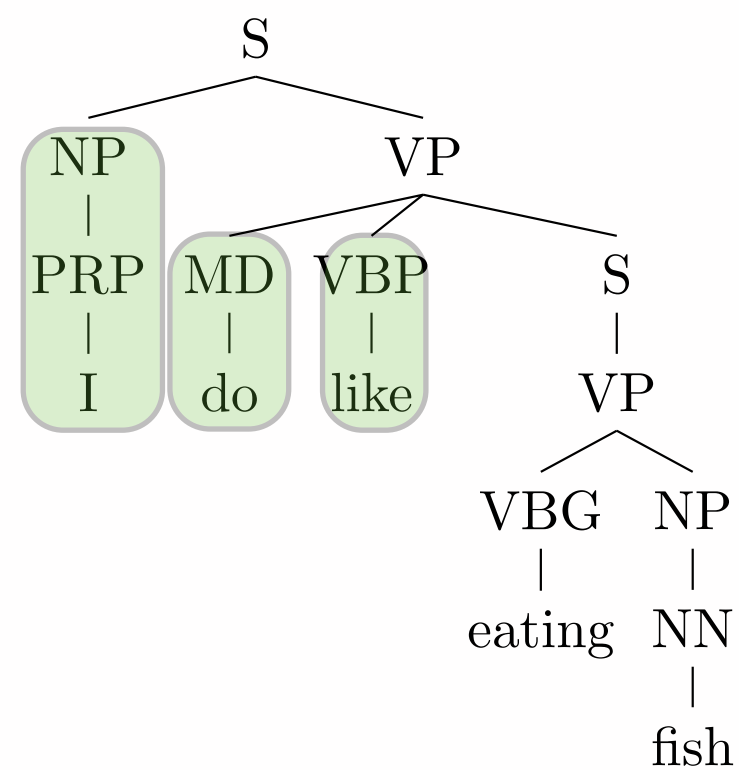
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

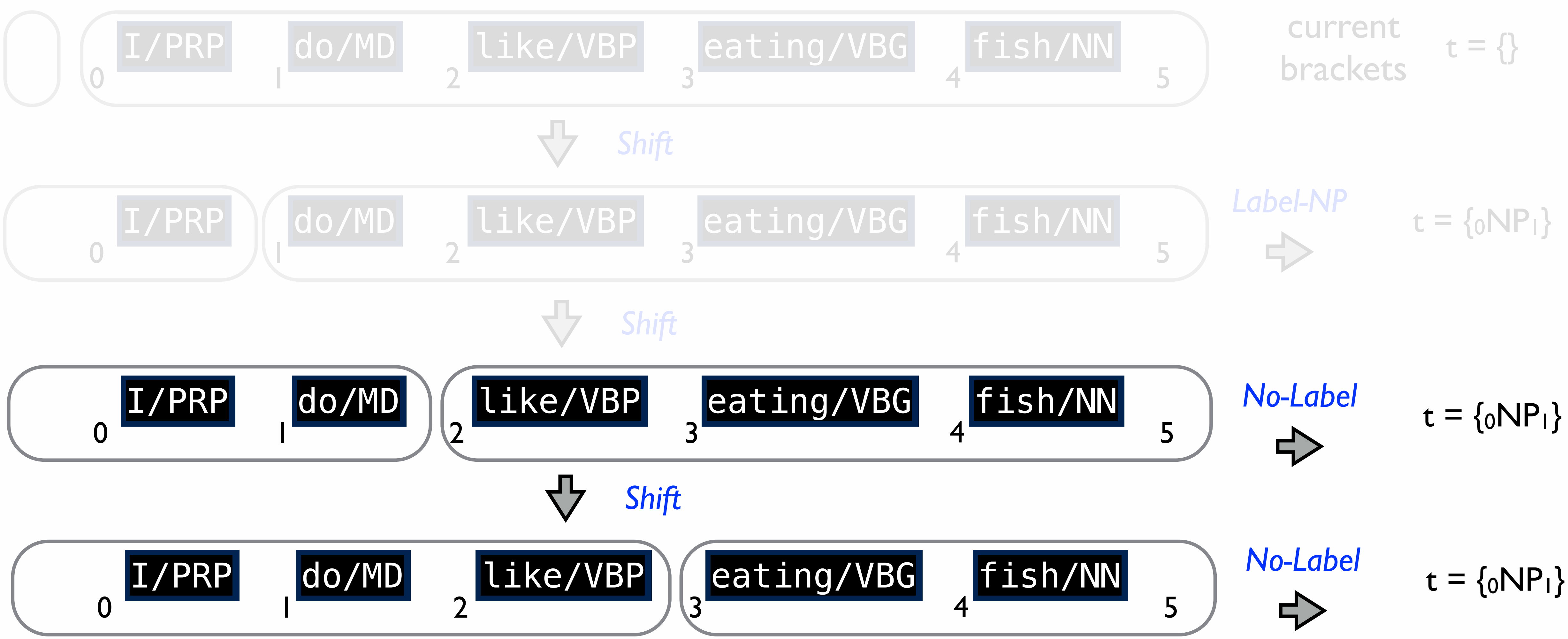




(Cross and Huang, EMNLP 2016)

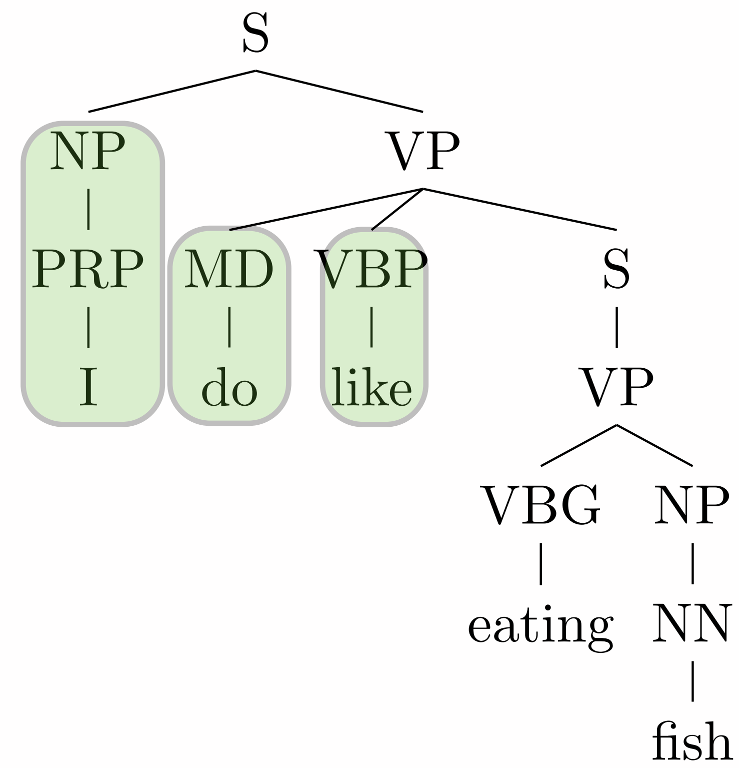
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label





(Cross and Huang, EMNLP 2016)

Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

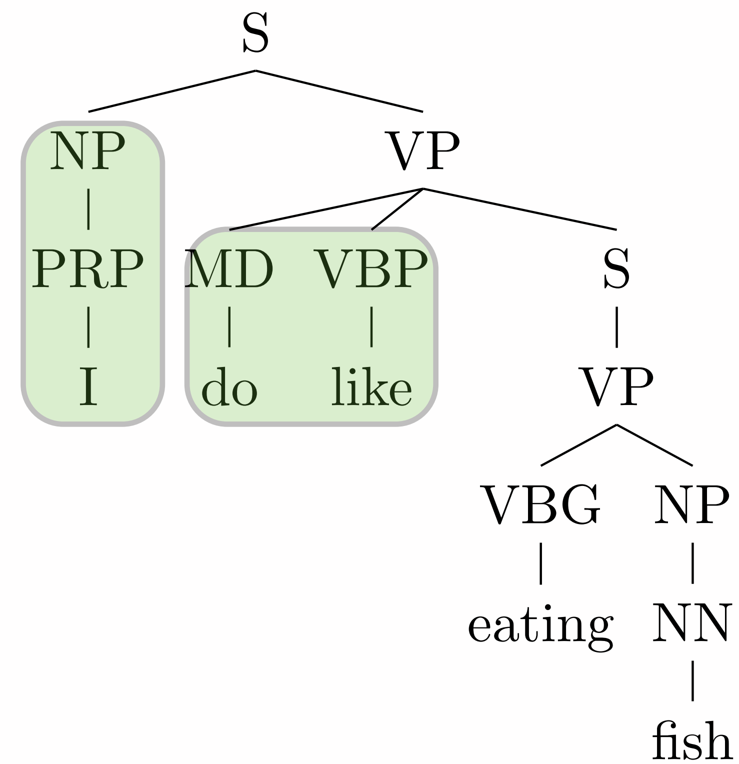


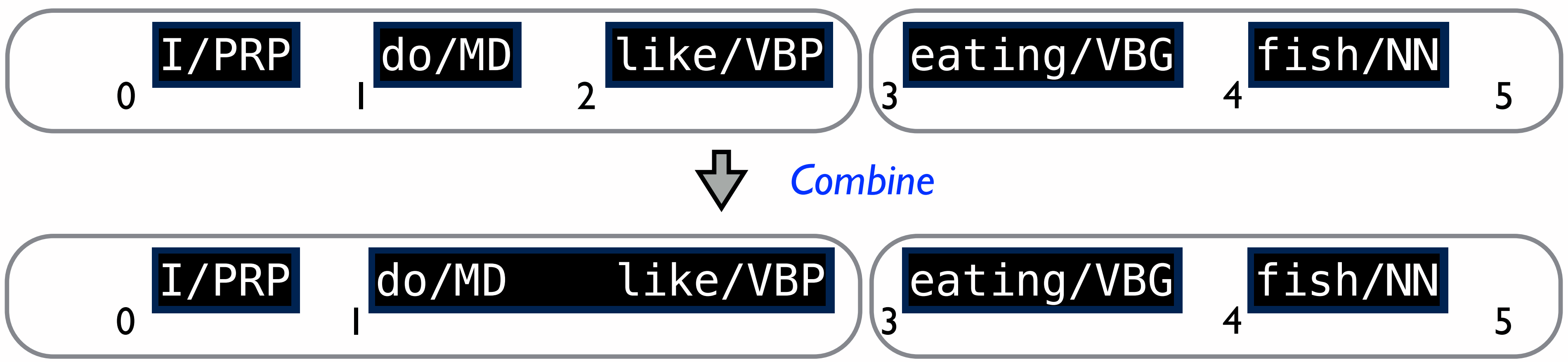
$_0$ I/PRP $_1$ do/MD $_2$ like/VBP $_3$ eating/VBG $_4$ fish/NN $_5$

$t = \{_0NP_1\}$

(Cross and Huang, EMNLP 2016)

Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

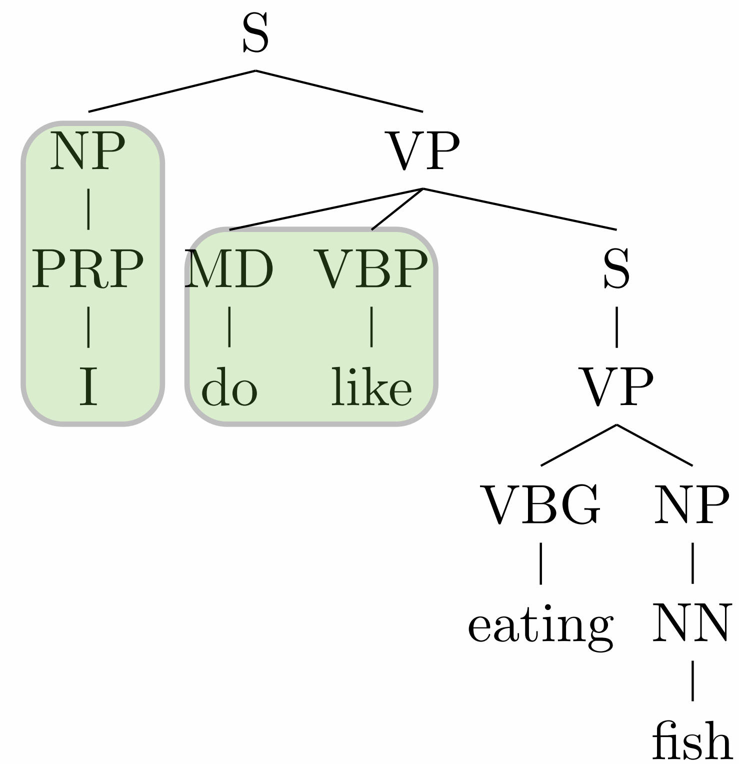


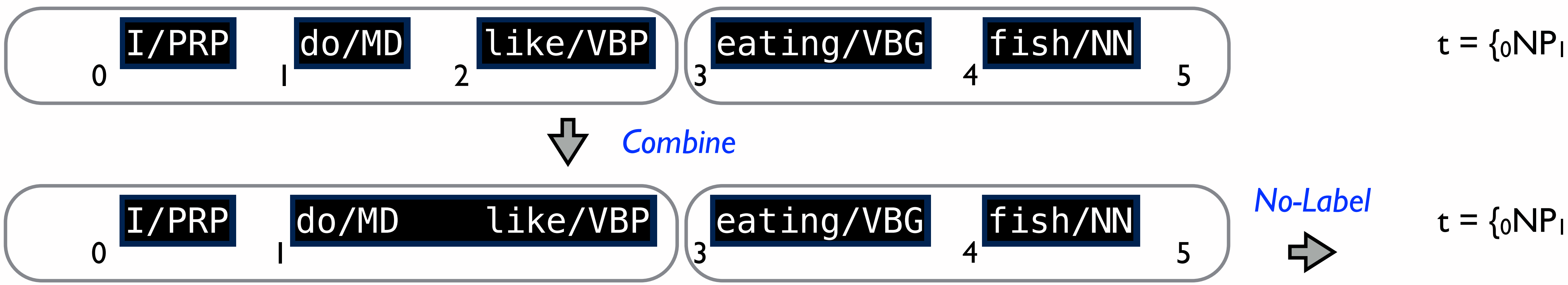


$t = \{{}_0NP_1\}$

(Cross and Huang, EMNLP 2016)

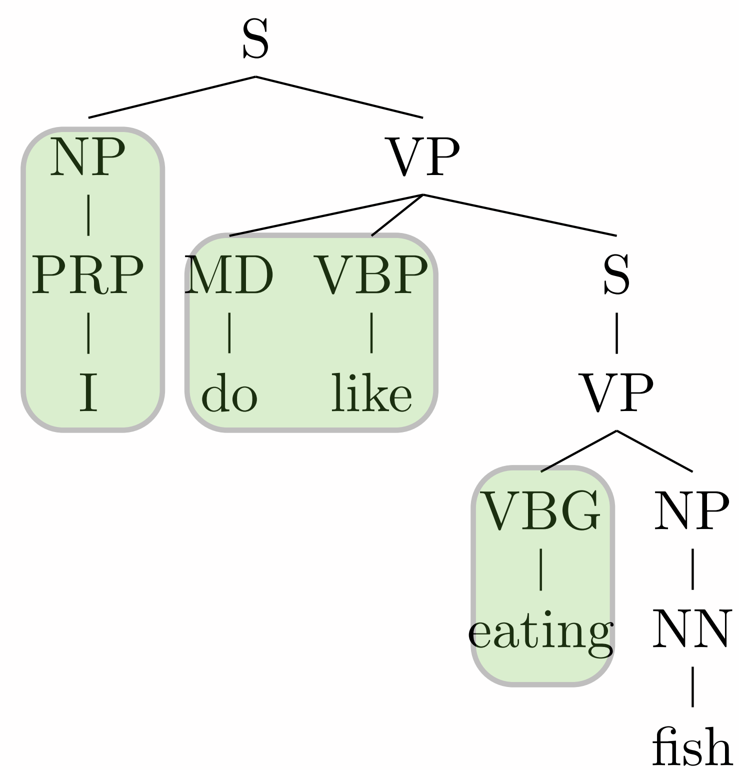
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

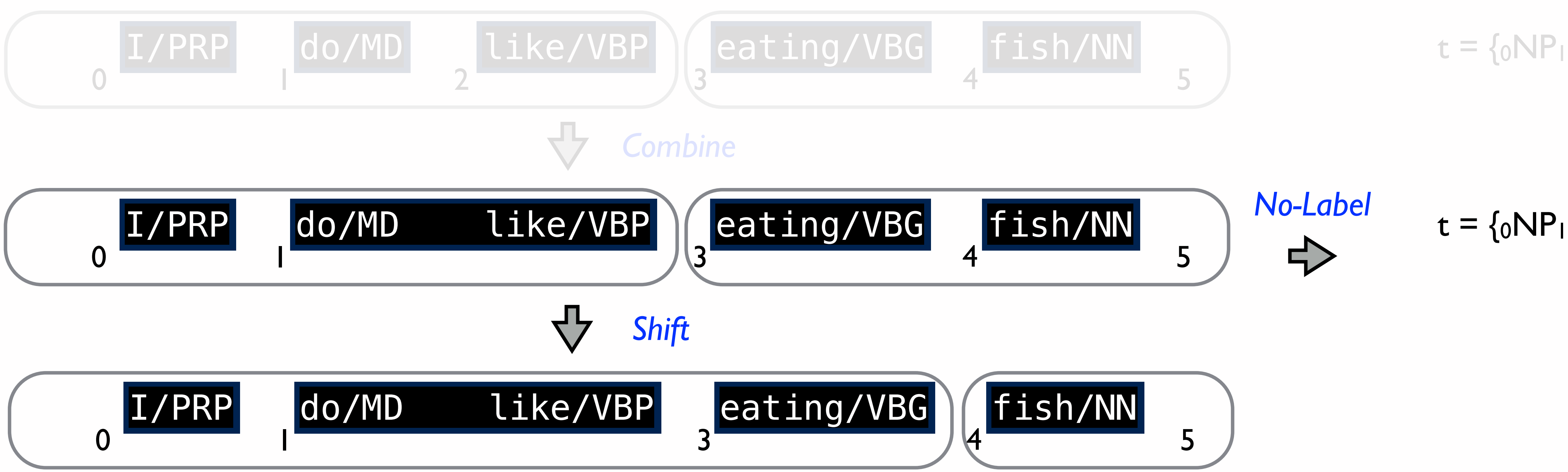




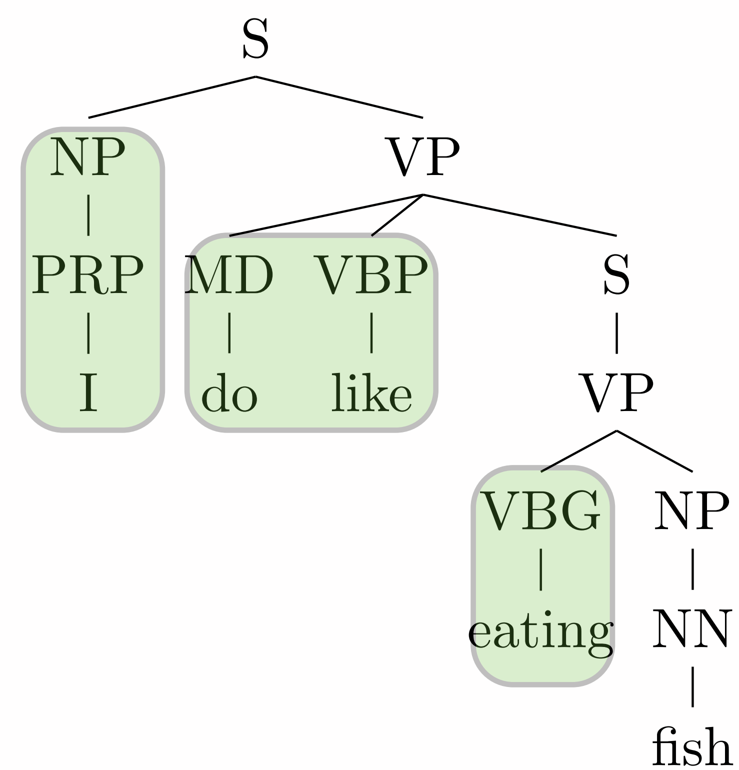
(Cross and Huang, EMNLP 2016)

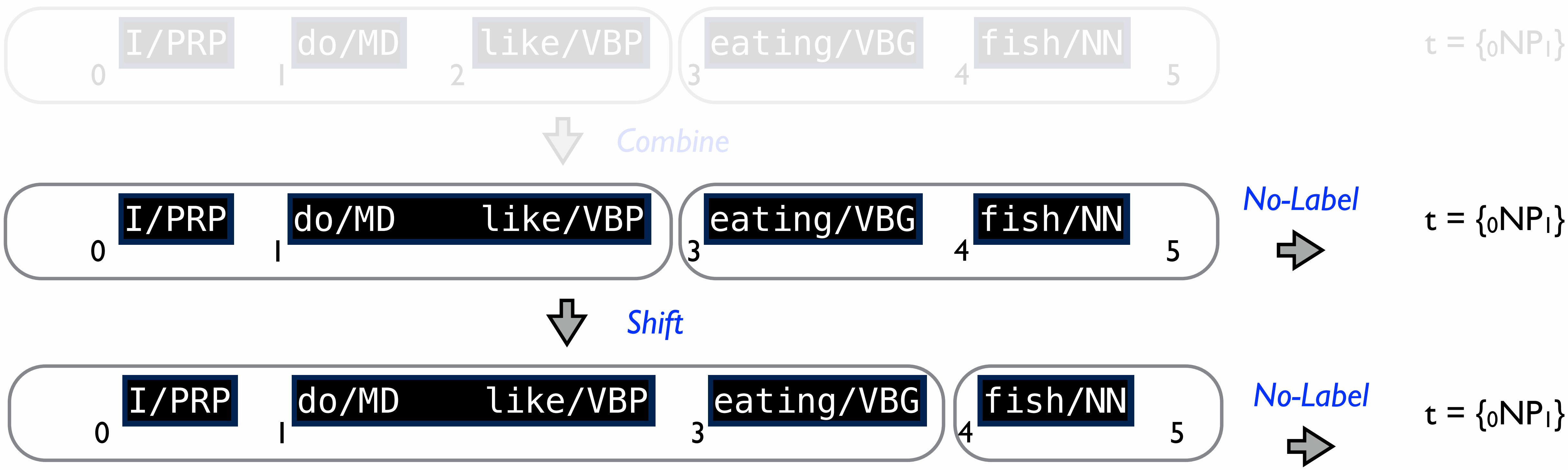
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label





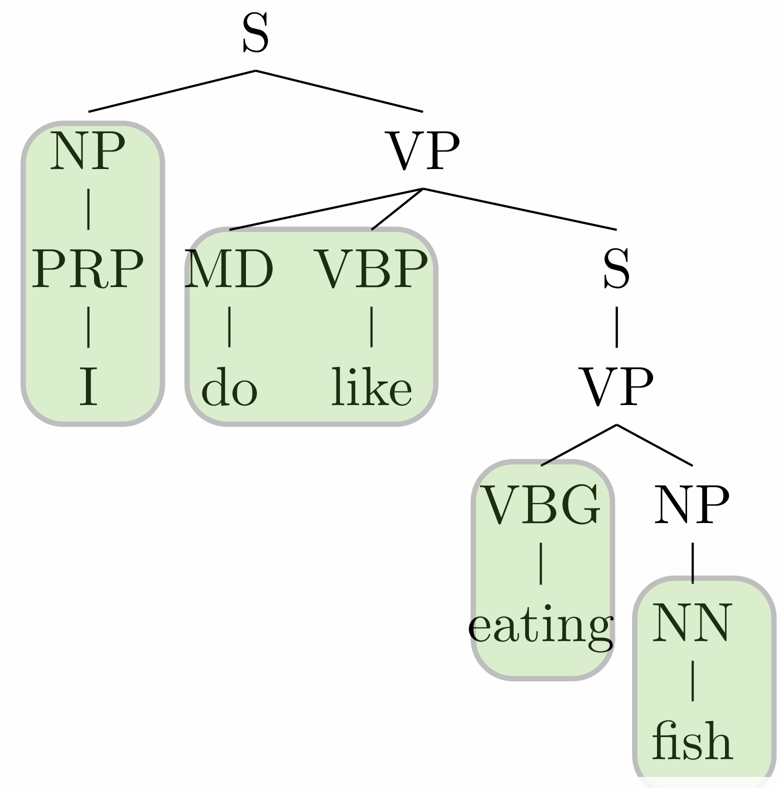
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

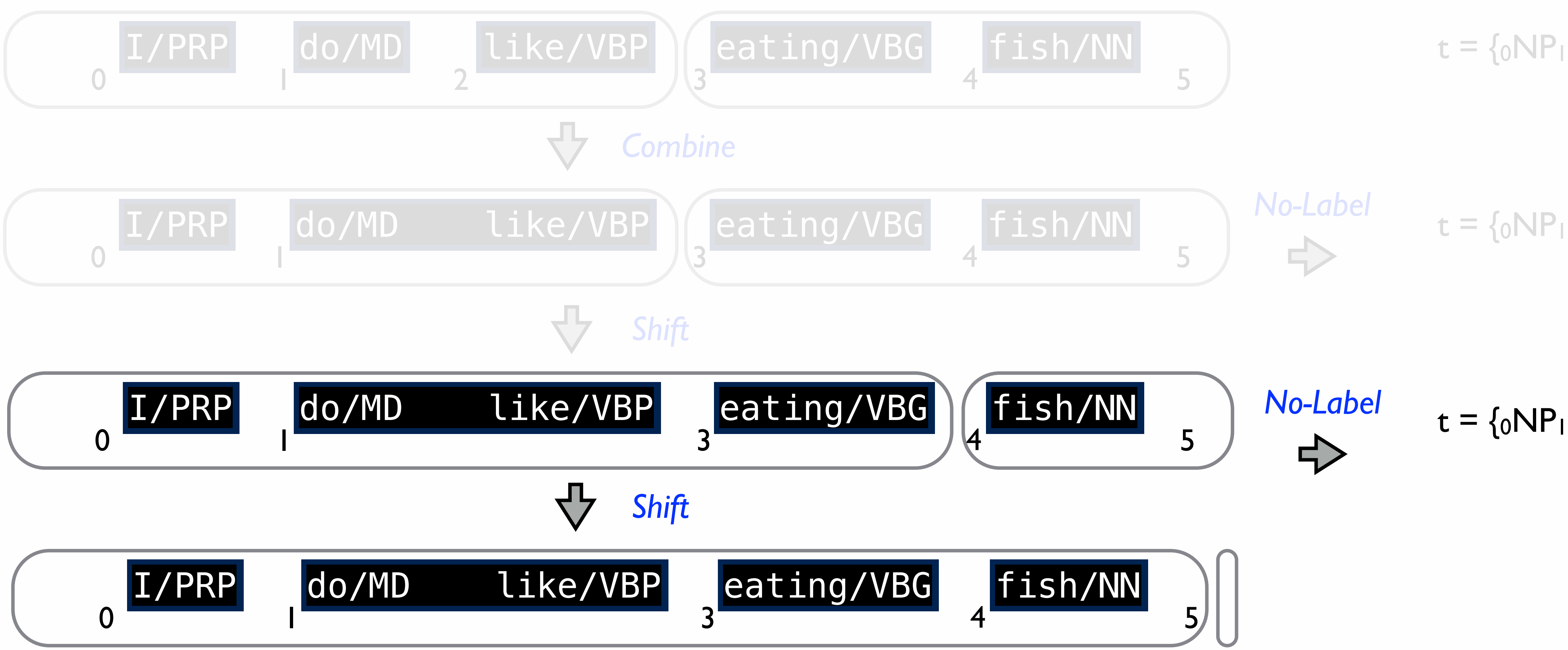




(Cross and Huang, EMNLP 2016)

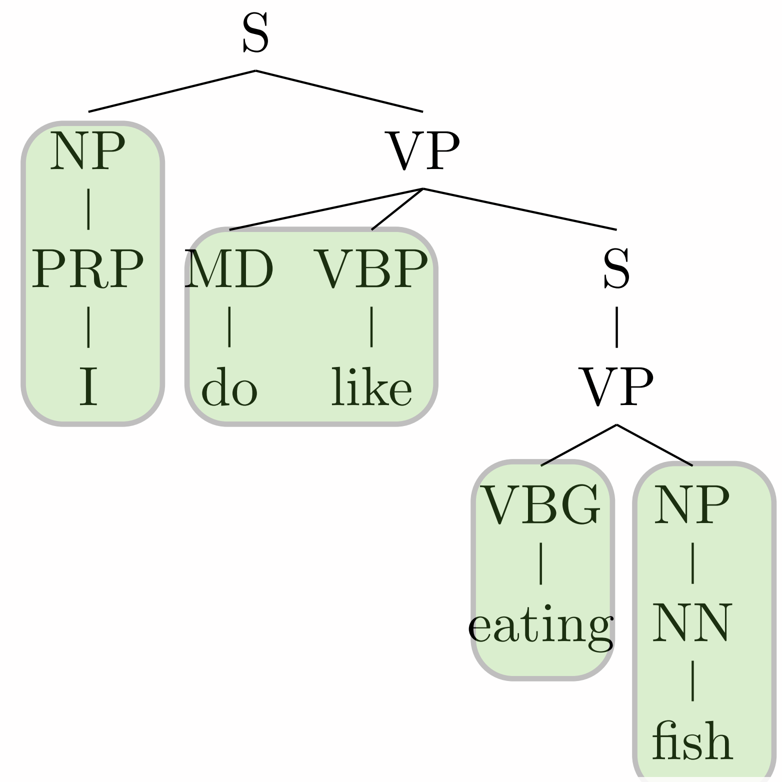
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

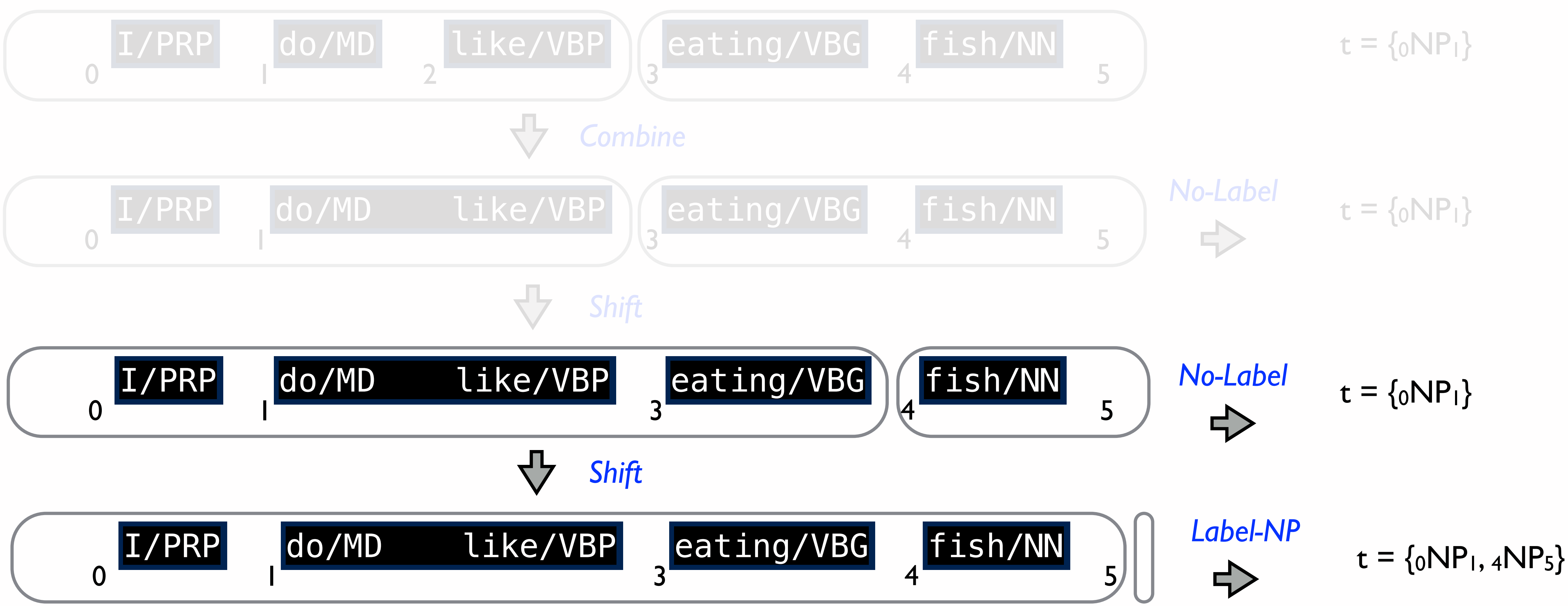




(Cross and Huang, EMNLP 2016)

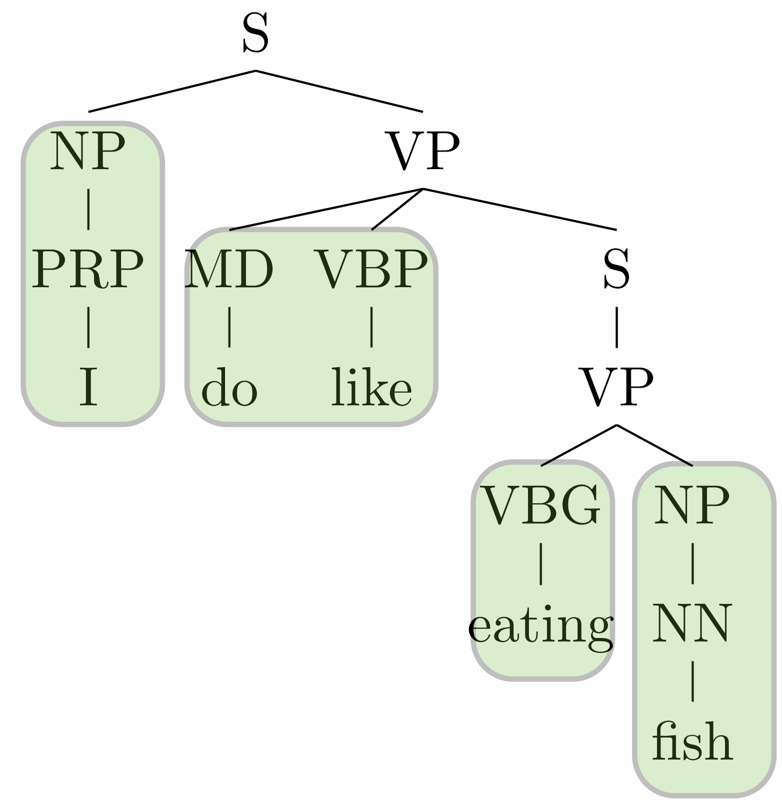
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label





(Cross and Huang, EMNLP 2016)

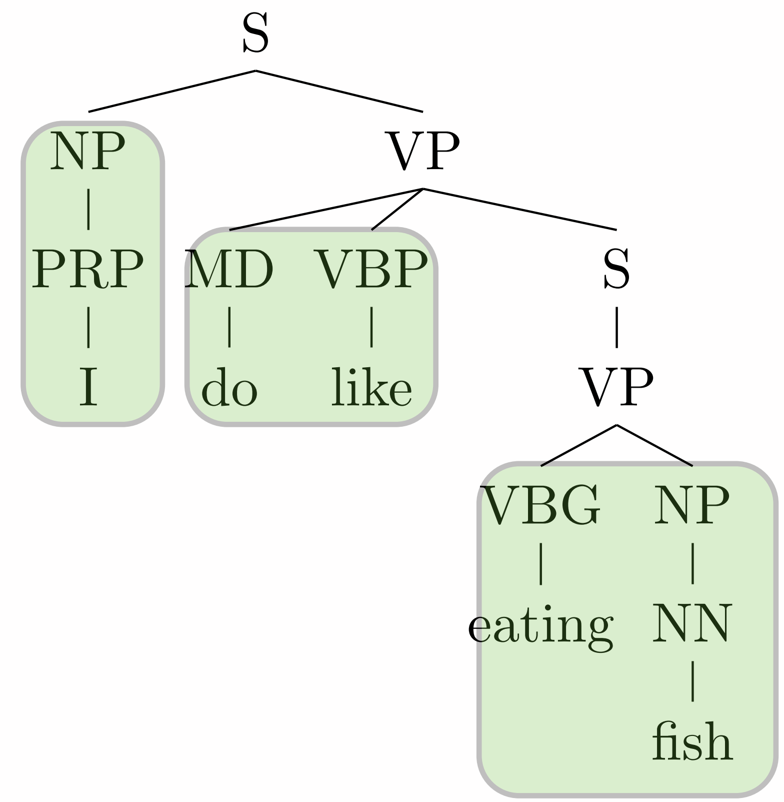
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

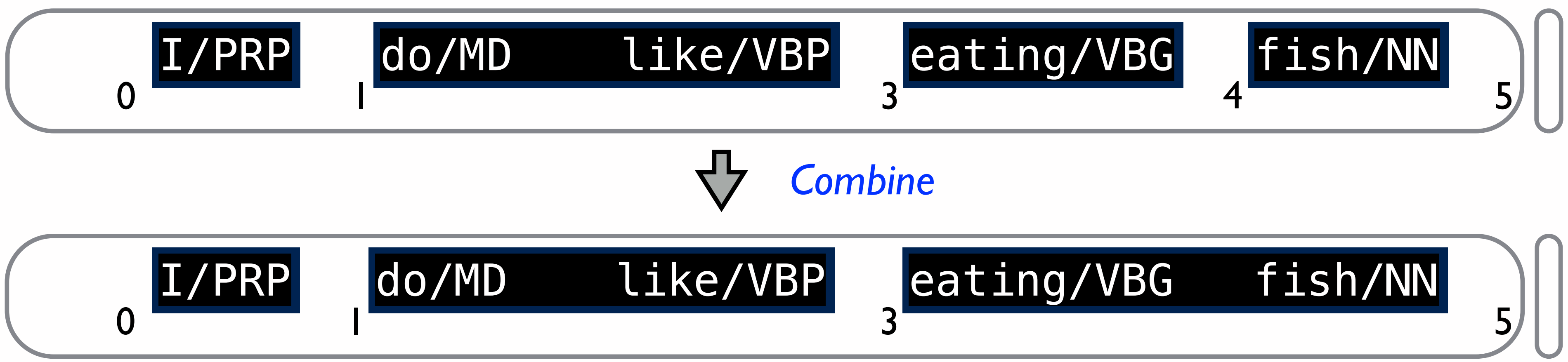


$t = \{{}_0NP_1, {}_4NP_5\}$

(Cross and Huang, EMNLP 2016)

Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

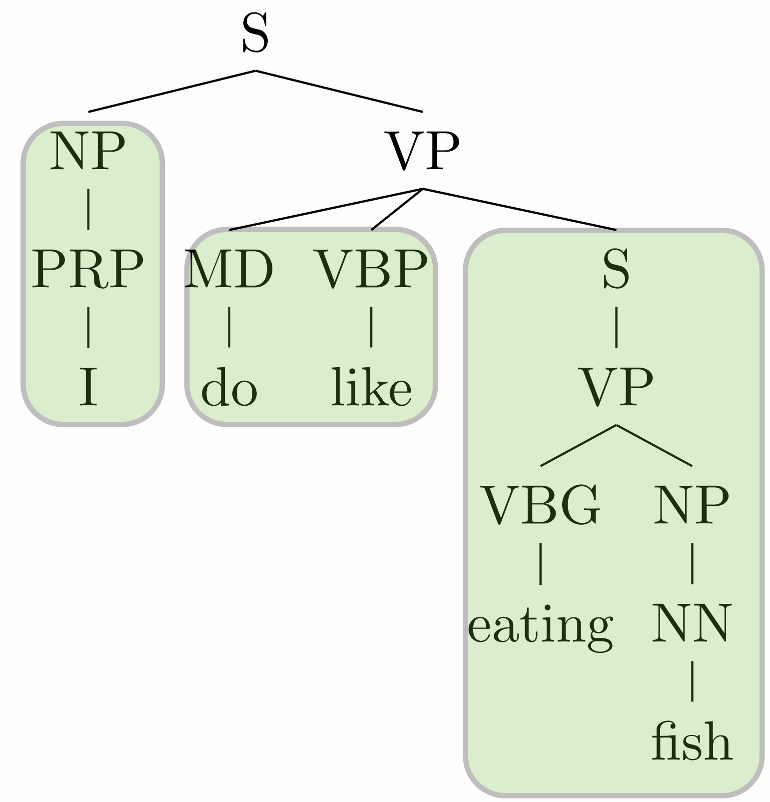


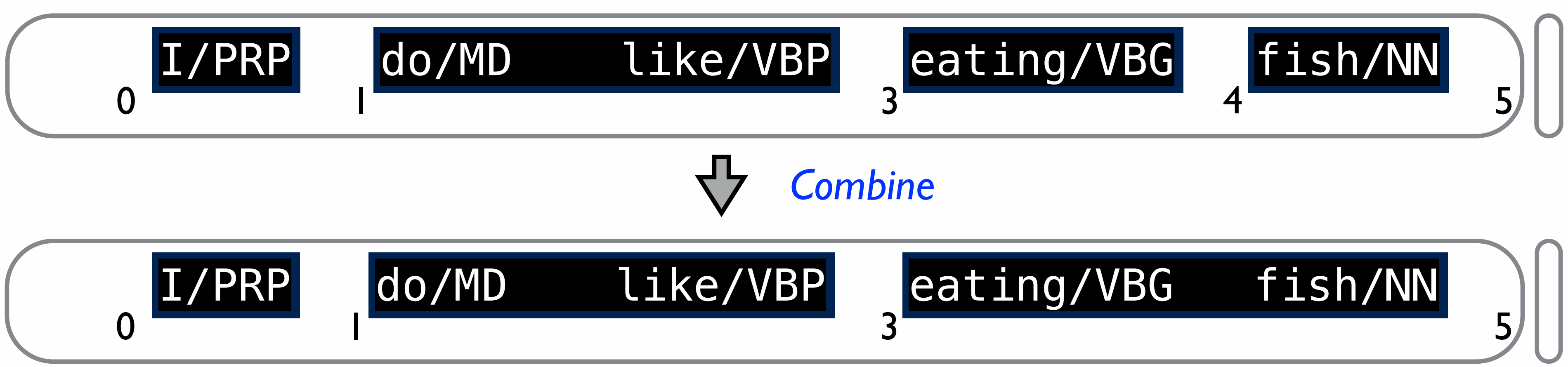


$t = \{{}_0NP_1, {}_4NP_5\}$

(Cross and Huang, EMNLP 2016)

Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label





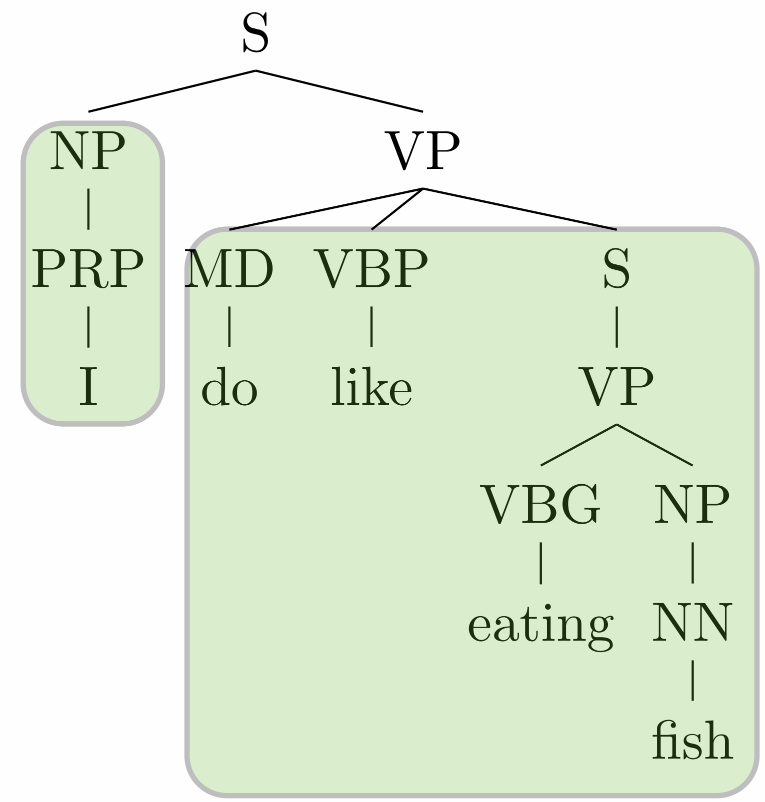
$t = \{{}_0NP_1, {}_4NP_5\}$

Label-S-VP

$t = \{{}_0NP_1, {}_4NP_5, {}_3S_5, {}_3VP_5\}$

(Cross and Huang, EMNLP 2016)

Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label



$_0$ I/PRP $_1$ do/MD like/VBP $_3$ eating/VBG $_4$ fish/NN $_5$

t = {$_0NP_1$, $_4NP_5$}

Combine

$_0$ I/PRP $_1$ do/MD like/VBP $_3$ eating/VBG fish/NN $_5$

Label-S-VP

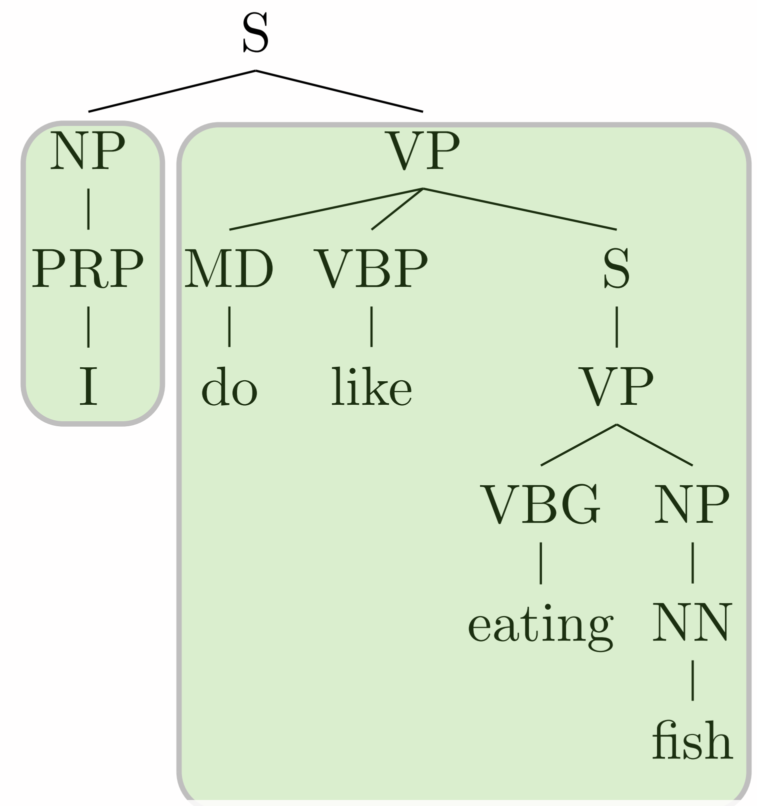
t = {$_0NP_1$, $_4NP_5$, $_3S_5$, $_3VP_5$}

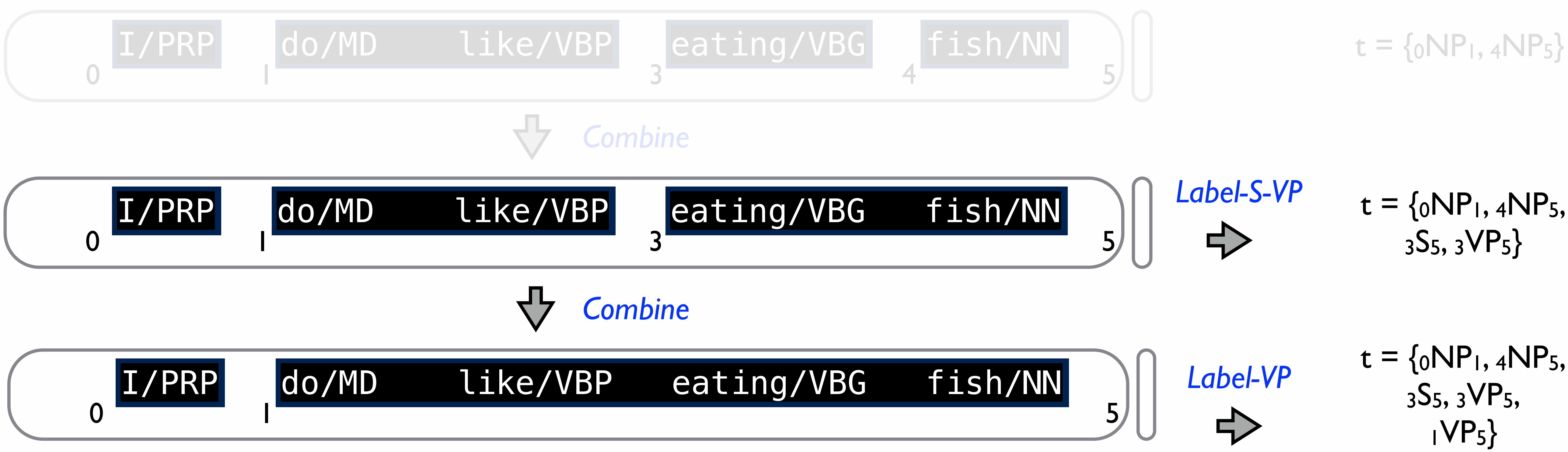
Combine

$_0$ I/PRP $_1$ do/MD like/VBP eating/VBG fish/NN $_5$

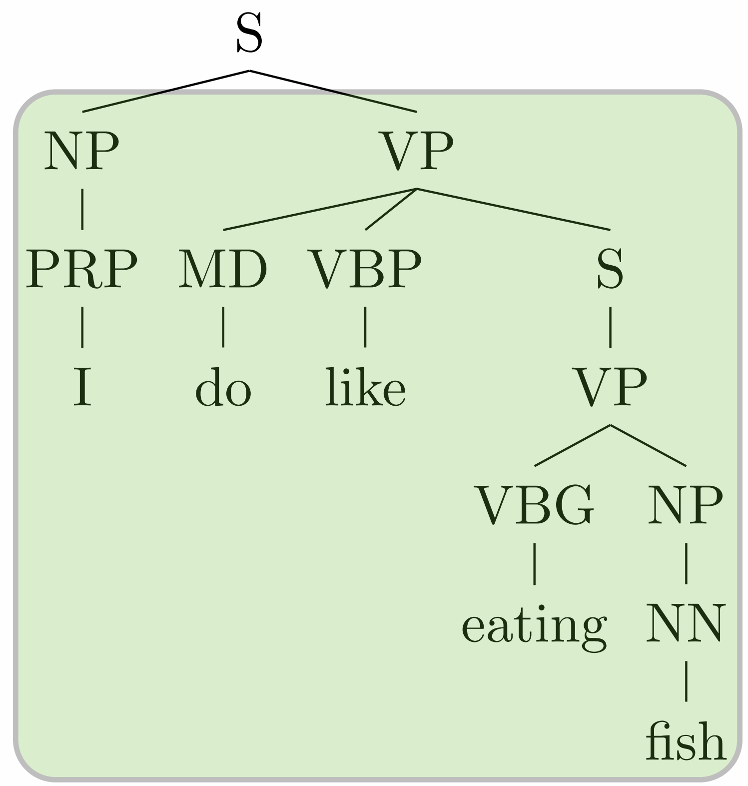
(Cross and Huang, EMNLP 2016)

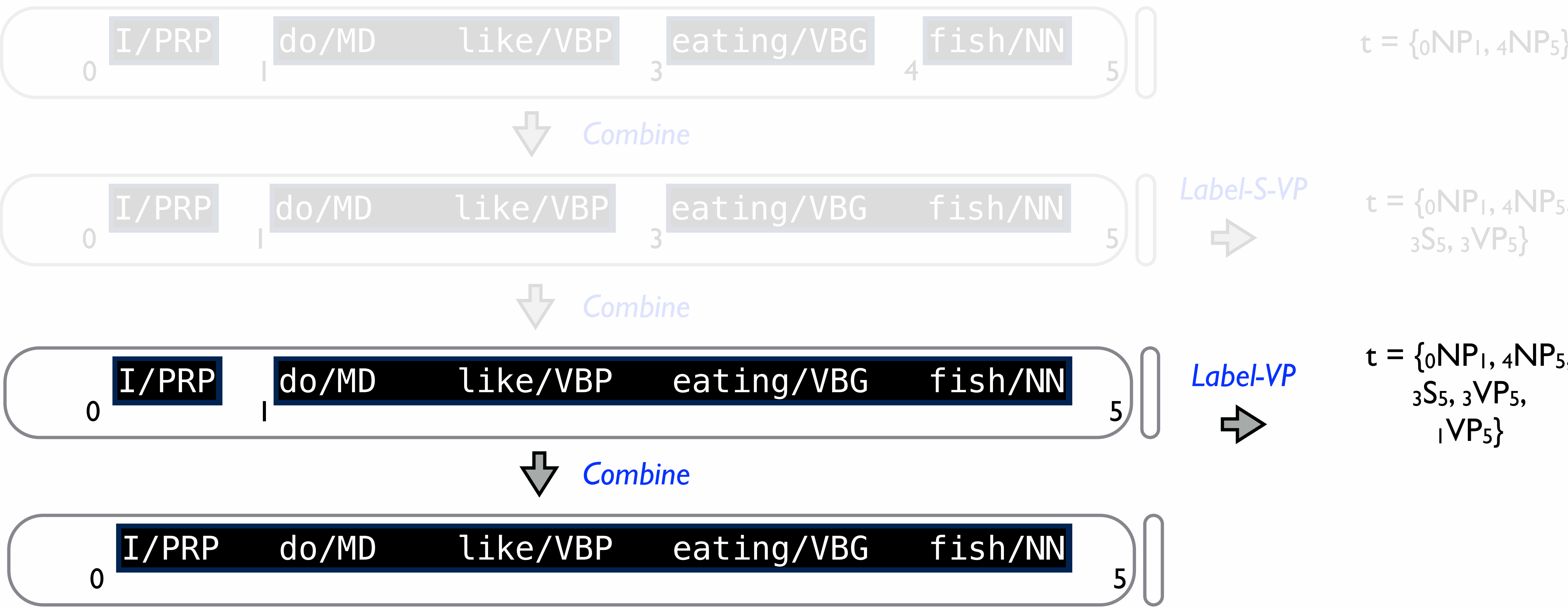
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label





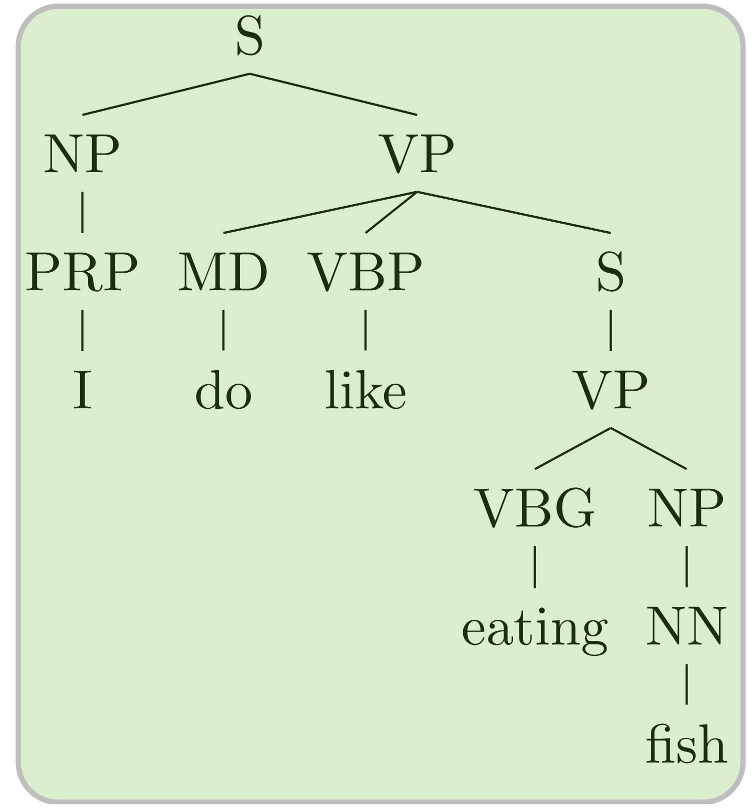
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label

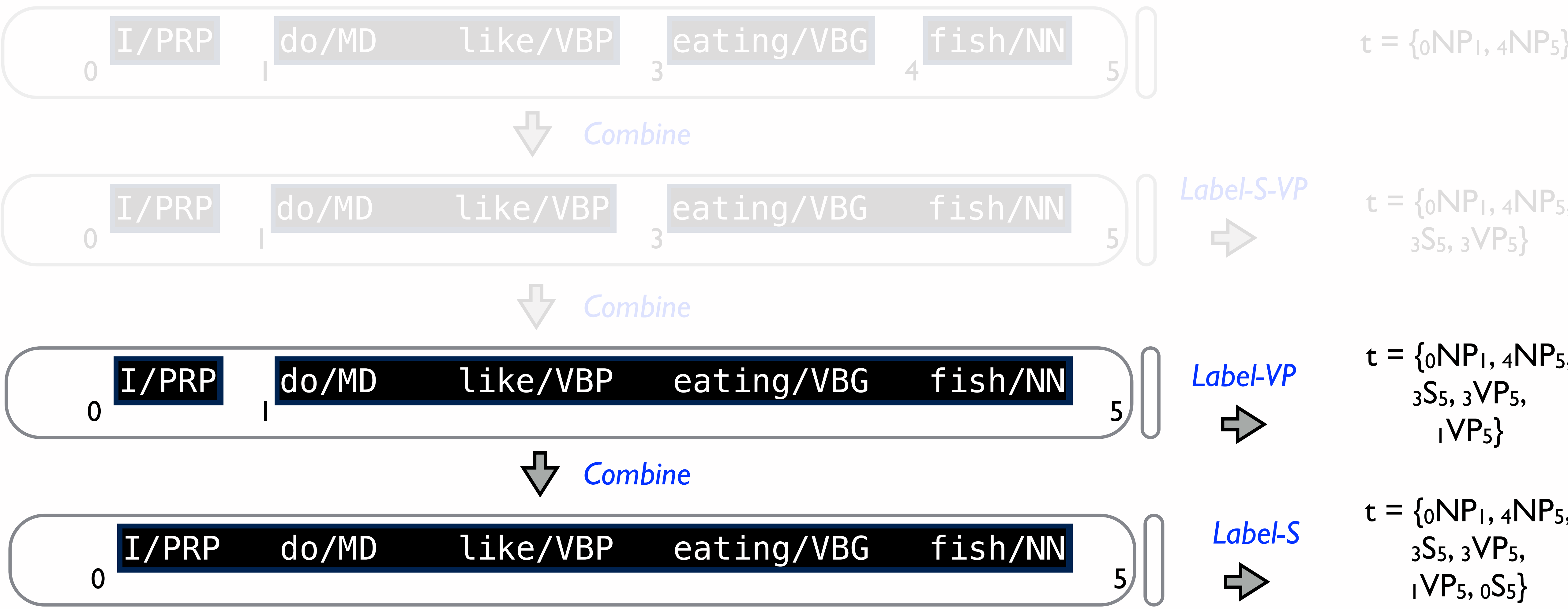




(Cross and Huang, EMNLP 2016)

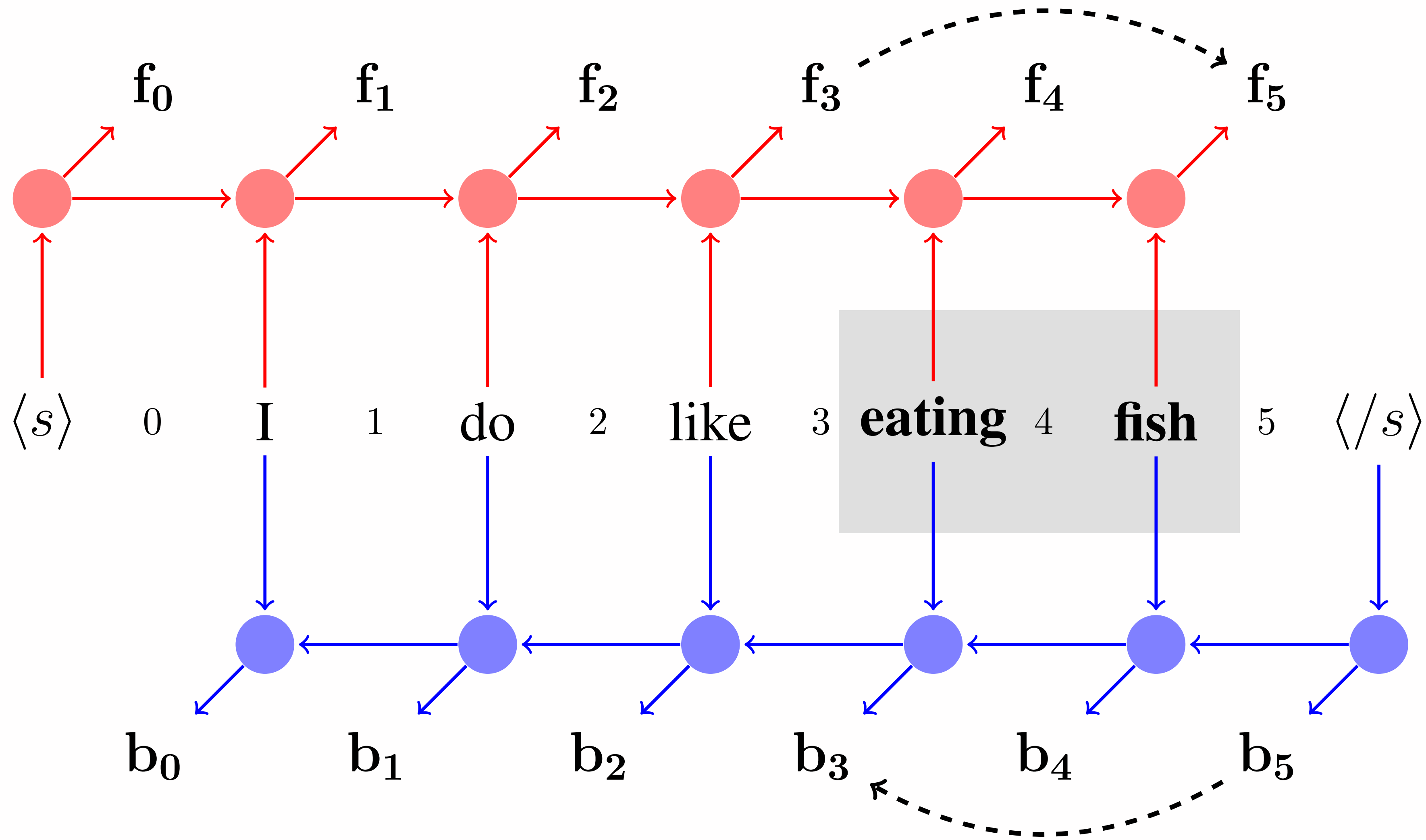
Structural (even step)	Shift
	Combine
Label (odd step)	Label-*X*
	No-Label





(Cross and Huang, EMNLP 2016)

Bi-LSTM Span Features

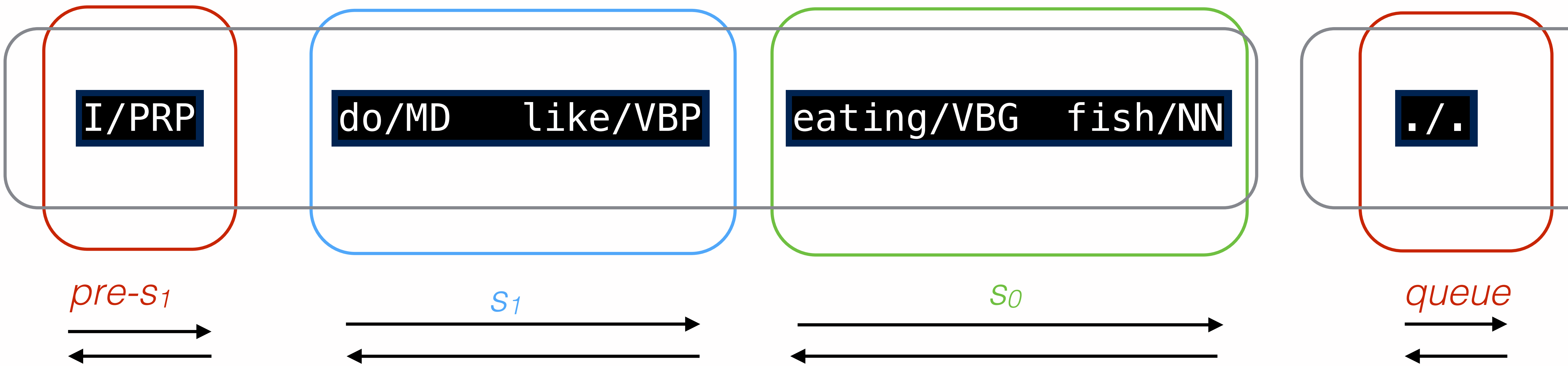


- Sentence segment "eating fish" represented by two vectors:
  - Forward component: f_5 - f_3 (Wang and Chang, ACL 2016)
  - Backward component: b_3 - b_5

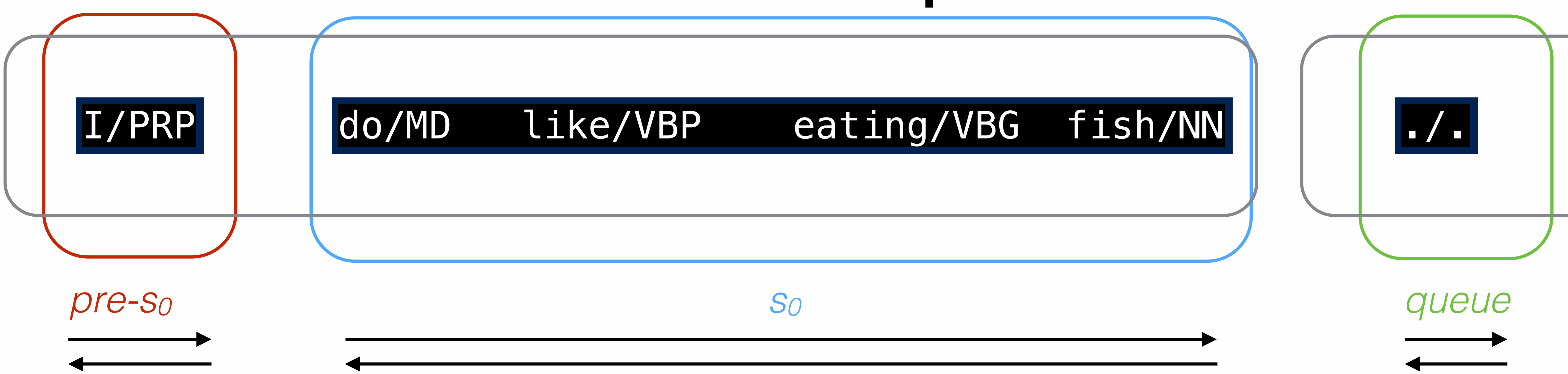
(Cross and Huang, EMNLP 2016)

Structural & Label Actions

Structural Action: 4 spans



Label Action: 3 spans



Results on Penn Treebank

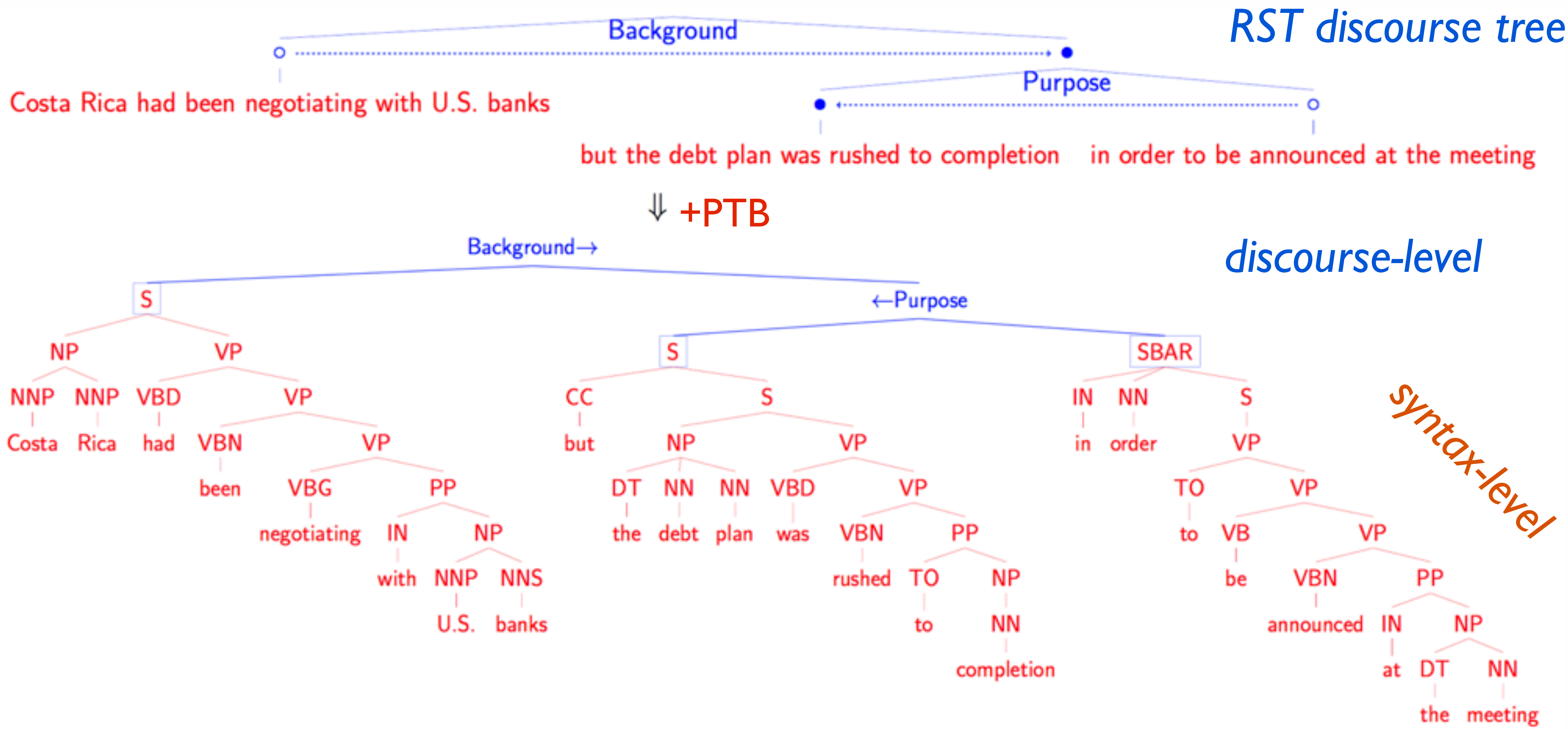
Parser	Search	Recall	Prec.	F_1
Carreras et al. (2008)	cubic	90.7	91.4	91.1
Shindo et al. (2012)	cubic			91.1
Thang et al. (2015)	~cubic			91.1
Watanabe et al. (2015)	beam			90.7
Static Oracle	**greedy**	90.7	91.4	91.0
Dynamic + Exploration	**greedy**	90.5	92.1	**91.3**

- state of the art despite simple system with greedy actions and small embeddings trained from scratch
- first neural constituency parser to outperform sparse features

(Cross and Huang, EMNLP 2016)

Extension: Joint Syntax-Discourse Parsing

- extend span-based parsing to discourse parsing
  - end-to-end, joint syntactic and discourse parsing



(Kai and Huang, EMNLP 2017)

In this talk...

- Background
- Dynamic Programming for Incremental Parsing
- Interlude: NN Features: from feedforward to recurrent
- Bidirectional RNNs: minimal features; no tree structures!
  - dependency parsing (Kiperwaser+Goldberg, 2016, Cross+Huang, 2016a)
  - span-based constituency parsing (Cross+Huang, 2016b)
- Marrying DP & RNNs *(mostly not my work!)*
  - minimal span-based constituency parsing (Stern et al, ACL 2017)
  - transition-based dependency parsing (Shi et al, EMNLP 2017)

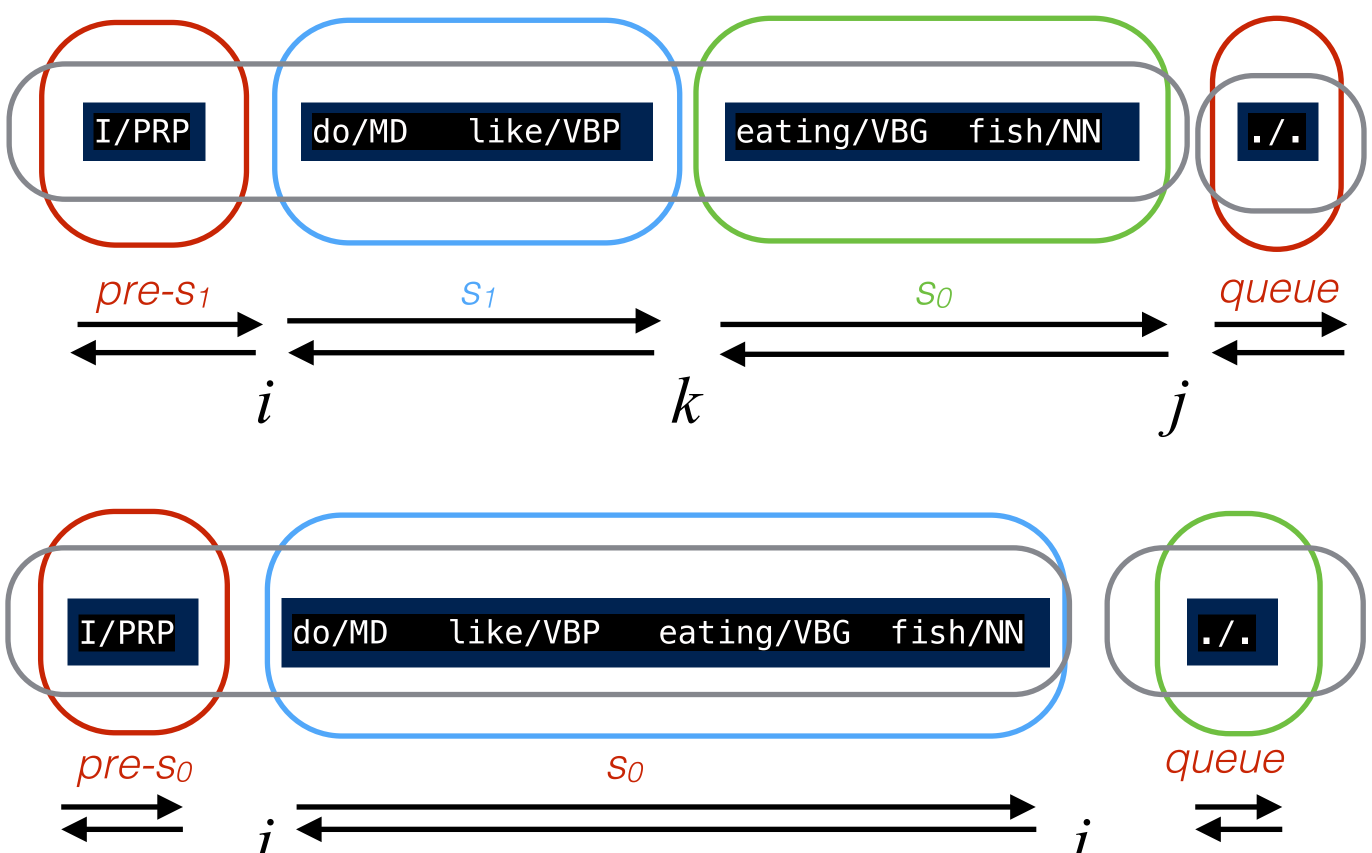
Minimal Span-based Const. Parsing

- chart-based bottom-up parsing instead of incremental
  - an even simpler score formulation
  - $O(n^3)$ exact DP (CKY) instead of greedy search
  - global loss-augmented training instead of local training

Minimal Span-based Const. Parsing

- chart-based bottom-up parsing instead of incremental
  - an even simpler score formulation
  - $O(n^3)$ exact DP (CKY) instead of greedy search
  - global loss-augmented training instead of local training

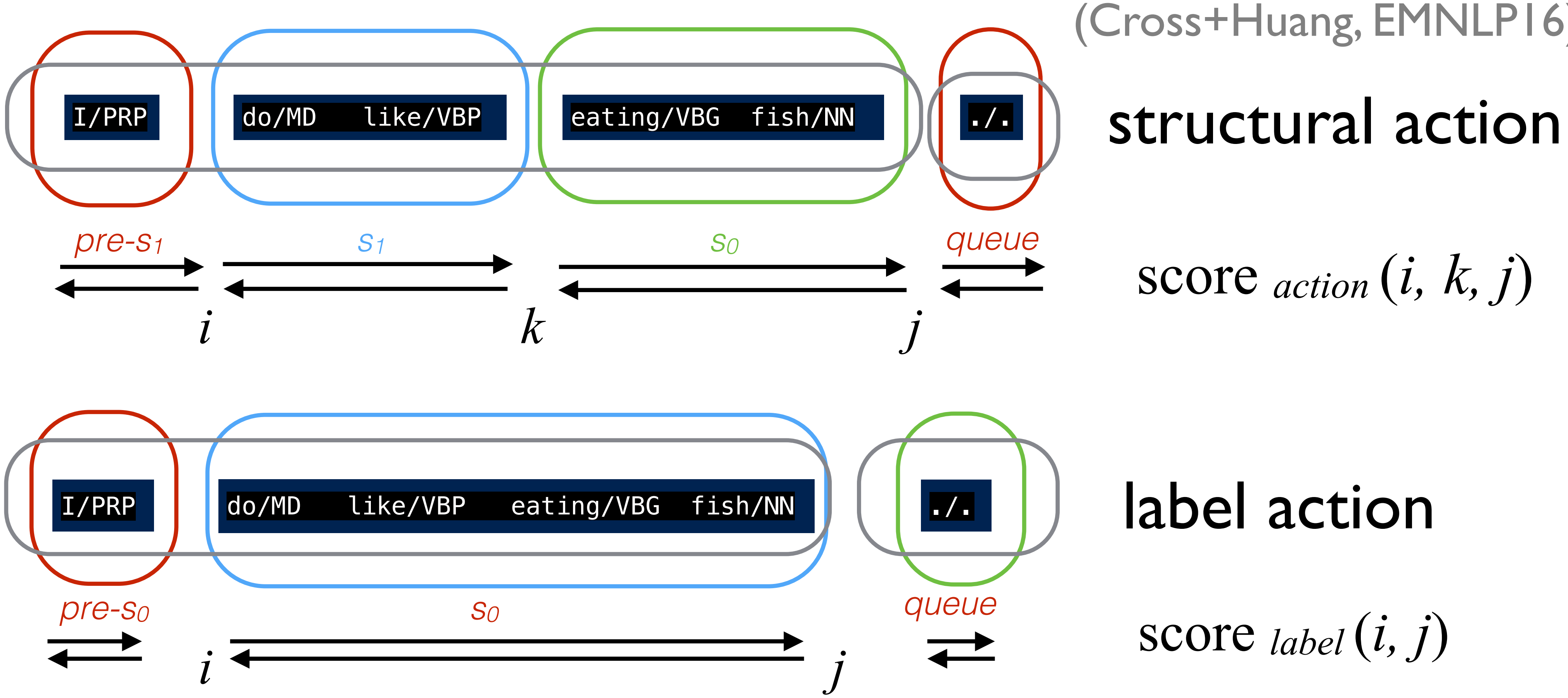
(Cross+Huang, EMNLP16)



Minimal Span-based Const. Parsing

- chart-based bottom-up parsing instead of incremental
  - an even simpler score formulation
  - $O(n^3)$ exact DP (CKY) instead of greedy search
  - global loss-augmented training instead of local training

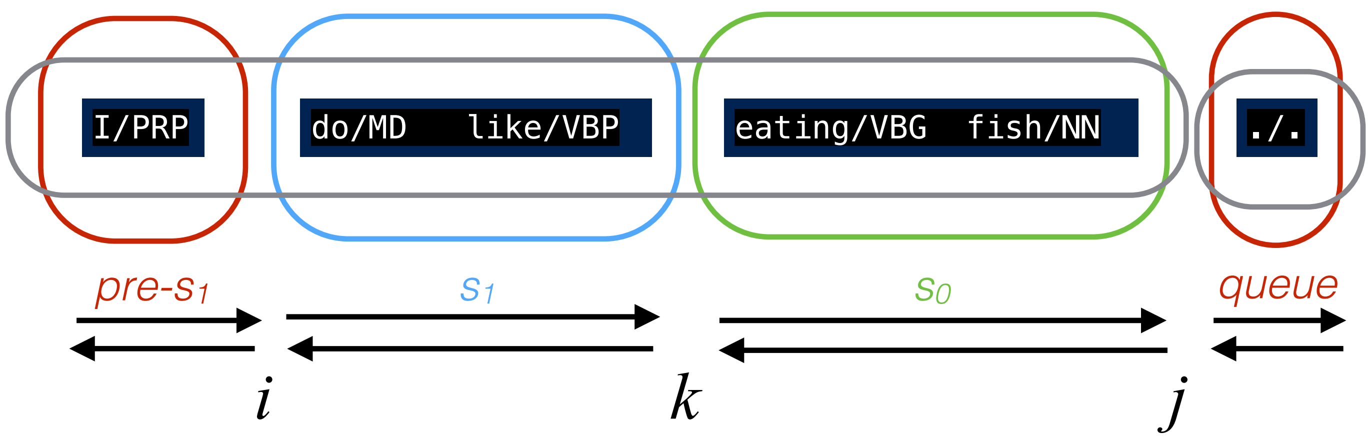
(Cross+Huang, EMNLP16)



Minimal Span-based Const. Parsing

- chart-based bottom-up parsing instead of incremental
  - an even simpler score formulation
  - $O(n^3)$ exact DP (CKY) instead of greedy search
  - global loss-augmented training instead of local training

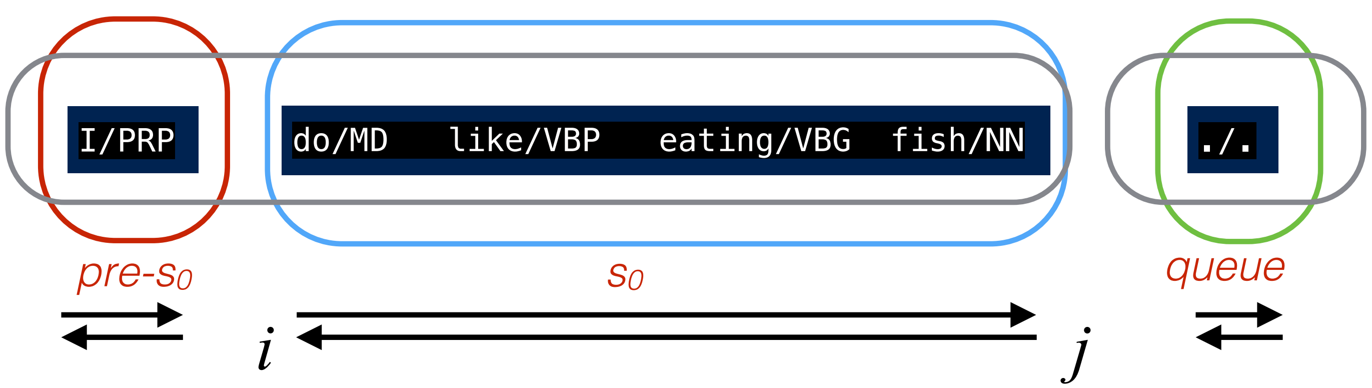
(Cross+Huang, EMNLP16) (Stern+, ACL 2017)



structural action

score $_{action}\,(i, k, j)$

$\max_k$ best (i, k)+best (k, j)



label action

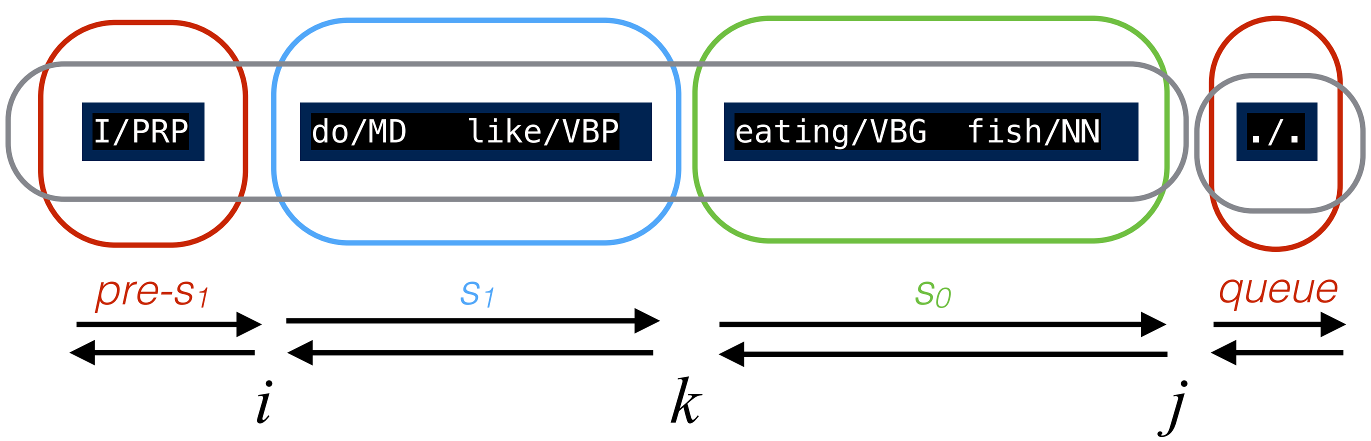
score $_{label}\,(i, j)$

$\max_{label}$ score $_{label}\,(i, j)$

Minimal Span-based Const. Parsing

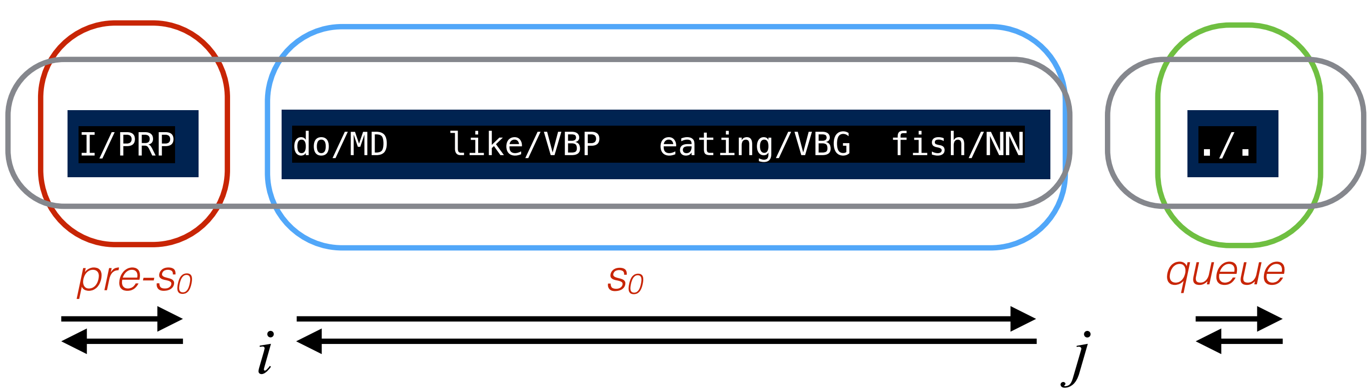
- chart-based bottom-up parsing instead of incremental
  - an even simpler score formulation
  - $O(n^3)$ exact DP (CKY) instead of greedy search
  - global loss-augmented training instead of local training

(Cross+Huang, EMNLP16)



structural action

score $_{action}(i, k, j)$



label action

score $_{label}(i, j)$

(Stern+, ACL 2017)

best $(i, j) =$

$$\max_k \text{best}(i, k) + \text{best}(k, j)$$

$$+$$

$$\max_{label} \text{score}_{label}(i, j)$$

Global Training & Loss-Augmented Decoding

want $s_{\text{tree}}(T^*) > s_{\text{tree}}(T)$ for all $T \neq T^*$

and larger margin for worse trees: $s_{\text{tree}}(T^*) \geq \Delta(T, T^*) + s_{\text{tree}}(T)$

loss-augmented decoding in training (find the most-violated tree, i.e., a *bad tree* with *good score*)

$$\hat{T} = \max_{T} [\underbrace{\Delta(T, T^*)}_{\text{bad tree}} + \underbrace{s_{\text{tree}}(T)}_{\text{good score}}]$$

loss-augmented decoding for Hamming loss (approximating F1):

simply replace $\text{score}_{\textit{label}}(i, j)$

with $\text{score}_{\textit{label}}(i, j) + \mathbf{1}(\textit{label} \neq \textit{label}^*_{ij})$

gold tree label for span (i, j)

(could be "nolabel")

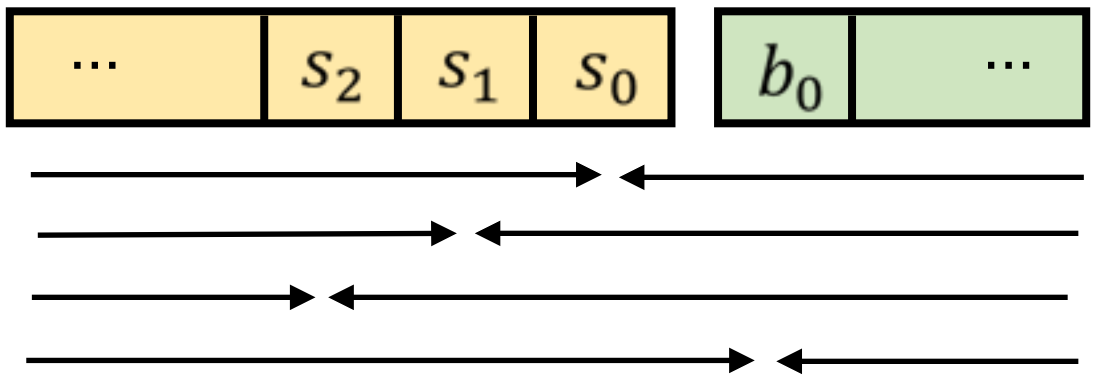
(Stern+, ACL 2017)

Penn Treebank Results

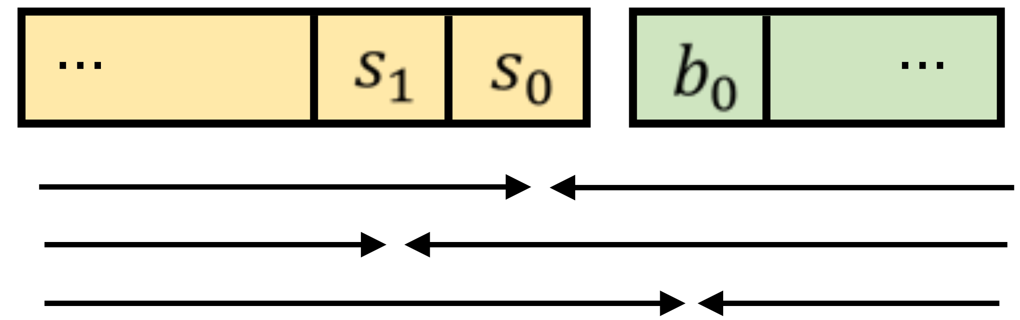
Parser	F1 Score
Hall et al. (2014)	89.2
Vinyals et al. (2015)	88.3
Cross and Huang (2016b)	91.3
Dyer et al. (2016) corrected	91.7
Liu and Zhang (2017)	91.7
Chart Parser	91.7
+refinement	91.8

(Stern+, ACL 2017)

Minimal Feats for Incremental Dep. Parsing

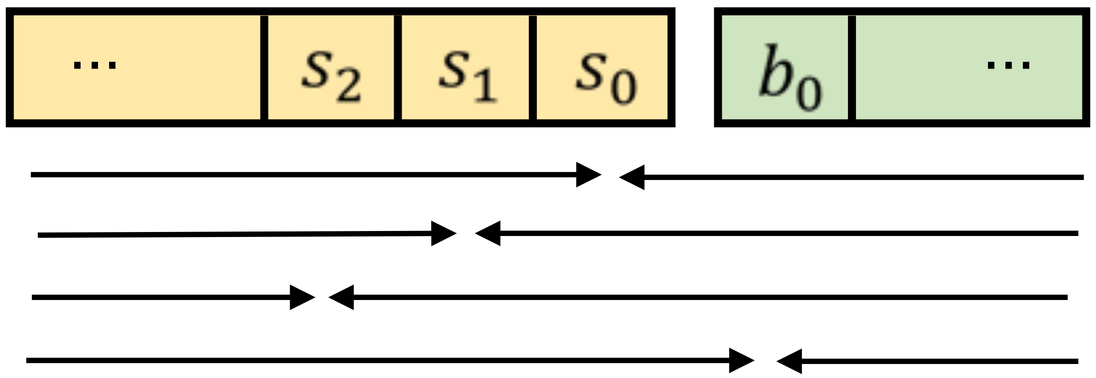


(Kiperwaser and Goldberg 2016)
arc-eager

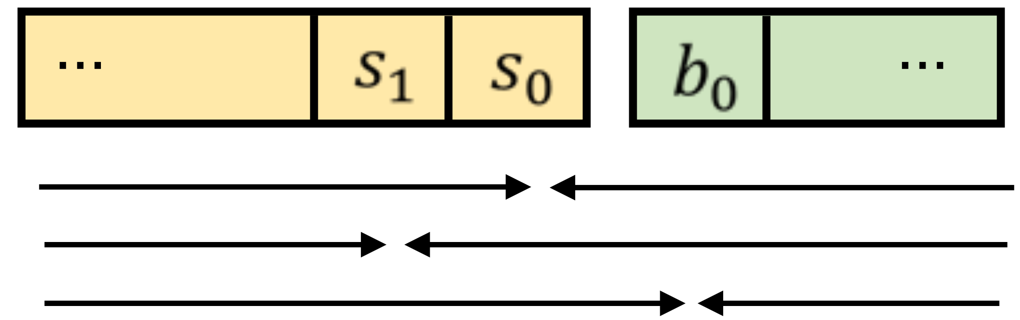


(Cross and Huang, ACL 2016)
arc-standard

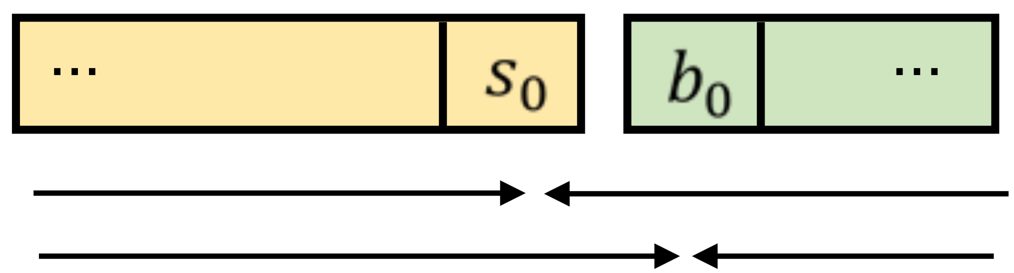
Minimal Feats for Incremental Dep. Parsing



(Kiperwaser and Goldberg 2016)
arc-eager



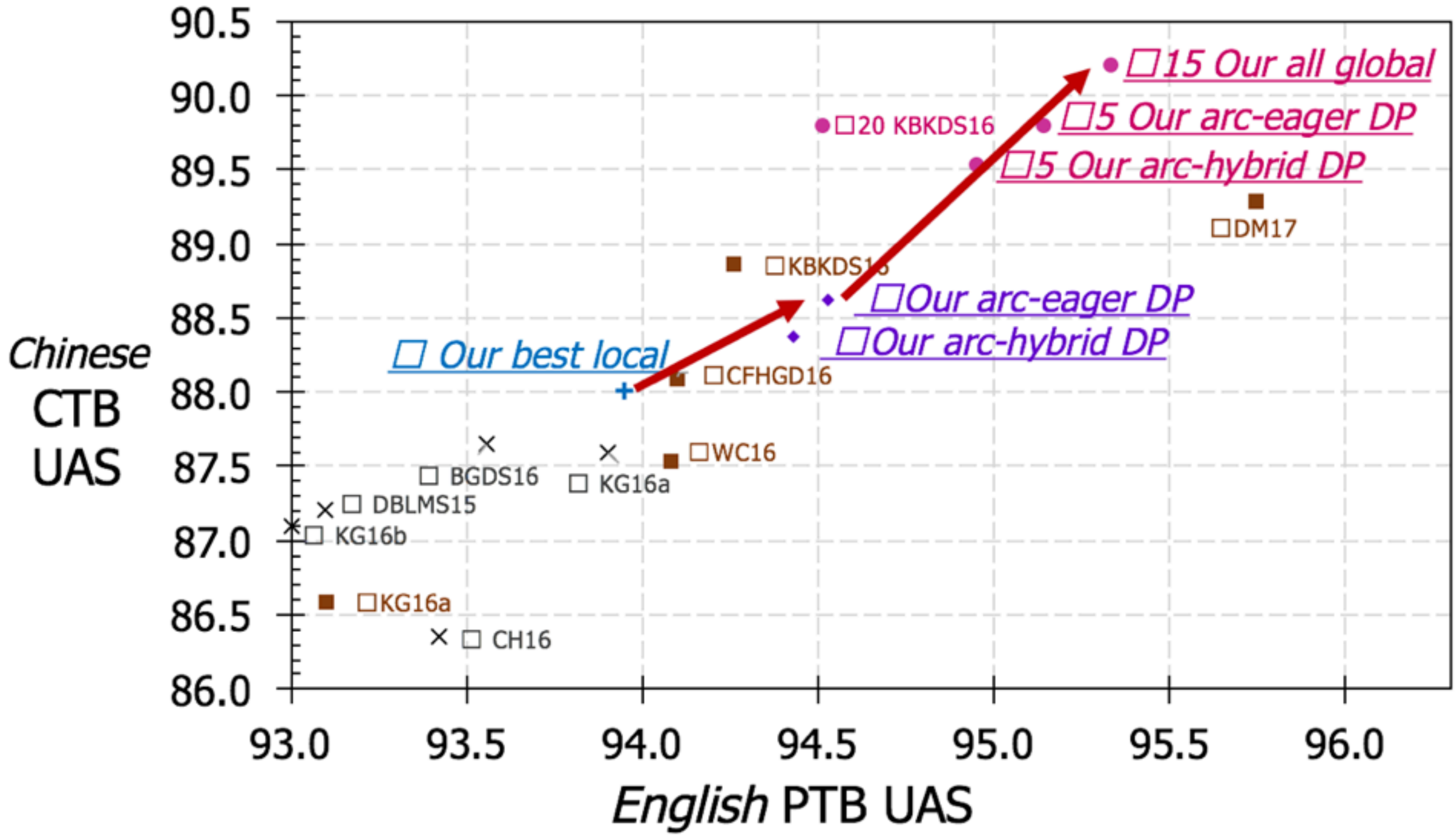
(Cross and Huang, ACL 2016)
arc-standard



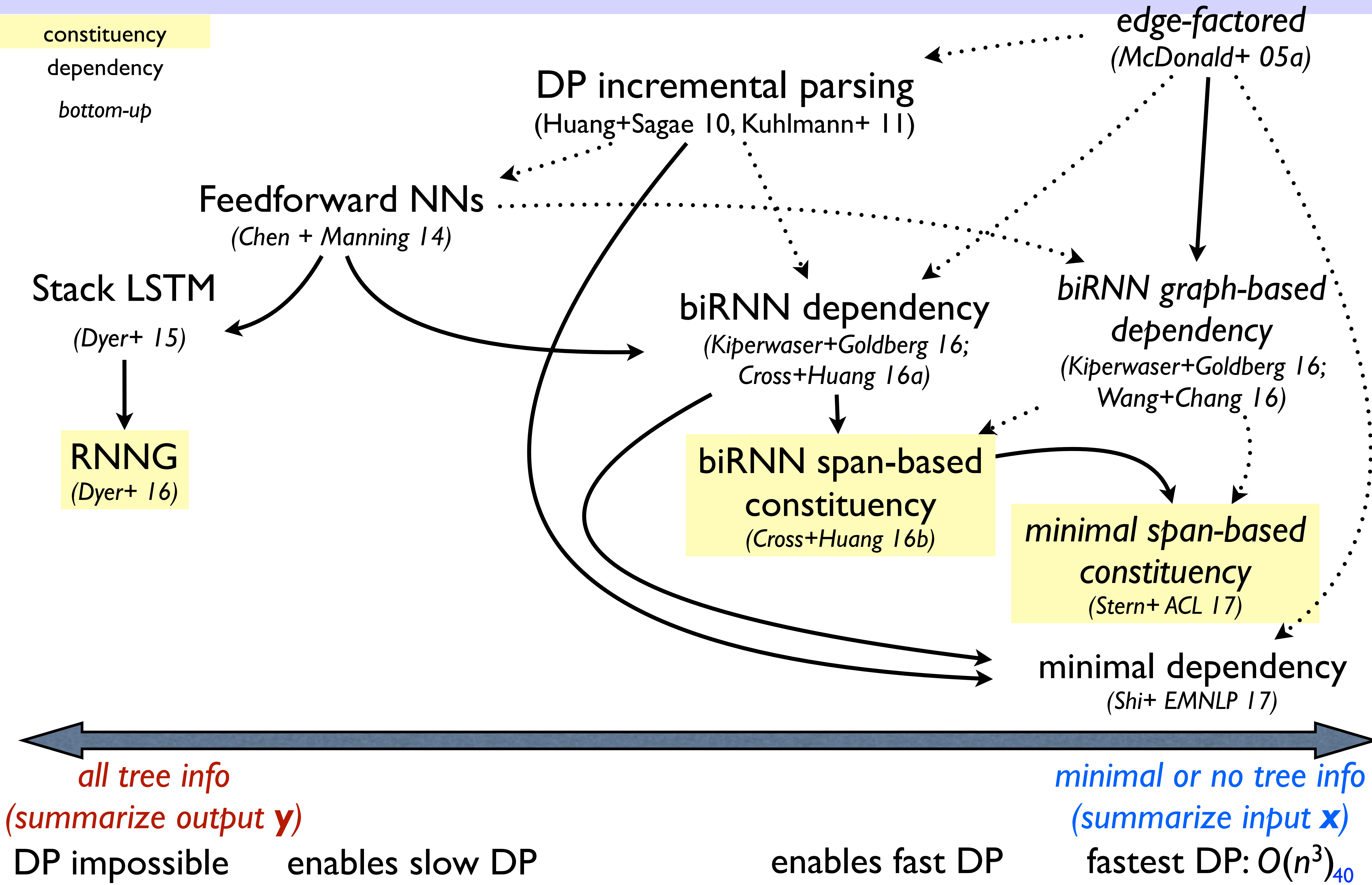
(Shi, Huang, Lee, EMNLP 2017)
Saturday talk!
arc-hybrid and arc-eager

works for both greedy and $O(n^3)$ DP

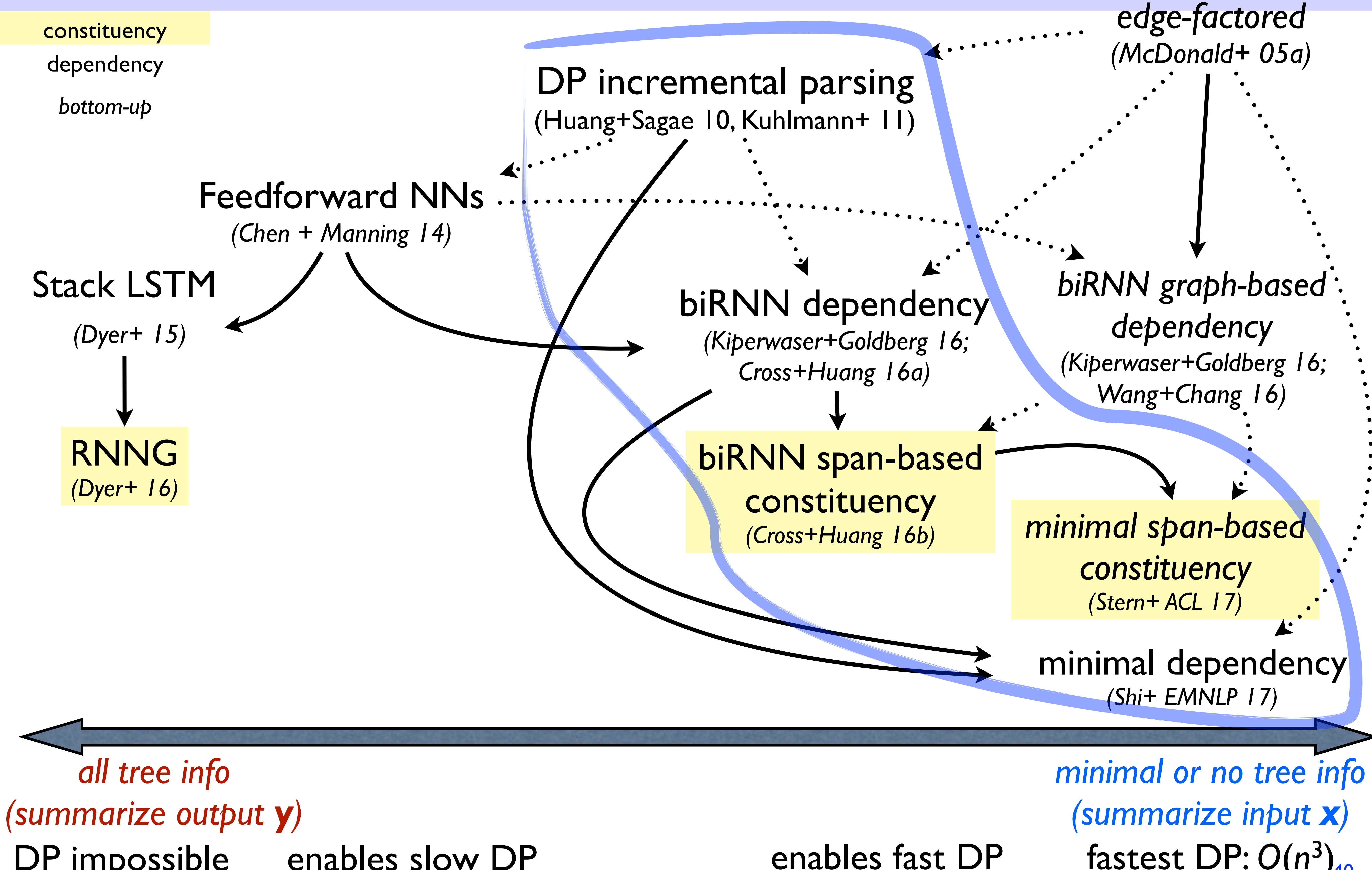
Minimal Feats for Incremental Dep. Parsing



Spectrum: Neural Incremental Parsing



Spectrum: Neural Incremental Parsing



Conclusions and Limitations

Conclusions and Limitations

- DP and RNNs can indeed be married, if done creatively
  - biRNN summarizing input **x** and not output structure **y**
  - this allows efficient DP with exact search
  - combine with global learning (loss-augmented decoding)

Conclusions and Limitations

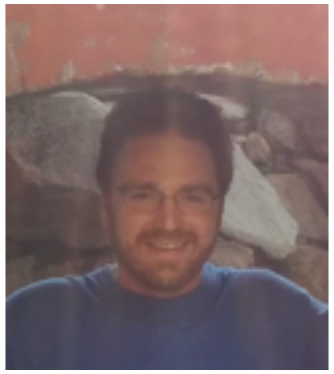
- DP and RNNs can indeed be married, if done creatively
  - biRNN summarizing input $\mathbf{x}$ and not output structure $\mathbf{y}$
  - this allows efficient DP with exact search
  - combine with global learning (loss-augmented decoding)
- but exact DP is still too slow
  - future work: linear-time beam search DP with biRNNs

Conclusions and Limitations

- DP and RNNs can indeed be married, if done creatively
  - biRNN summarizing input **x** and not output structure **y**
  - this allows efficient DP with exact search
  - combine with global learning (loss-augmented decoding)
- but exact DP is still too slow
  - future work: linear-time beam search DP with biRNNs
- what if we want strictly incremental parsing? no biRNN...
  - DP search could compensate for loss of lookahead

Conclusions and Limitations

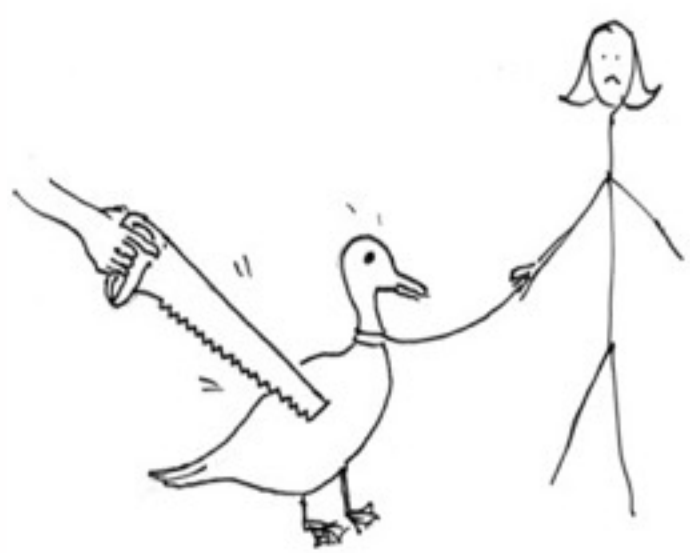
- DP and RNNs can indeed be married, if done creatively
  - biRNN summarizing input **x** and not output structure **y**
  - this allows efficient DP with exact search
  - combine with global learning (loss-augmented decoding)
- but exact DP is still too slow
  - future work: linear-time beam search DP with biRNNs
- what if we want strictly incremental parsing? no biRNN...
  - DP search could compensate for loss of lookahead
- what about translation? we do need to model **y** directly...

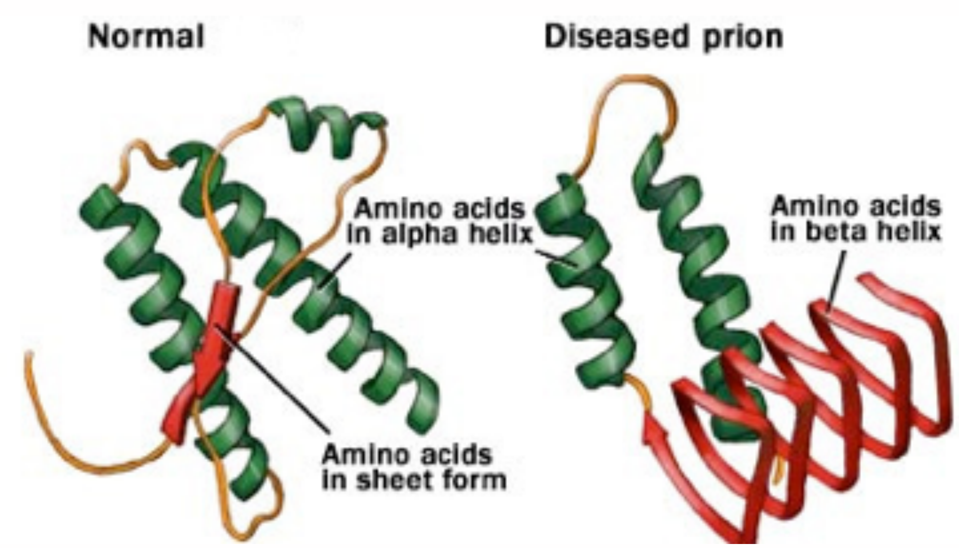


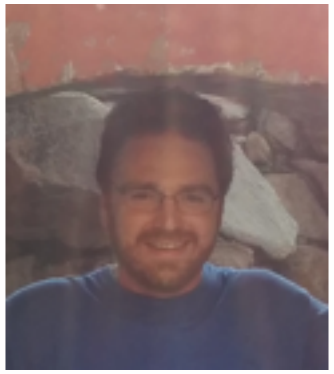
James Cross

非常 感谢！

fēi cháng gǎn xiè



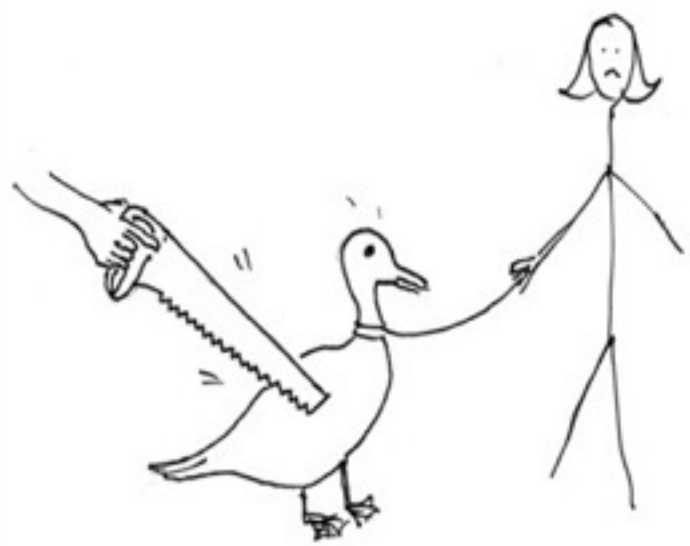


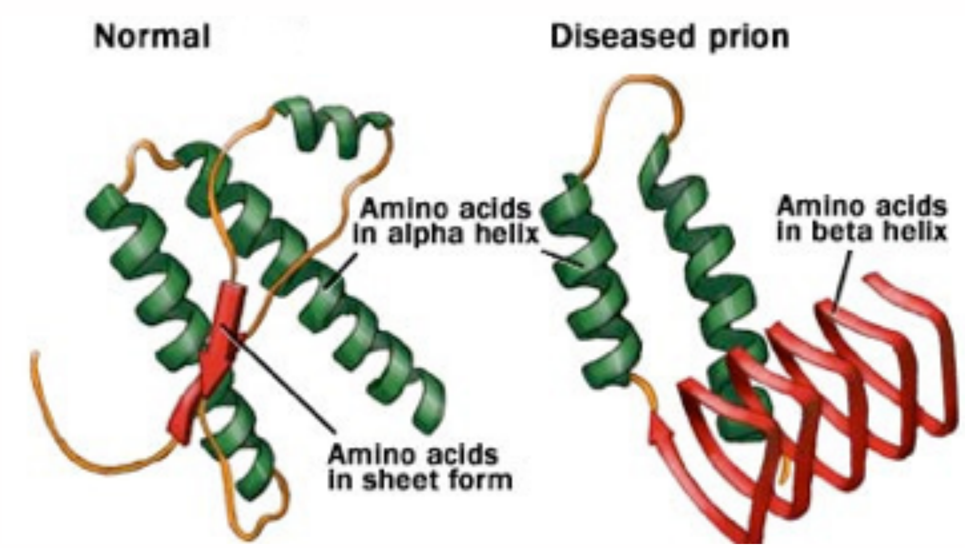


James Cross

非常 感谢！

fēi cháng gǎn xiè





Thank you very much !