A Transition-Based Directed Acyclic Graph Parser for Universal Conceptual Cognitive Annotation

Daniel Hershcovich, Omri Abend and Ari Rappoport

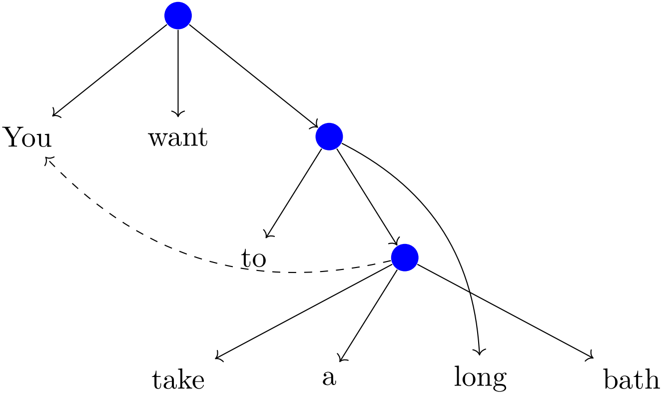
האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM

ACL 2017

TUPA — Transition-based UCCA Parser

The **first parser** to support the combination of three properties:

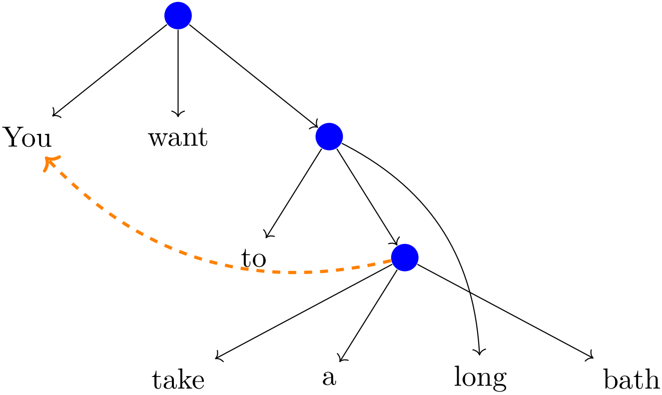
1. Non-terminal nodes — entities and events over the text



TUPA — Transition-based UCCA Parser

The **first parser** to support the combination of three properties:

1. Non-terminal nodes — entities and events over the text
2. Reentrancy — allow argument sharing

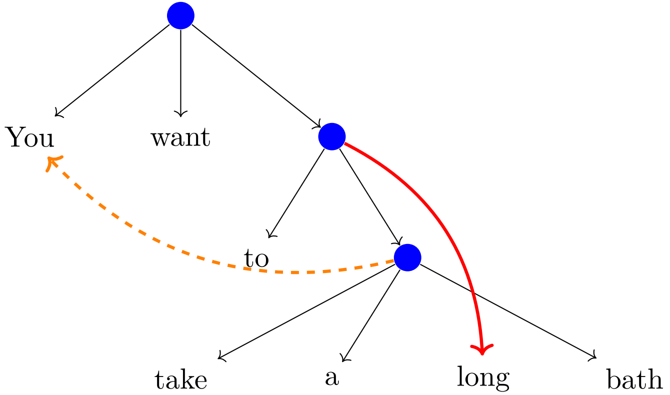


TUPA — Transition-based UCCA Parser

The **first parser** to support the combination of three properties:

1. Non-terminal nodes — entities and events over the text
2. Reentrancy — allow argument sharing
3. Discontinuity — conceptual units are split

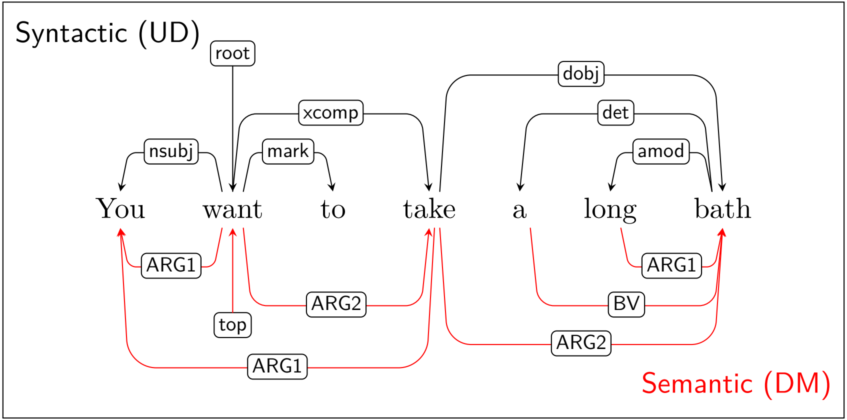
— needed for many semantic schemes (e.g. AMR, UCCA).



Introduction

Linguistic Structure Annotation Schemes

- Syntactic dependencies
- Semantic dependencies (Oepen et al., 2016)



Bilexical dependencies.

Linguistic Structure Annotation Schemes

- Syntactic dependencies
- Semantic dependencies (Oepen et al., 2016)
- Semantic role labeling (PropBank, FrameNet)
- AMR (Banarescu et al., 2013)
- UCCA (Abend and Rappoport, 2013)
- Other semantic representation schemes[1]

Semantic representation schemes attempt to abstract away from syntactic detail that does not affect meaning:

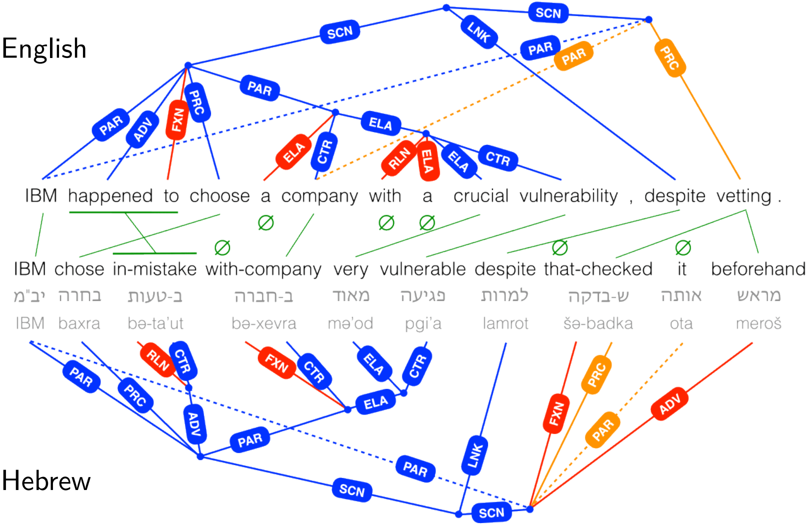
$$\boxed{\ldots\text{bathed}} = \boxed{\ldots\text{took a bath}}$$

[1]See recent survey (Abend and Rappoport, 2017)

The UCCA Semantic Representation Scheme

Universal Conceptual Cognitive Annotation (UCCA)

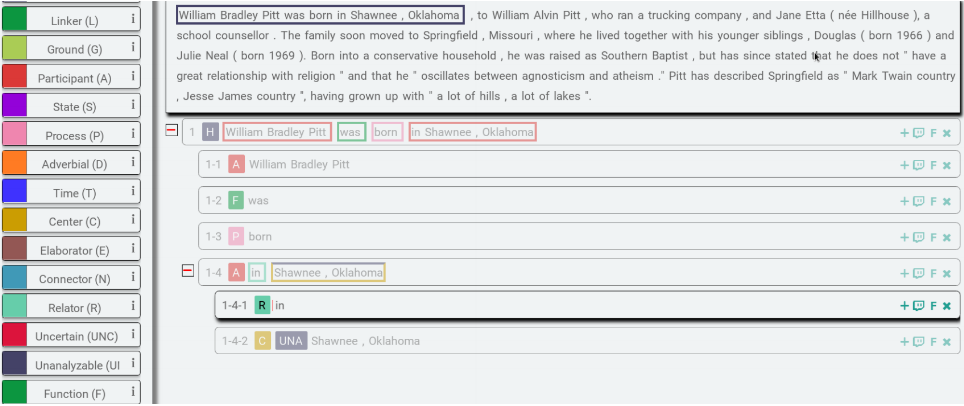
Cross-linguistically applicable (Abend and Rappoport, 2013).
Stable in translation (Sulem et al., 2015).



Universal Conceptual Cognitive Annotation (UCCA)

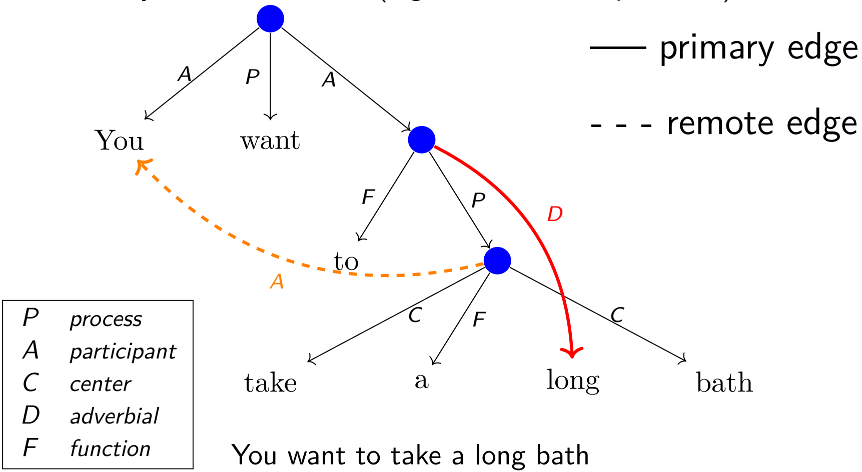
Rapid and intuitive annotation interface (Abend et al., 2017).
Usable by non-experts. `ucca-demo.cs.huji.ac.il`

Facilitates semantics-based human evaluation of machine translation (Birch et al., 2016). `ucca.cs.huji.ac.il/mteval`



Graph Structure

UCCA generates a directed acyclic graph (DAG).
Text tokens are terminals, complex units are non-terminal nodes.
Remote edges enable reentrancy for argument sharing.
Phrases may be discontinuous (e.g., multi-word expressions).



Transition-based UCCA Parsing

Transition-Based Parsing

First used for dependency parsing (Nivre, 2004).
Parse text $w_1 \ldots w_n$ to graph G incrementally by applying transitions to the parser state: stack, buffer and constructed graph.

Transition-Based Parsing

First used for dependency parsing (Nivre, 2004).
Parse text $w_1 \dots w_n$ to graph G incrementally by applying transitions to the parser state: stack, buffer and constructed graph.

Initial state:

stack

●

buffer

You	want	to	take	a	long	bath

Transition-Based Parsing

First used for dependency parsing (Nivre, 2004).
Parse text $w_1 \ldots w_n$ to graph G incrementally by applying transitions to the parser state: stack, buffer and constructed graph.

Initial state:

stack | buffer

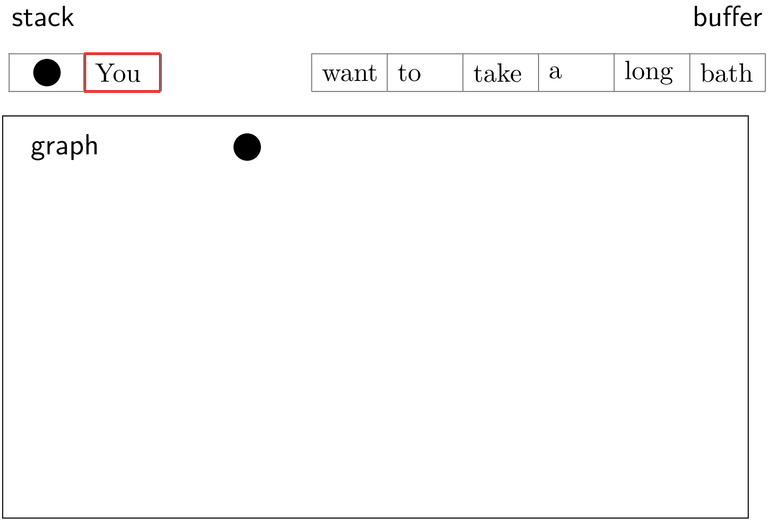
●		You	want	to	take	a	long	bath

TUPA transitions:
$\{$Shift, Reduce, Node$_X$, Left-Edge$_X$, Right-Edge$_X$, Left-Remote$_X$, Right-Remote$_X$, Swap, Finish$\}$

Support non-terminal nodes, reentrancy and discontinuity.

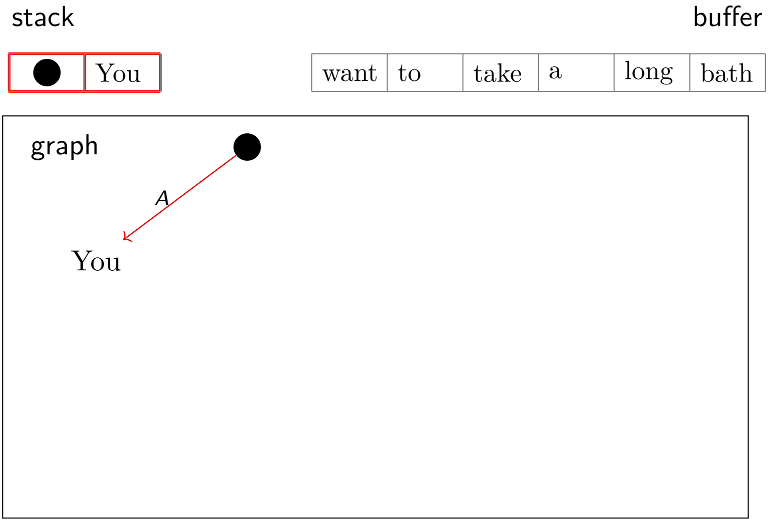
Example

$\Rightarrow$ SHIFT



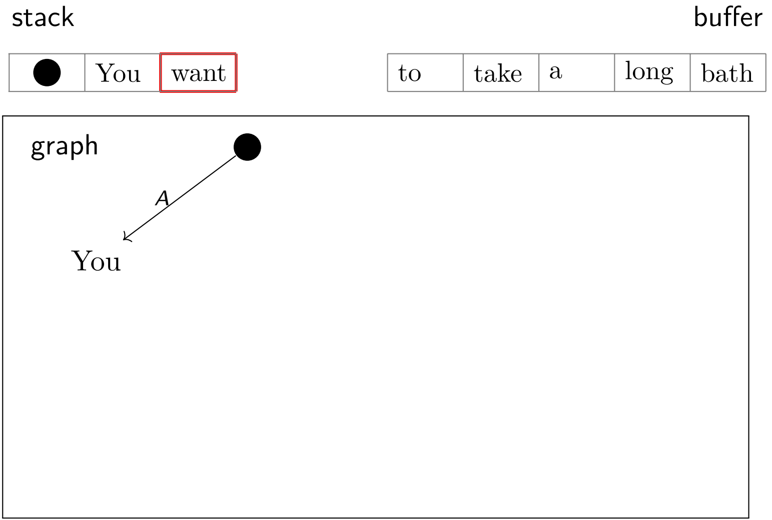
Example

$\Rightarrow$ RIGHT-EDGE$_A$



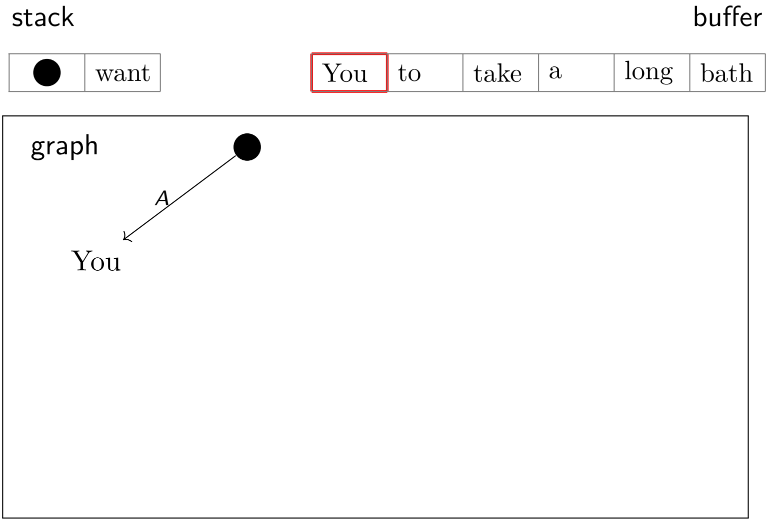
Example

⇒ Shift



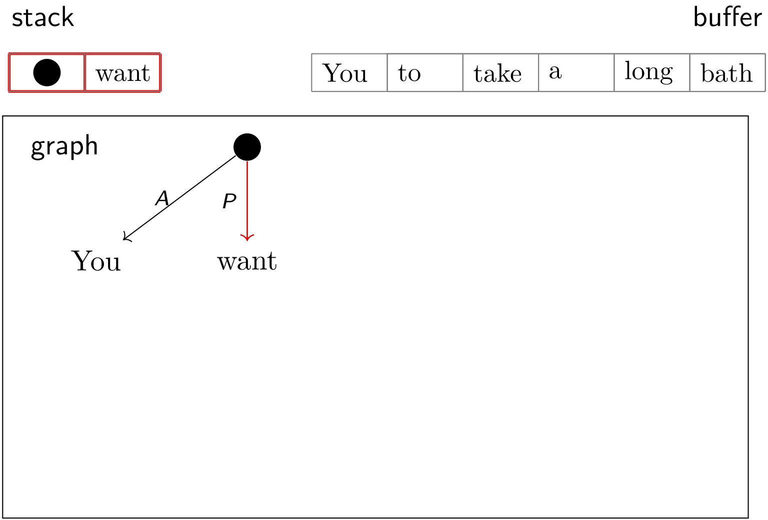
Example

$\Rightarrow$ SWAP



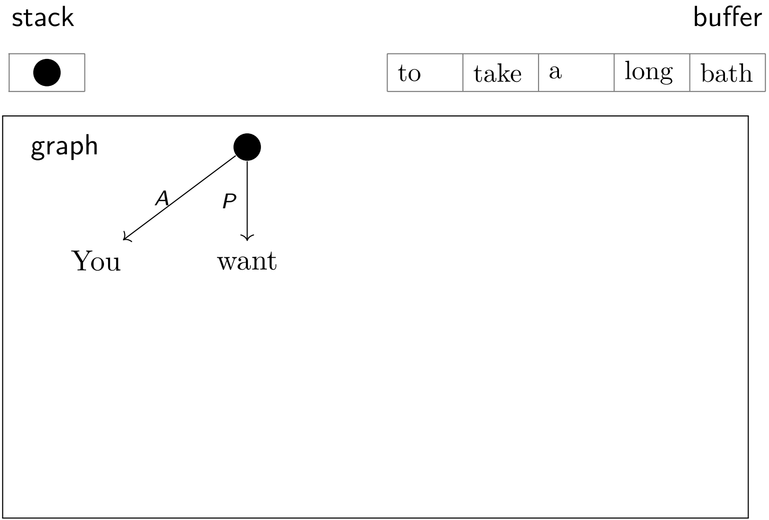
Example

$\Rightarrow$ RIGHT-EDGE$_P$



Example

⇒ REDUCE



Example

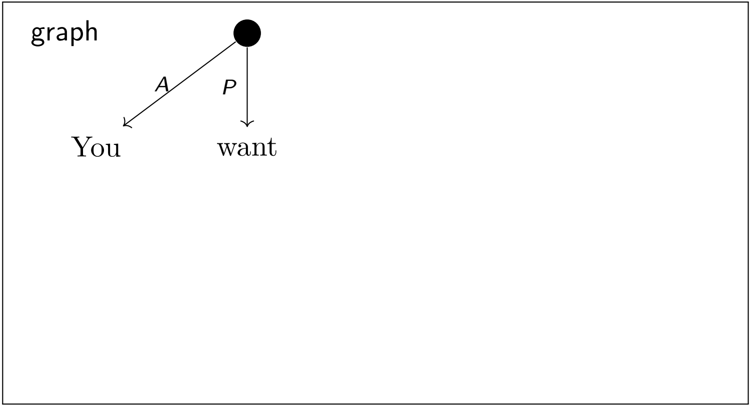
$\Rightarrow$ SHIFT

stack

●	You

buffer

to	take	a	long	bath



Example

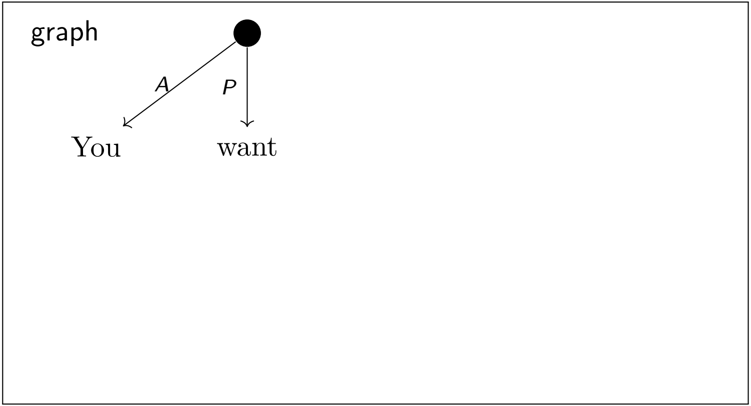
$\Rightarrow$ SHIFT

stack

●	You	to

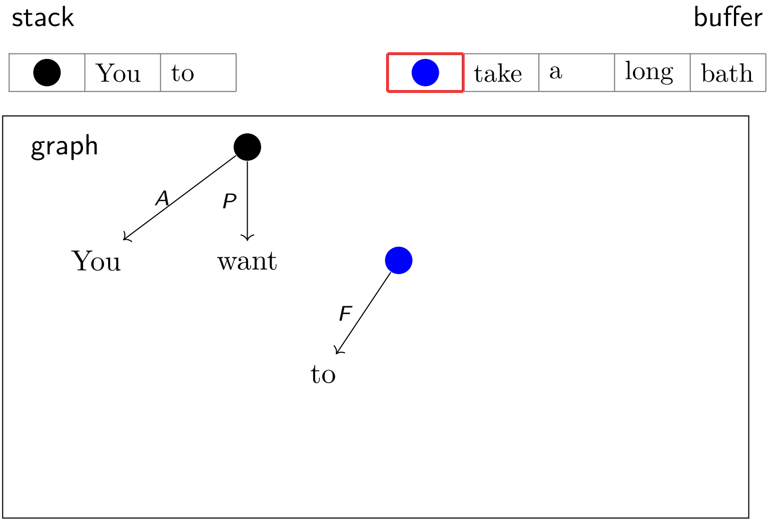
buffer

take	a	long	bath



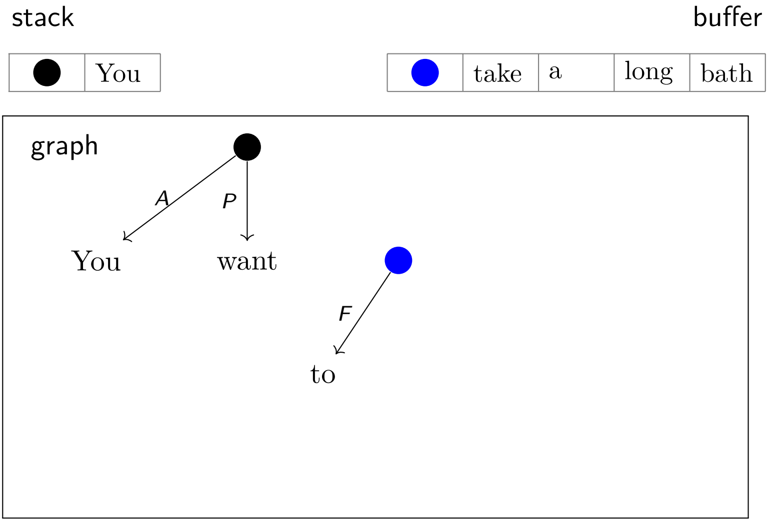
Example

$\Rightarrow$ $\textsc{Node}_F$



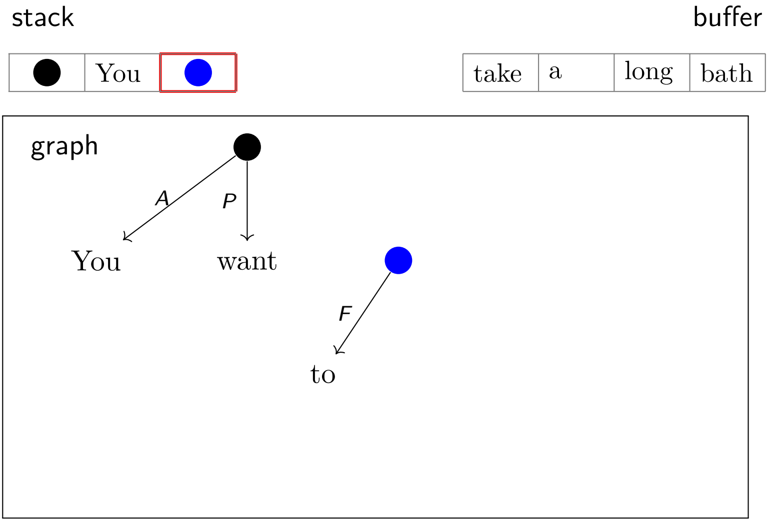
Example

⇒ REDUCE



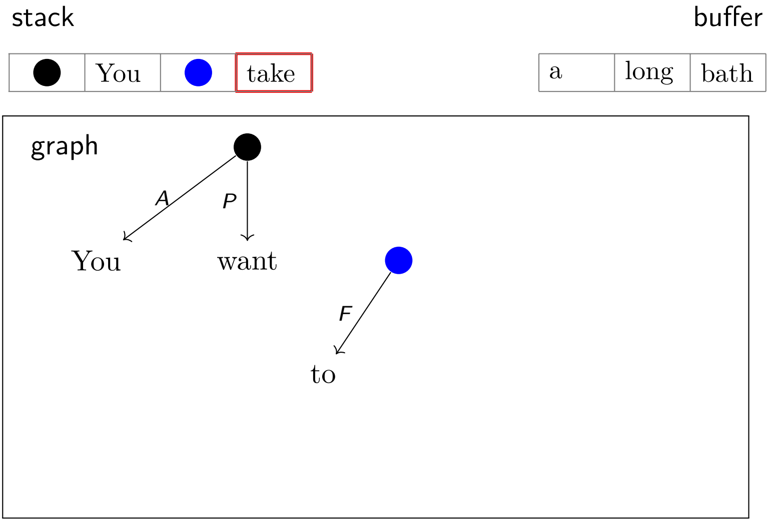
Example

⇒ Shift



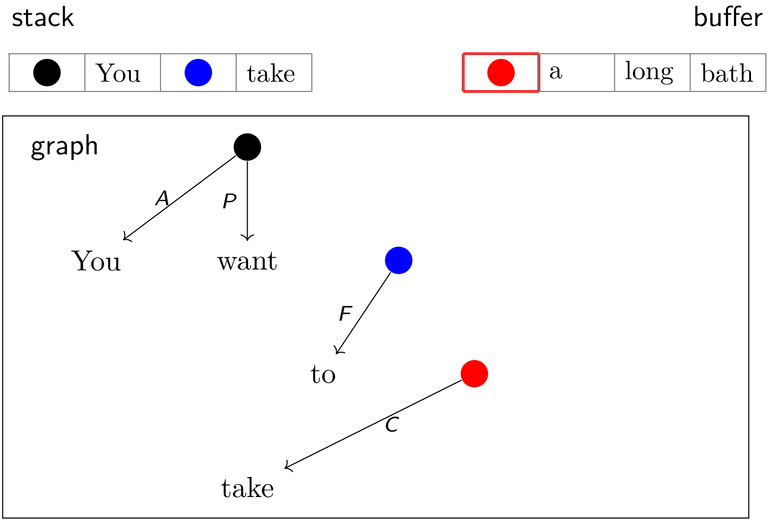
Example

$\Rightarrow$ Shift



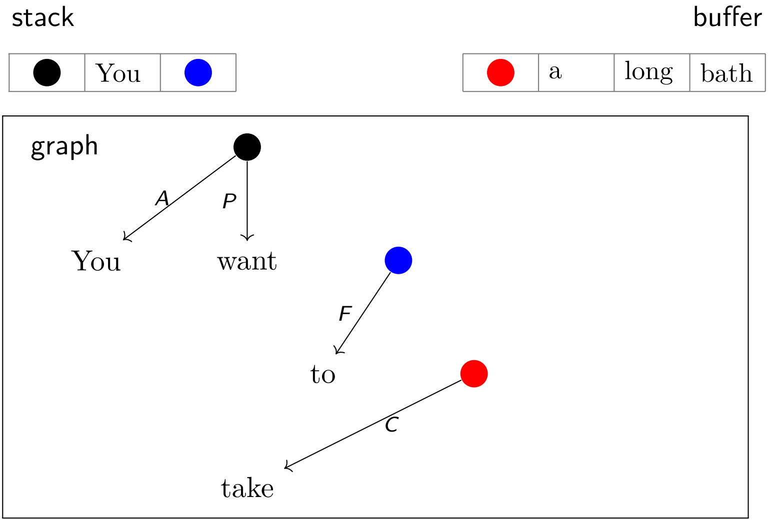
Example

$\Rightarrow$ NODE$_C$



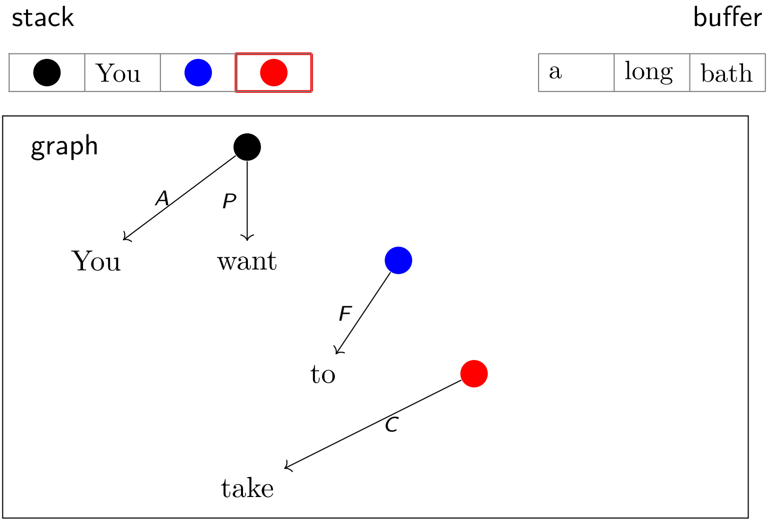
Example

$\Rightarrow$ Reduce



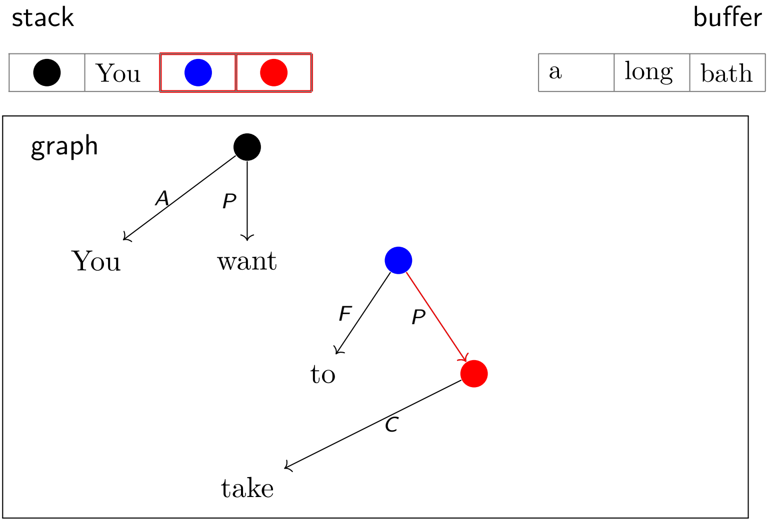
Example

$\Rightarrow$ Shift



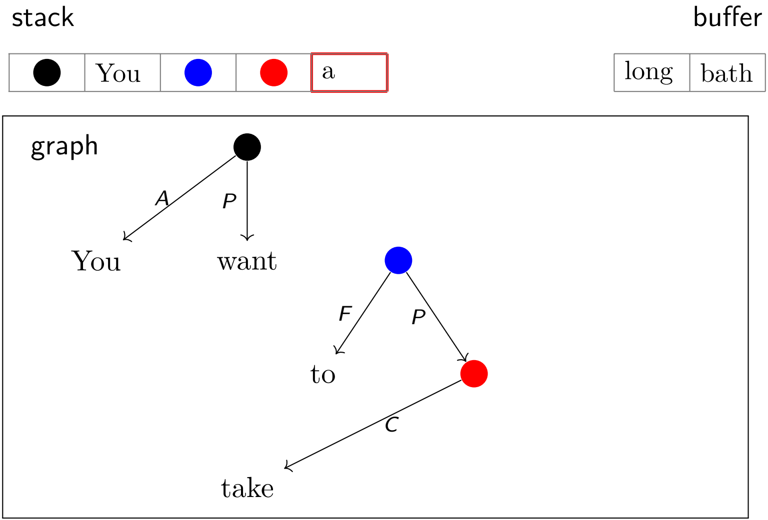
Example

$\Rightarrow$ RIGHT-EDGE$_P$



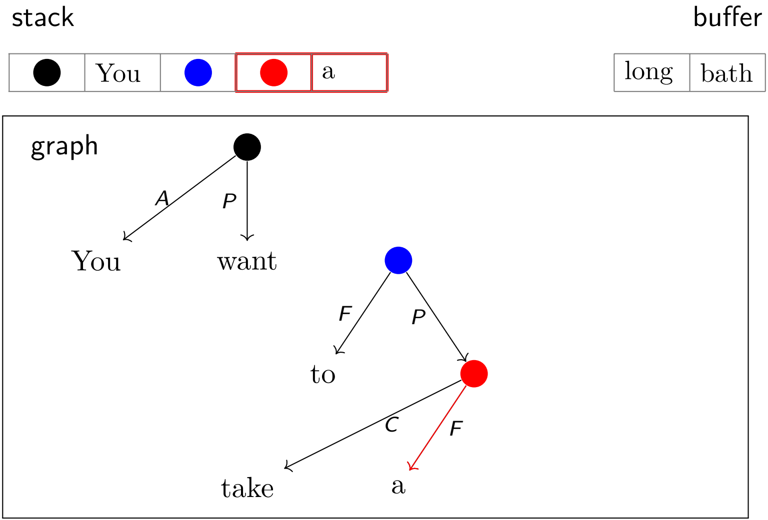
Example

$\Rightarrow$ Shift



Example

$\Rightarrow$ RIGHT-EDGE$_F$



Example

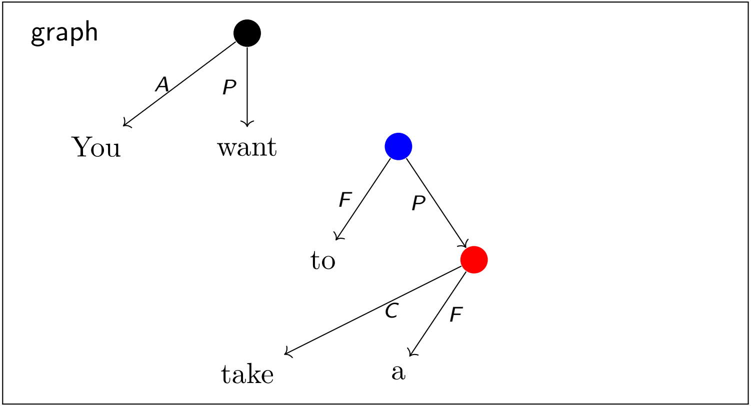
$\Rightarrow$ Reduce

stack

●	You	●	●

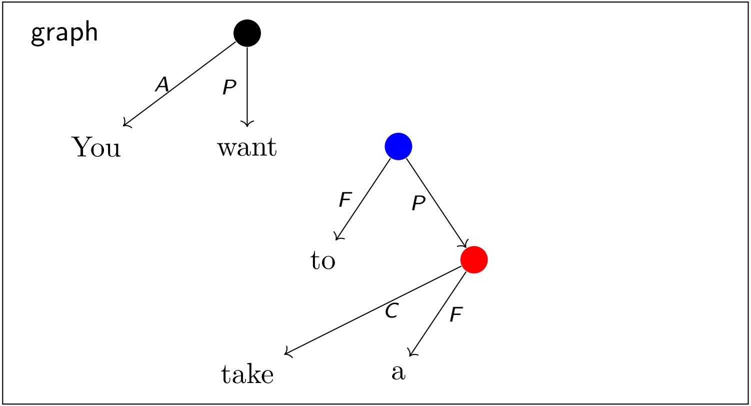
buffer

long	bath



Example

$\Rightarrow$ Shift



Example

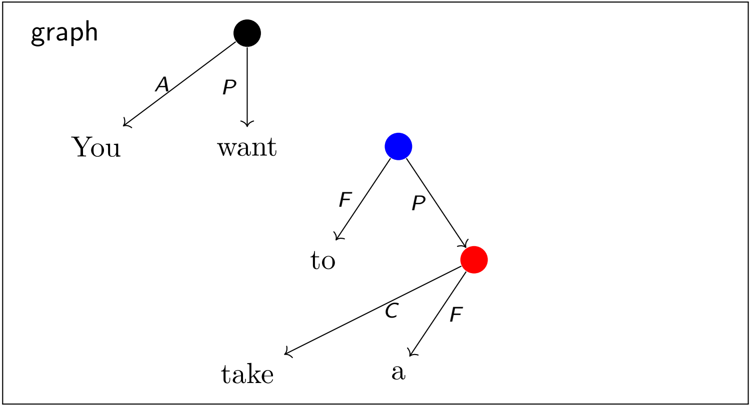
$\Rightarrow$ SWAP

stack

●	You	●	long

buffer

●	bath



Example

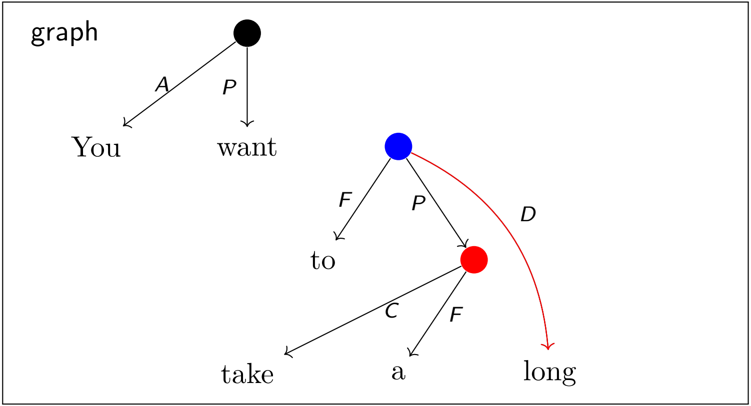
⇒ RIGHT-EDGE$_D$

stack

●	You	●	long

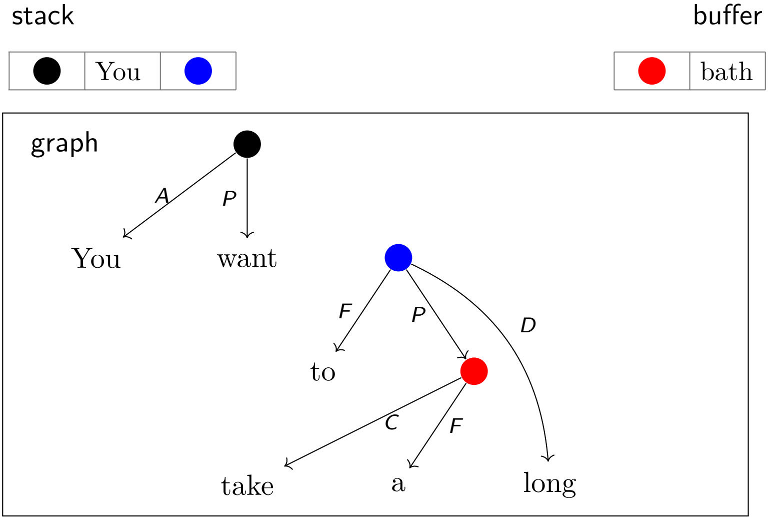
buffer

●	bath



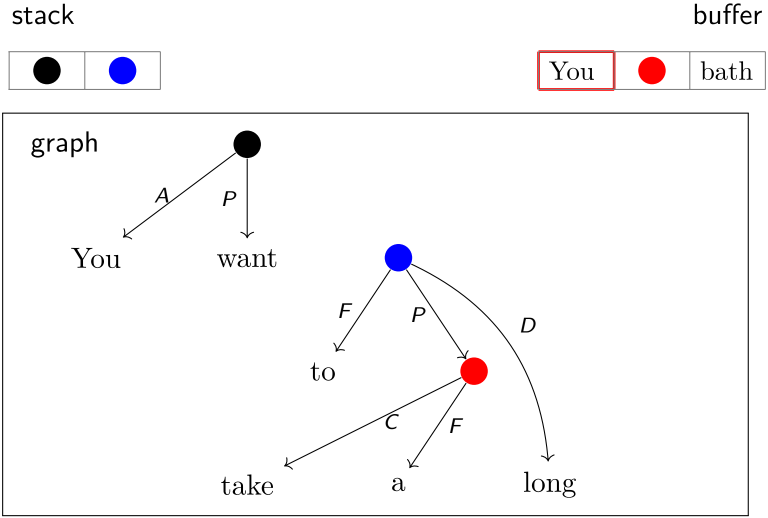
Example

⇒ REDUCE



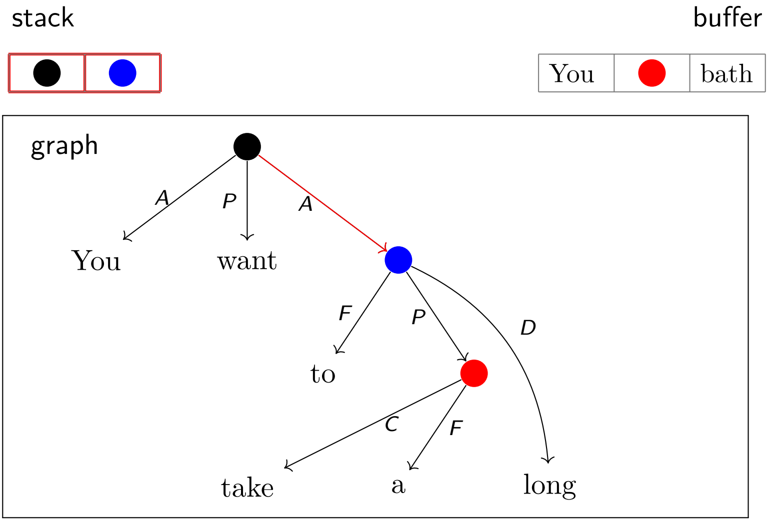
Example

⇒ Swap



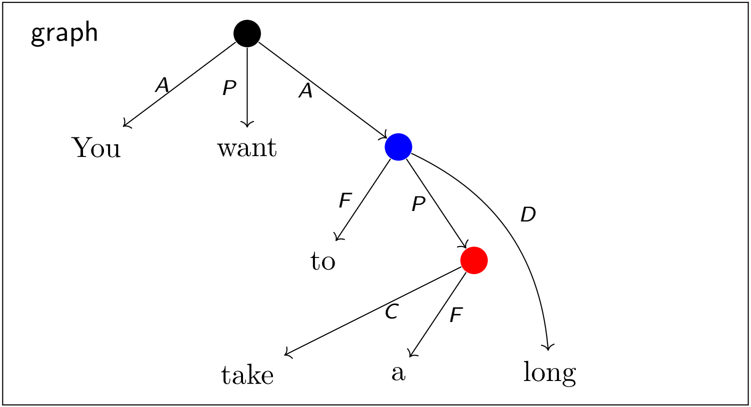
Example

$\Rightarrow$ Right-Edge$_A$



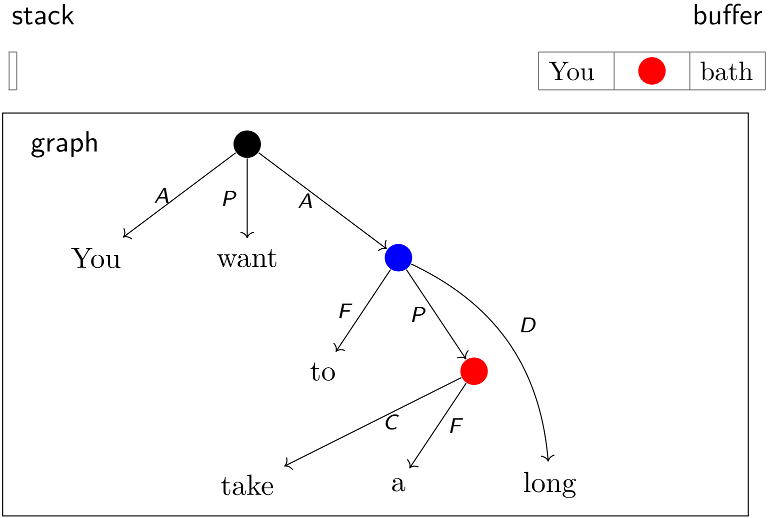
Example

$\Rightarrow$ REDUCE



Example

⇒ REDUCE



Example

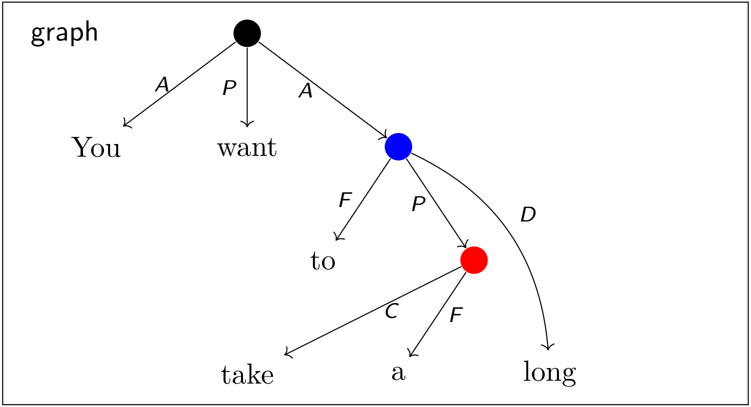
$\Rightarrow$ SHIFT

stack

You

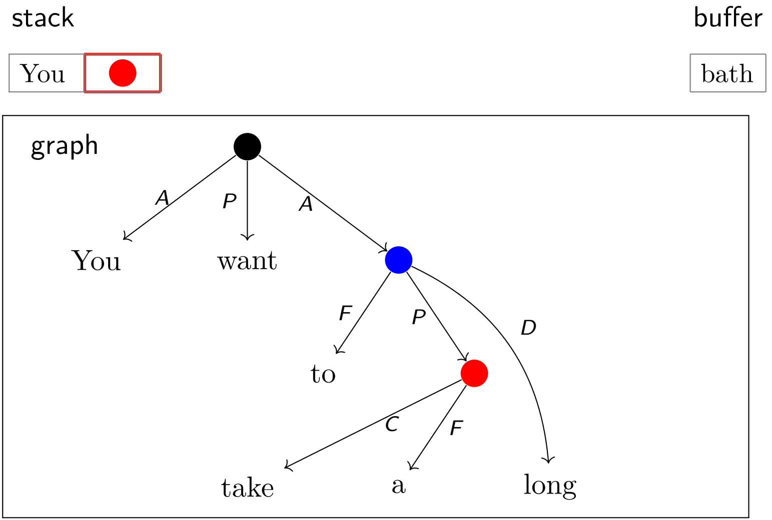
buffer

	bath



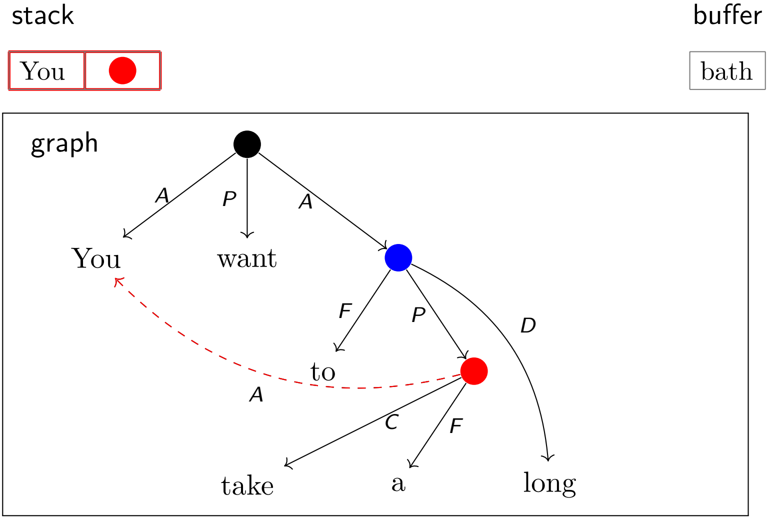
Example

$\Rightarrow$ SHIFT



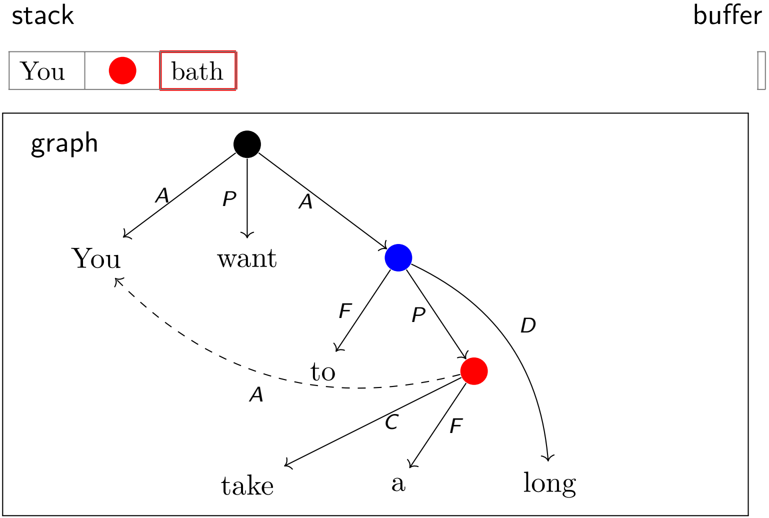
Example

$\Rightarrow$ LEFT-REMOTE$_A$



Example

⇒ Shift

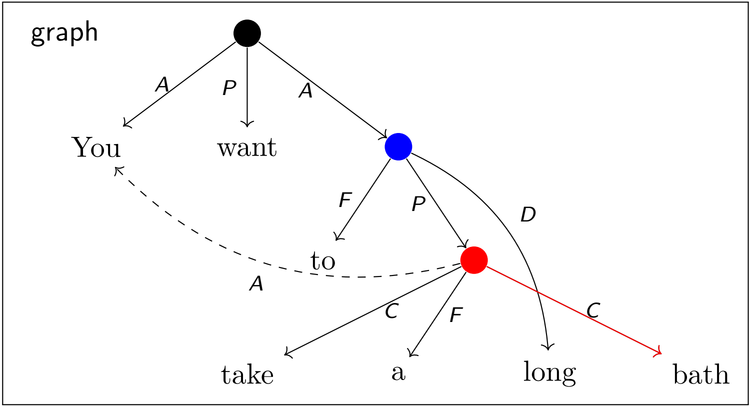


Example

$\Rightarrow$ Right-Edge$_C$

stack

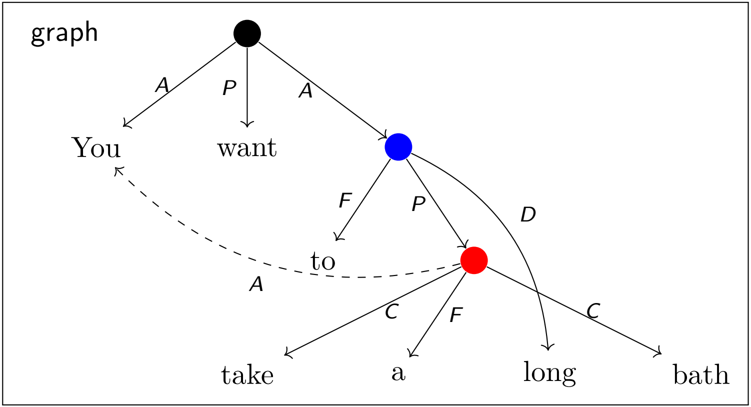
buffer



Example

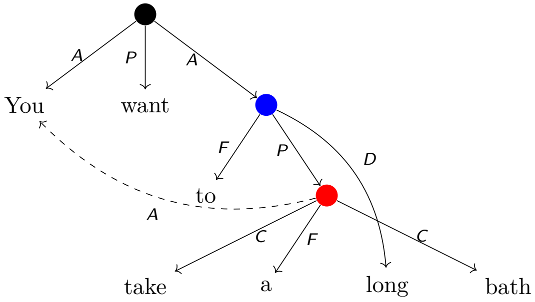
⇒ FINISH

stack buffer



Training

An *oracle* provides the transition sequence given the correct graph:



$\Downarrow$

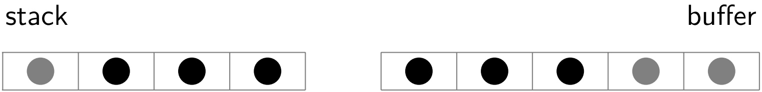
Shift, Right-Edge$_A$, Shift, Swap, Right-Edge$_P$, Reduce, Shift, Shift, Node$_F$, Reduce, Shift, Shift, Node$_C$, Reduce, Shift, Right-Edge$_P$, Shift, Right-Edge$_F$, Reduce, Shift, Swap, Right-Edge$_D$, Reduce, Swap, Right-Edge$_A$, Reduce, Reduce, Shift, Shift, Left-Remote$_A$, Shift, Right-Edge$_C$, Finish

TUPA Model

Learn to greedily predict transition based on current state.
Experimenting with three classifiers:

Sparse	Perceptron with sparse features (Zhang and Nivre, 2011).
MLP	Embeddings + feedforward NN (Chen and Manning, 2014).
BiLSTM	Embeddings + deep bidirectional LSTM + MLP (Kiperwasser and Goldberg, 2016).

Features: words, POS, syntactic dependencies, existing edge labels from the stack and buffer + parents, children, grandchildren; ordinal features (height, number of parents and children)

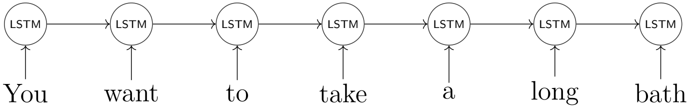


TUPA Model

Learn to greedily predict transition based on current state.
Experimenting with three classifiers:

Sparse	Perceptron with sparse features (Zhang and Nivre, 2011).
MLP	Embeddings + feedforward NN (Chen and Manning, 2014).
BiLSTM	Embeddings + **deep bidirectional LSTM** + MLP (Kiperwasser and Goldberg, 2016).

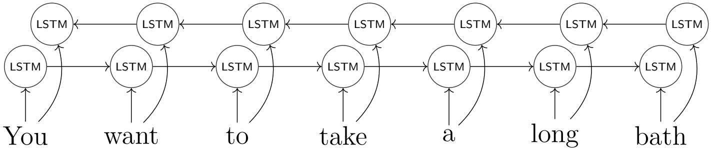
Effective "lookahead" encoded in the representation.



TUPA Model

Learn to greedily predict transition based on current state.
Experimenting with three classifiers:

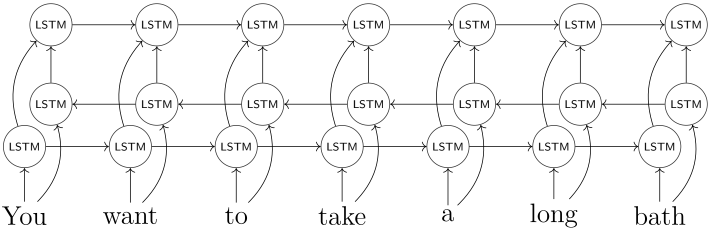
Sparse	Perceptron with sparse features (Zhang and Nivre, 2011).
MLP	Embeddings + feedforward NN (Chen and Manning, 2014).
BiLSTM	Embeddings + **deep bidirectional LSTM** + MLP (Kiperwasser and Goldberg, 2016).



TUPA Model

Learn to greedily predict transition based on current state.
Experimenting with three classifiers:

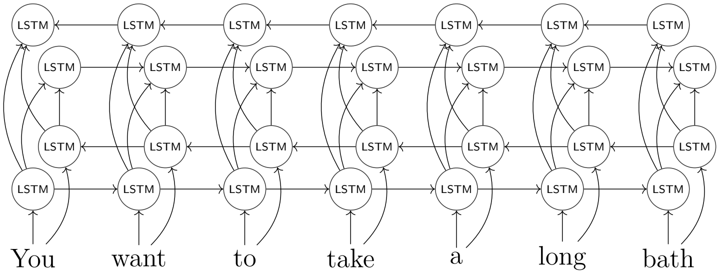
Sparse	Perceptron with sparse features (Zhang and Nivre, 2011).
MLP	Embeddings + feedforward NN (Chen and Manning, 2014).
BiLSTM	Embeddings + **deep bidirectional LSTM** + MLP (Kiperwasser and Goldberg, 2016).

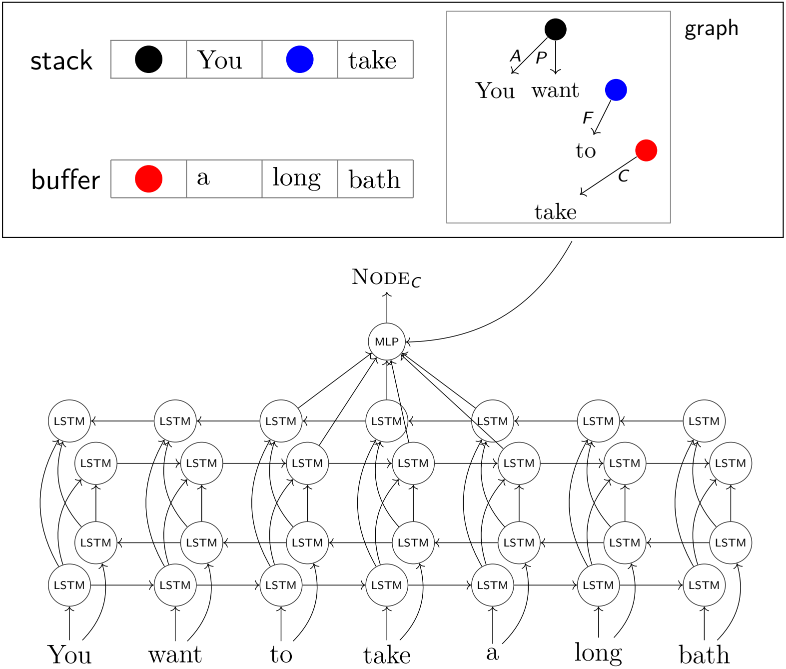


TUPA Model

Learn to greedily predict transition based on current state.
Experimenting with three classifiers:

Sparse	Perceptron with sparse features (Zhang and Nivre, 2011).
MLP	Embeddings + feedforward NN (Chen and Manning, 2014).
BiLSTM	Embeddings + **deep bidirectional LSTM** + MLP (Kiperwasser and Goldberg, 2016).

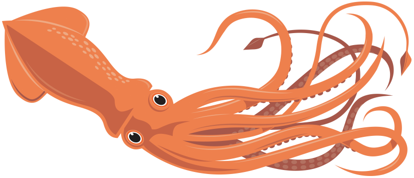



stack
You
take
buffer
a
long
bath
graph
A
P
You
want
F
to
C
take
NODE$_C$
MLP
LSTM
You
want
to
take
a
long
bath

Experiments

Experimental Setup

- UCCA Wikipedia corpus ($\overset{\text{train}}{4268} + \overset{\text{dev}}{454} + \overset{\text{test}}{503}$ sentences).
- Out-of-domain: English part of English-French parallel corpus, *Twenty Thousand Leagues Under the Sea* (506 sentences).



Baselines

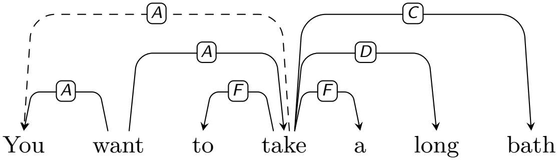
No existing UCCA parsers ⇒ conversion-based approximation.

Bilexical DAG parsers (allow reentrancy):

- DAGParser (Ribeyre et al., 2014): transition-based.
- TurboParser (Almeida and Martins, 2015): graph-based.

Tree parsers (all transition-based):

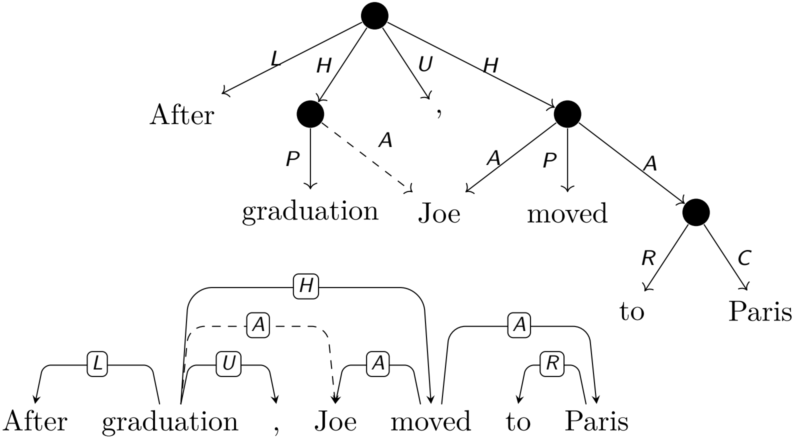
- MaltParser (Nivre et al., 2007): bilexical tree parser.
- Stack LSTM Parser (Dyer et al., 2015): bilexical tree parser.
- UPARSE (Maier, 2015): allows non-terminals, discontinuity.



UCCA bilexical DAG approximation (for tree, delete remote edges).

Bilexical Graph Approximation

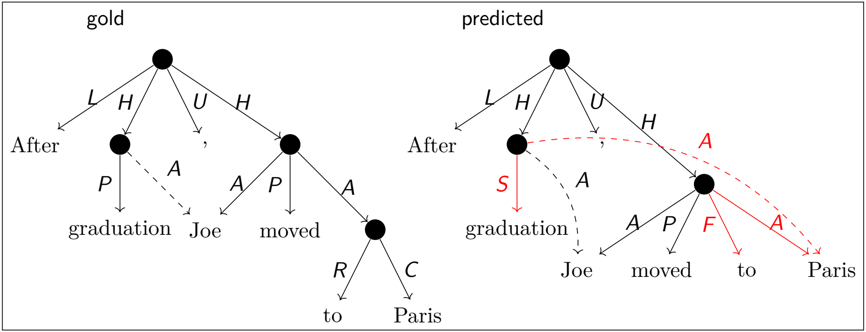
1. Convert UCCA to bilexical dependencies.
2. Train bilexical parsers and apply to test sentences.
3. Reconstruct UCCA graphs and compare with gold standard.



Evaluation

Comparing graphs over the same sequence of tokens,

- Match edges by their terminal yield and label.
- Calculate **labeled precision, recall and F1** scores.
- Separate primary and remote edges.



Primary:

LP	LR	LF
$\frac{6}{9} = 67\%$	$\frac{6}{10} = 60\%$	64%

Remote:

LP	LR	LF
$\frac{1}{2} = 50\%$	$\frac{1}{1} = 100\%$	67%

Results

TUPA$_{\text{BiLSTM}}$ obtains the highest F-scores in all metrics:

	Primary edges			Remote edges		
	LP	**LR**	**LF**	**LP**	**LR**	**LF**
TUPA$_{\text{Sparse}}$	64.5	63.7	64.1	19.8	13.4	16
TUPA$_{\text{MLP}}$	65.2	64.6	64.9	23.7	13.2	16.9
TUPA$_{\text{BiLSTM}}$	74.4	72.7	**73.5**	47.4	51.6	**49.4**
Bilexical DAG			(91)			(58.3)
DAGParser	61.8	55.8	58.6	9.5	0.5	1
TurboParser	57.7	46	51.2	77.8	1.8	3.7
Bilexical tree			(91)			–
MaltParser	62.8	57.7	60.2	–	–	–
Stack LSTM	73.2	66.9	69.9	–	–	–
Tree			(100)			–
UPARSE	60.9	61.2	61.1	–	–	–

Results on the Wiki test set.

Results

Comparable on out-of-domain test set:

	Primary edges			Remote edges		
	LP	**LR**	**LF**	**LP**	**LR**	**LF**
$\text{TUPA}_{\text{Sparse}}$	59.6	59.9	59.8	22.2	7.7	11.5
TUPA_{MLP}	62.3	62.6	62.5	20.9	6.3	9.7
$\text{TUPA}_{\text{BiLSTM}}$	68.7	68.5	**68.6**	38.6	18.8	**25.3**
Bilexical DAG			(91.3)			(43.4)
DAGParser	56.4	50.6	53.4	–	0	0
TurboParser	50.3	37.7	43.1	100	0.4	0.8
Bilexical tree			(91.3)			–
MaltParser	57.8	53	55.3	–	–	–
Stack LSTM	66.1	61.1	63.5	–	–	–
Tree			(100)			–
UPARSE	52.7	52.8	52.8	–	–	–

Results on the 20K Leagues out-of-domain set.

Conclusion

- UCCA's semantic distinctions require a graph structure including non-terminals, reentrancy and discontinuity.
- TUPA is an accurate transition-based UCCA parser, and the **first** to support UCCA and any DAG over the text tokens.
- Outperforms strong conversion-based baselines.

Code: `github.com/danielhers/tupa`
Demo: `bit.ly/tupademo`
Corpora: `cs.huji.ac.il/~oabend/ucca.html`

Conclusion

- UCCA's semantic distinctions require a graph structure including non-terminals, reentrancy and discontinuity.
- TUPA is an accurate transition-based UCCA parser, and the **first** to support UCCA and any DAG over the text tokens.
- Outperforms strong conversion-based baselines.

Future Work:

- More languages (German corpus construction is underway).
- Parsing other schemes, such as AMR.
- Compare semantic representations through conversion.
- Text simplification, MT evaluation and other applications.

Code: `github.com/danielhers/tupa`
Demo: `bit.ly/tupademo`
Corpora: `cs.huji.ac.il/~oabend/ucca.html`

Conclusion

- UCCA's semantic distinctions require a graph structure including non-terminals, reentrancy and discontinuity.
- TUPA is an accurate transition-based UCCA parser, and the **first** to support UCCA and any DAG over the text tokens.
- Outperforms strong conversion-based baselines.

Future Work:

- More languages (German corpus construction is underway).
- Parsing other schemes, such as AMR.
- Compare semantic representations through conversion.
- Text simplification, MT evaluation and other applications.

Code: `github.com/danielhers/tupa`
Demo: `bit.ly/tupademo`
Corpora: `cs.huji.ac.il/~oabend/ucca.html`

Thank you!

References I

Abend, O. and Rappoport, A. (2013).
Universal Conceptual Cognitive Annotation (UCCA).
In *Proc. of ACL*, pages 228–238.

Abend, O. and Rappoport, A. (2017).
The state of the art in semantic representation.
In *Proc. of ACL*.
to appear.

Abend, O., Yerushalmi, S., and Rappoport, A. (2017).
UCCAApp: Web-application for syntactic and semantic phrase-based annotation.
In *Proc. of ACL: System Demonstration Papers*.
to appear.

Almeida, M. S. C. and Martins, A. F. T. (2015).
Lisbon: Evaluating TurboSemanticParser on multiple languages and out-of-domain data.
In *Proc. of SemEval*, pages 970–973.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Palmer, M., and Schneider, N. (2013).
Abstract Meaning Representation for sembanking.
In *Proc. of the Linguistic Annotation Workshop*.

Birch, A., Abend, O., Bojar, O., and Haddow, B. (2016).
HUME: Human UCCA-based evaluation of machine translation.
In *Proc. of EMNLP*, pages 1264–1274.

Chen, D. and Manning, C. (2014).
A fast and accurate dependency parser using neural networks.
In *Proc. of EMNLP*, pages 740–750.

References II

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015).
Transition-based dependeny parsing with stack long short-term memory.
In *Proc. of ACL*, pages 334–343.

Kiperwasser, E. and Goldberg, Y. (2016).
Simple and accurate dependency parsing using bidirectional LSTM feature representations.
TACL, 4:313–327.

Maier, W. (2015).
Discontinuous incremental shift-reduce parsing.
In *Proc. of ACL*, pages 1202–1212.

Nivre, J. (2004).
Incrementality in deterministic dependency parsing.
In Keller, F., Clark, S., Crocker, M., and Steedman, M., editors, *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain. Association for Computational Linguistics.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007).
MaltParser: A language-independent system for data-driven dependency parsing.
Natural Language Engineering, 13(02):95–135.

Oepen, S., Kuhlmann, M., Miyao, Y., Zeman, D., Cinková, S., Flickinger, D., Hajic, J., Ivanova, A., and Uresová, Z. (2016).
Towards comparability of linguistic graph banks for semantic parsing.
In *LREC*.

Ribeyre, C., Villemonte de la Clergerie, E., and Seddah, D. (2014).
Alpage: Transition-based semantic graph parsing with syntactic features.
In *Proc. of SemEval*, pages 97–103.

References III

Sulem, E., Abend, O., and Rappoport, A. (2015).
Conceptual annotations preserve structure across translations: A French-English case study.
In *Proc. of S2MT*, pages 11–22.

Zhang, Y. and Nivre, J. (2011).
Transition-based dependency parsing with rich non-local features.
In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193.

Backup

UCCA Corpora

	Wiki Train	Wiki Dev	Wiki Test	20K Leagues
# passages	300	34	33	154
# sentences	4268	454	503	506
# nodes	298,993	33,704	35,718	29,315
% terminal	42.96	43.54	42.87	42.09
% non-term.	58.33	57.60	58.35	60.01
% **discont.**	**0.54**	**0.53**	**0.44**	**0.81**
% **reentrant**	**2.38**	**1.88**	**2.15**	**2.03**
# edges	287,914	32,460	34,336	27,749
% primary	98.25	98.75	98.74	97.73
% remote	1.75	1.25	1.26	2.27
Average per non-terminal node				
# children	1.67	1.68	1.66	1.61

Corpus statistics.

Evaluation

Mutual edges between predicted graph $G_p = (V_p, E_p, \ell_p)$ and gold graph $G_g = (V_g, E_g, \ell_g)$, both over terminals $W = \{w_1, \ldots, w_n\}$:

$$M(G_p, G_g) = \left\{(e_1, e_2) \in E_p \times E_g \;\middle|\; y(e_1) = y(e_2) \wedge \ell_p(e_1) = \ell_g(e_2)\right\}$$

The yield $y(e) \subseteq W$ of an edge $e = (u, v)$ in either graph is the set of terminals in W that are descendants of v. ℓ is the edge label.

Labeled precision, recall and F-score are then defined as:

$$\mathrm{LP} = \frac{|M(G_p, G_g)|}{|E_p|}, \quad \mathrm{LR} = \frac{|M(G_p, G_g)|}{|E_g|},$$

$$\mathrm{LF} = \frac{2 \cdot \mathrm{LP} \cdot \mathrm{LR}}{\mathrm{LP} + \mathrm{LR}}.$$

Two variants: one for primary edges, and another for remote edges.