Fast(er) Exact Decoding and Global Training for Transition-Based Dependency Parsing via a Minimal Feature Set

Tianze Shi* Liang Huang† Lillian Lee*

* Cornell University † Oregon State University

$O(n^3)$

Theoretical

Minimal

Feature Set

$O(n^3)$

~~$O(n^6)$~~

Practical

Short Version

- Transition-based dependency parsing has an exponentially-large search space
- $O(n^3)$ exact solutions exist 😃
- In practice, however, we needed rich features $\Longrightarrow O(n^6)$ 😟
- (This work) with bi-LSTMs, now we can do $O(n^3)$! 😃
- And we get state-of-the-art results

Short Version

- Transition-based dependency parsing has an exponentially-large search space
- $O(n^3)$ exact solutions exist 😃
- In practice, however, we needed rich features $\Longrightarrow O(n^6)$ 😟
- (This work) with bi-LSTMs, now we can do $O(n^3)$! 😃
- And we get state-of-the-art results

Short Version

- Transition-based dependency parsing has an exponentially-large search space
- $O(n^3)$ exact solutions exist 😃
- In practice, however, we needed rich features $\Longrightarrow$ $O(n^6)$ 😣
- (This work) with bi-LSTMs, now we can do $O(n^3)$! 😃
- And we get state-of-the-art results

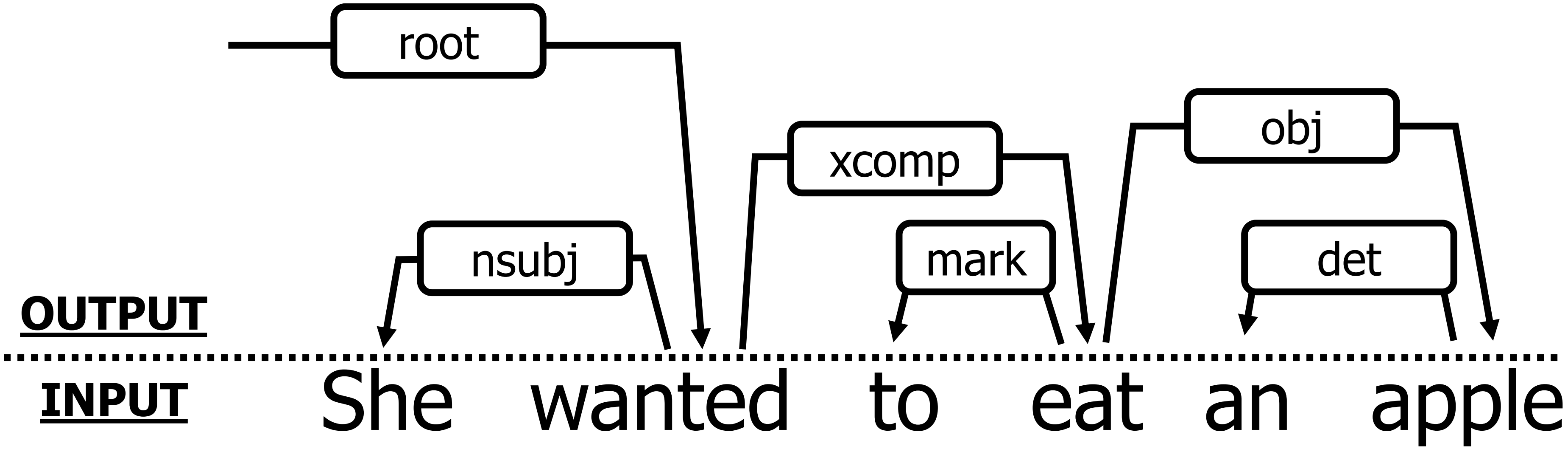
Short Version

- Transition-based dependency parsing has an exponentially-large search space
- $O(n^3)$ exact solutions exist 😀
- In practice, however, we needed rich features $\Longrightarrow O(n^6)$ 😟
- (This work) with bi-LSTMs, now we can do $O(n^3)$! 😀
- And we get state-of-the-art results

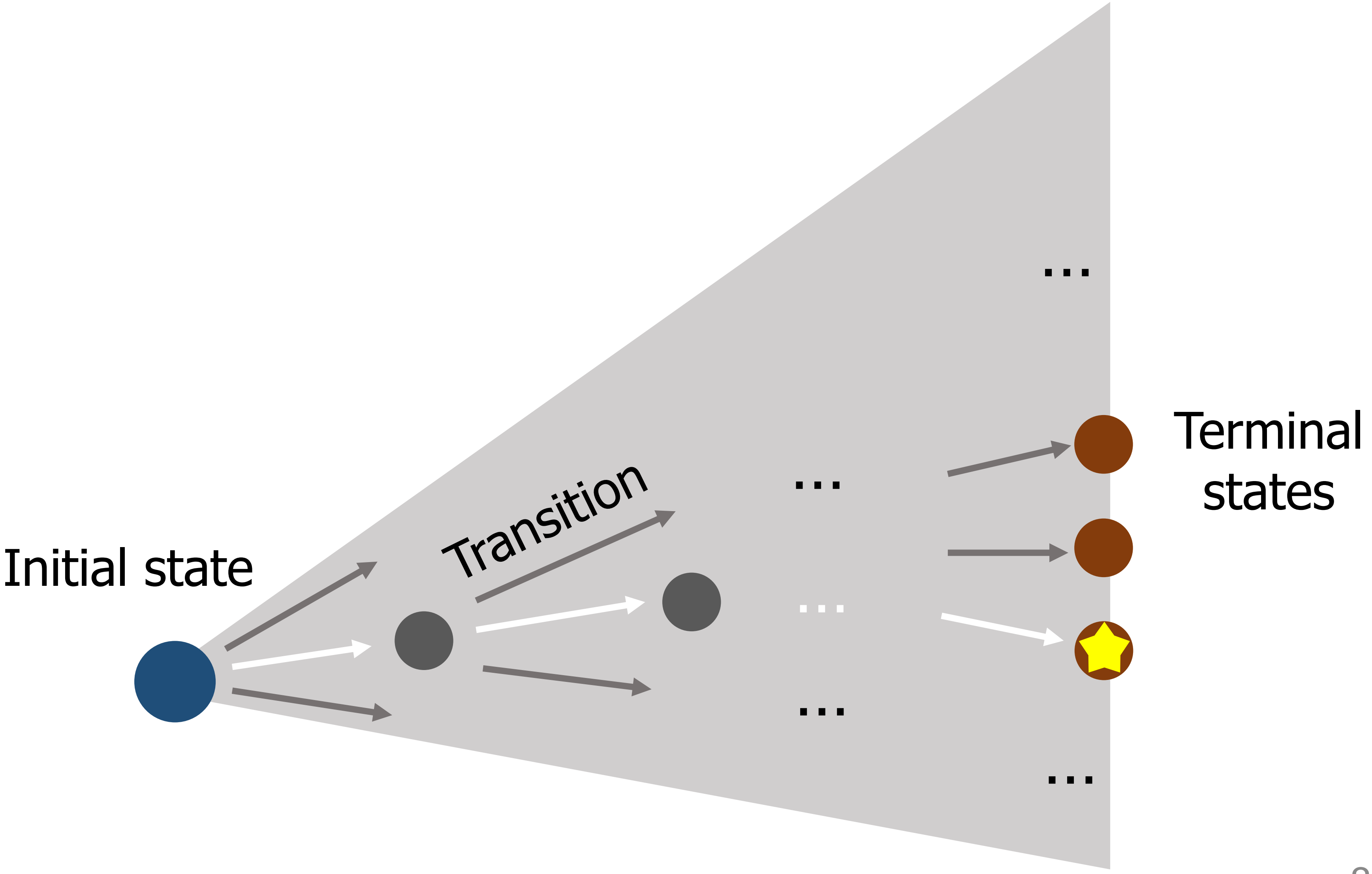
Short Version

- Transition-based dependency parsing has an exponentially-large search space
- $O(n^3)$ exact solutions exist 😃
- In practice, however, we needed rich features $\Longrightarrow O(n^6)$ 😔
- (This work) with bi-LSTMs, now we can do $O(n^3)$! 😃
- And we get state-of-the-art results

Dependency Parsing



Transition-based Dependency Parsing

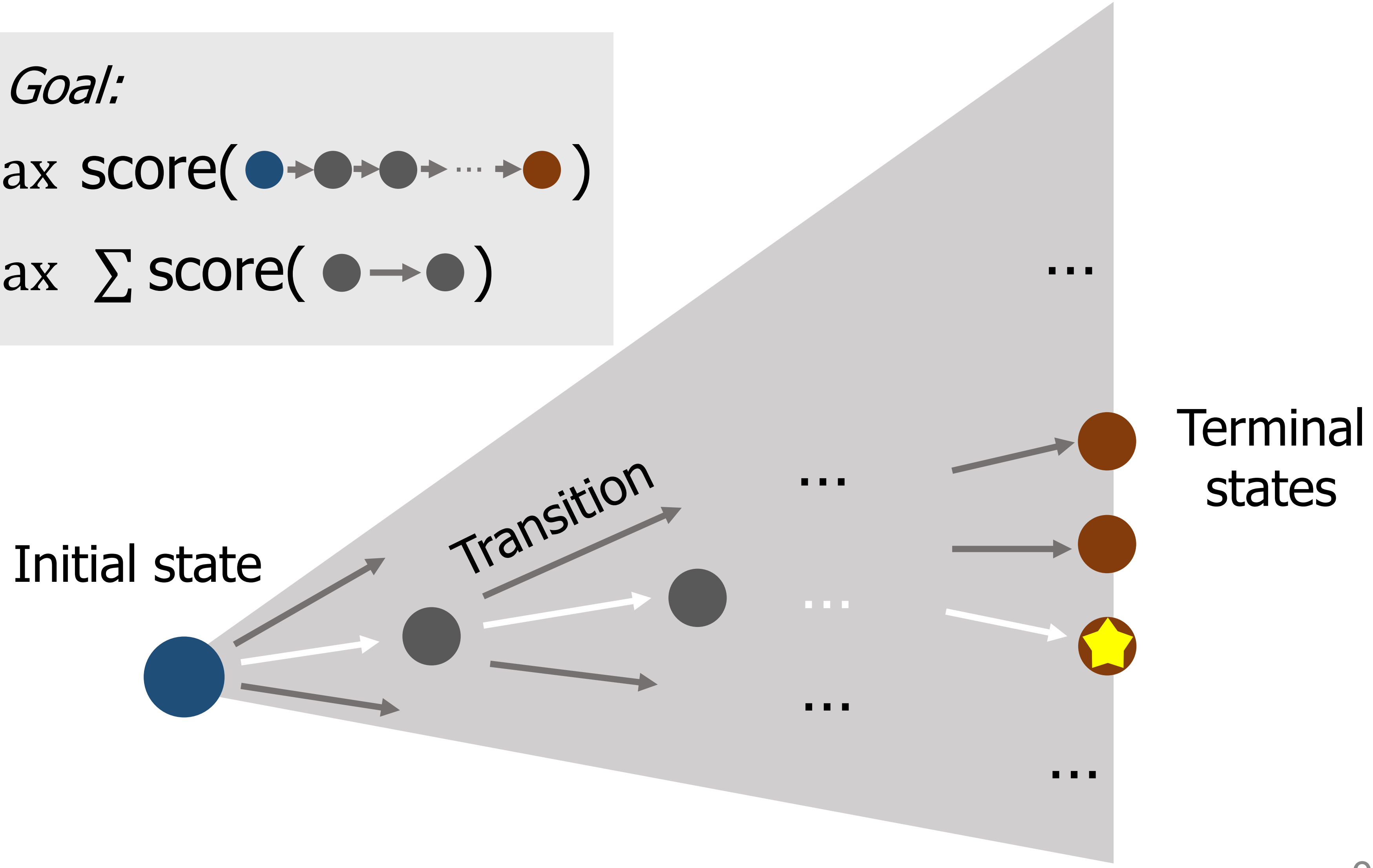


Transition-based Dependency Parsing

Goal:

max score(● → ● → ● → ⋯ → ●)

= max ∑ score(● → ●)



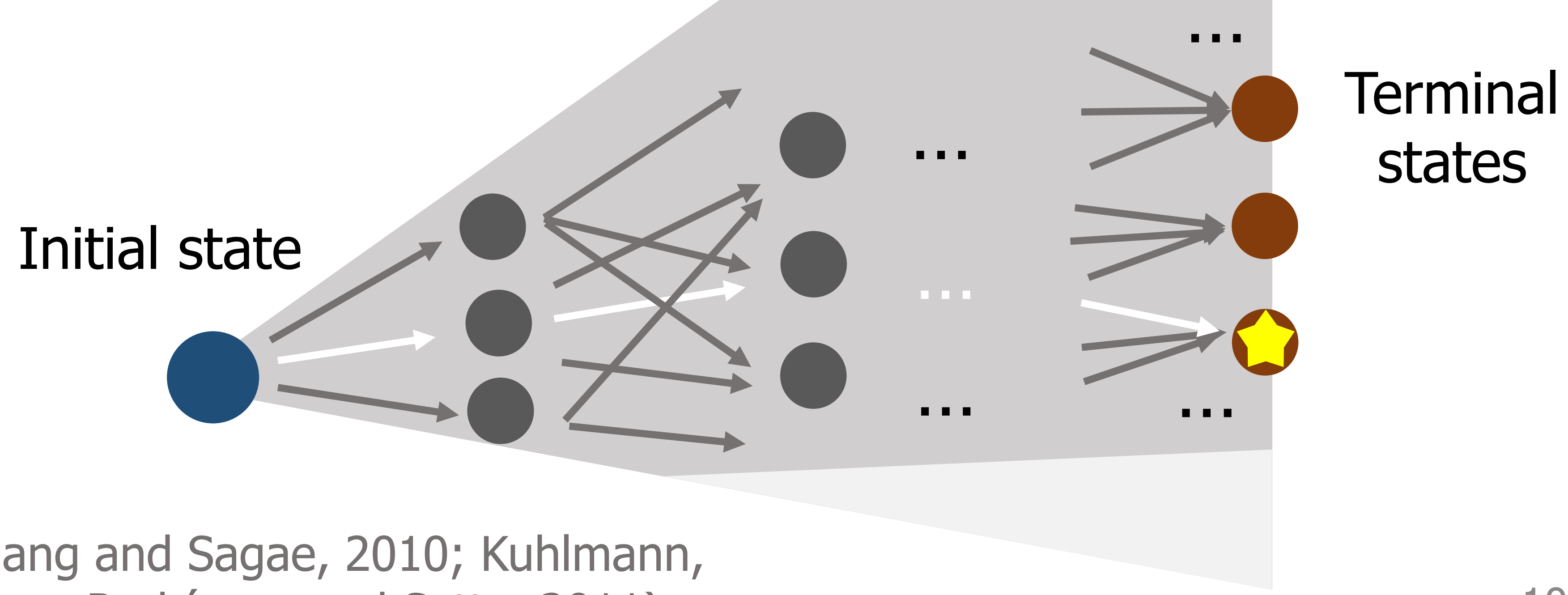
Exact Decoding with Dynamic Programming

Goal:

max score(● → ● → ● → ⋯ → ●)

= max ∑ score(● → ●)

Exponential to polynomial



(Huang and Sagae, 2010; Kuhlmann, Gómez-Rodríguez and Satta, 2011)

Transition Systems

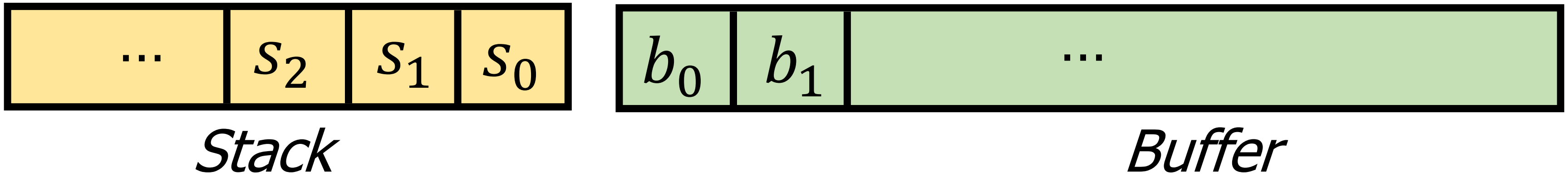
	DP Complexity	# Action Types
Arc-standard	$O(n^4)$	**3**
Arc-eager	$\boldsymbol{O(n^3)}$	4
Arc-hybrid	$\boldsymbol{O(n^3)}$	**3**

In our paper

Presentational convenience

Arc-hybrid Transition System

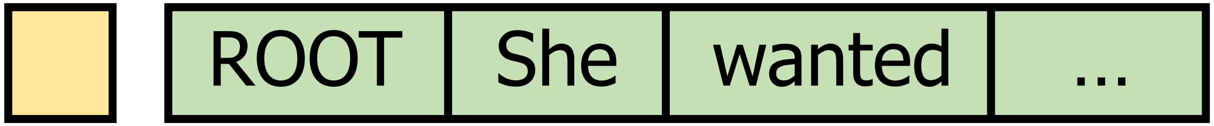
State



…	s_2	s_1	s_0	b_0	b_1	…	

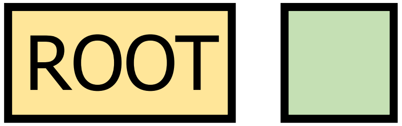
Stack *Buffer*

Initial State



	ROOT	She	wanted	…

Terminal State

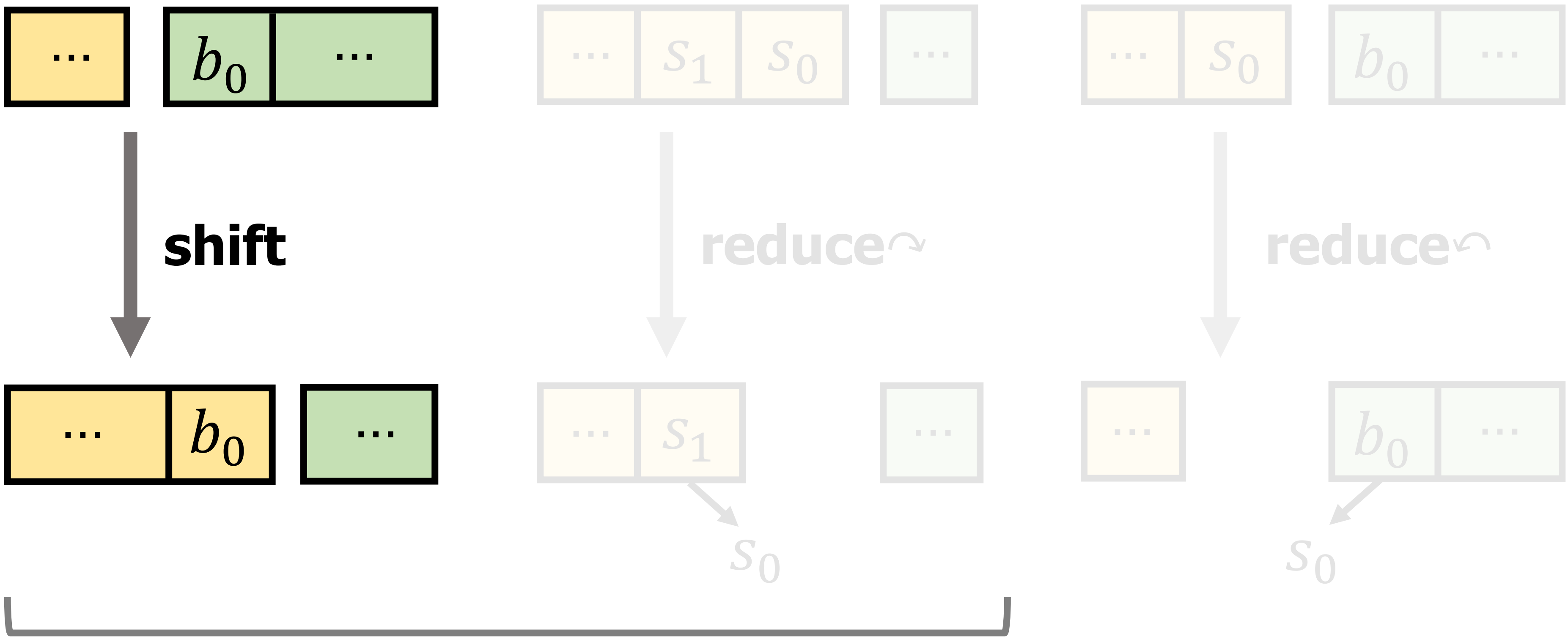


ROOT	

(Yamada and Matsumoto, 2003)
(Gómez-Rodríguez et al., 2008)
(Kuhlmann et al., 2011)

Arc-hybrid Transition System

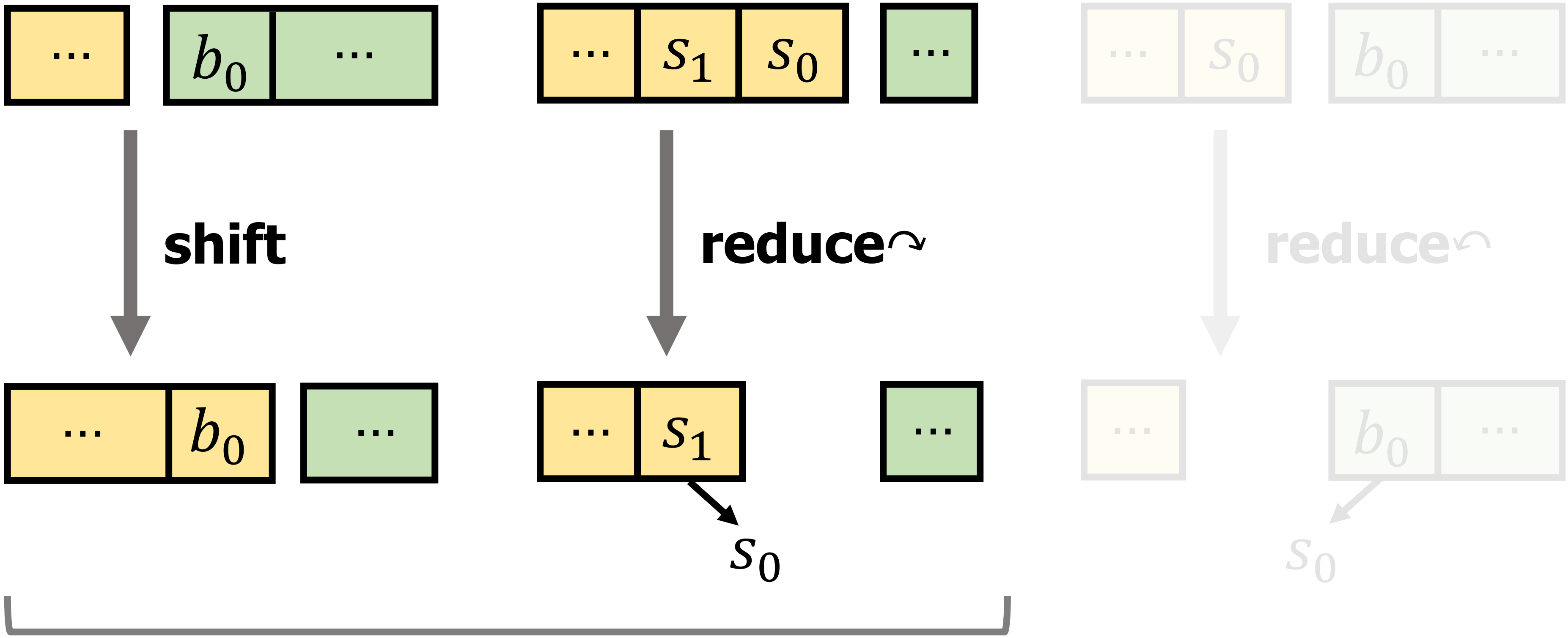
Transitions



Same as arc-standard

Arc-hybrid Transition System

Transitions



shift

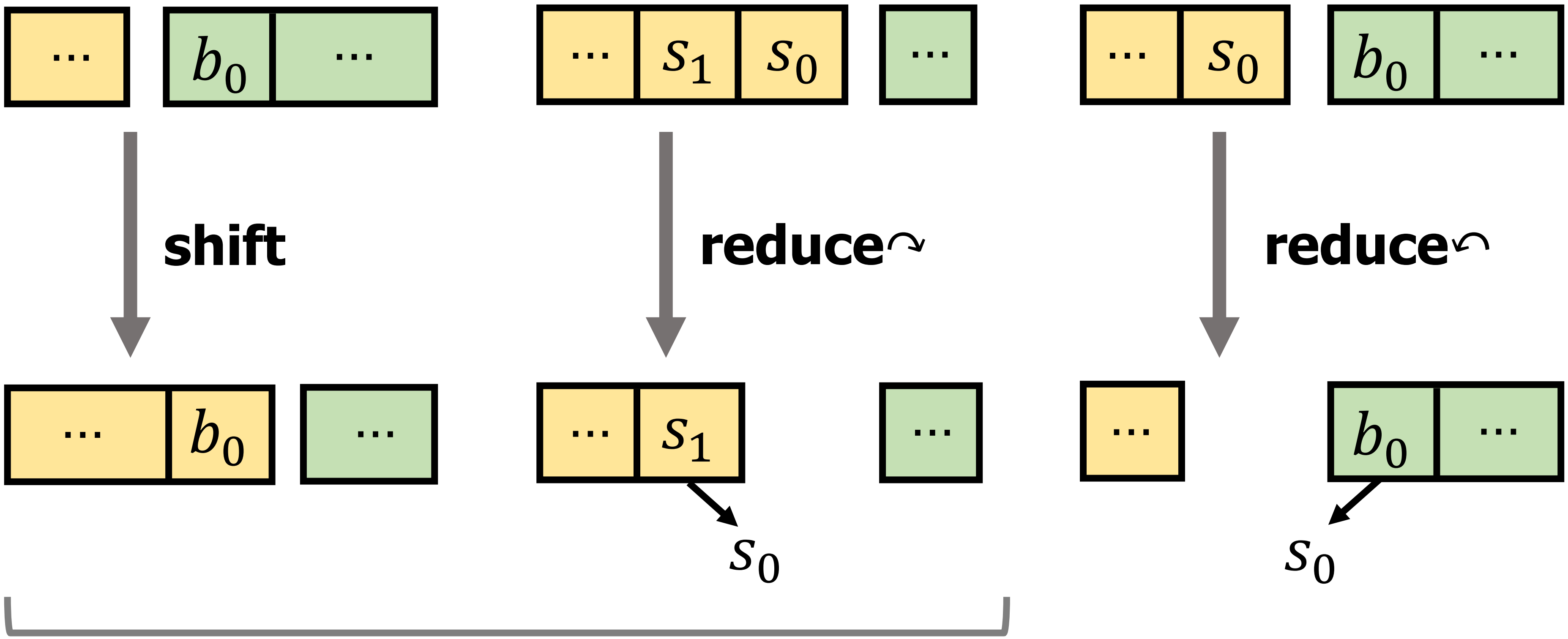
reduce↷

reduce↶

Same as arc-standard

Arc-hybrid Transition System

Transitions



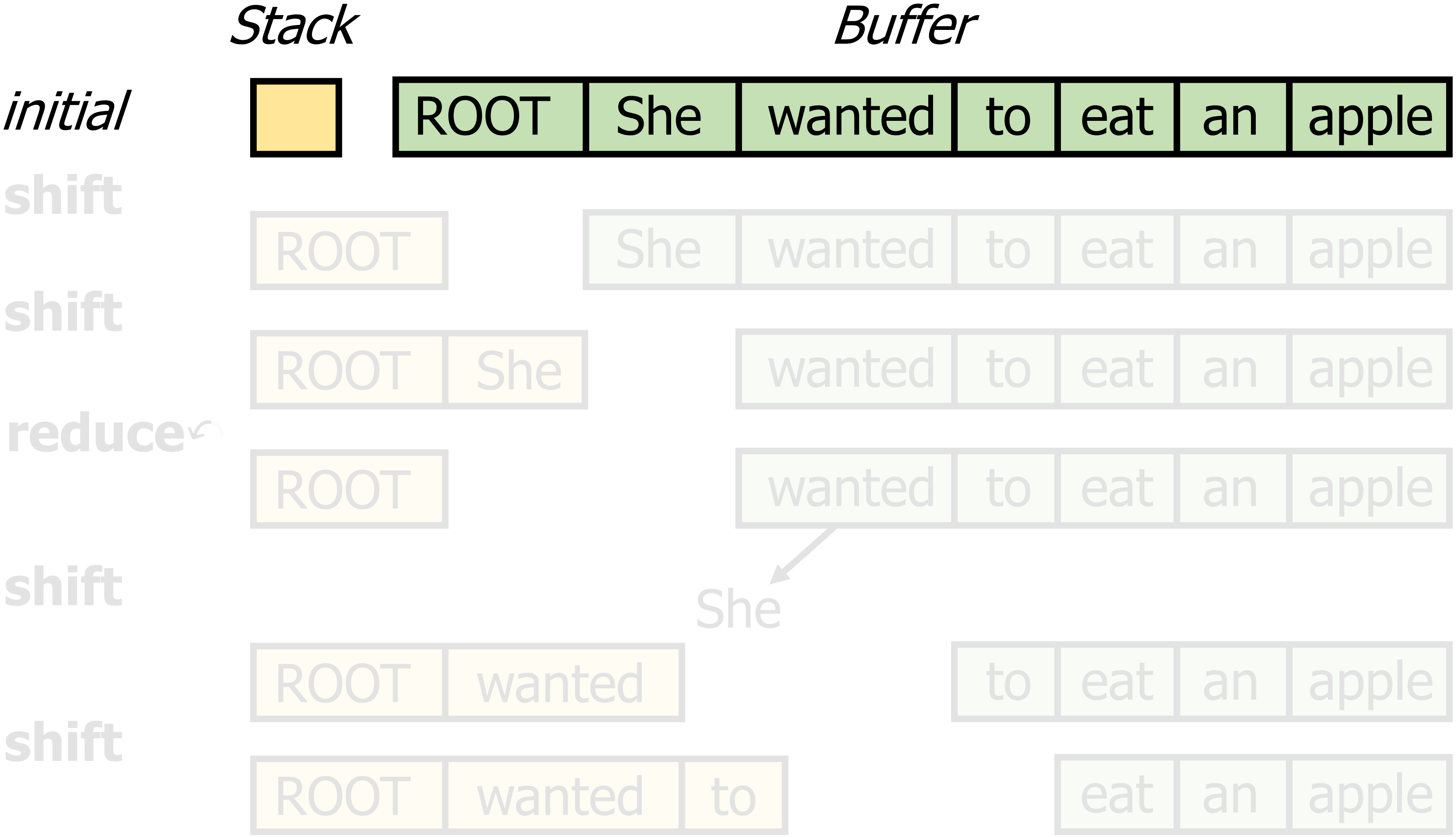
shift

reduce↷

reduce↶

Same as arc-standard

Arc-hybrid Transition System

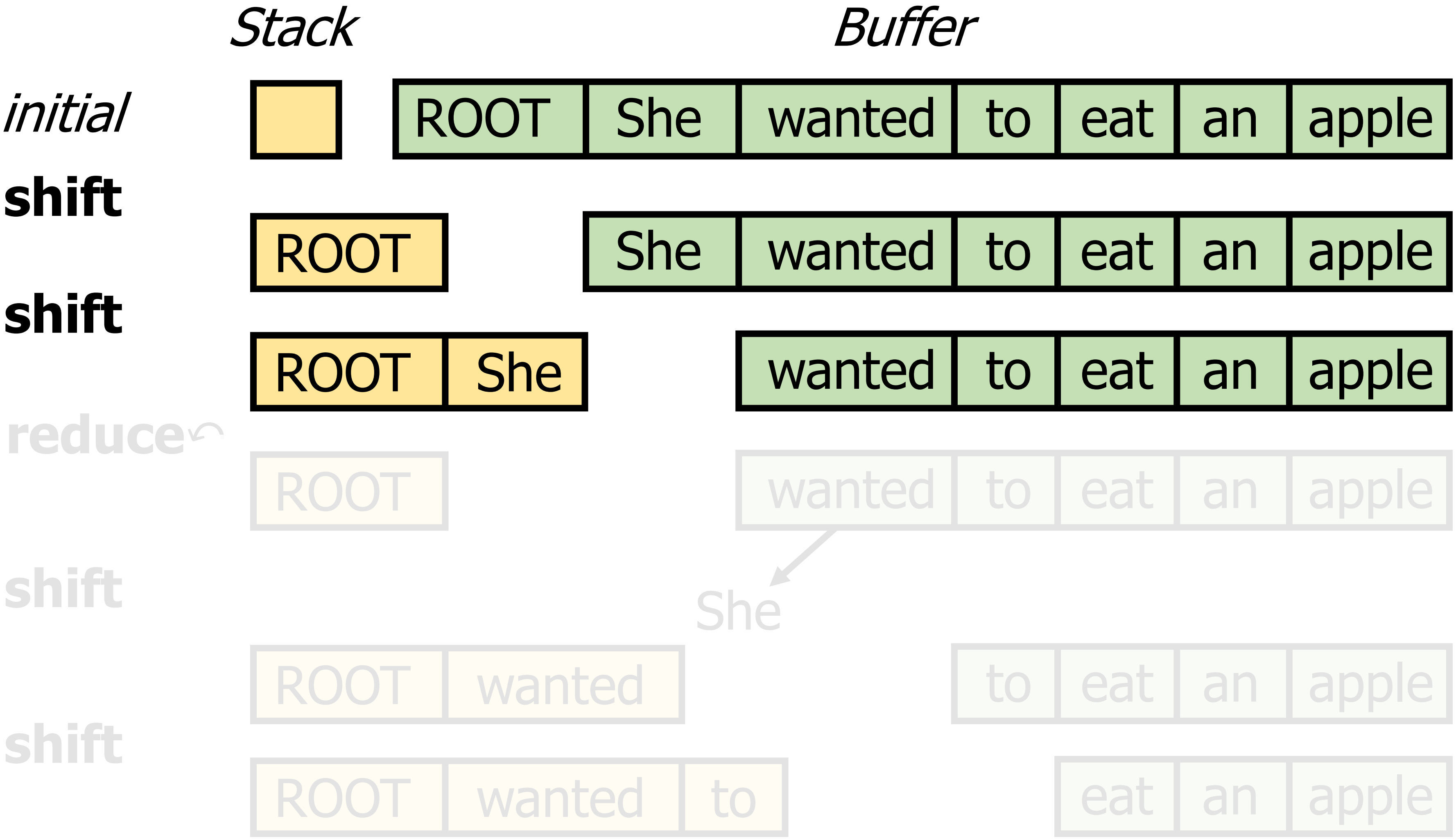


	Stack	*Buffer*
initial	□	ROOT She wanted to eat an apple
shift	ROOT	She wanted to eat an apple
shift	ROOT She	wanted to eat an apple
reduce↶	ROOT	wanted to eat an apple
shift	ROOT wanted	to eat an apple
shift	ROOT wanted to	eat an apple

Arc-hybrid Transition System

	Stack	*Buffer*
initial		ROOT | She | wanted | to | eat | an | apple
shift	ROOT	She | wanted | to | eat | an | apple
shift	ROOT | She	wanted | to | eat | an | apple
reduce↶	ROOT	wanted | to | eat | an | apple
		She
shift	ROOT | wanted	to | eat | an | apple
shift	ROOT | wanted | to	eat | an | apple

Arc-hybrid Transition System



Stack *Buffer*

initial | □ | ROOT She wanted to eat an apple

shift | ROOT | She wanted to eat an apple

shift | ROOT She | wanted to eat an apple

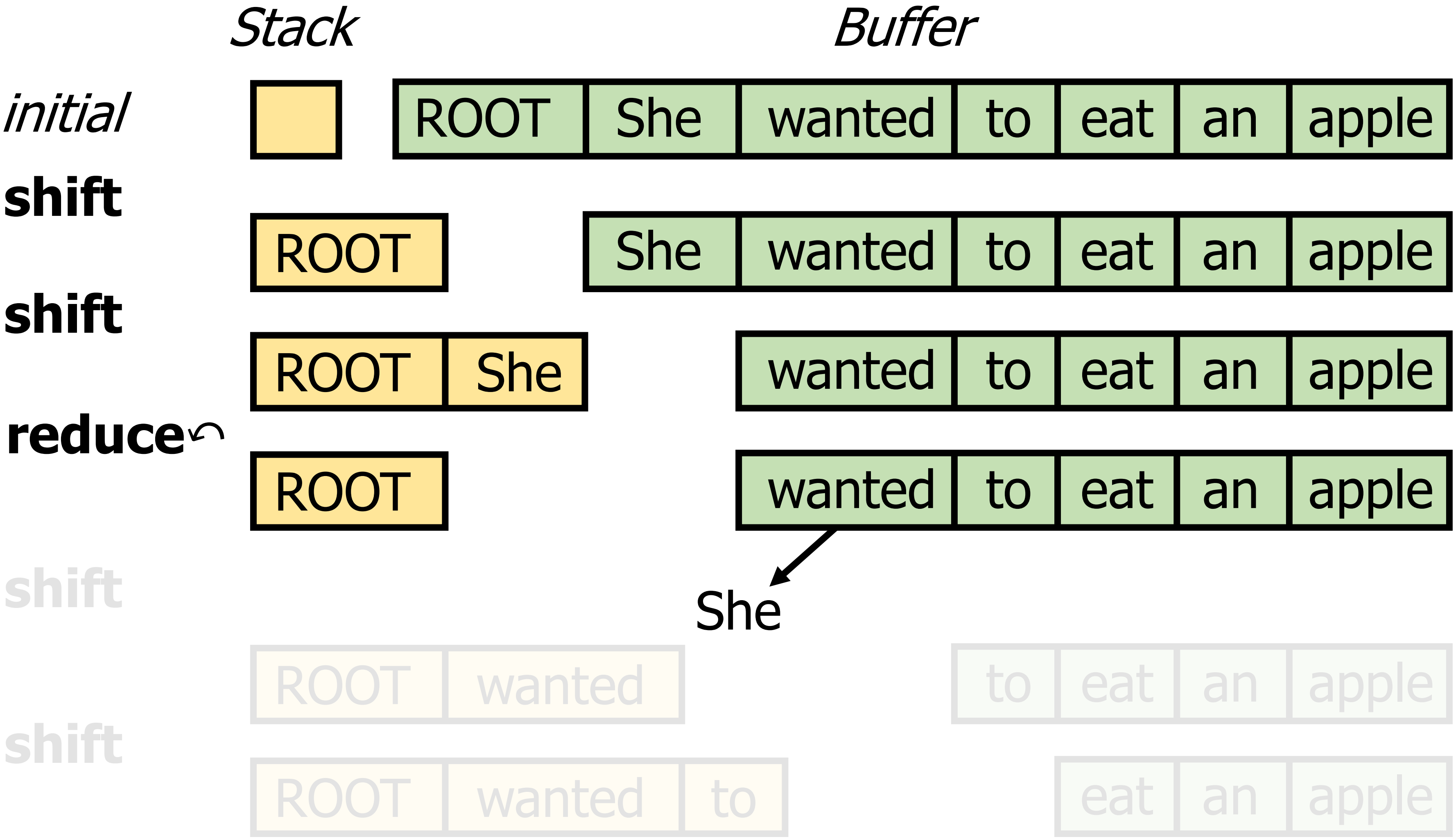
reduce↶ | ROOT | wanted to eat an apple

She

shift | ROOT wanted | to eat an apple

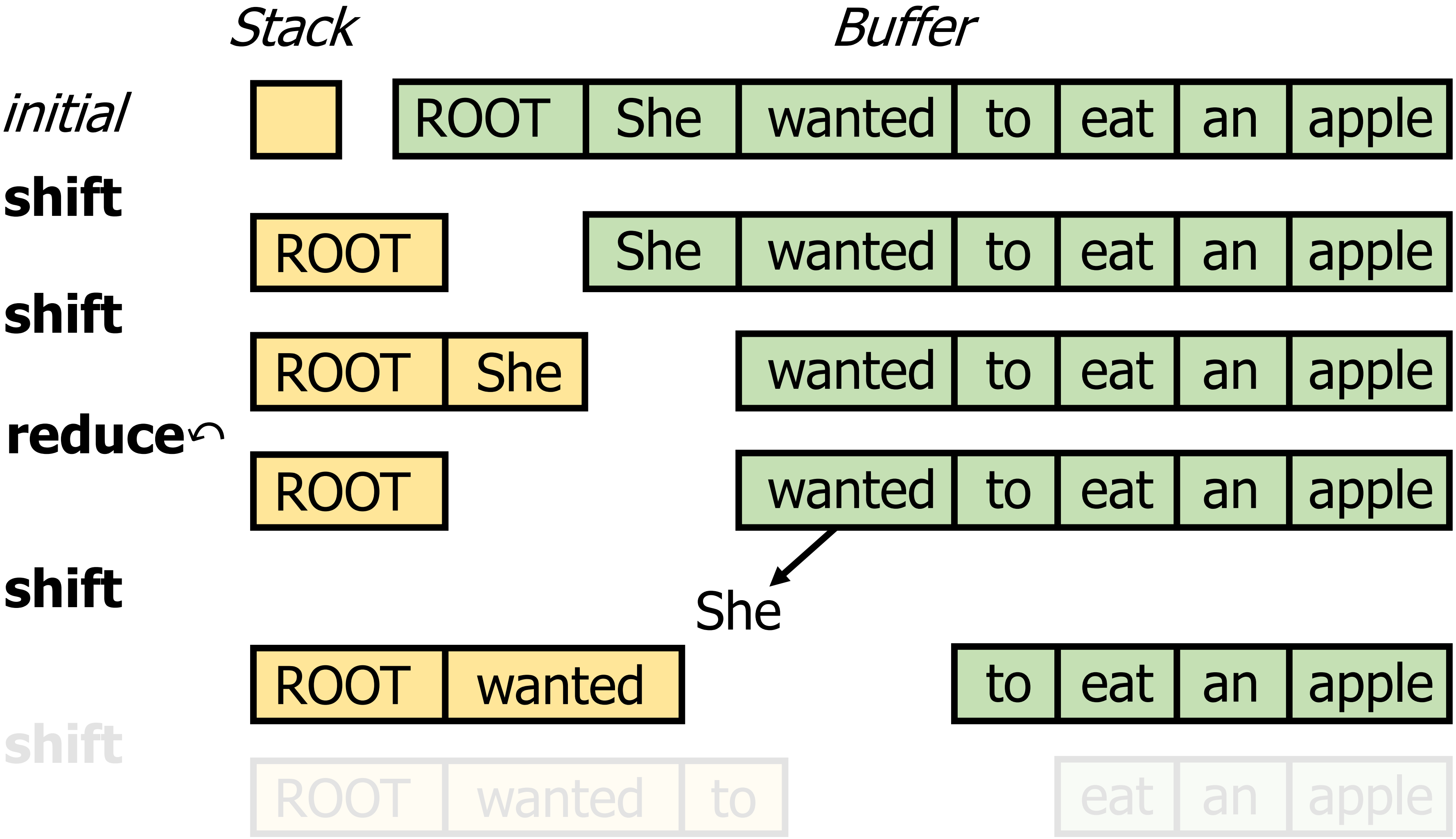
shift | ROOT wanted to | eat an apple

Arc-hybrid Transition System



	Stack	*Buffer*
initial		ROOT She wanted to eat an apple
shift	ROOT	She wanted to eat an apple
shift	ROOT She	wanted to eat an apple
reduce↶	ROOT	wanted to eat an apple
shift	ROOT wanted	to eat an apple
shift	ROOT wanted to	eat an apple

Arc-hybrid Transition System



Stack *Buffer*

initial | | ROOT She wanted to eat an apple

shift

ROOT | She wanted to eat an apple

shift

ROOT She | wanted to eat an apple

reduce↶

ROOT | wanted to eat an apple

She

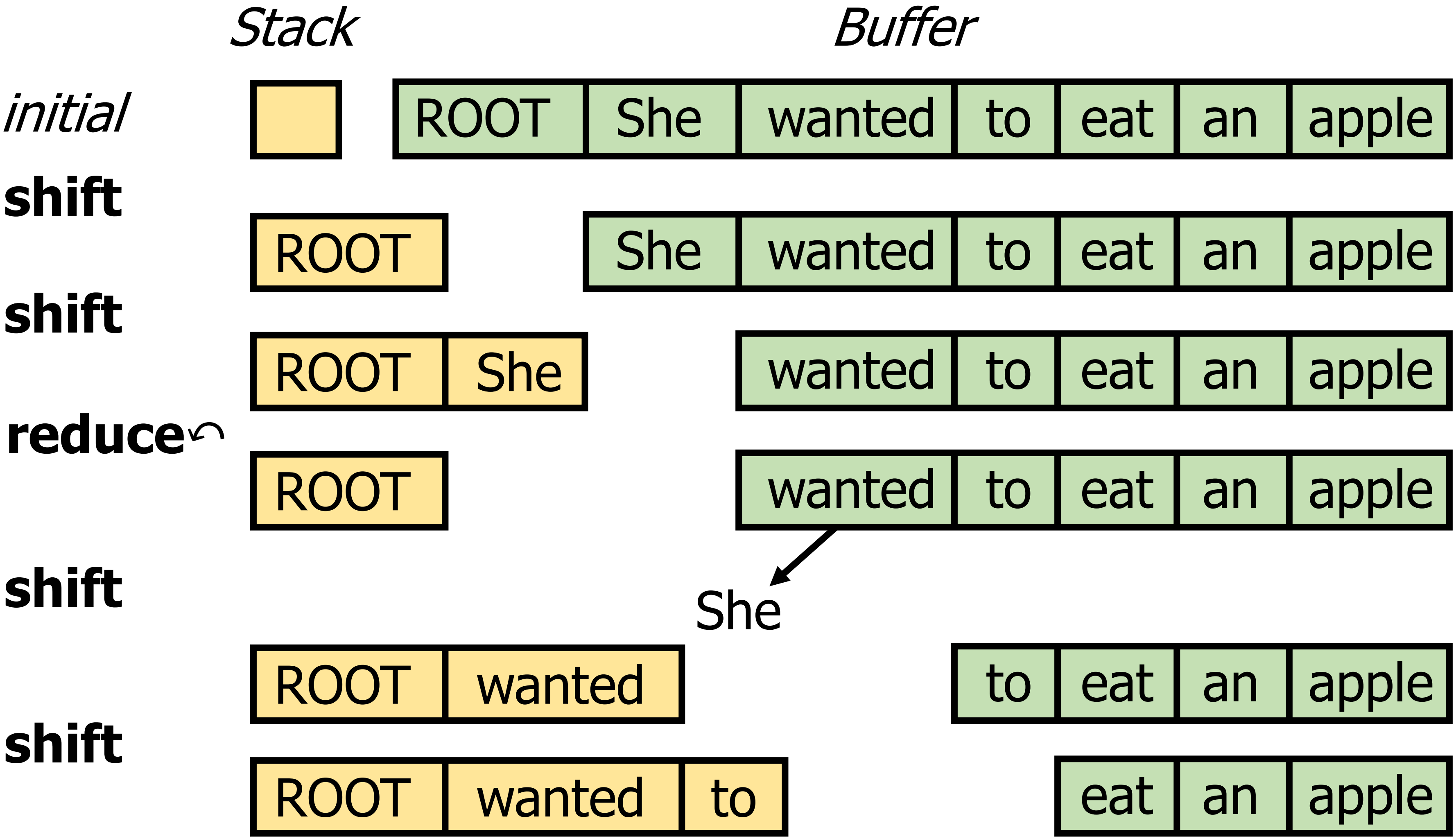
shift

ROOT wanted | to eat an apple

shift

ROOT wanted to | eat an apple

Arc-hybrid Transition System



Stack *Buffer*

initial — Stack: (empty) — Buffer: ROOT She wanted to eat an apple

shift

Stack: ROOT — Buffer: She wanted to eat an apple

shift

Stack: ROOT She — Buffer: wanted to eat an apple

reduce↶

Stack: ROOT — Buffer: wanted to eat an apple

She

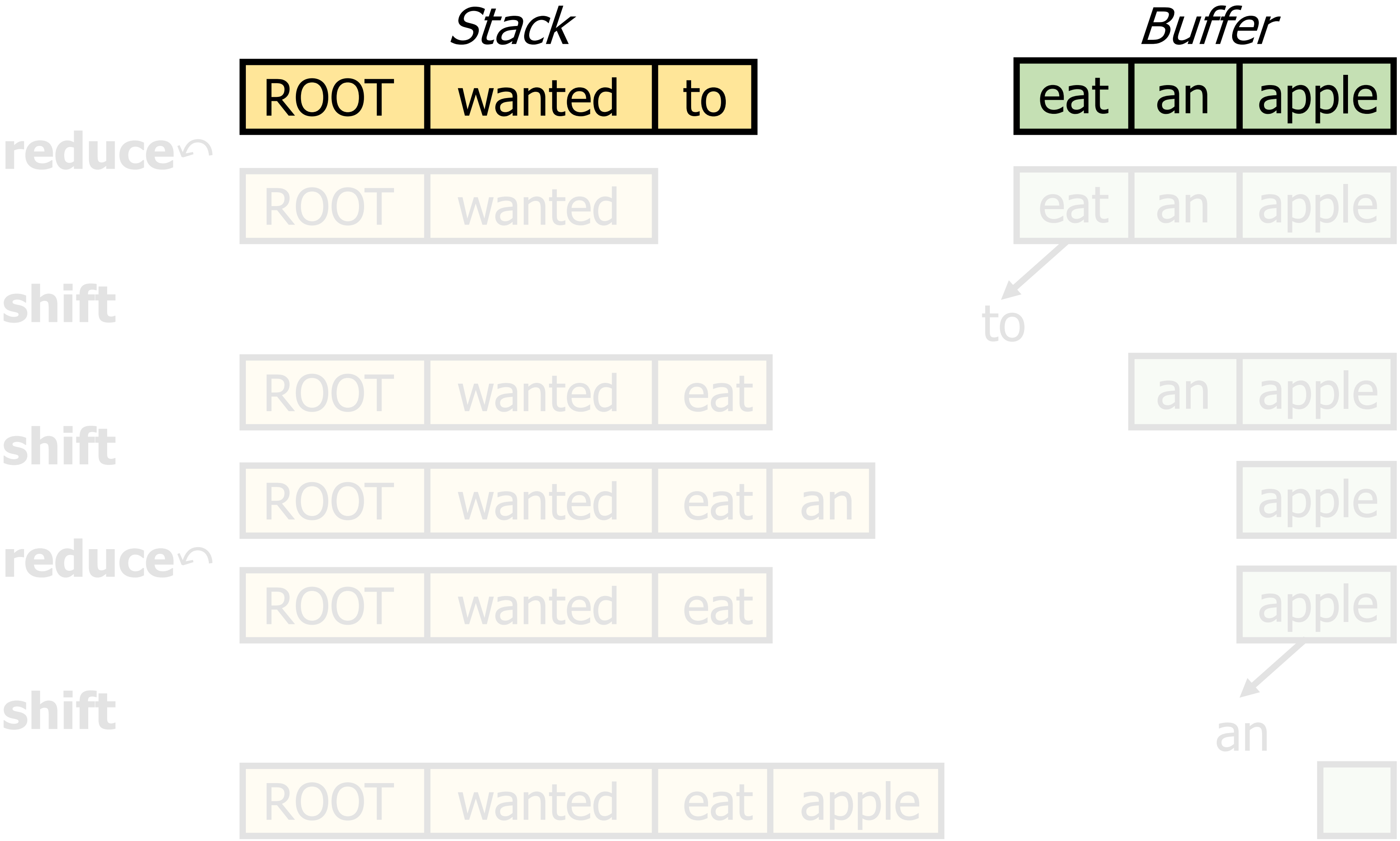
shift

Stack: ROOT wanted — Buffer: to eat an apple

shift

Stack: ROOT wanted to — Buffer: eat an apple

Arc-hybrid Transition System



Stack *Buffer*

ROOT wanted to | eat an apple

reduce↶

ROOT wanted | eat an apple

to

shift

ROOT wanted eat | an apple

shift

ROOT wanted eat an | apple

reduce↶

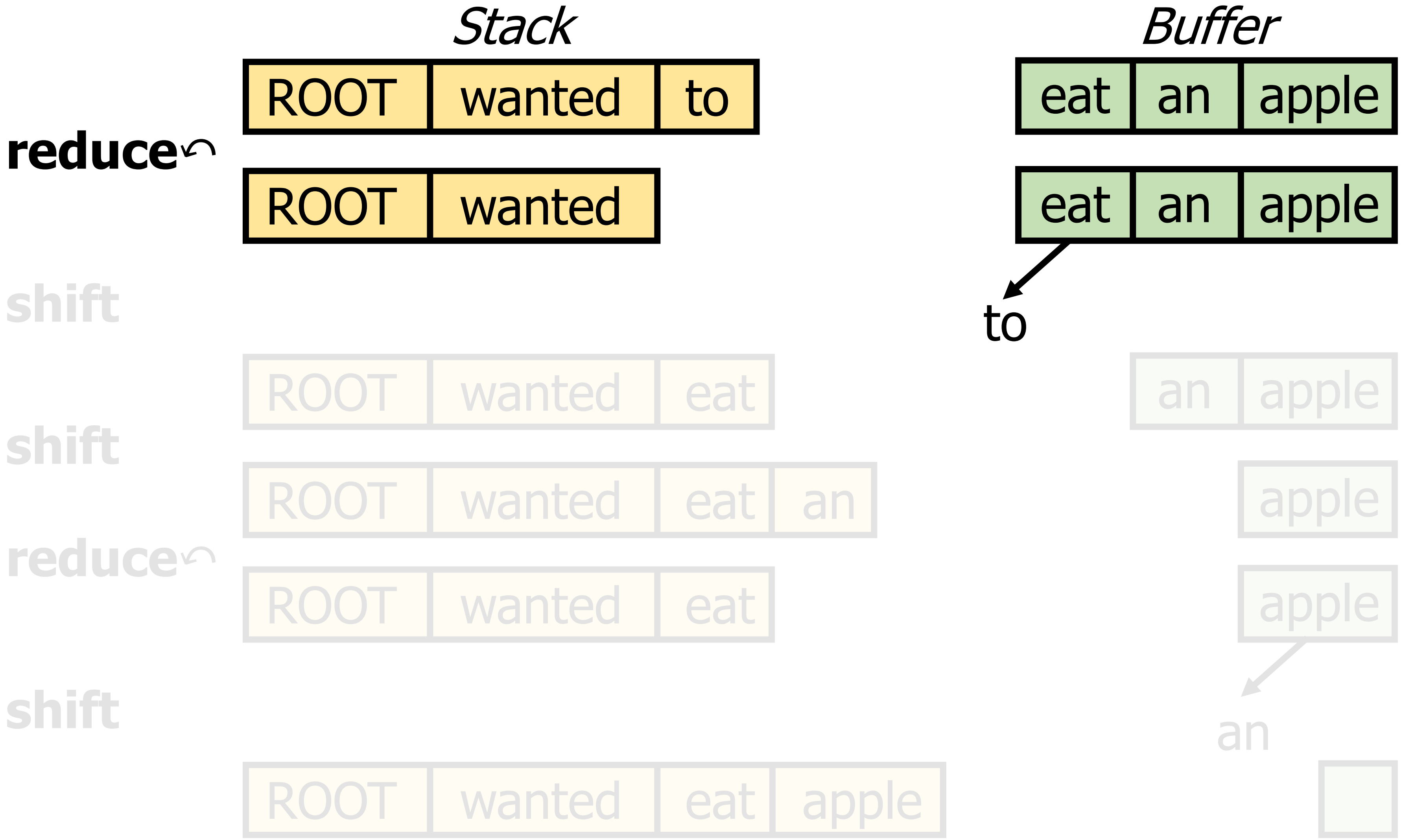
ROOT wanted eat | apple

an

shift

ROOT wanted eat apple |

Arc-hybrid Transition System



Stack

Buffer

ROOT wanted to | eat an apple

reduce↶

ROOT wanted | eat an apple

to

shift

ROOT wanted eat | an apple

shift

ROOT wanted eat an | apple

reduce↶

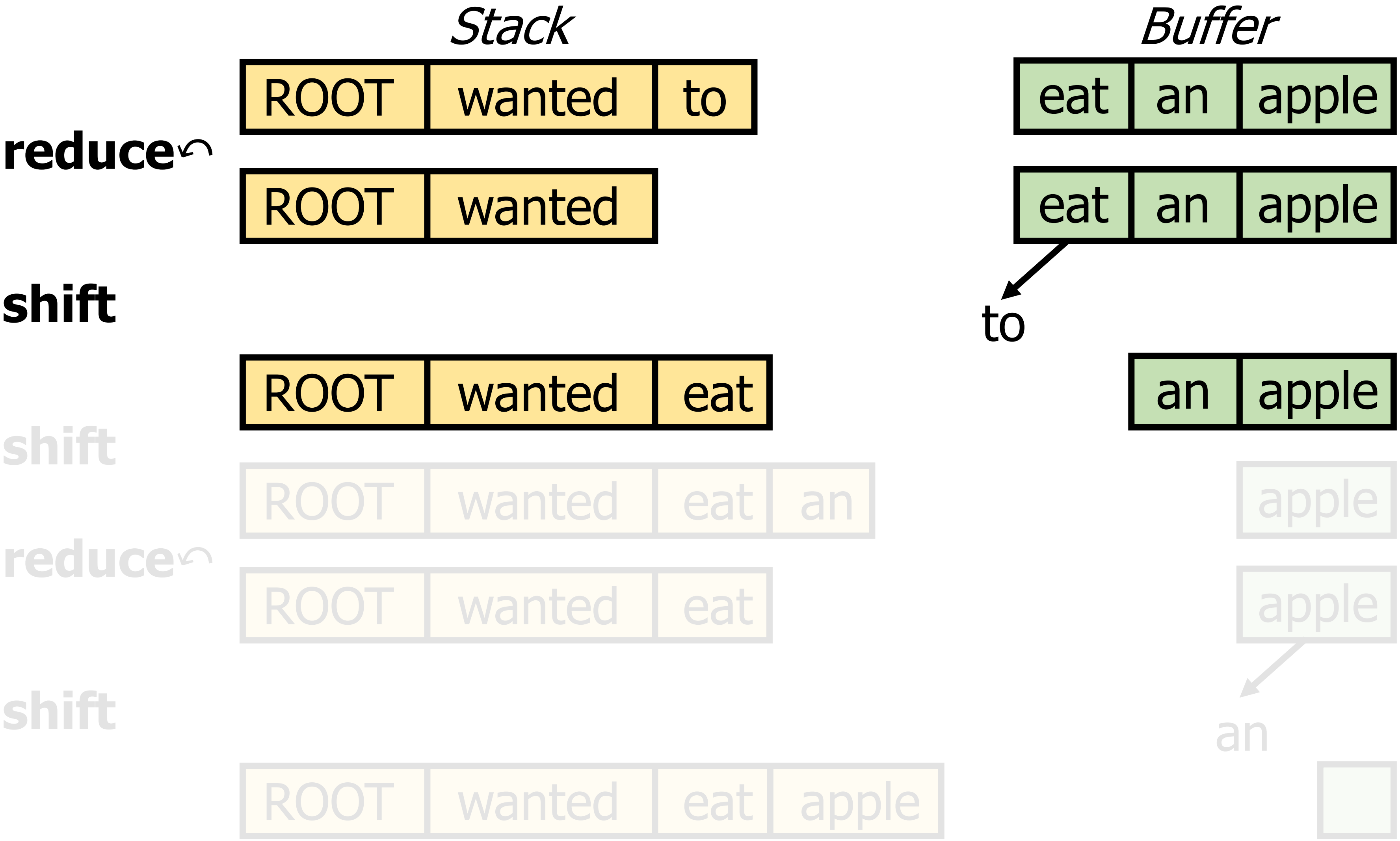
ROOT wanted eat | apple

an

shift

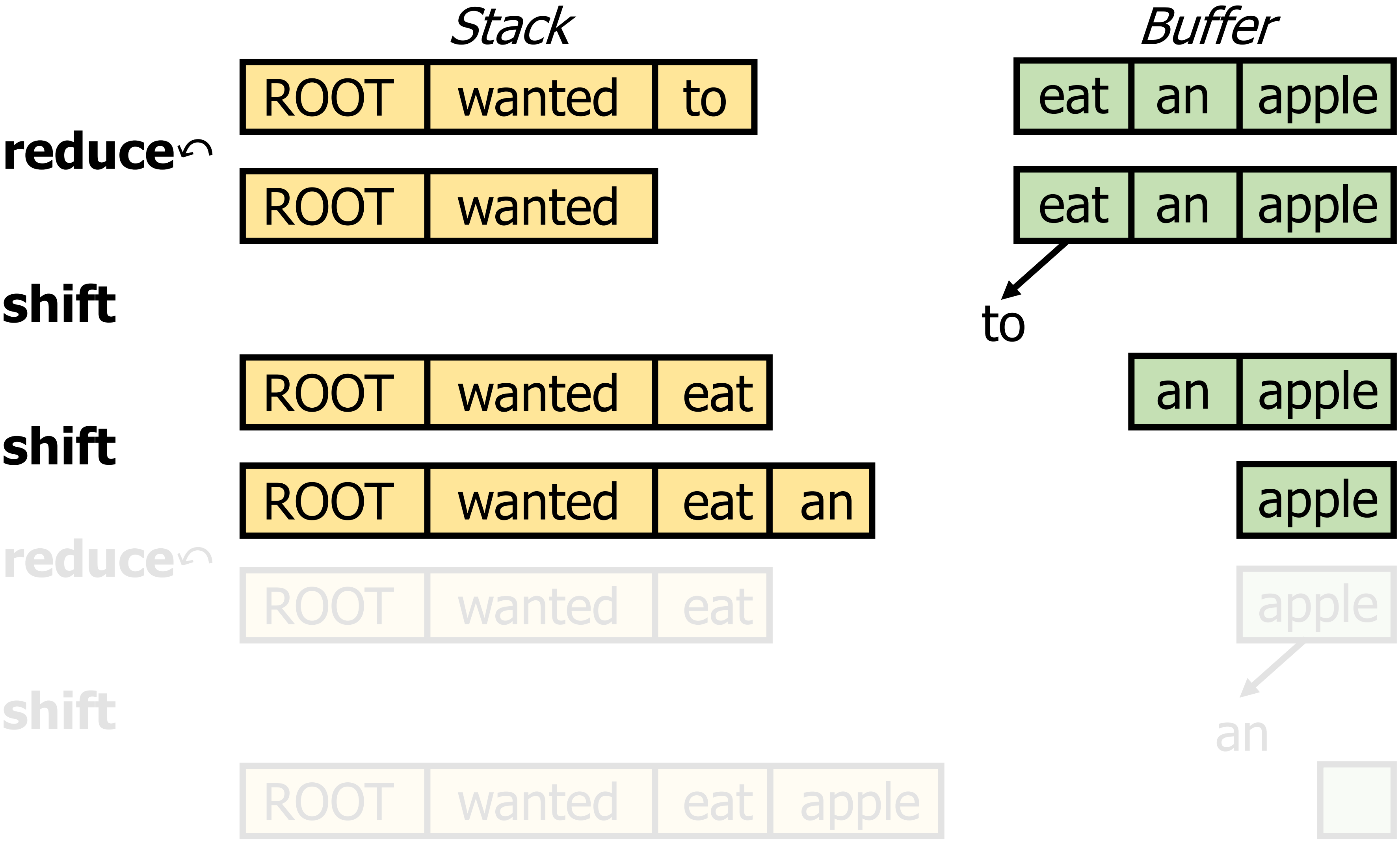
ROOT wanted eat apple |

Arc-hybrid Transition System



	Stack	*Buffer*
	ROOT | wanted | to	eat | an | apple
reduce↶	ROOT | wanted	eat | an | apple
shift		to
	ROOT | wanted | eat	an | apple
shift	ROOT | wanted | eat | an	apple
reduce↶	ROOT | wanted | eat	apple
shift		an
	ROOT | wanted | eat | apple	

Arc-hybrid Transition System



Stack *Buffer*

ROOT | wanted | to   eat | an | apple

reduce↶

ROOT | wanted   eat | an | apple

shift

to

ROOT | wanted | eat   an | apple

shift

ROOT | wanted | eat | an   apple

reduce↶

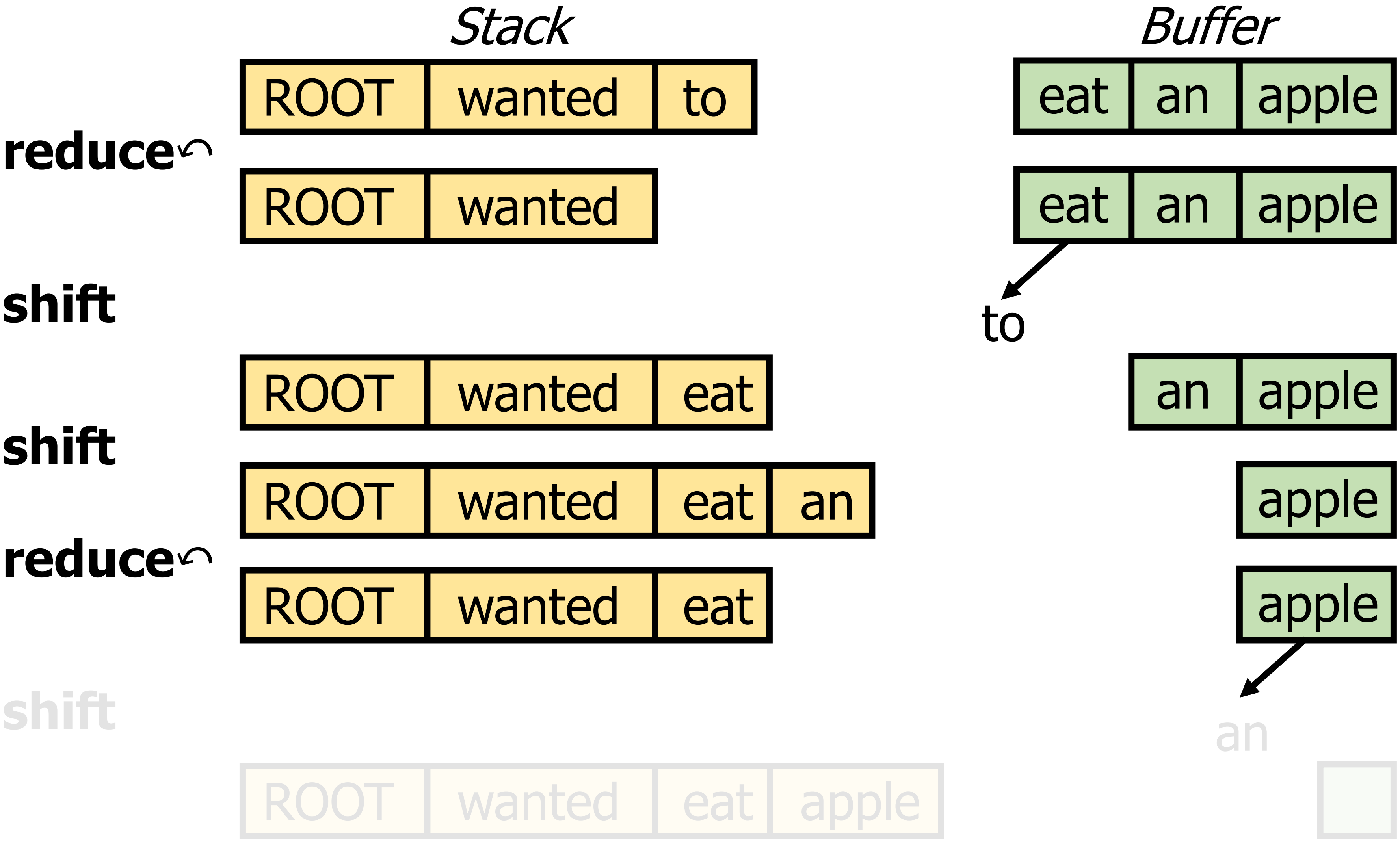
ROOT | wanted | eat   apple

shift

an

ROOT | wanted | eat | apple

Arc-hybrid Transition System



Stack

Buffer

ROOT wanted to | eat an apple

reduce↶

ROOT wanted | eat an apple

to

shift

ROOT wanted eat | an apple

shift

ROOT wanted eat an | apple

reduce↶

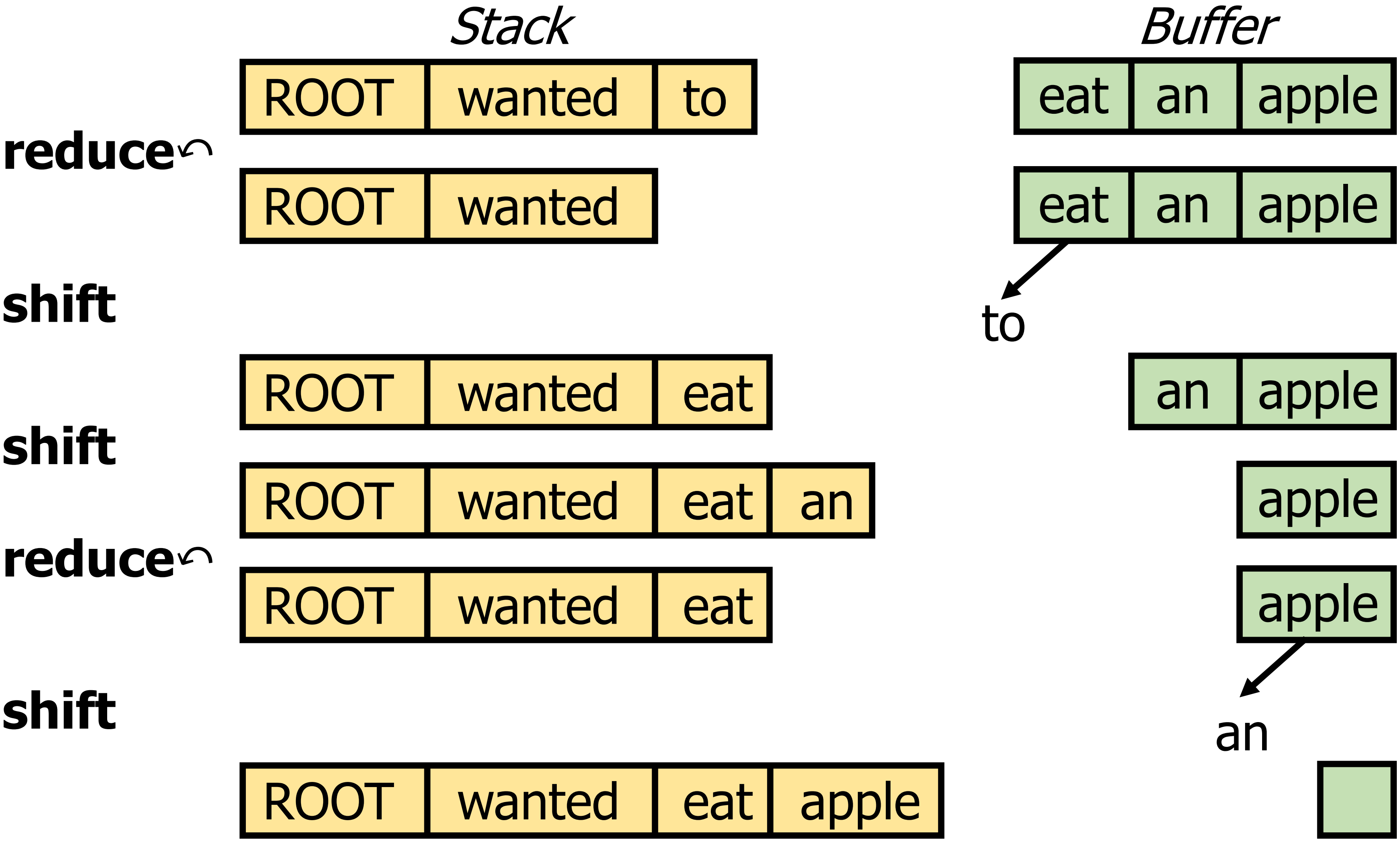
ROOT wanted eat | apple

an

shift

ROOT wanted eat apple |

Arc-hybrid Transition System



Stack

Buffer

ROOT | wanted | to

eat | an | apple

reduce↶

ROOT | wanted

eat | an | apple

to

shift

ROOT | wanted | eat

an | apple

shift

ROOT | wanted | eat | an

apple

reduce↶

ROOT | wanted | eat

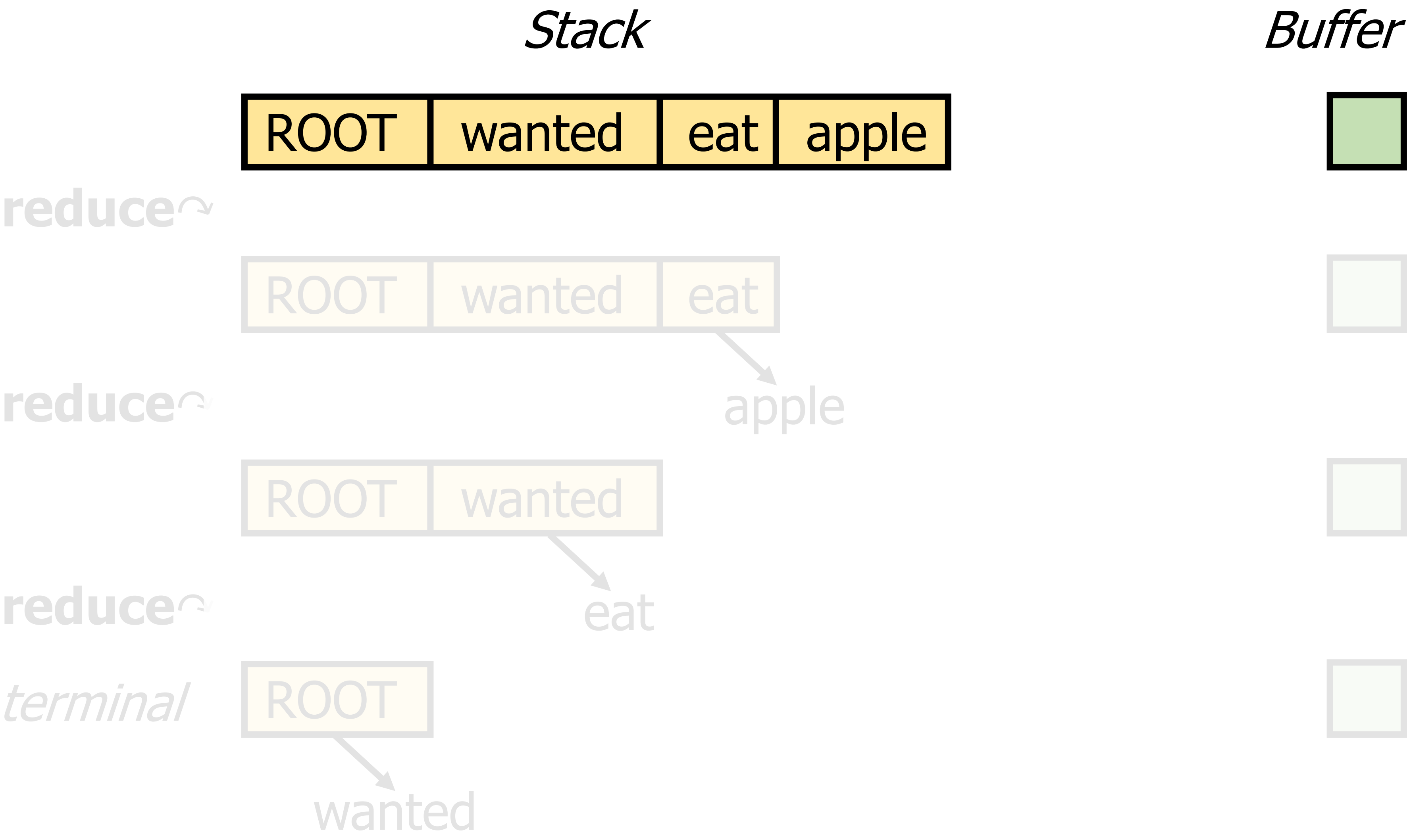
apple

an

shift

ROOT | wanted | eat | apple

Arc-hybrid Transition System



Stack *Buffer*

ROOT | wanted | eat | apple

reduce↷

ROOT | wanted | eat

apple

reduce↷

ROOT | wanted

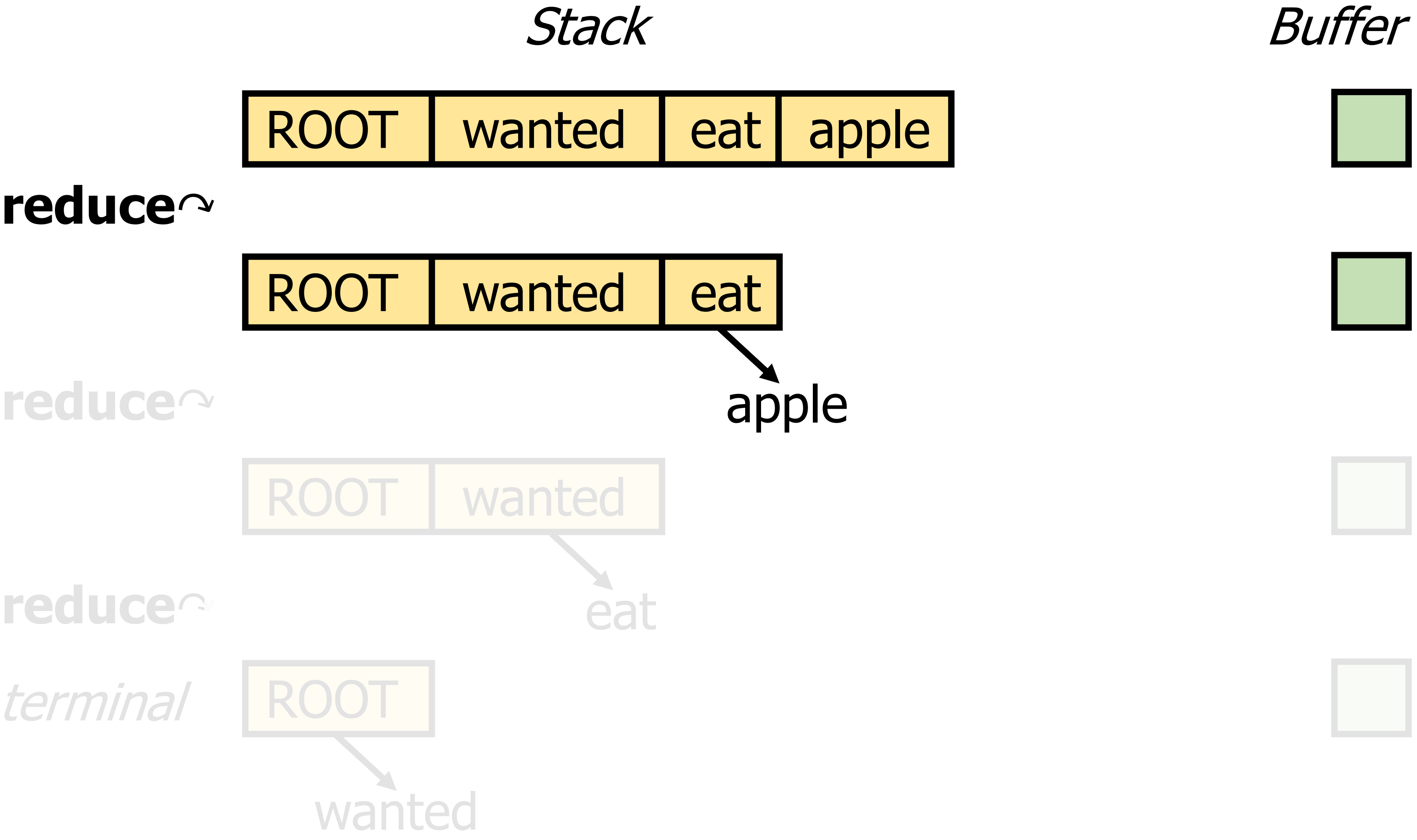
eat

reduce↷

terminal ROOT

wanted

Arc-hybrid Transition System



Stack

Buffer

ROOT | wanted | eat | apple

reduce↷

ROOT | wanted | eat

apple

reduce↷

ROOT | wanted

eat

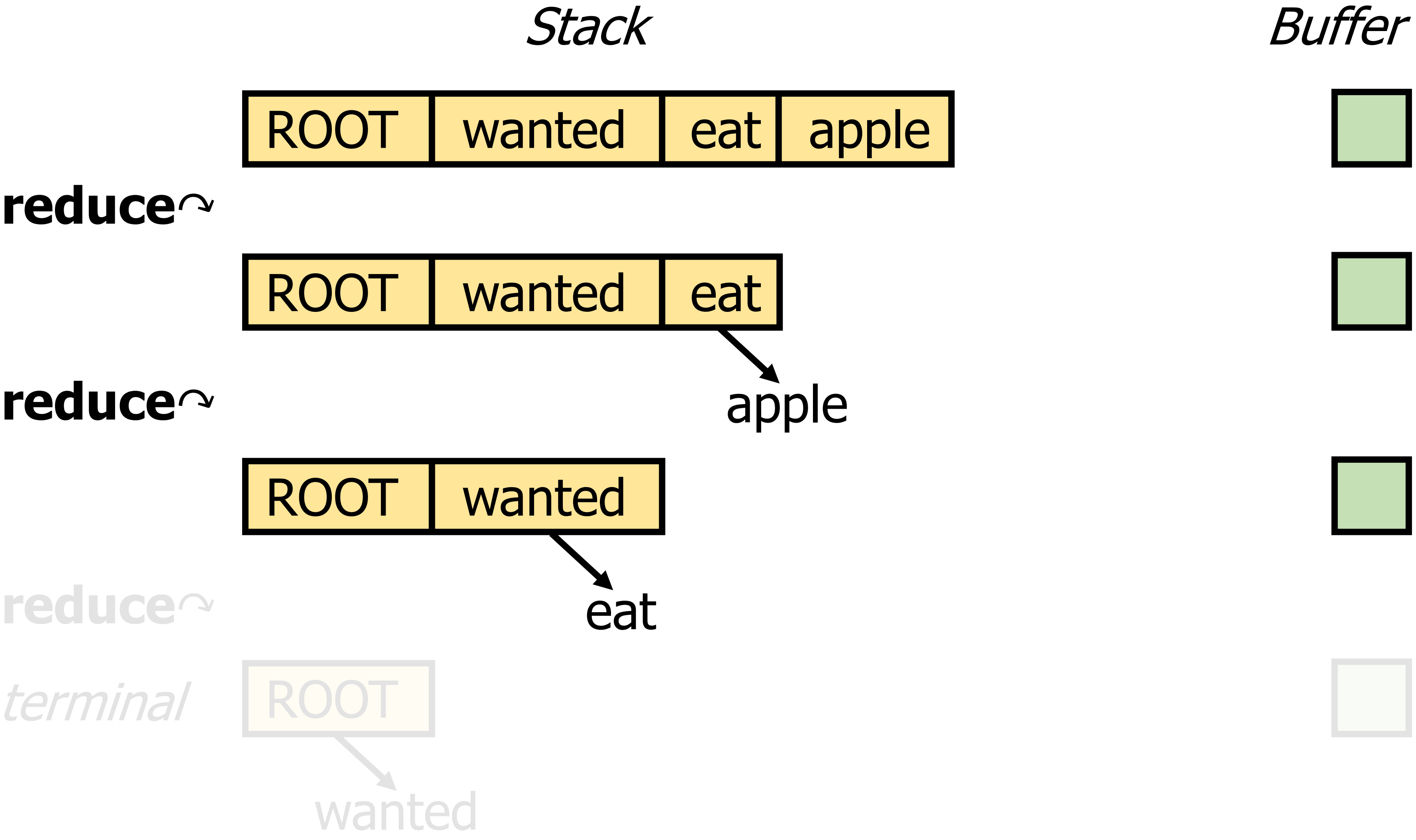
reduce↷

terminal

ROOT

wanted

Arc-hybrid Transition System



Stack *Buffer*

ROOT wanted eat apple

reduce↷

ROOT wanted eat

apple

reduce↷

ROOT wanted

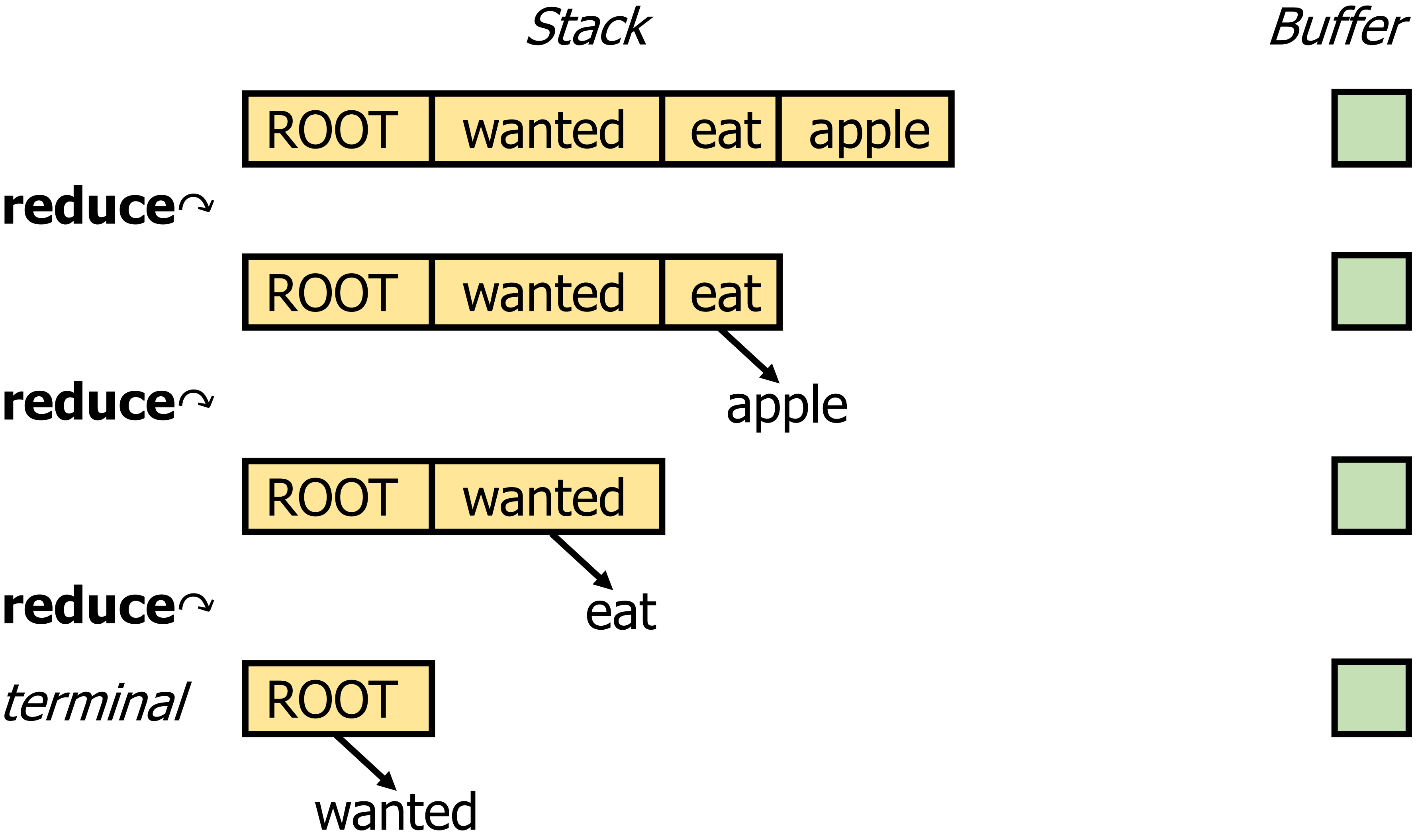
eat

reduce↷

terminal ROOT

wanted

Arc-hybrid Transition System



Stack

Buffer

ROOT | wanted | eat | apple

reduce↷

ROOT | wanted | eat

apple

reduce↷

ROOT | wanted

eat

reduce↷

terminal ROOT

wanted

Dynamic Programming for Arc-hybrid

0	1	2	3	4	5	6 (n)
ROOT	She	wanted	to	eat	an	apple

Stack   *Buffer*

- Deduction Item

$\cdots$	i

j	$\cdots$

$[i, j]$

- Goal

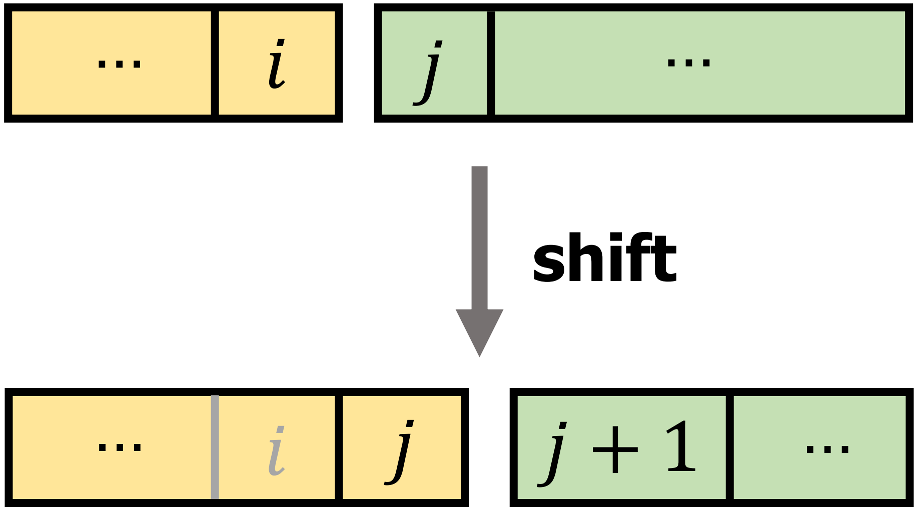
0

$n+1$

$[0, n+1]$

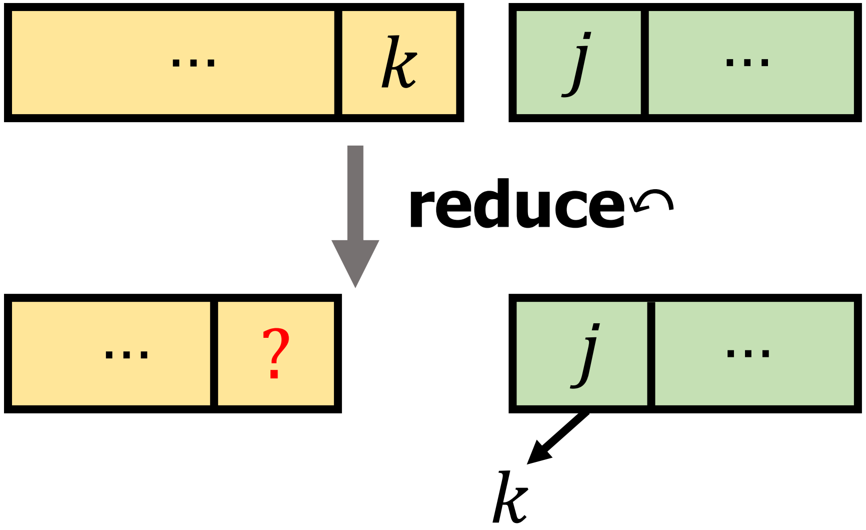
Dynamic Programming for Arc-hybrid

shift $$\frac{[i, j]}{[j, j+1]}$$



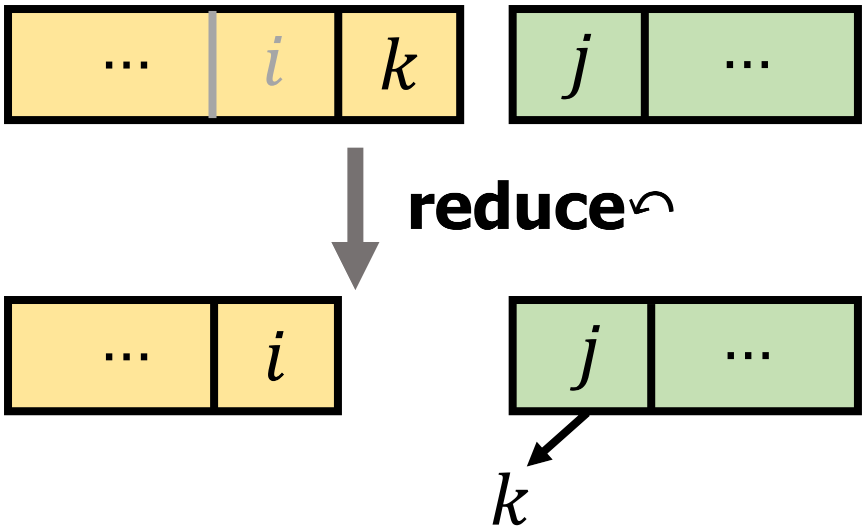
Dynamic Programming for Arc-hybrid

reduce↶ $$\frac{[k, j]}{[?\,, j]}$$



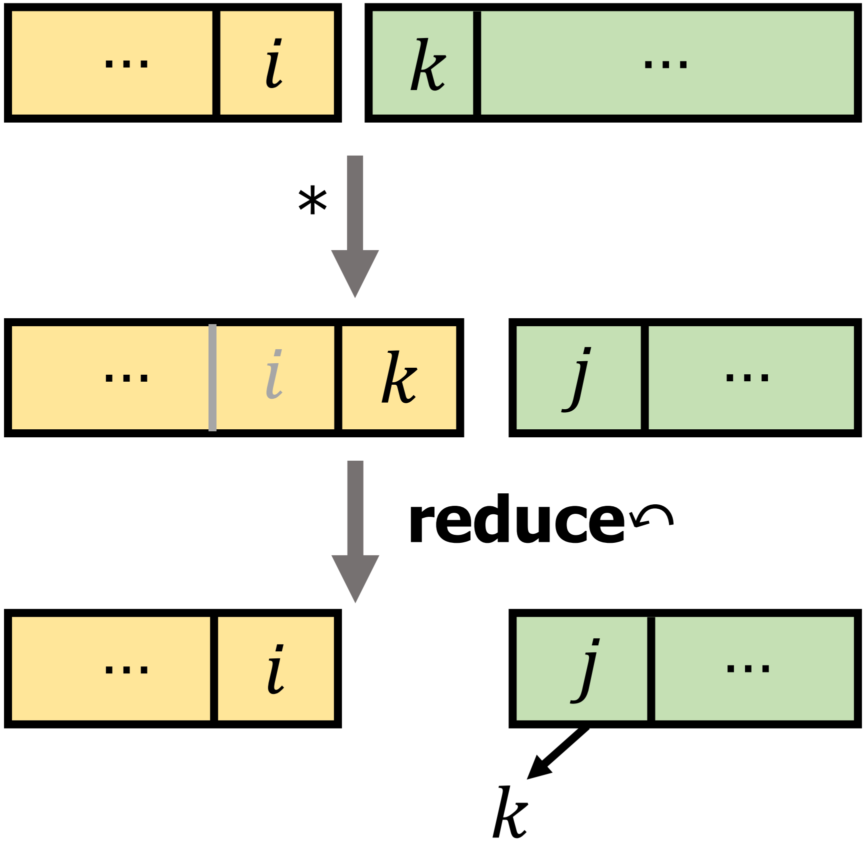
Dynamic Programming for Arc-hybrid

reduce↶ $\dfrac{[k, j]}{[i, j]}$



Dynamic Programming for Arc-hybrid

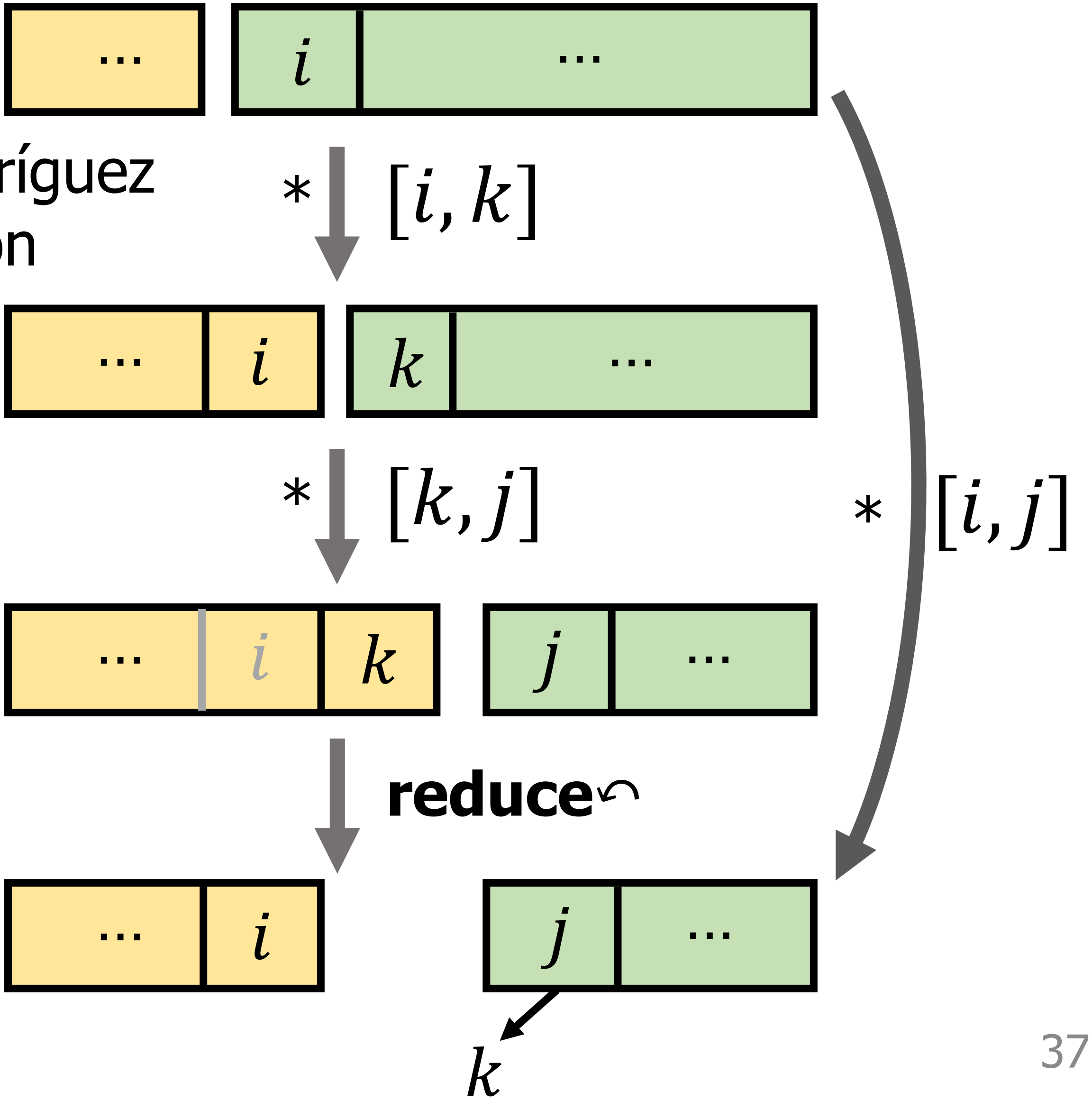
reduce↶ $$\frac{[i,k]\ [k,j]}{[i,j]}$$



Dynamic Programming for Arc-hybrid

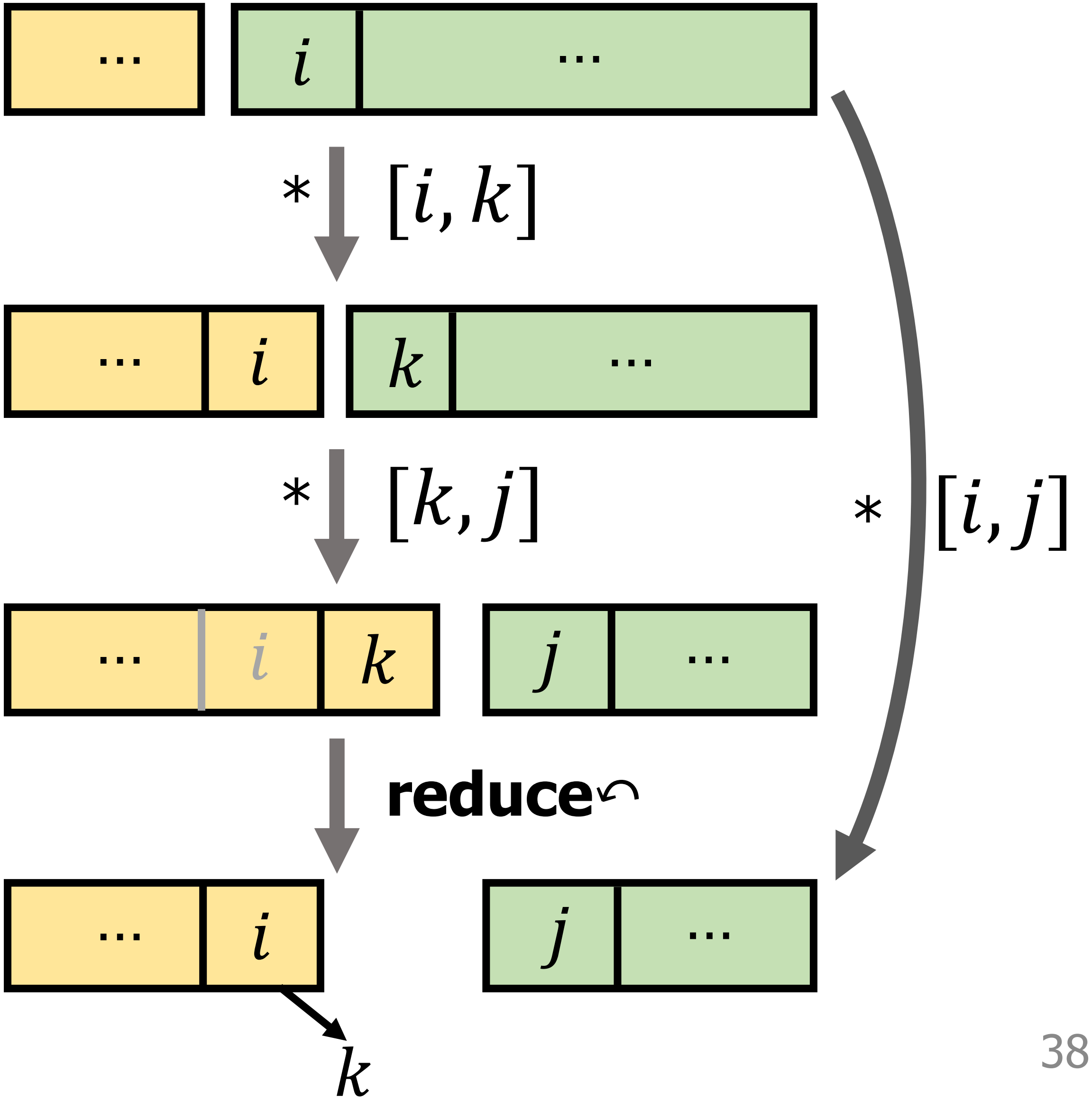
In Kuhlmann, Gómez-Rodríguez and Satta (2011)’s notation

reduce↶ $$\frac{[i,k]\ [k,j]}{[i,j]}$$



Dynamic Programming for Arc-hybrid

reduce↷ $\dfrac{[i,k]\ [k,j]}{[i,j]}$



Dynamic Programming for Arc-hybrid

shift $$\frac{[i,j]}{[j,j+1]}$$

Goal: $[0,n+1]$

reduce$\curvearrowleft$ $$\frac{[i,k]\;[k,j]}{[i,j]}\quad k \curvearrowleft j$$

reduce$\curvearrowright$ $$\frac{[i,k]\;[k,j]}{[i,j]}\quad i \curvearrowright k$$

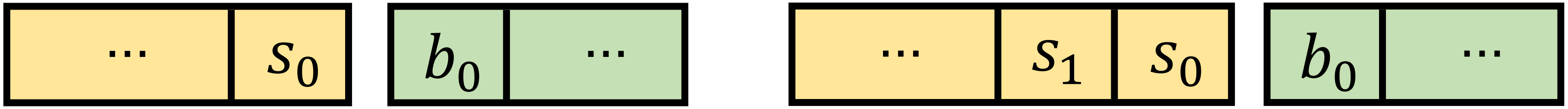
$O(n^3)$

Time Complexity in Practice

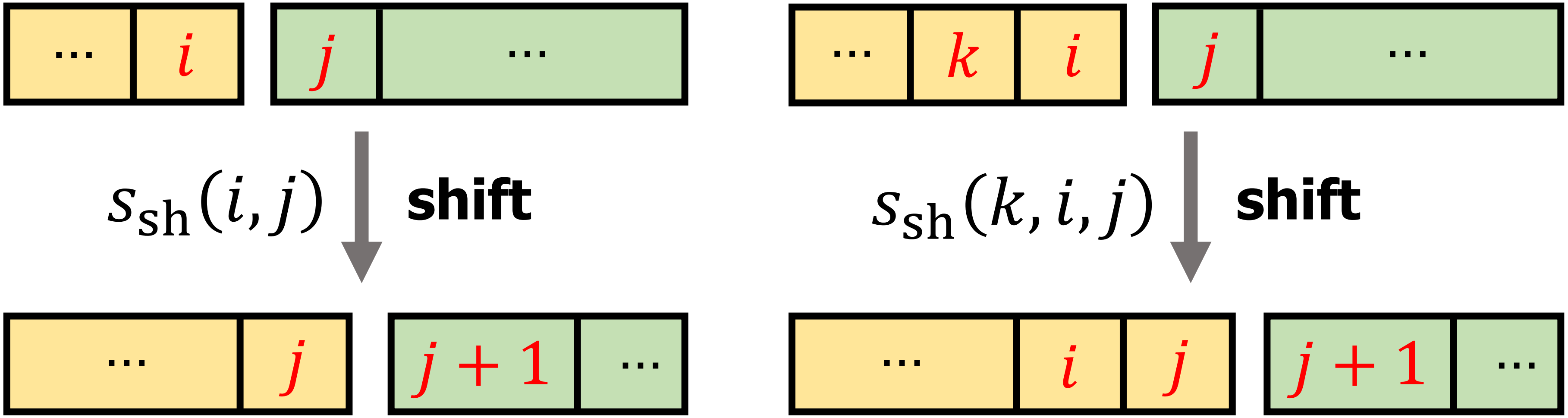
- Complexity depends on feature representation!
- Typical feature representation:
  - Feature templates look at specific *positions* in the stack and in the buffer

Time Complexity in Practice

- Compare the following features

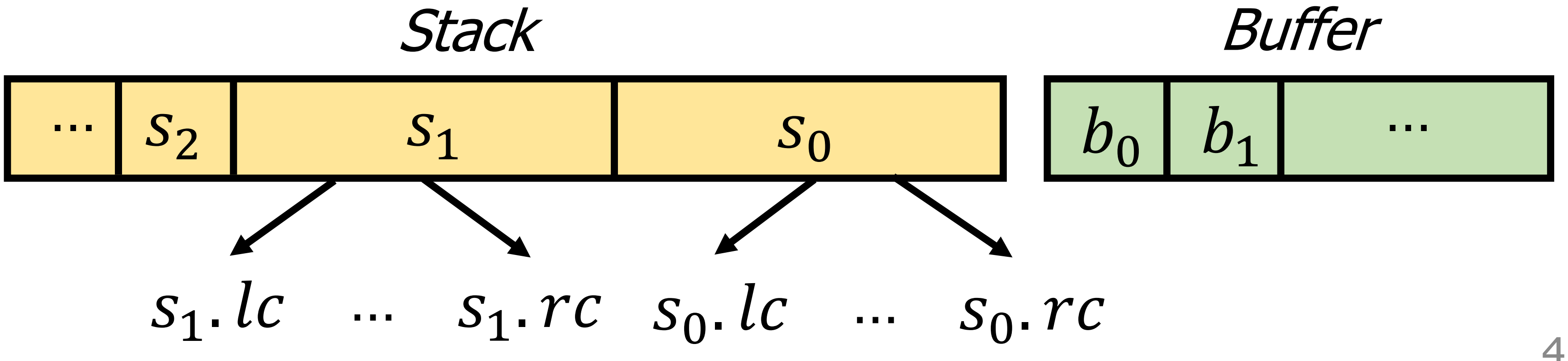


- Time complexities are different!!!



Time Complexity in Practice

- Complexity depends on feature representation!
- Typical feature representation:
  - Feature templates look at specific *positions* in the stack and in the buffer
- Best-known complexity in practice: $O(n^6)$ (Huang and Sagae, 2010)



Stack

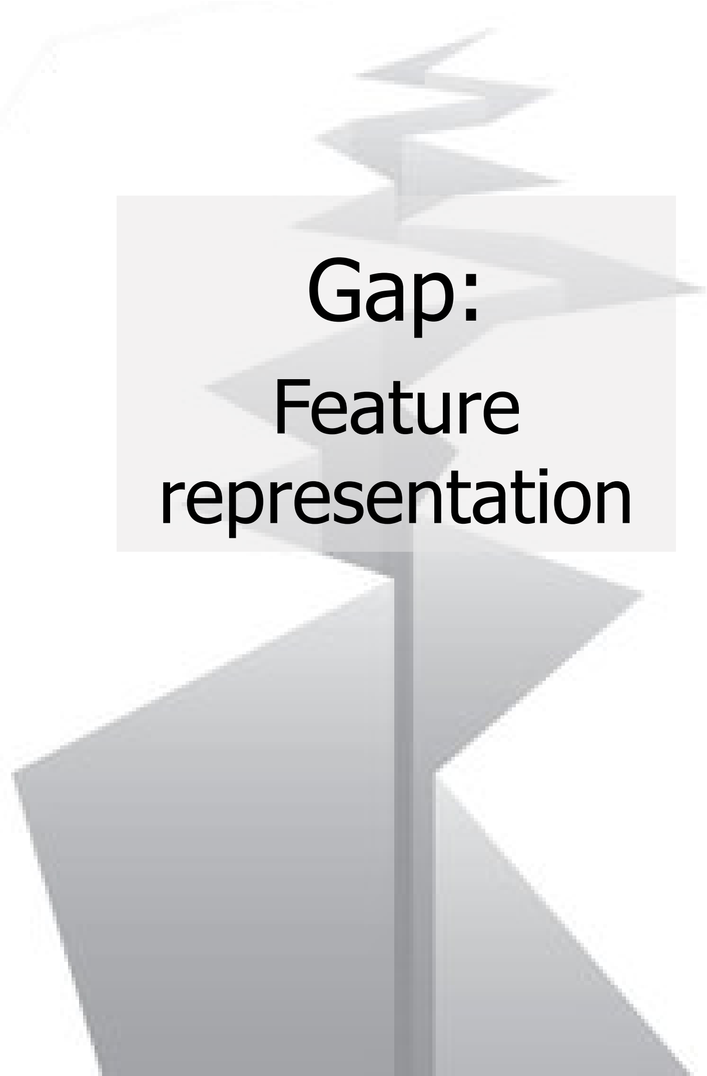
Buffer

$\cdots$ s_2 s_1 s_0

b_0 b_1 $\cdots$

$s_1.lc$ $\cdots$ $s_1.rc$ $s_0.lc$ $\cdots$ $s_0.rc$

Best-known Time Complexities (recap)



$$O(n^3)$$

Theoretical

Gap:

Feature representation

$$O(n^6)$$

Practical

In Practice, Instead of Exact Decoding …

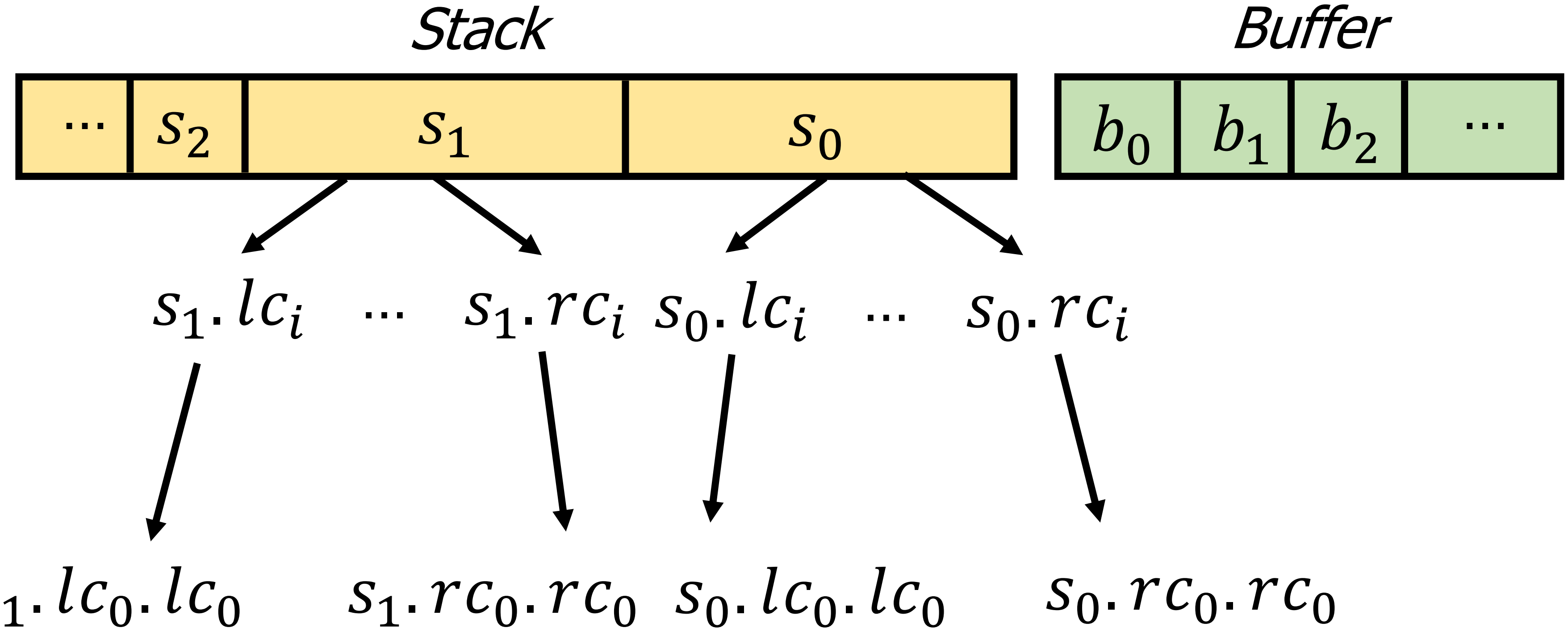
- Greedy search (Nivre, 2003, 2004, 2008; Chen and Manning, 2014)
- Beam search (Zhang and Clark, 2011; Weiss et al.,2015)
- Best-first search (Sagae and Lavie, 2006; Sagae and Tsujii, 2007; Zhao et al., 2013)
- Dynamic oracles (Goldberg and Nivre, 2012, 2013)
- "Global" normalization on the beam (Zhou et al., 2015; Andor et al., 2016)
- Reinforcement learning (Lê and Fokkens, 2017)
- Learning to search (Daumé III and Marcu, 2005; Chang et al., 2016; Wiseman and Rush, 2016)
- …

How Many Positional Features Do We Need?

Non-neural (manual engineering)

↓

☞ Chen and Manning (2014)



How Many Positional Features Do We Need?

Non-neural (manual engineering)

☞ Chen and Manning (2014)

Stack LSTM

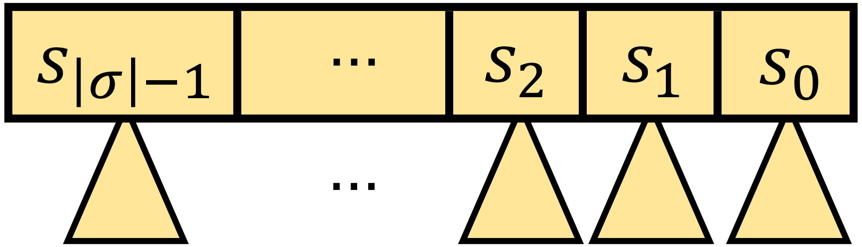
☞ Dyer et al. (2015)

Bi-LSTM

☞ Kiperwasser and Goldberg (2016)

☞ Cross and Huang (2016)

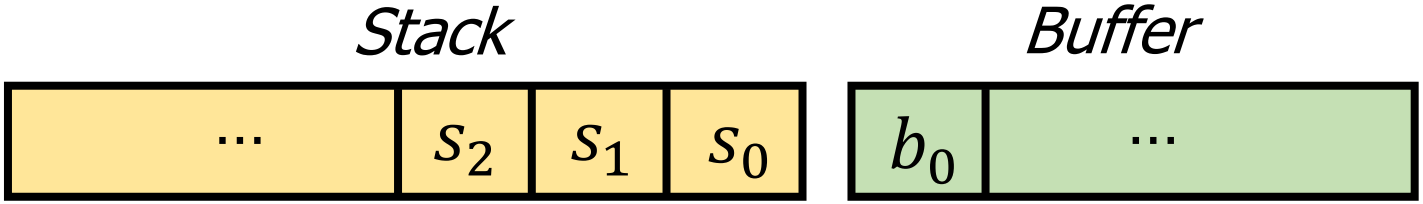
Stack



How Many Positional Features Do We Need?

- Bi-LSTMs give compact feature representations (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016)

- Features used in Kiperwasser and Goldberg (2016)



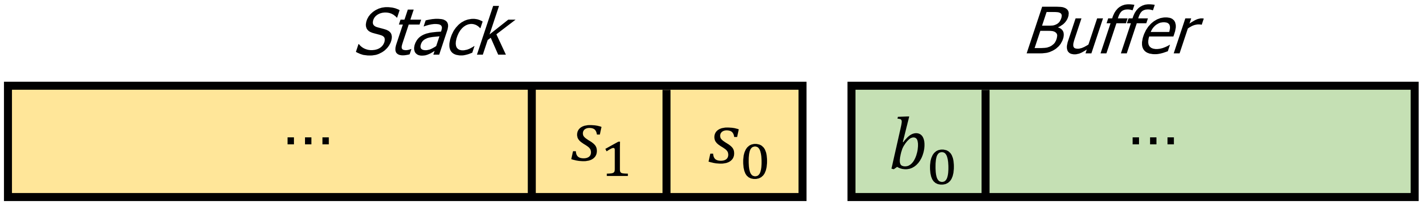
Stack

… s_2 s_1 s_0

Buffer

b_0 …

- Features used in Cross and Huang (2016)



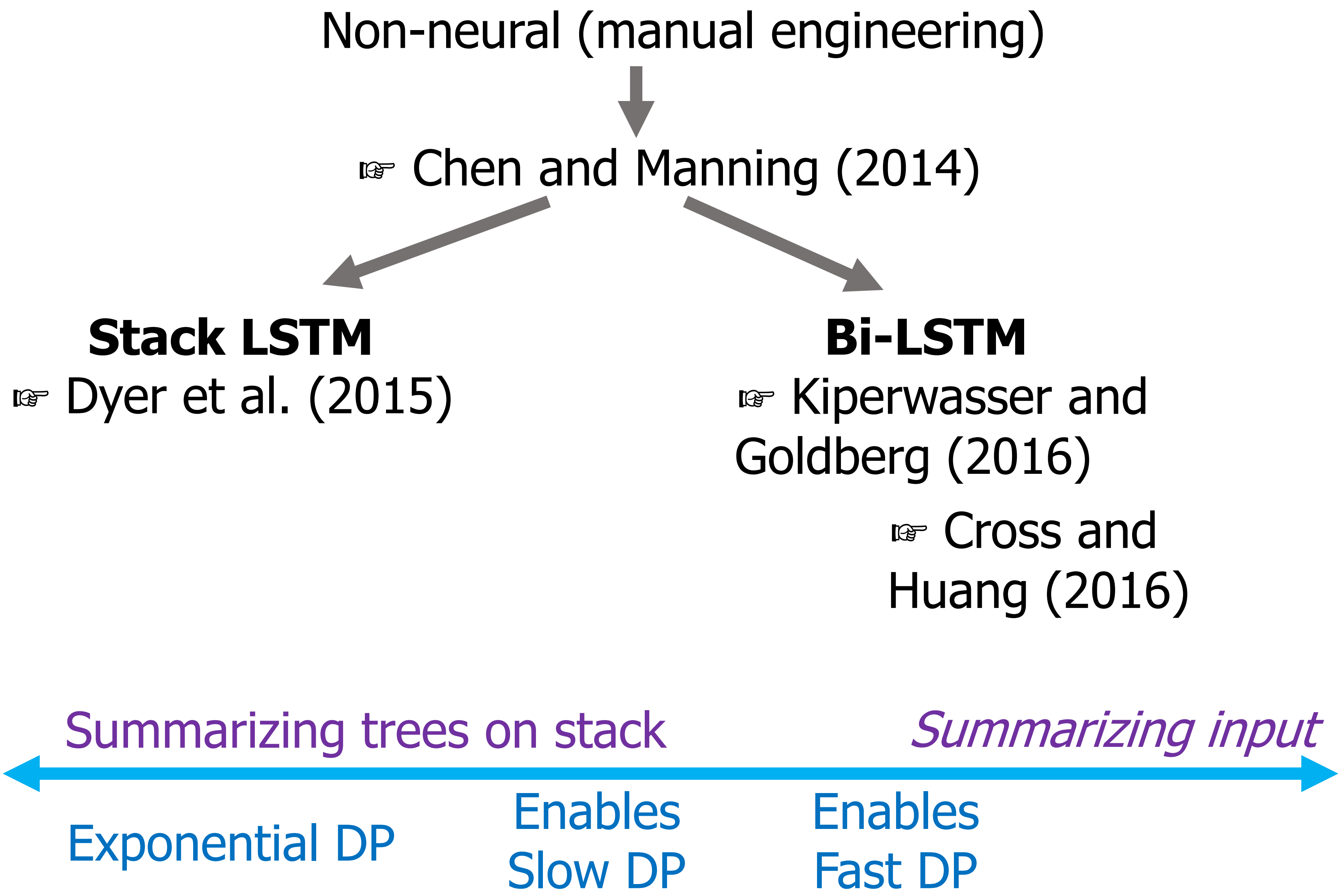
Stack

… s_1 s_0

Buffer

b_0 …

How Many Positional Features Do We Need?



Non-neural (manual engineering)

☞ Chen and Manning (2014)

Stack LSTM

☞ Dyer et al. (2015)

Bi-LSTM

☞ Kiperwasser and Goldberg (2016)

☞ Cross and Huang (2016)

Summarizing trees on stack

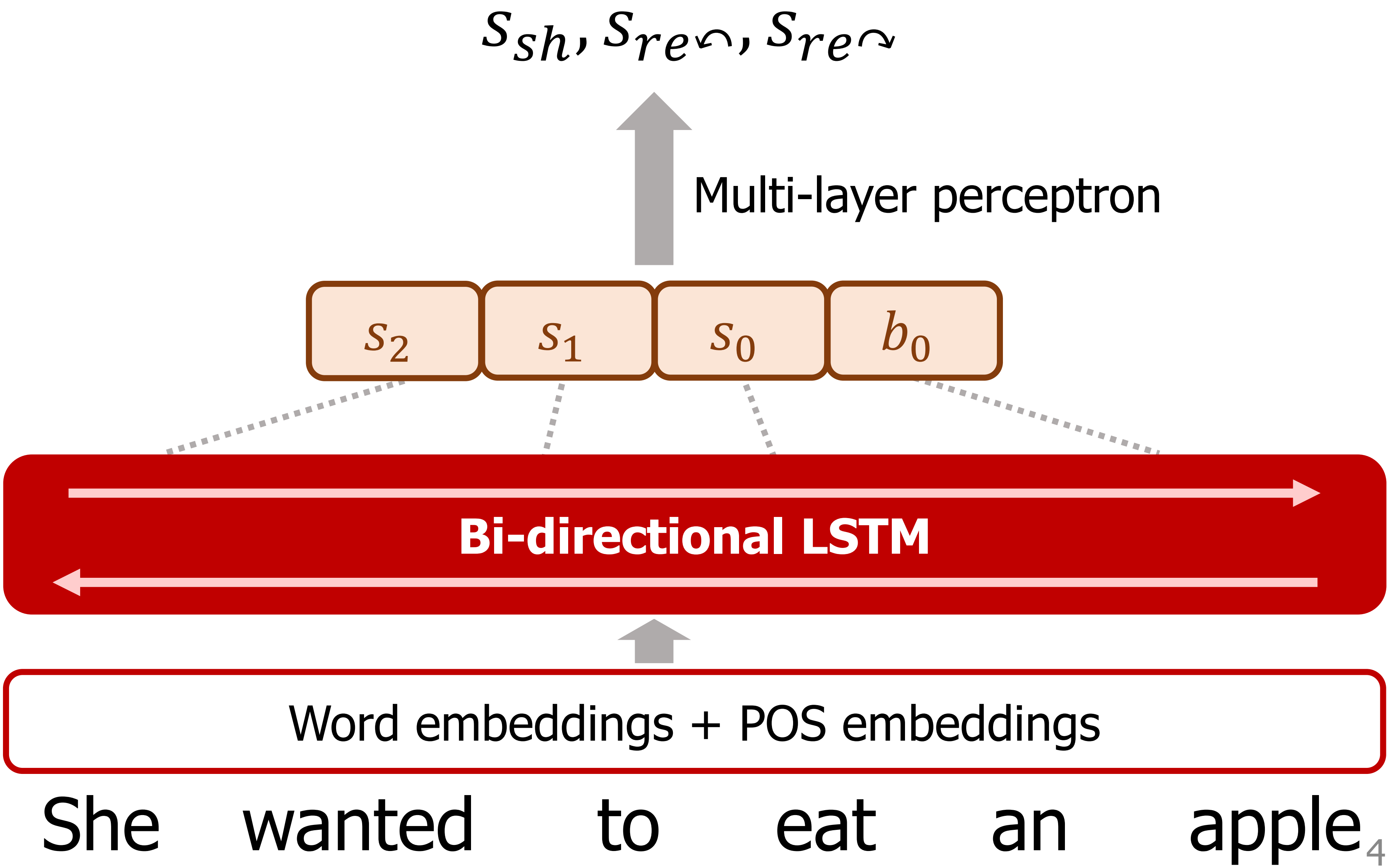
Summarizing input

Exponential DP

Enables Slow DP

Enables Fast DP

Model Architecture



$s_{sh}, s_{re\curvearrowleft}, s_{re\curvearrowright}$

Multi-layer perceptron

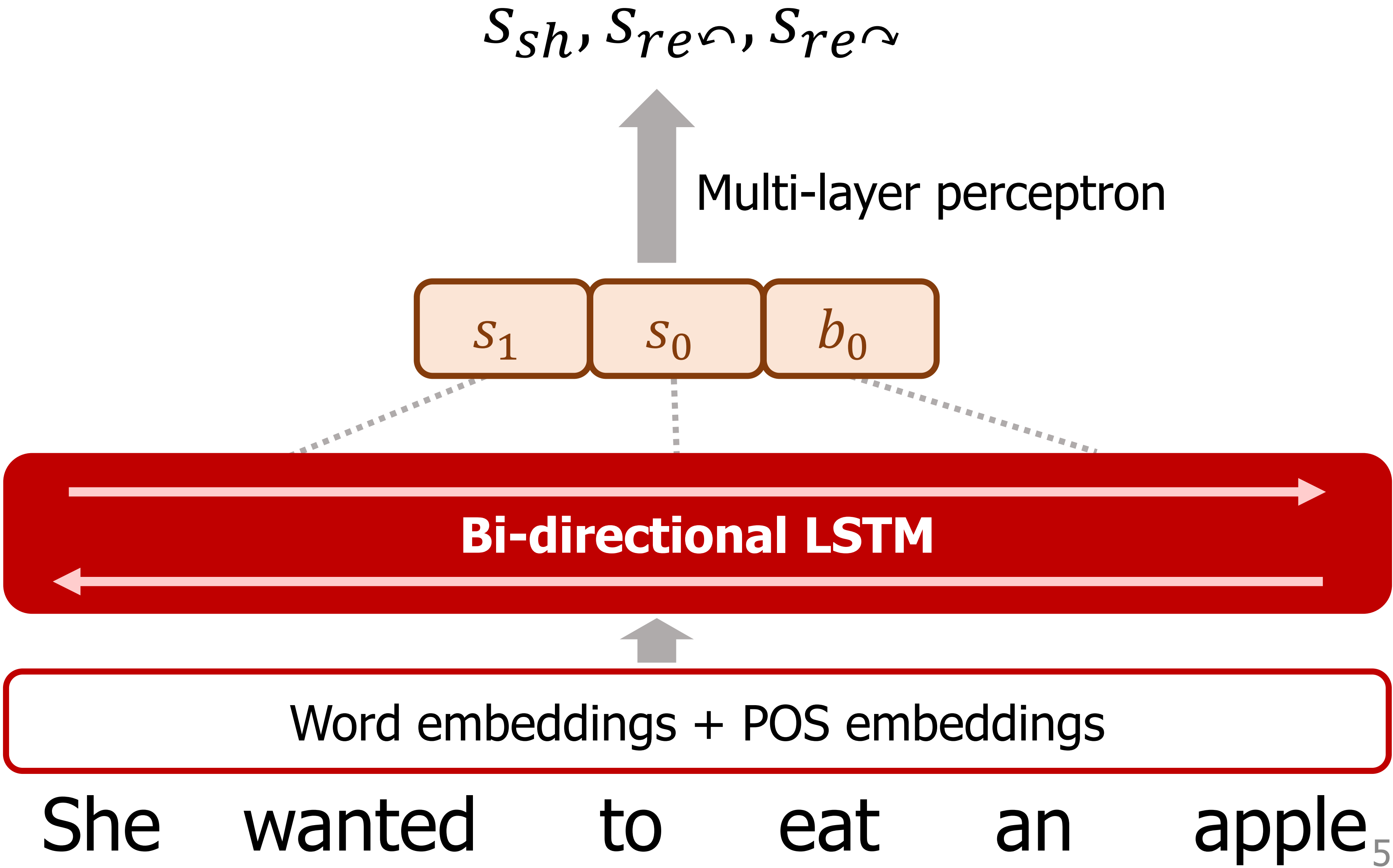
s_2 s_1 s_0 b_0

Bi-directional LSTM

Word embeddings + POS embeddings

She wanted to eat an apple

Model Architecture



$s_{sh}, s_{re\curvearrowleft}, s_{re\curvearrowright}$

Multi-layer perceptron

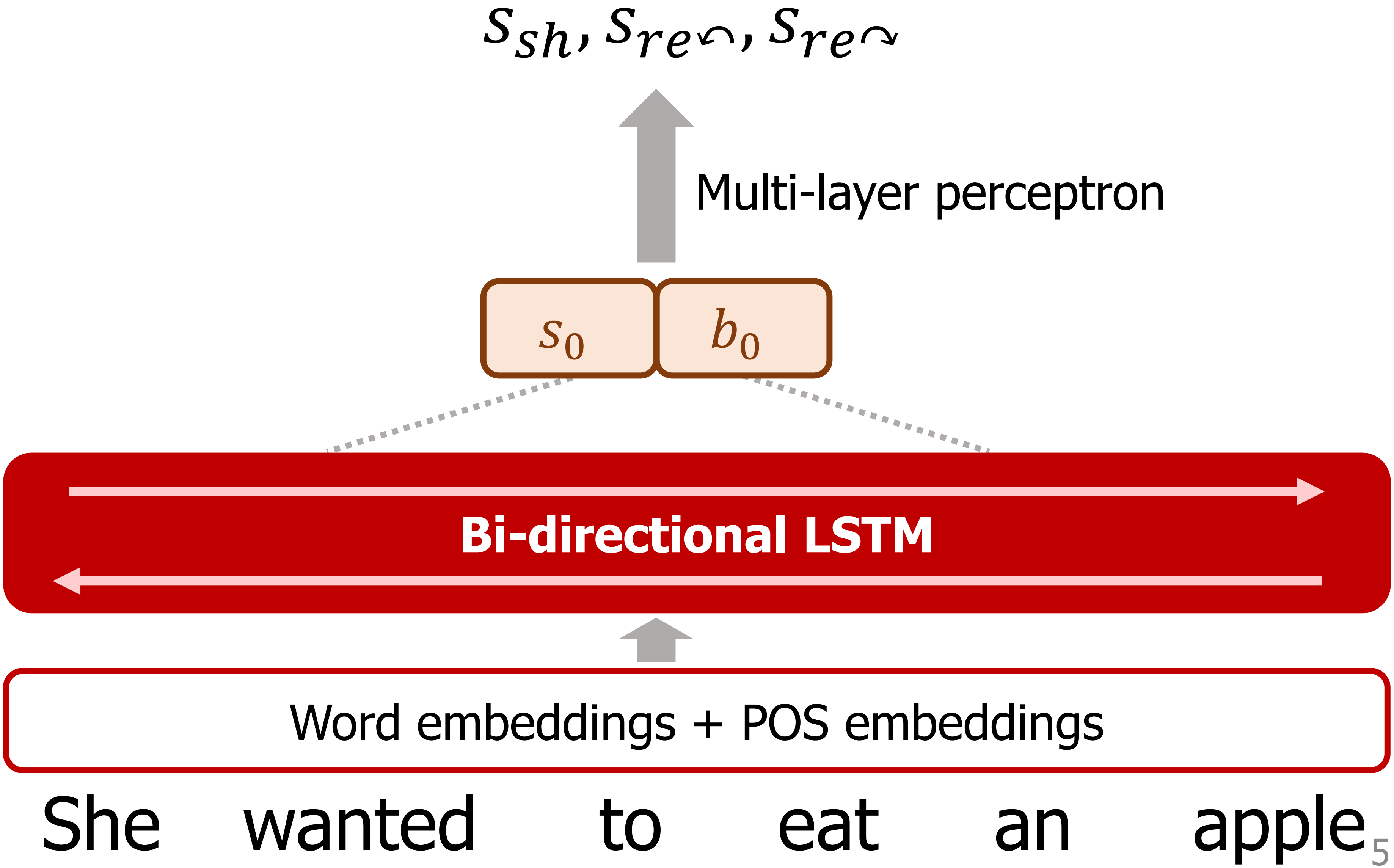
s_1 s_0 b_0

Bi-directional LSTM

Word embeddings + POS embeddings

She wanted to eat an apple

Model Architecture



$s_{sh}, s_{re\curvearrowleft}, s_{re\curvearrowright}$

Multi-layer perceptron

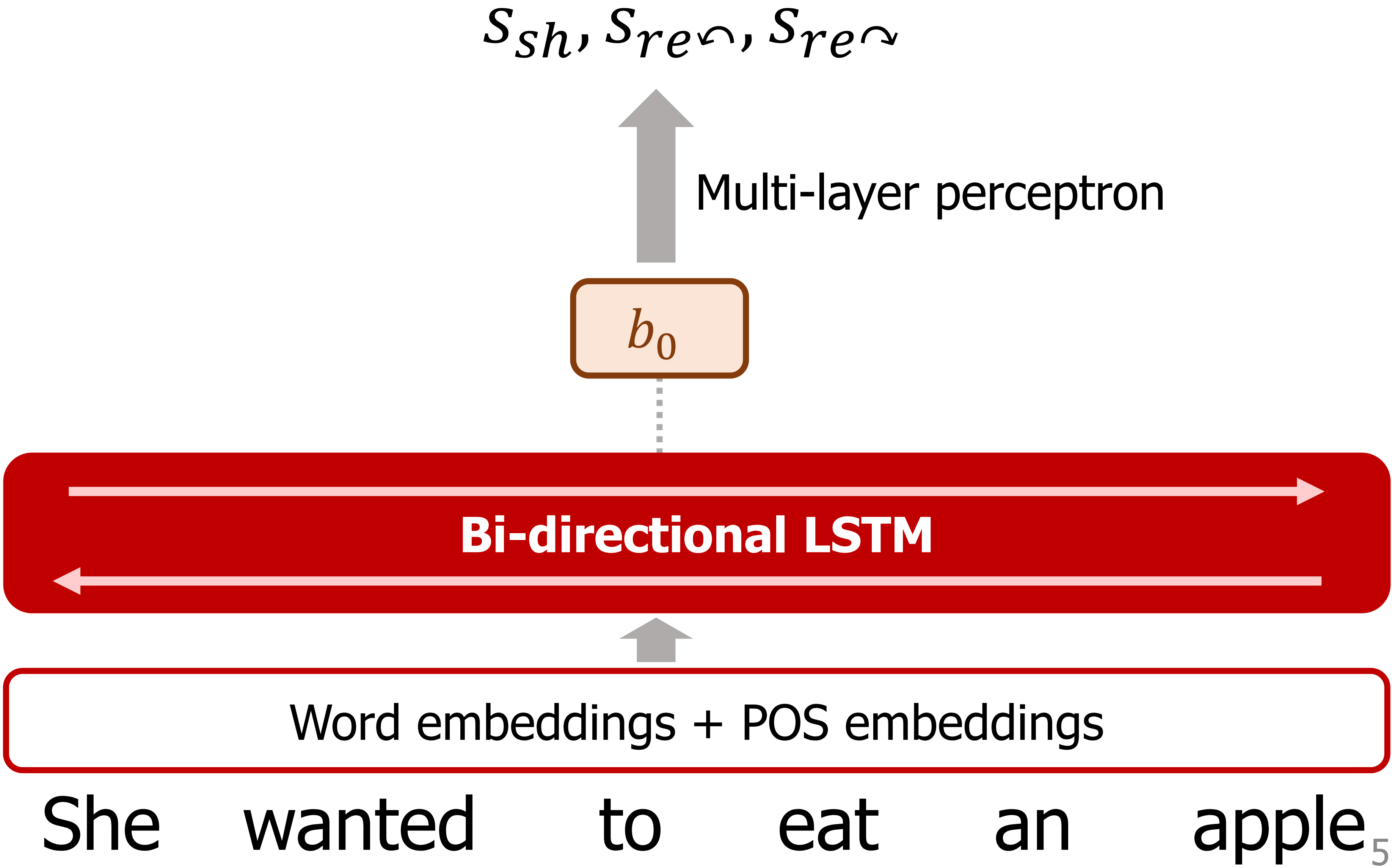
s_0 b_0

Bi-directional LSTM

Word embeddings + POS embeddings

She wanted to eat an apple

Model Architecture



$s_{sh}, s_{re\curvearrowleft}, s_{re\curvearrowright}$

Multi-layer perceptron

b_0

Bi-directional LSTM

Word embeddings + POS embeddings

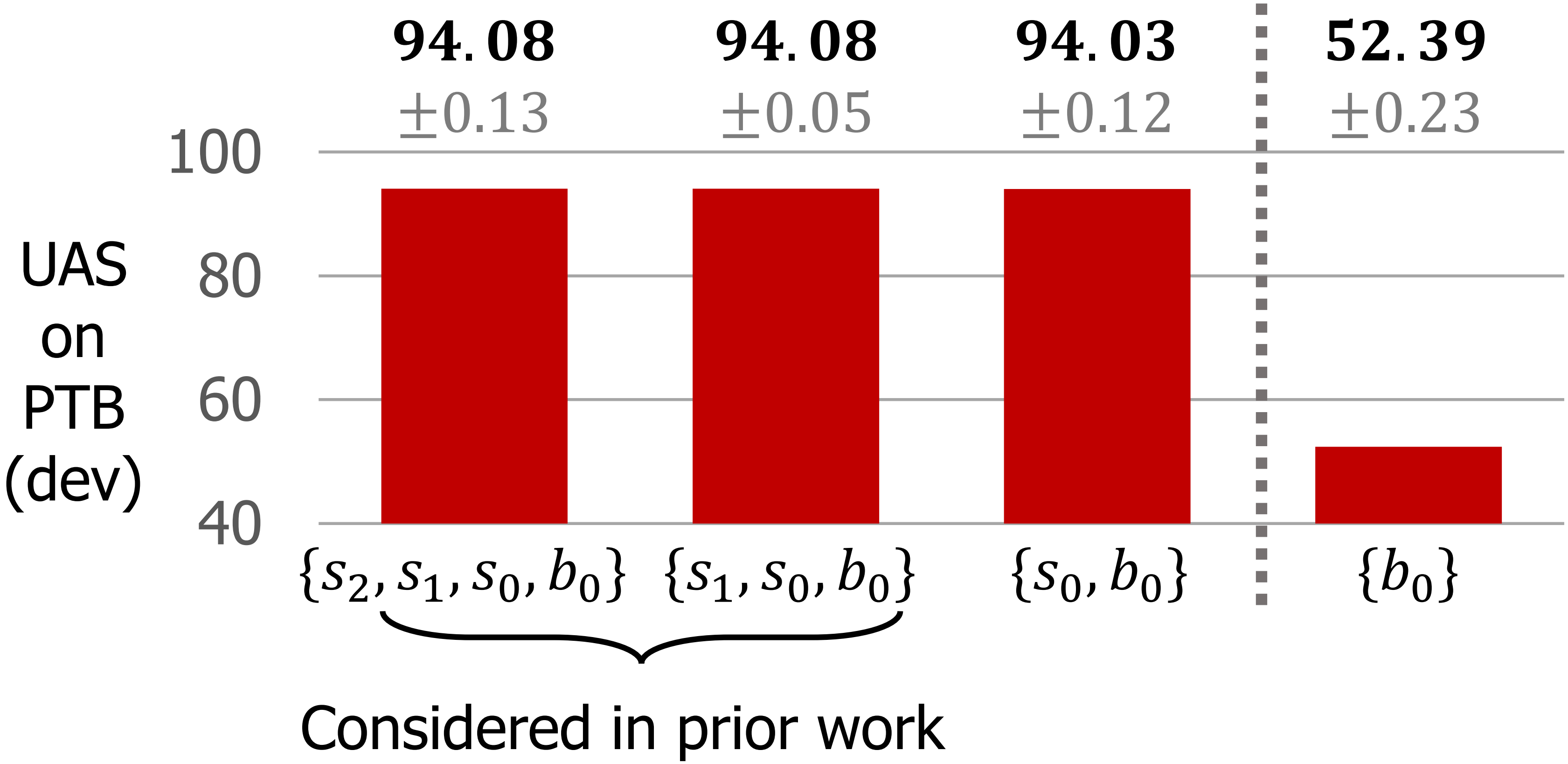
She wanted to eat an apple

How Many Positional Features Do We Need?

- We answer the question empirically

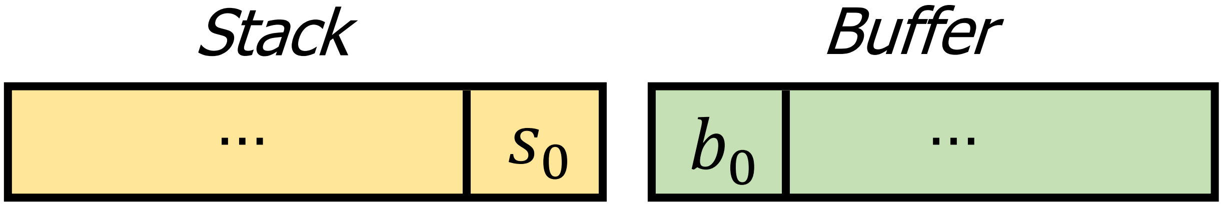
  … experimented with greedy decoding

- Two positional feature vectors are enough!



How Many Positional Features Do We Need?

- Our minimal feature set



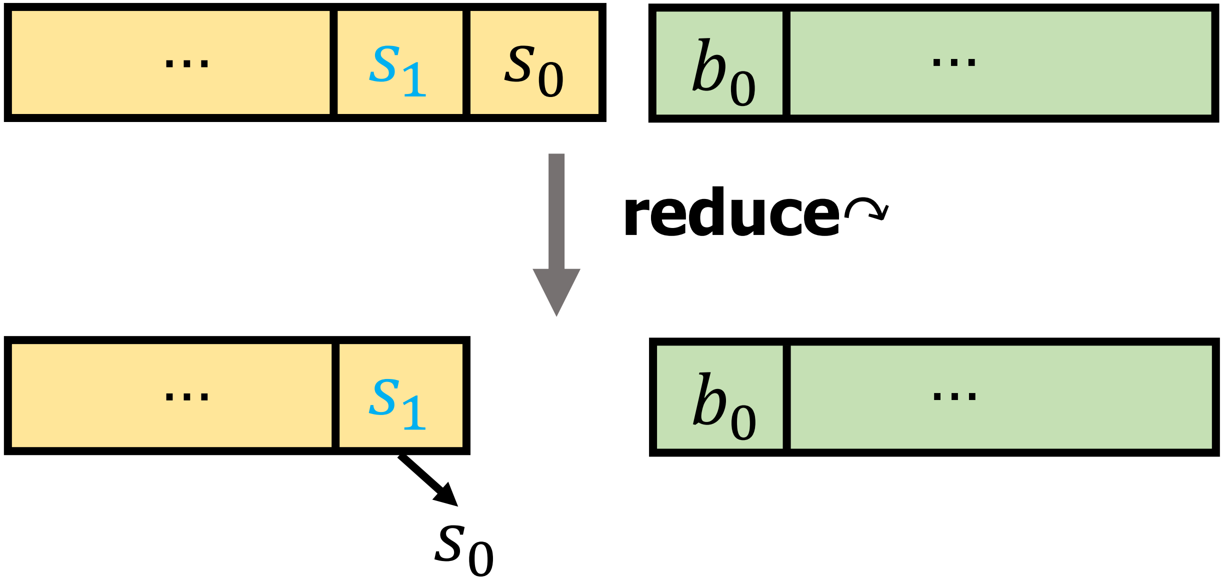
Stack

Buffer

… s_0

b_0 …

- Counter-intuitive, but works for *greedy decoding*



… s_1 s_0

b_0 …

reduce$\curvearrowright$

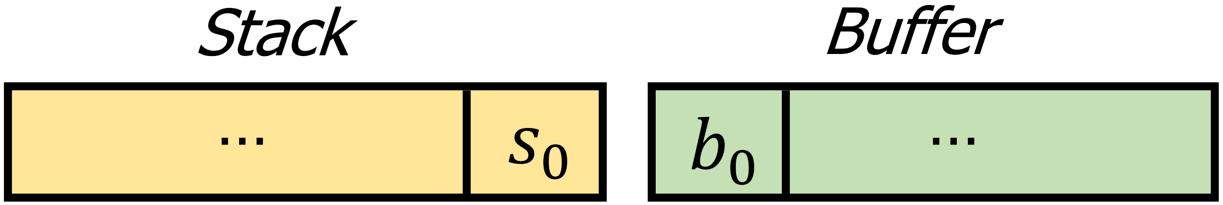
… s_1

b_0 …

s_0

How Many Positional Features Do We Need?

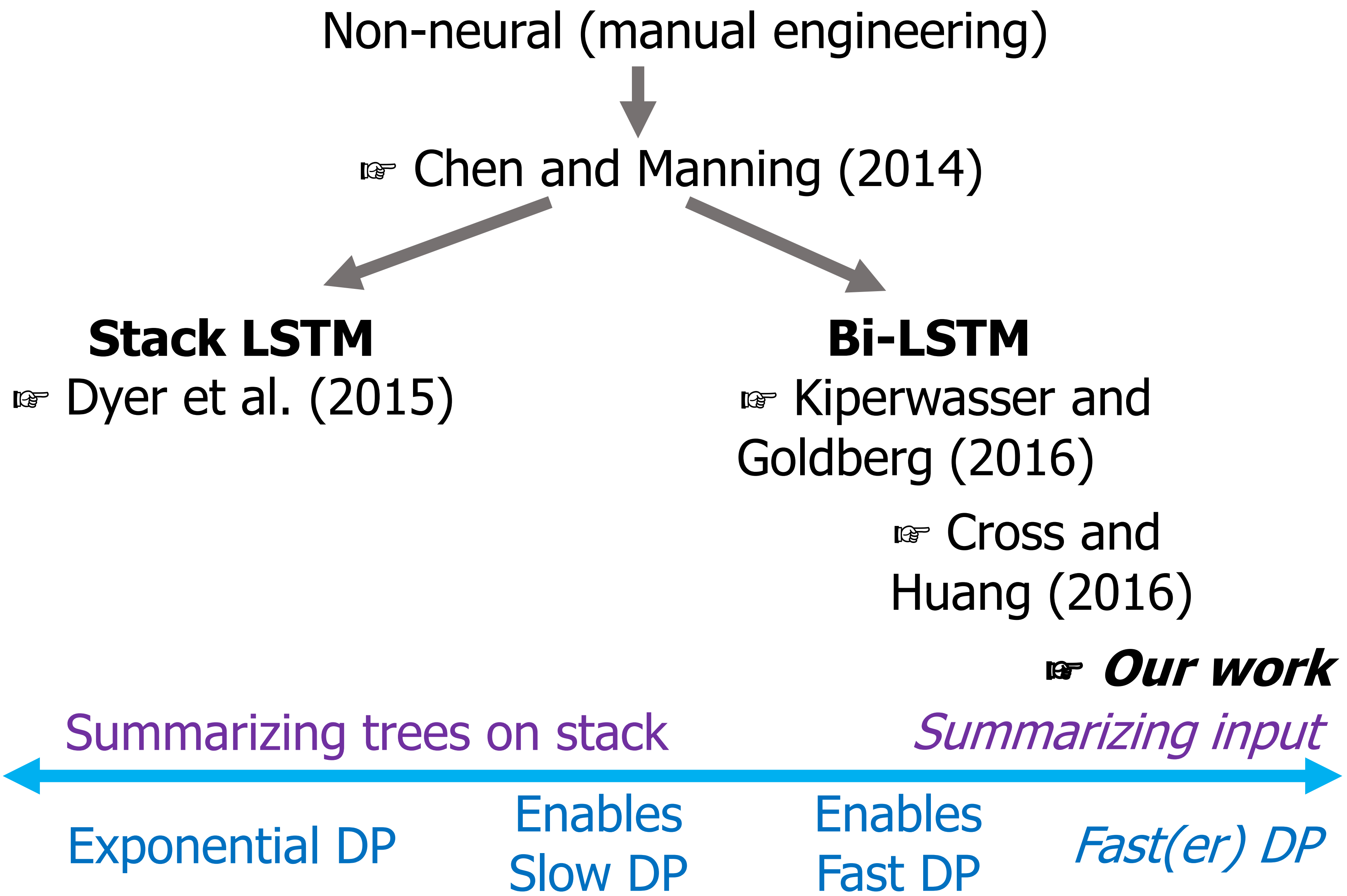
- Our minimal feature set



- Counter-intuitive, but works for *greedy decoding*

- The bare deduction items already contain enough information to extract features for DP
- Leads to the first $O(n^3)$ implementation of global decoders!

How Many Positional Features Do We Need?



Non-neural (manual engineering)

☞ Chen and Manning (2014)

Stack LSTM

☞ Dyer et al. (2015)

Bi-LSTM

☞ Kiperwasser and Goldberg (2016)

☞ Cross and Huang (2016)

☞ ***Our work***

Summarizing trees on stack

Summarizing input

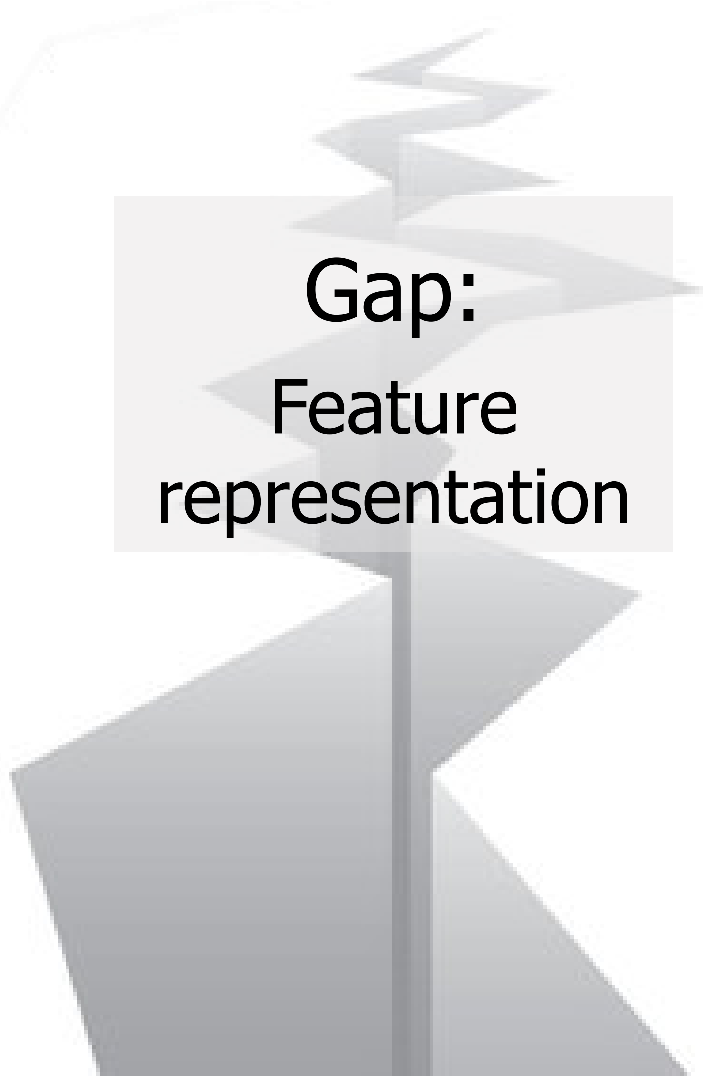
Exponential DP

Enables Slow DP

Enables Fast DP

Fast(er) DP

Best-known Time Complexities (recap)



$O(n^3)$

Theoretical

Gap:

Feature representation

$O(n^6)$

Practical

Our contribution

$O(n^3)$

Theoretical

Minimal

Feature Set

$O(n^3)$

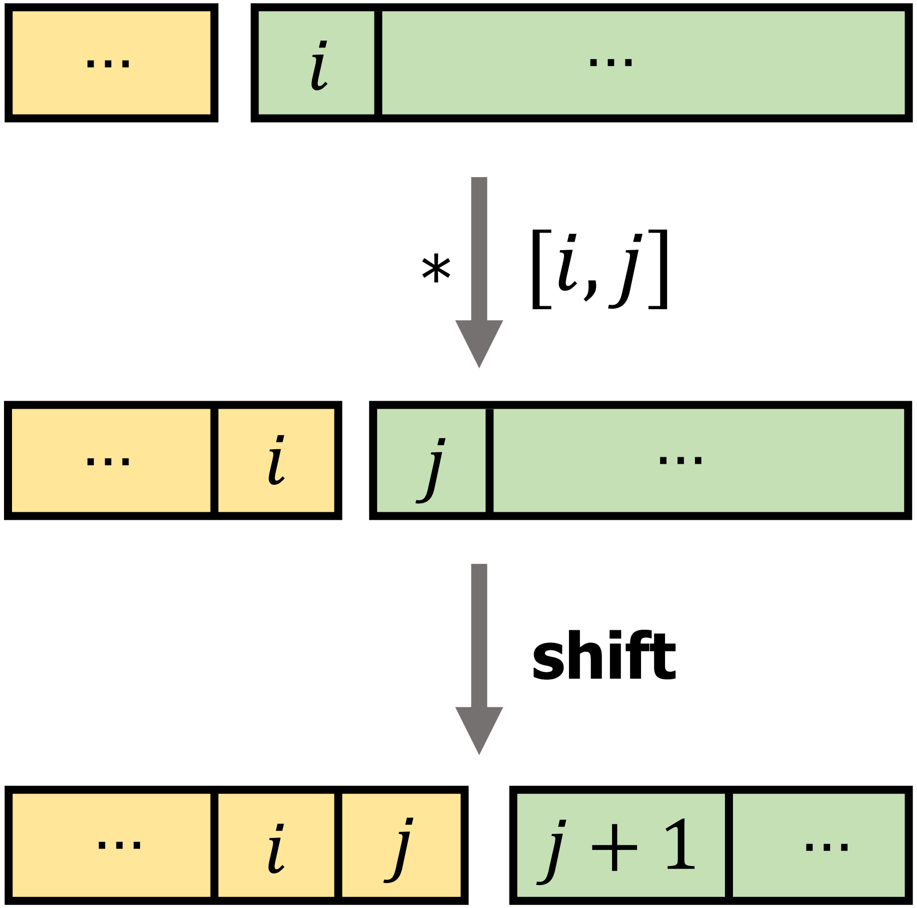
~~$O(n^6)$~~

Practical

Decoding

Score of the sub-sequence

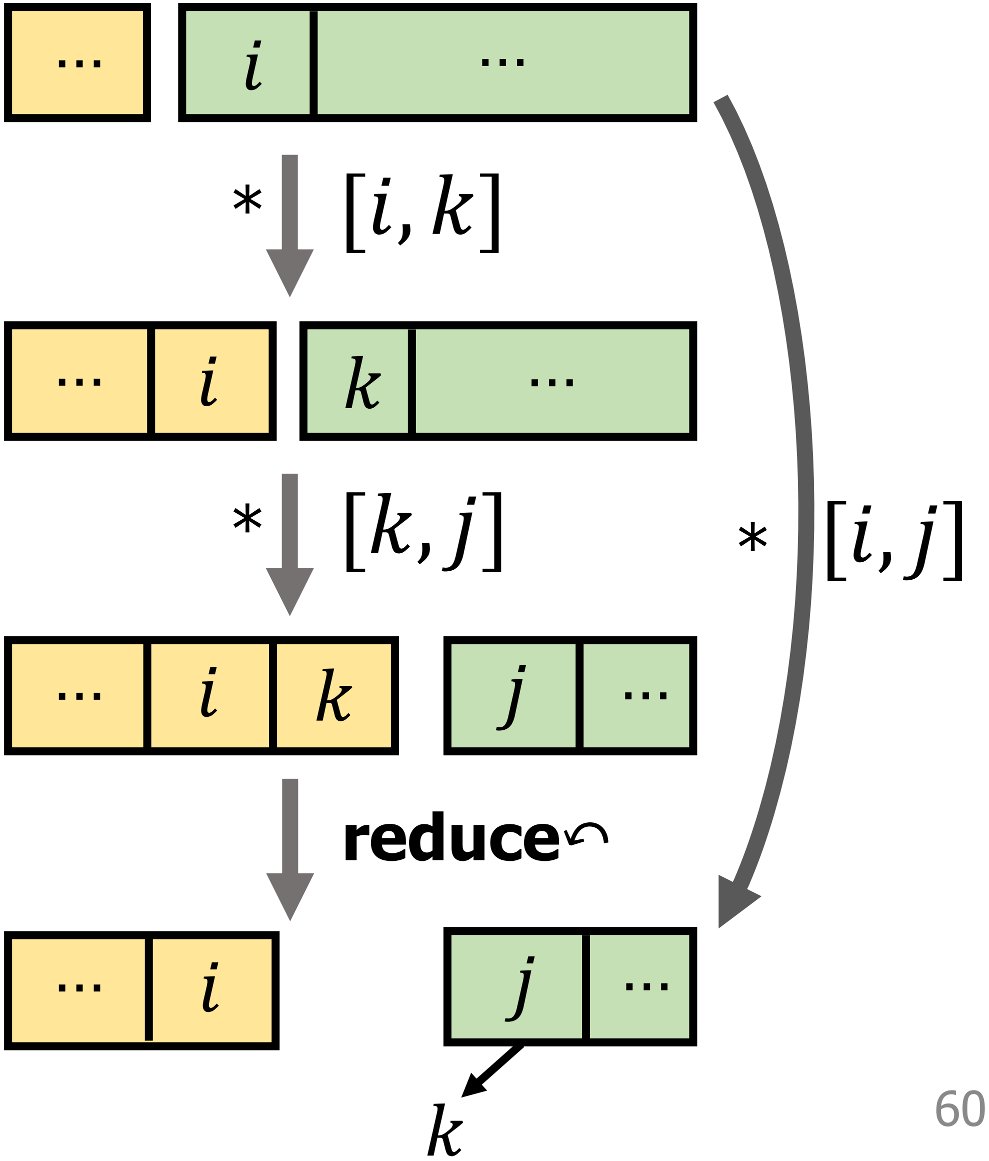
$$\textbf{shift}\quad \frac{[i,j]: \boxed{v}}{[j, j+1]: 0}$$



Decoding

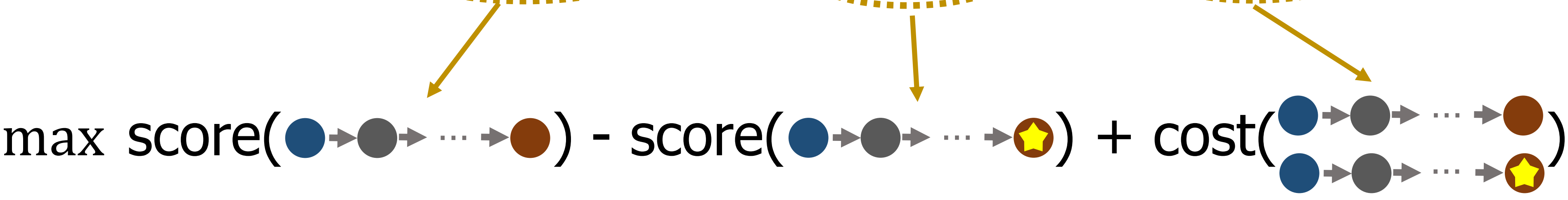
$$\textbf{reduce}{\curvearrowleft}\ \frac{[i,k]{:}\,v_1 \quad [k,j]{:}\,v_2}{[i,j]{:}\,v_1+v_2+\Delta}$$

$$\Delta = s_{\text{sh}}(i,k) + s_{\text{re}\curvearrowleft}(k,j)$$



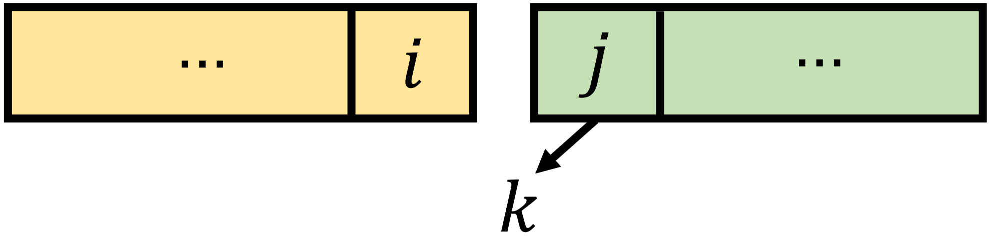
Training

- Separate incorrect from correct by a margin

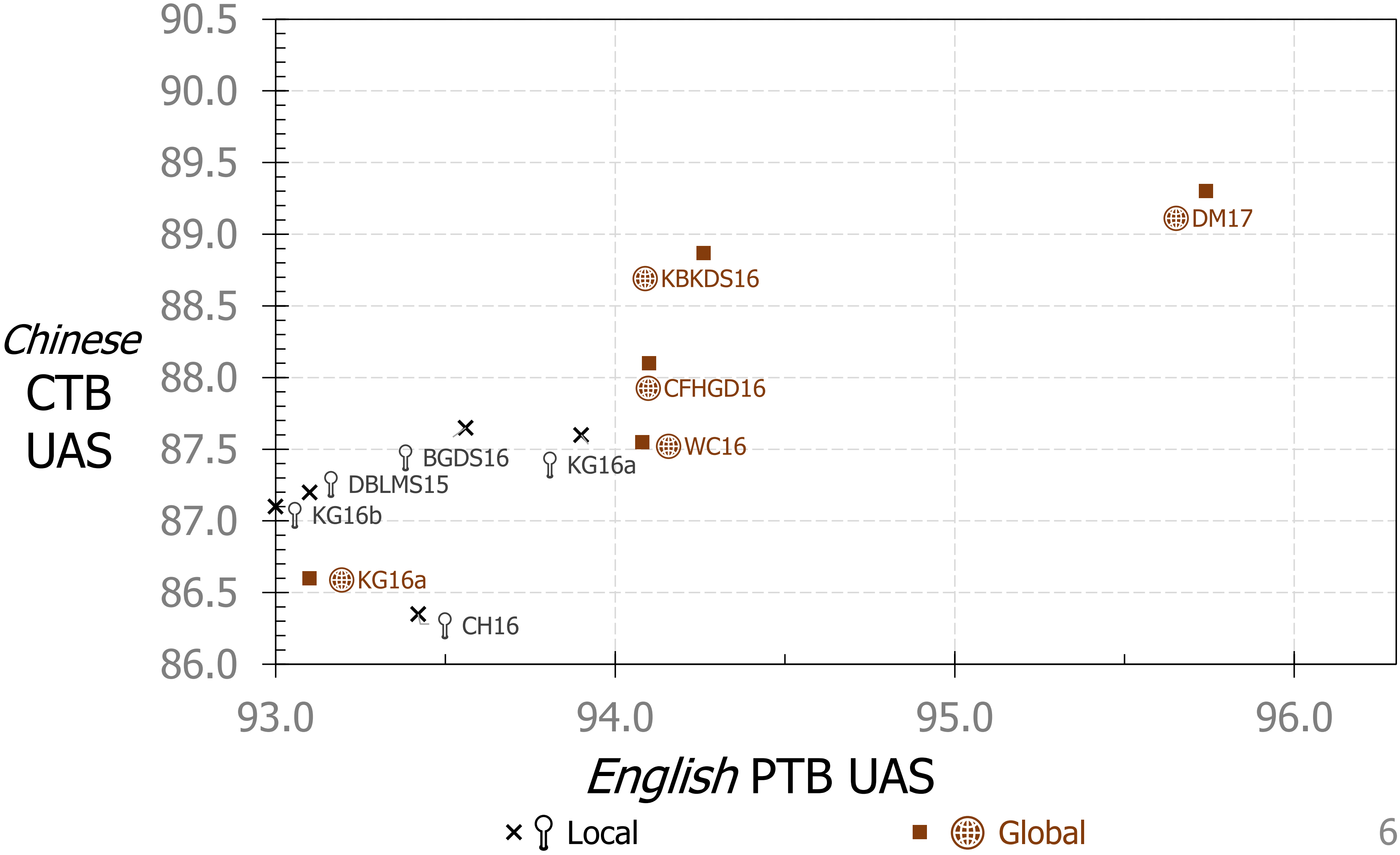


- Cost-augmented decoding (Taskar et al., 2005)

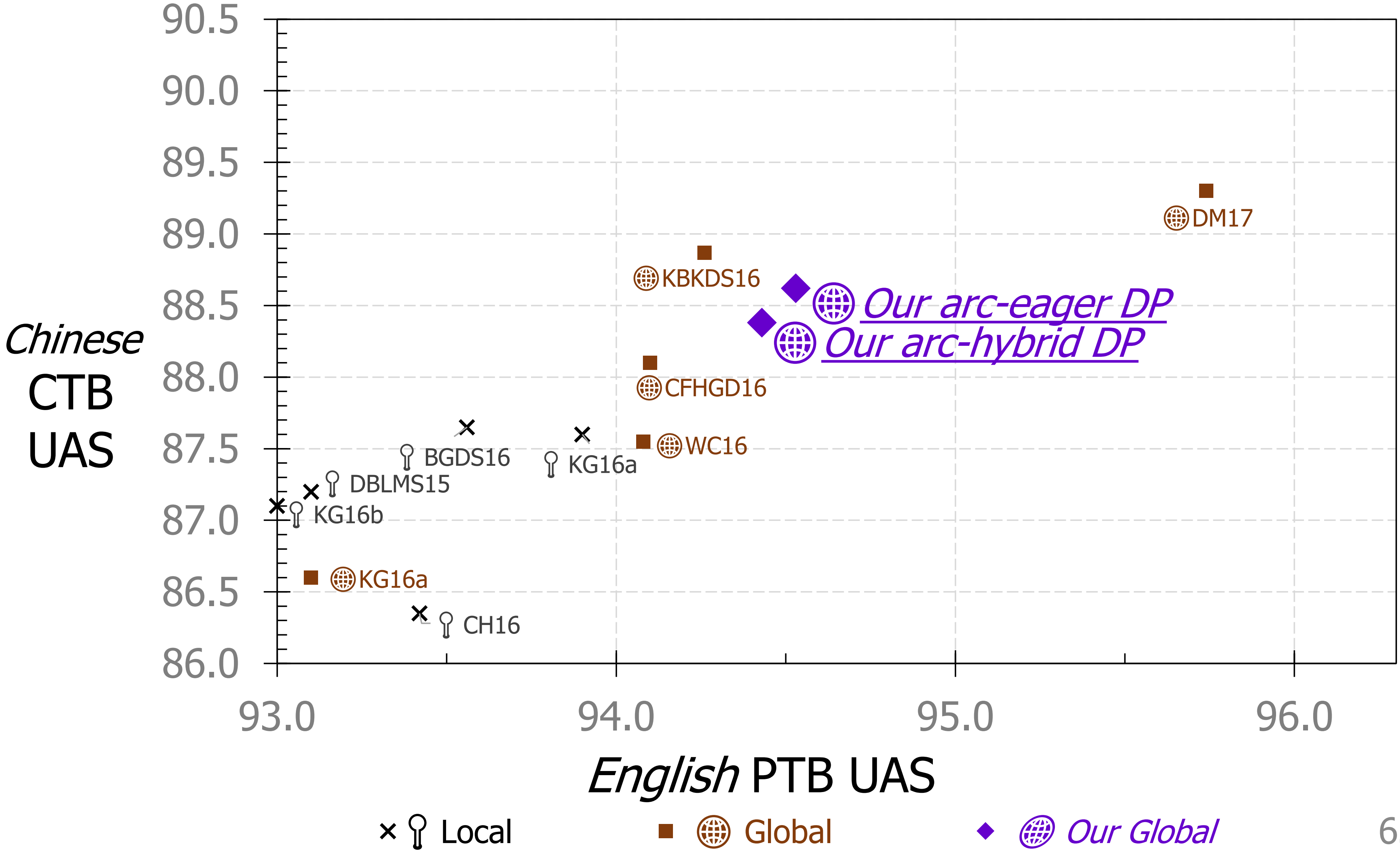
$$\textbf{reduce}\curvearrowleft \quad \frac{[i,k]: v_1 \qquad [k,j]: v_2}{[i,j]: v_1 + v_2 + s_{\mathrm{sh}}(i,k) + s_{\mathrm{re}\curvearrowleft}(k,j) + \mathbf{1}(\mathrm{head}\star(k) \neq j)}$$



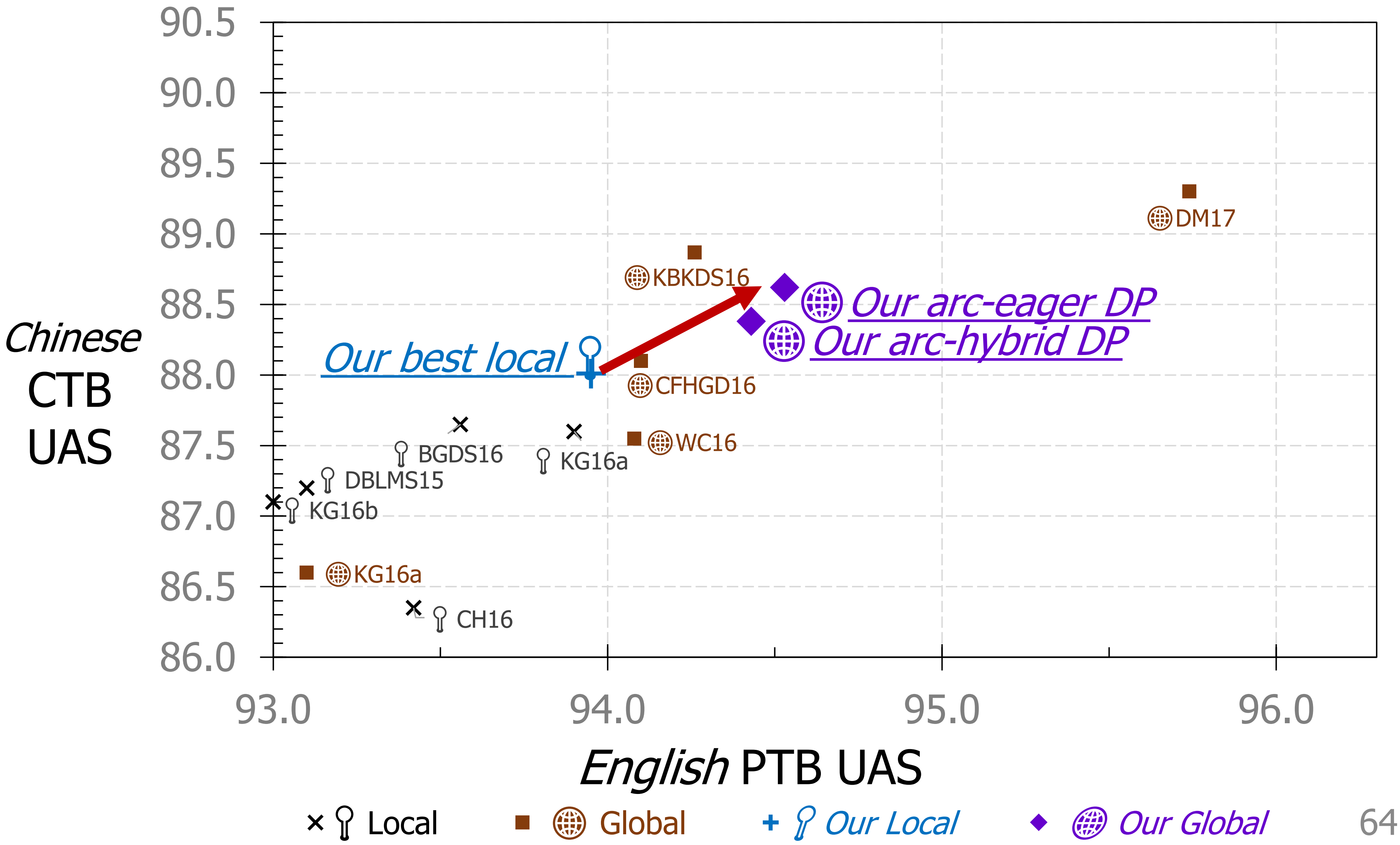
Comparing with State-of-the-art



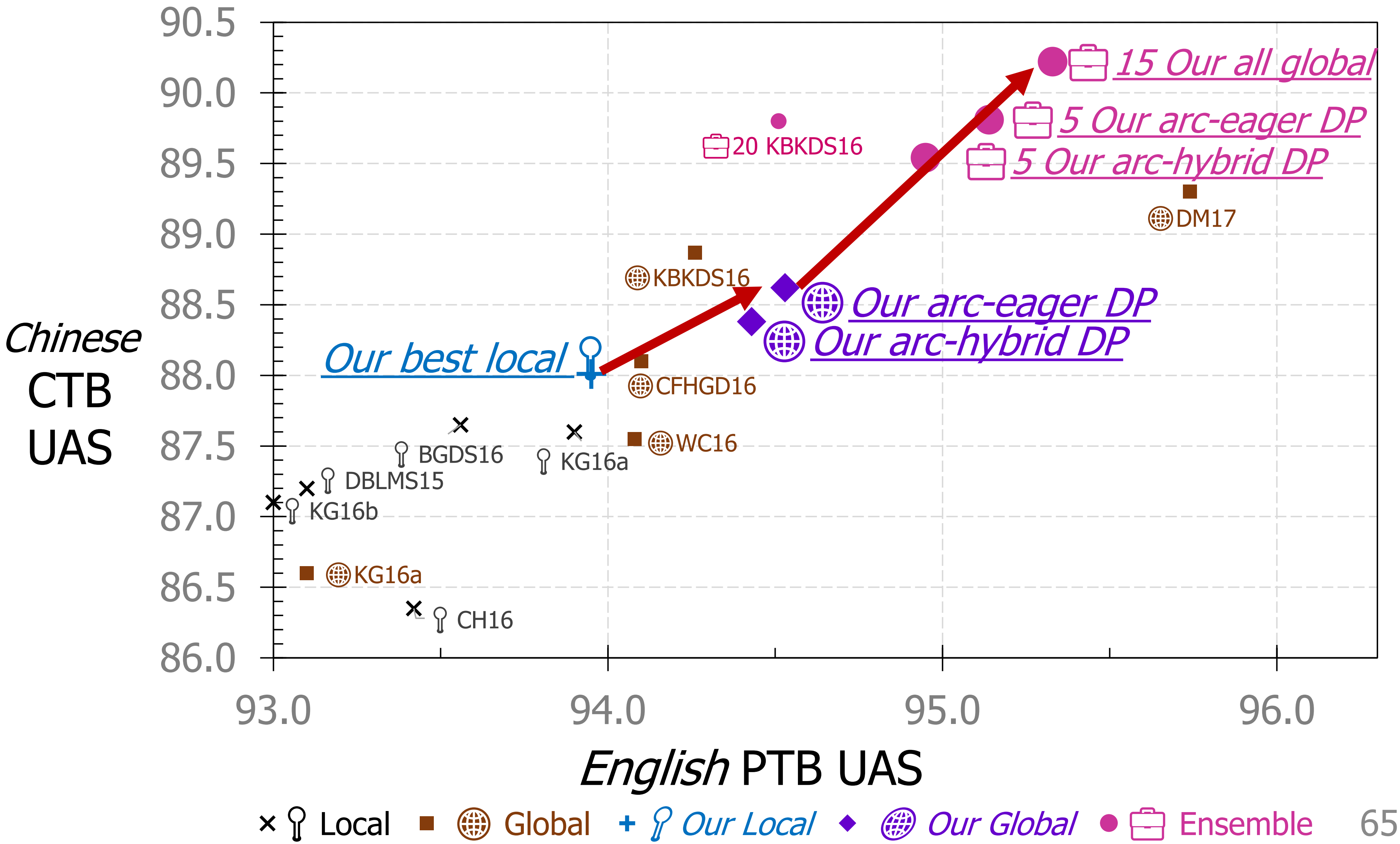
Comparing with State-of-the-art



Comparing with State-of-the-art

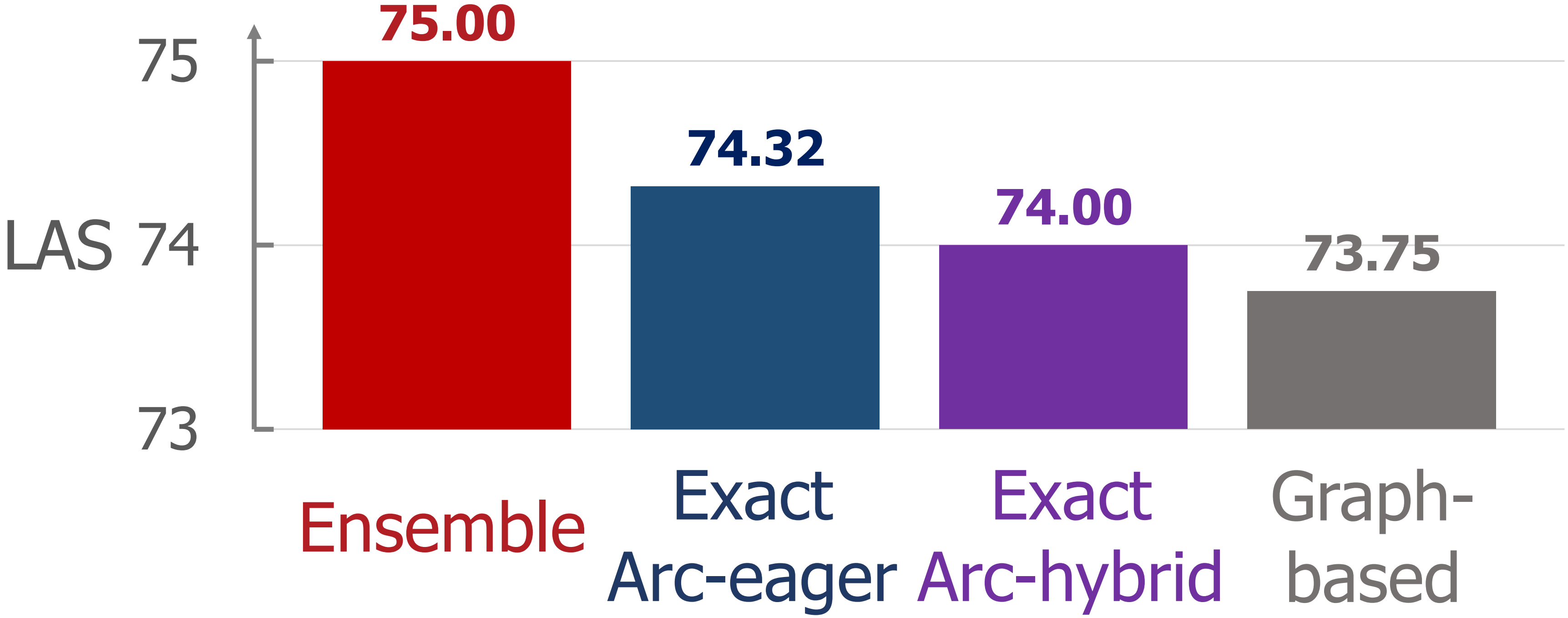


Comparing with State-of-the-art



Results – CoNLL’17 Shared Task

- Macro-average of 81 treebanks in 49 languages
- 2nd–highest overall performance



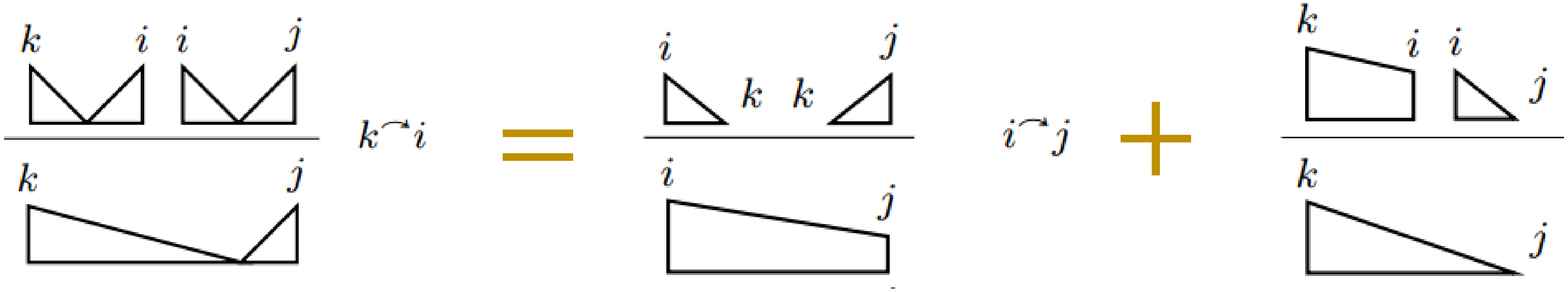
(Shi, Wu, Chen and Cheng, 2017; Zeman et al., 2017)

Conclusion

- Bi-LSTM feature set is minimal yet highly effective
- First $O(n^3)$ implementation of exact decoders
- Global training and decoding gave high performance

More in Our Paper

- Description and analysis of three transition systems (arc-standard, arc-hybrid, arc-eager)
- CKY-style representations of the deduction systems



- Theoretical analysis of the global methods
  - Arc-eager models can “simulate” arc-hybrid models
  - Arc-eager models can “simulate” edge-factored models

Fast(er) Exact Decoding and Global Training for Transition-Based Dependency Parsing via a Minimal Feature Set

https://github.com/tzshi/dp-parser-emnlp17

Tianze Shi*   Liang Huang†   Lillian Lee*

* Cornell University

† Oregon State University

CKY-style Visualization

Axiom $[0, 1]$

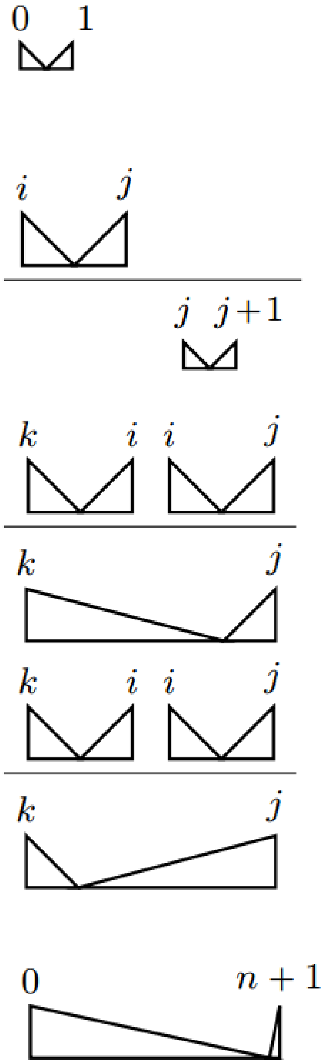
Inference Rules

sh $\dfrac{[i, j]}{[j, j+1]}$ $j \leqslant n$

re$_{\curvearrowright}$ $\dfrac{[k, i] \quad [i, j]}{[k, j]}$ $k \curvearrowright i$

re$_{\curvearrowleft}$ $\dfrac{[k, i] \quad [i, j]}{[k, j]}$ $i \curvearrowleft j$

Goal $[0, n+1]$



(b) Arc-hybrid

Axiom $[0^0, 1]$

Inference Rules

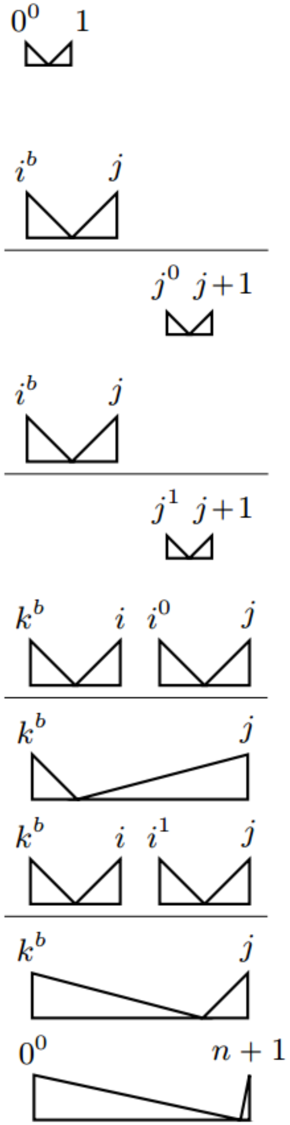
sh $\dfrac{[i^b, j]}{[j^0, j+1]}$ $j \leqslant n$

ra $\dfrac{[i^b, j]}{[j^1, j+1]}$ $i \curvearrowright j$ $j \leqslant n$

re$_{\curvearrowleft}$ $\dfrac{[k^b, i] \quad [i^0, j]}{[k^b, j]}$ $i \curvearrowleft j$

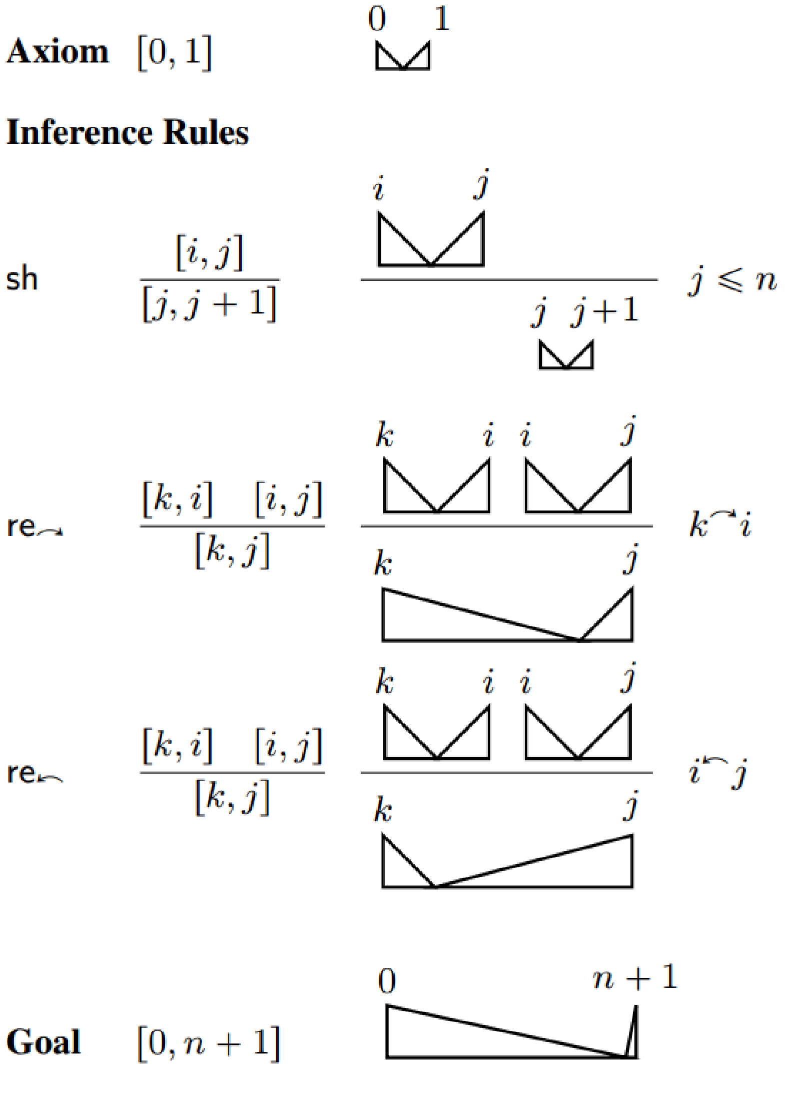
re $\dfrac{[k^b, i] \quad [i^1, j]}{[k^b, j]}$

Goal $[0^0, n+1]$

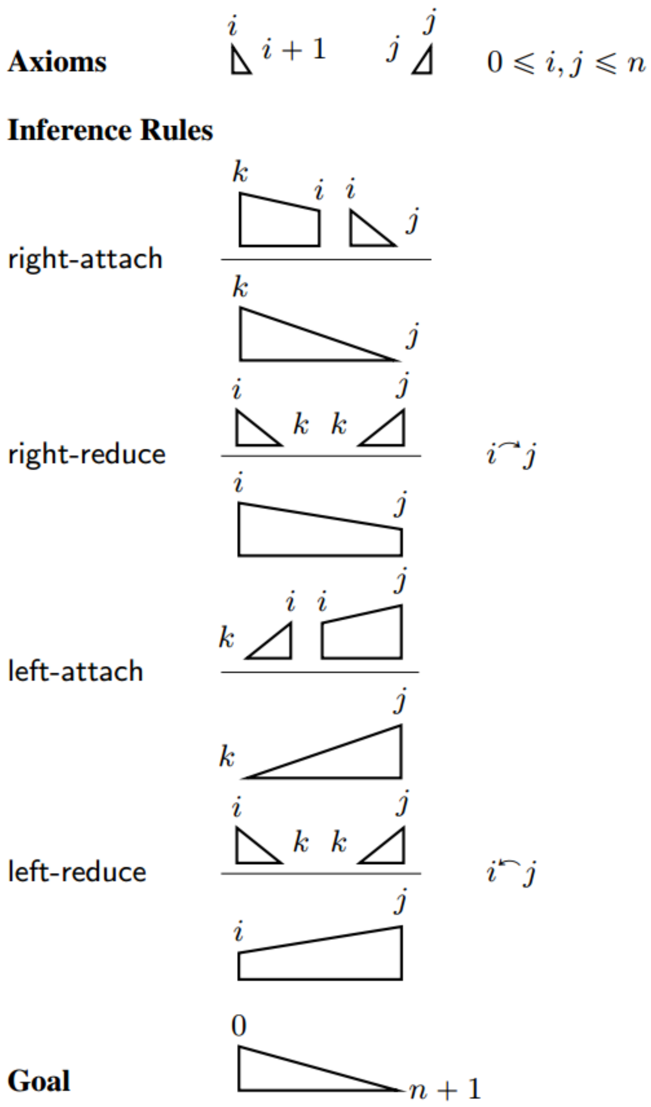


(c) Arc-eager

CKY-style Visualization

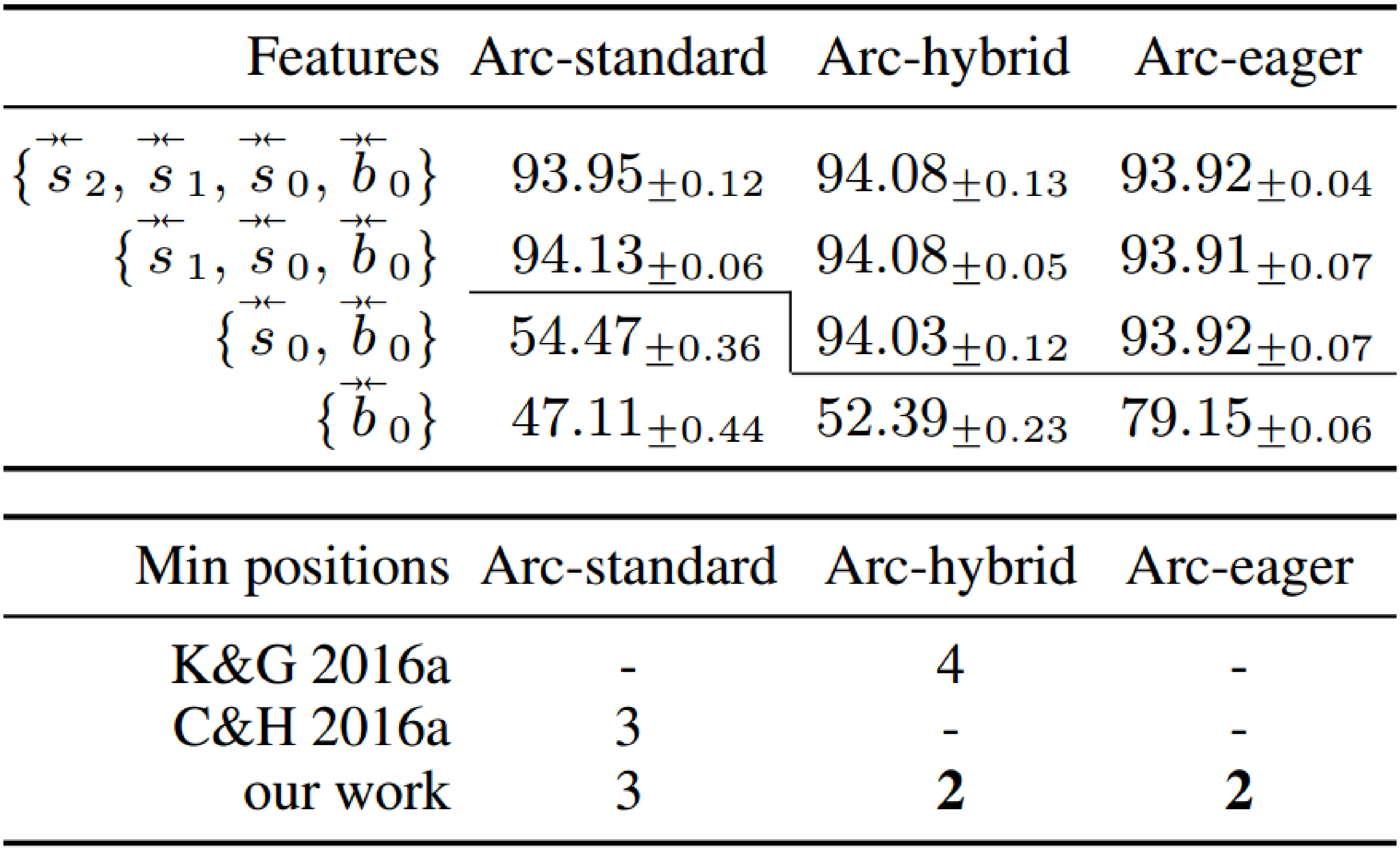


(b) Arc-hybrid



(d) Edge-factored graph-based parsing.

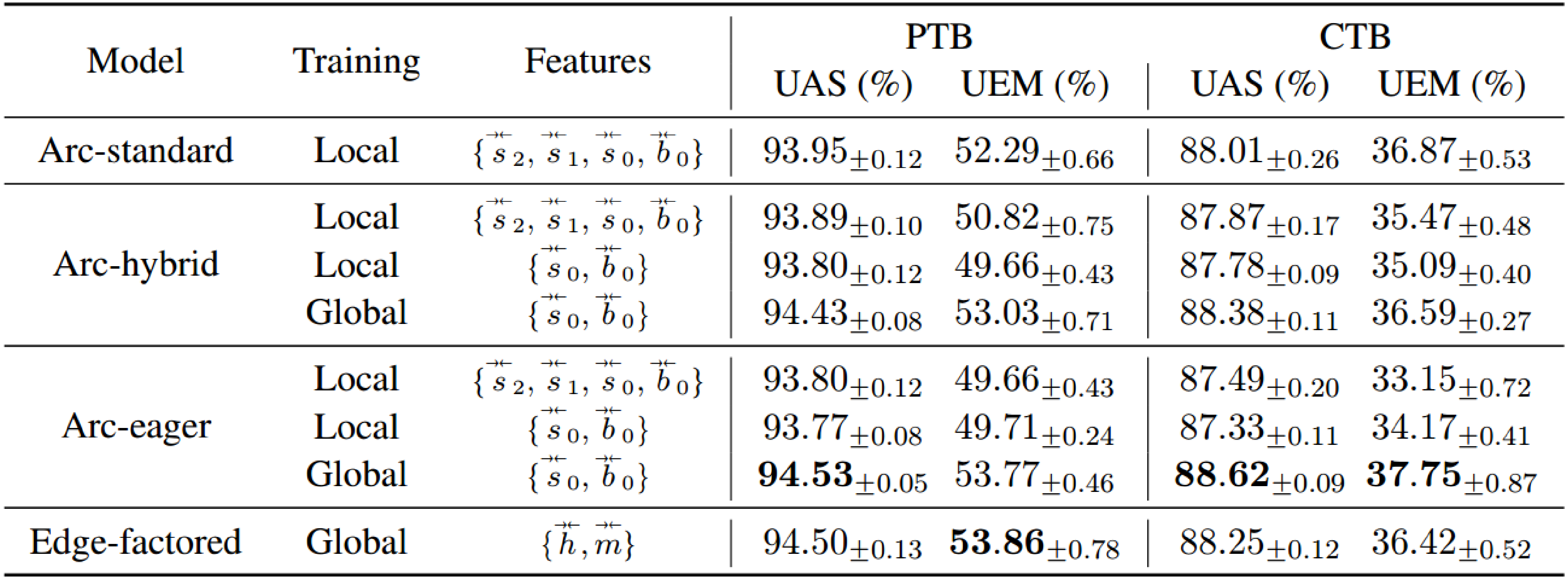
Results with Arc-eager and Arc-standard



Features	Arc-standard	Arc-hybrid	Arc-eager
$\{\overset{\rightarrow\leftarrow}{s}_2, \overset{\rightarrow\leftarrow}{s}_1, \overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.95_{\pm 0.12}$	$94.08_{\pm 0.13}$	$93.92_{\pm 0.04}$
$\{\overset{\rightarrow\leftarrow}{s}_1, \overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$94.13_{\pm 0.06}$	$94.08_{\pm 0.05}$	$93.91_{\pm 0.07}$
$\{\overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$54.47_{\pm 0.36}$	$94.03_{\pm 0.12}$	$93.92_{\pm 0.07}$
$\{\overset{\rightarrow\leftarrow}{b}_0\}$	$47.11_{\pm 0.44}$	$52.39_{\pm 0.23}$	$79.15_{\pm 0.06}$

Min positions	Arc-standard	Arc-hybrid	Arc-eager
K&G 2016a	-	4	-
C&H 2016a	3	-	-
our work	3	**2**	**2**

Results with Arc-eager and Arc-standard



Model	Training	Features	PTB UAS (%)	PTB UEM (%)	CTB UAS (%)	CTB UEM (%)
Arc-standard	Local	$\{\overset{\rightarrow\leftarrow}{s}_2, \overset{\rightarrow\leftarrow}{s}_1, \overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.95_{\pm 0.12}$	$52.29_{\pm 0.66}$	$88.01_{\pm 0.26}$	$36.87_{\pm 0.53}$
Arc-hybrid	Local	$\{\overset{\rightarrow\leftarrow}{s}_2, \overset{\rightarrow\leftarrow}{s}_1, \overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.89_{\pm 0.10}$	$50.82_{\pm 0.75}$	$87.87_{\pm 0.17}$	$35.47_{\pm 0.48}$
	Local	$\{\overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.80_{\pm 0.12}$	$49.66_{\pm 0.43}$	$87.78_{\pm 0.09}$	$35.09_{\pm 0.40}$
	Global	$\{\overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$94.43_{\pm 0.08}$	$53.03_{\pm 0.71}$	$88.38_{\pm 0.11}$	$36.59_{\pm 0.27}$
Arc-eager	Local	$\{\overset{\rightarrow\leftarrow}{s}_2, \overset{\rightarrow\leftarrow}{s}_1, \overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.80_{\pm 0.12}$	$49.66_{\pm 0.43}$	$87.49_{\pm 0.20}$	$33.15_{\pm 0.72}$
	Local	$\{\overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.77_{\pm 0.08}$	$49.71_{\pm 0.24}$	$87.33_{\pm 0.11}$	$34.17_{\pm 0.41}$
	Global	$\{\overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$\mathbf{94.53}_{\pm 0.05}$	$53.77_{\pm 0.46}$	$\mathbf{88.62}_{\pm 0.09}$	$\mathbf{37.75}_{\pm 0.87}$
Edge-factored	Global	$\{\overset{\rightarrow\leftarrow}{h}, \overset{\rightarrow\leftarrow}{m}\}$	$94.50_{\pm 0.13}$	$\mathbf{53.86}_{\pm 0.78}$	$88.25_{\pm 0.12}$	$36.42_{\pm 0.52}$