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

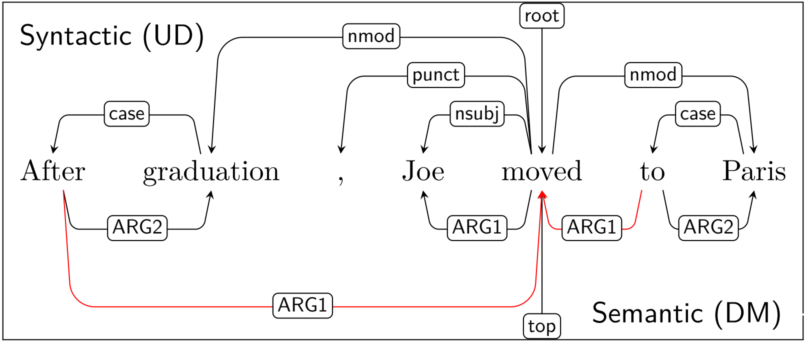
ACL 2017

Linguistic Structure Annotation Schemes

- Dependency grammar: bilexical tree, purely syntactic.

Semantic representation (Abend and Rappoport, 2017):

- Semantic dependencies: bilexical graph, coupled with syntax.



Bilexical dependencies.

Linguistic Structure Annotation Schemes

- Dependency grammar: bilexical tree, purely syntactic.

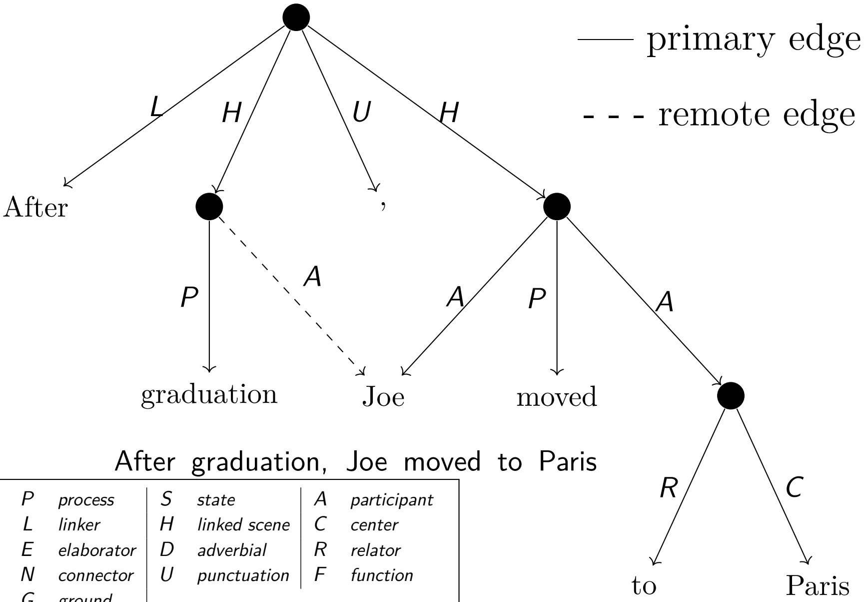
Semantic representation (Abend and Rappoport, 2017):

- Semantic dependencies: bilexical graph, coupled with syntax.
- AMR (Banarescu et al., 2013), UCCA (Abend and Rappoport, 2013): direct semantic representation.

$\boxed{\text{...showered}} = \boxed{\text{...took a shower}}$

$\boxed{\text{...'s war against crime}} = \boxed{\text{...fights crime}}$

Universal Conceptual Cognitive Annotation (UCCA)

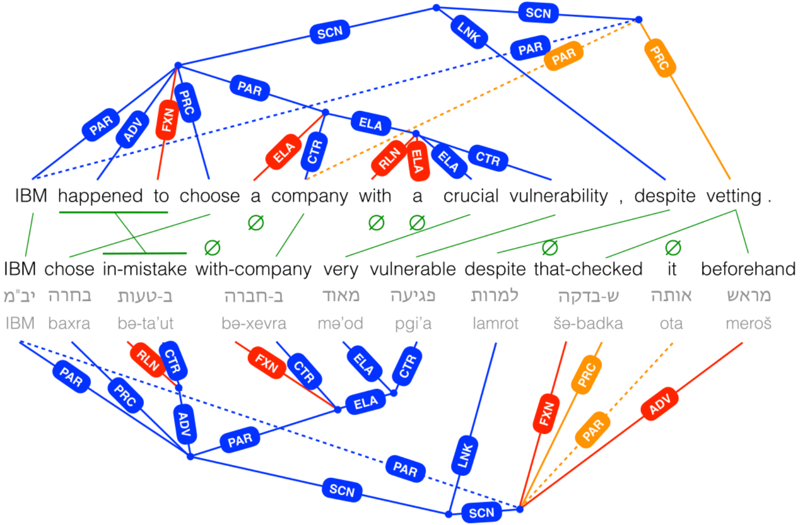


After graduation, Joe moved to Paris

P	*process*	*S*	*state*	*A*	*participant*
L	*linker*	*H*	*linked scene*	*C*	*center*
E	*elaborator*	*D*	*adverbial*	*R*	*relator*
N	*connector*	*U*	*punctuation*	*F*	*function*
G	*ground*				

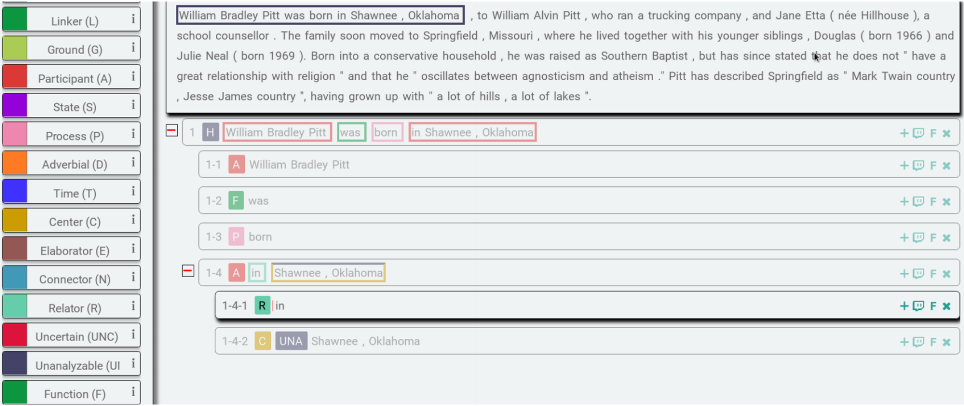
The UCCA Semantic Representation Scheme

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



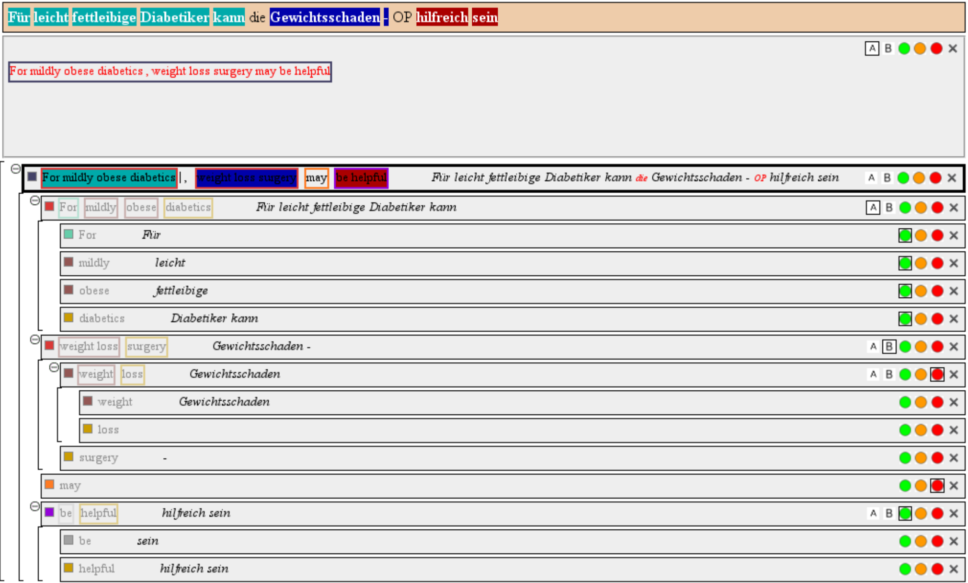
UCCAApp

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



HUME

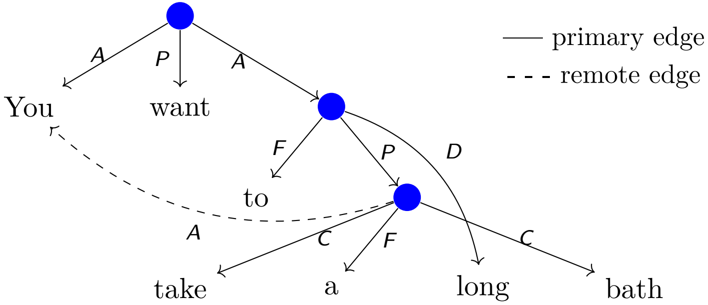
UCCA facilitates semantics-based human evaluation of machine translation (Birch et al., 2016). `http://ucca.cs.huji.ac.il/mteval`



Graph Structure

UCCA forms a directed acyclic graph (DAG). Tokens are terminals.
Structural properties:

1. Non-terminal nodes

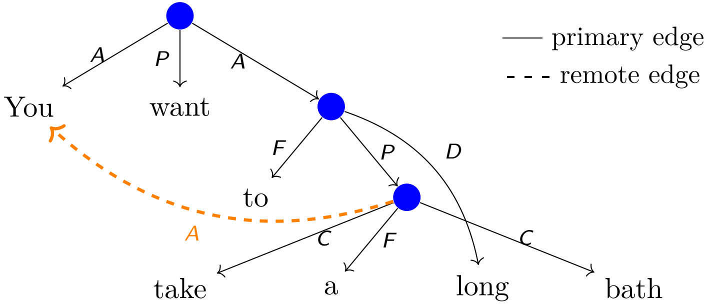


You want to take a long bath

Graph Structure

UCCA forms a directed acyclic graph (DAG). Tokens are terminals. Structural properties:

1. Non-terminal nodes
2. Reentrancy

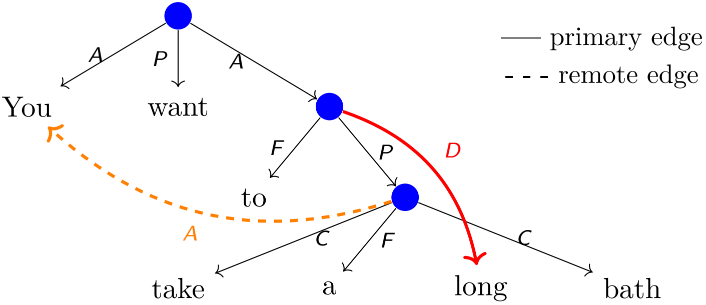


You want to take a long bath

Graph Structure

UCCA forms a directed acyclic graph (DAG). Tokens are terminals.
Structural properties:

1. Non-terminal nodes
2. Reentrancy
3. Discontinuity



You want to take a long bath

Transition-Based Parsing

- Parse text $w_1 \ldots w_n$ to graph $G = (V, E, \ell)$ incrementally.
- Classifier determines transition to apply at each step.
- Trained by an oracle based on gold-standard graph.

Transition-Based Parsing

- Parse text $w_1 \dots w_n$ to graph $G = (V, E, \ell)$ incrementally.
- Classifier determines transition to apply at each step.
- Trained by an oracle based on gold-standard graph.

Initial state:

stack

●

buffer

You	want	to	take	a	long	bath

Transition-Based Parsing

- Parse text $w_1 \ldots w_n$ to graph $G = (V, E, \ell)$ incrementally.
- Classifier determines transition to apply at each step.
- Trained by an oracle based on gold-standard 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.

Transition-Based UCCA Parsing

⇒ SHIFT

stack

●	You

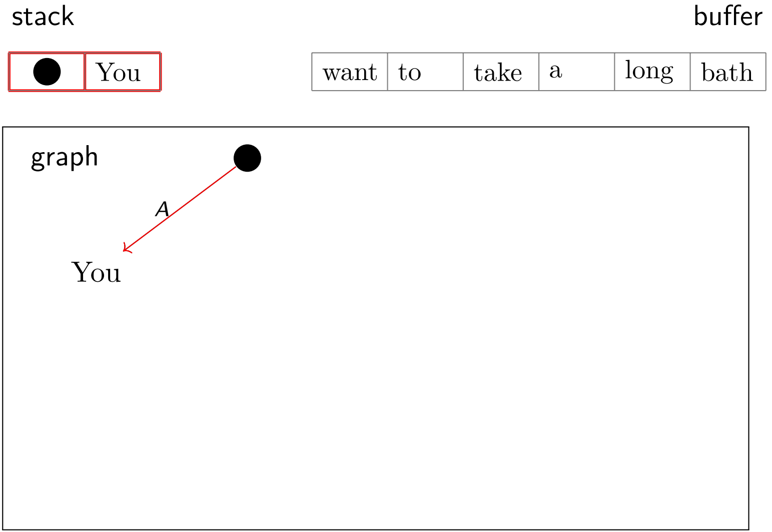
buffer

want	to	take	a	long	bath

graph

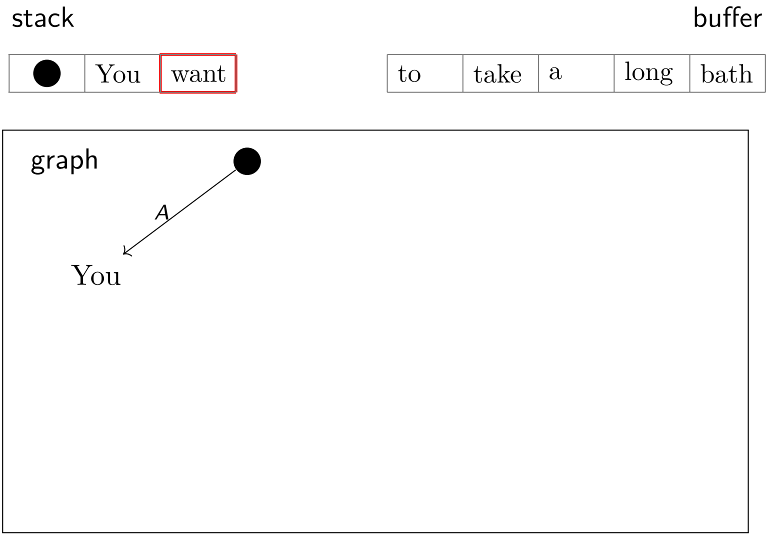
Transition-Based UCCA Parsing

$\Rightarrow$ RIGHT-EDGE$_{\text{A}}$



Transition-Based UCCA Parsing

⇒ Shift



Transition-Based UCCA Parsing

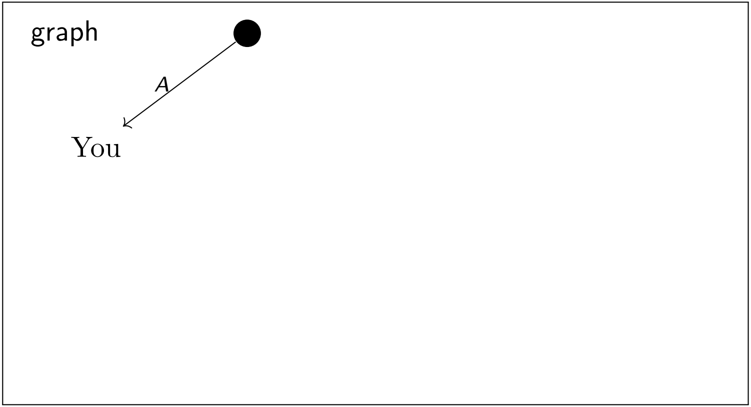
⇒ Swap

stack

●	want

buffer

You	to	take	a	long	bath



Transition-Based UCCA Parsing

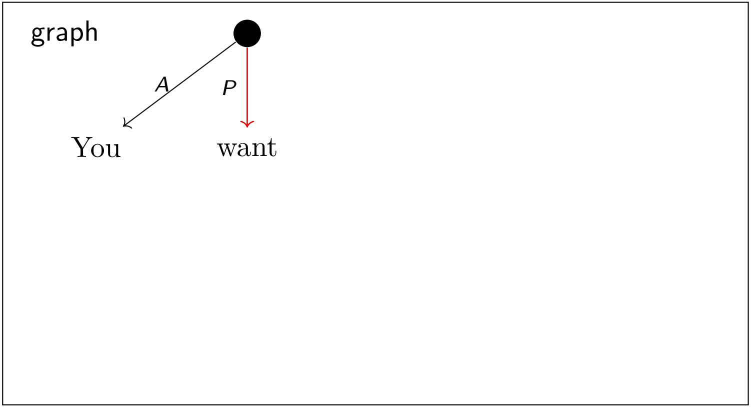
⇒ $\textsc{Right-Edge}_P$

stack

●	want

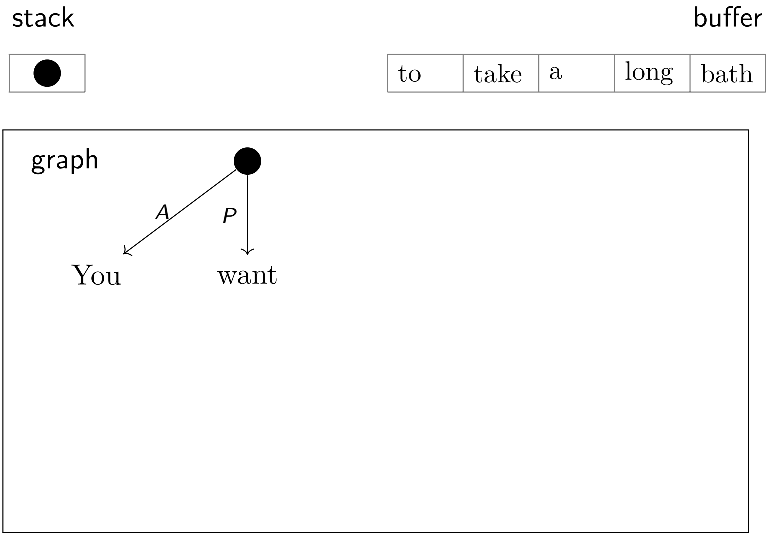
buffer

You	to	take	a	long	bath



Transition-Based UCCA Parsing

⇒ Reduce



Transition-Based UCCA Parsing

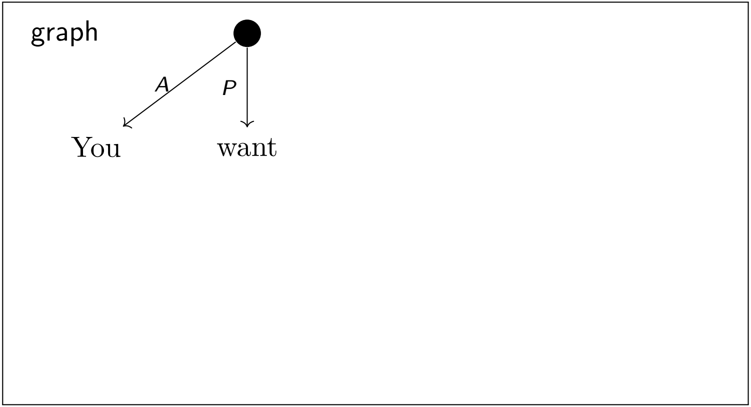
⇒ SHIFT

stack

●	You

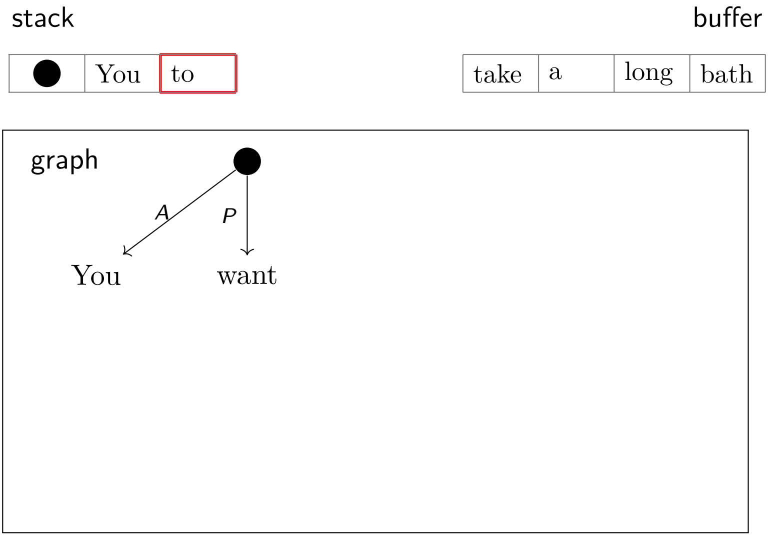
buffer

to	take	a	long	bath



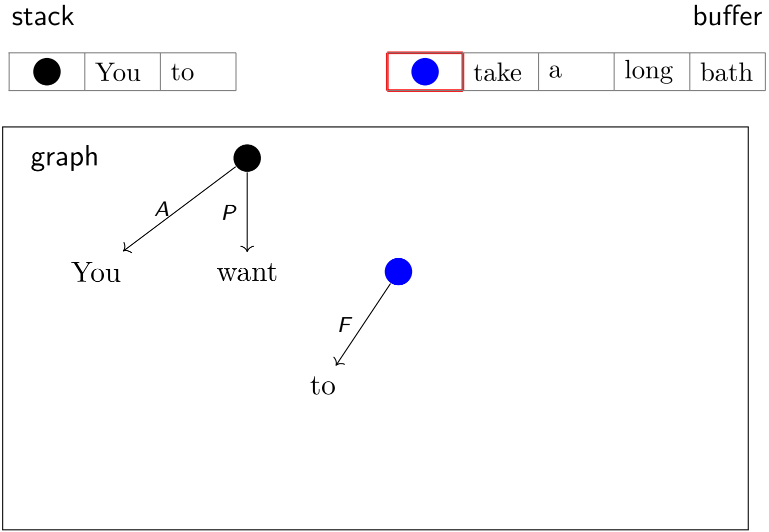
Transition-Based UCCA Parsing

⇒ SHIFT



Transition-Based UCCA Parsing

$\Rightarrow$ $\textsc{Node}_{\text{F}}$

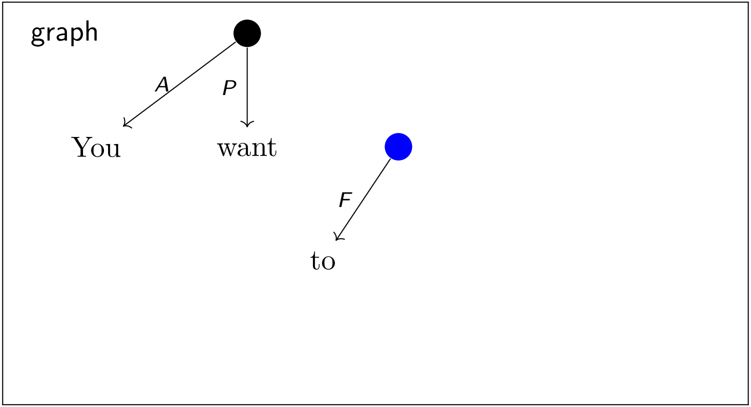


Transition-Based UCCA Parsing

⇒ Reduce

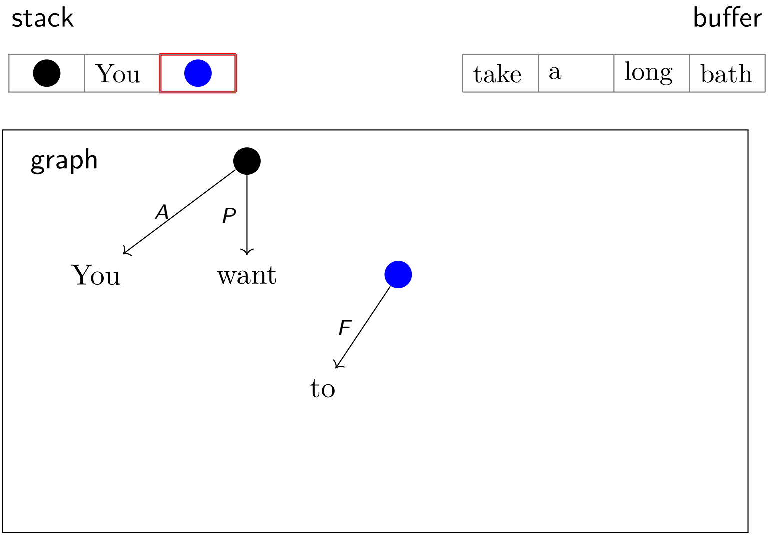
stack	
●	You

buffer				
●	take	a	long	bath



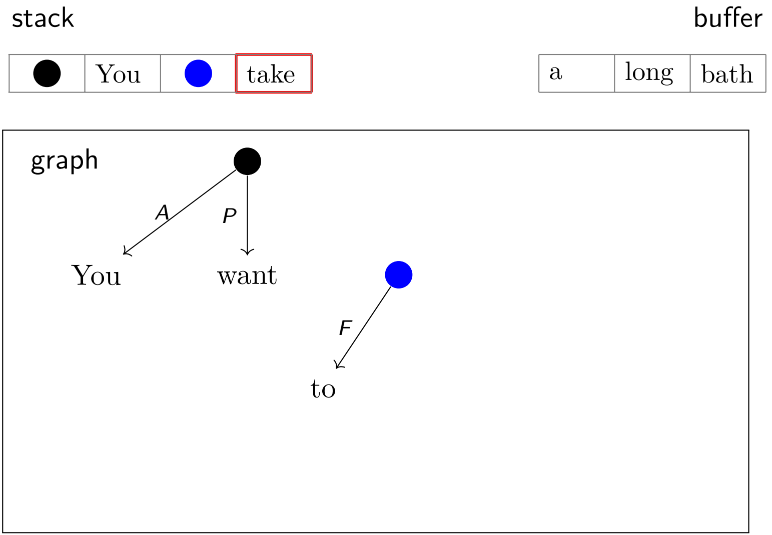
Transition-Based UCCA Parsing

⇒ Shift



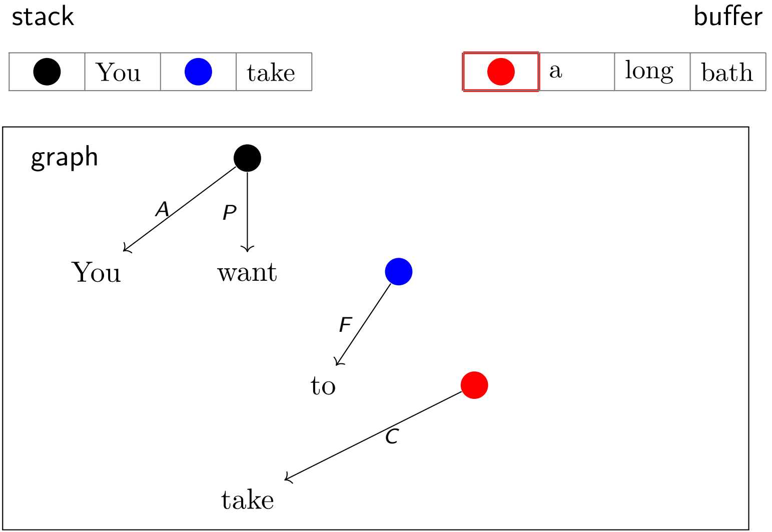
Transition-Based UCCA Parsing

$\Rightarrow$ SHIFT



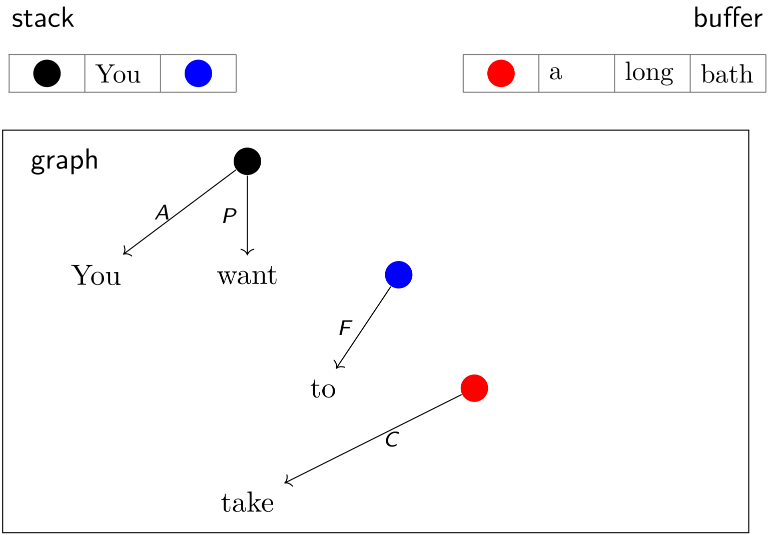
Transition-Based UCCA Parsing

$\Rightarrow$ $\textsc{Node}_C$



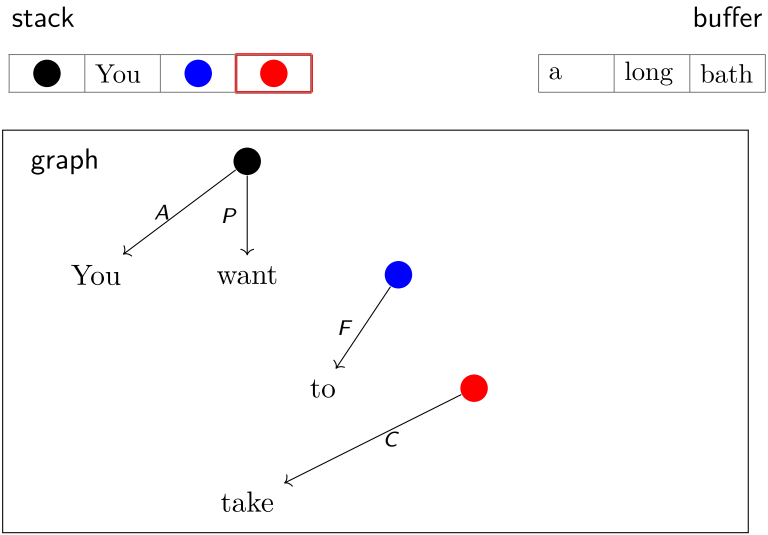
Transition-Based UCCA Parsing

⇒ REDUCE



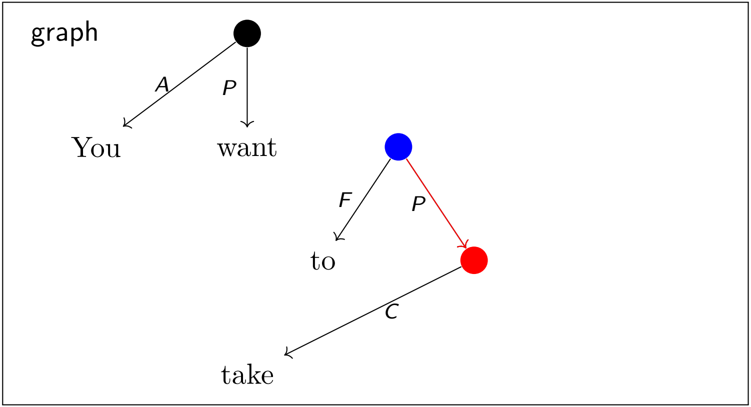
Transition-Based UCCA Parsing

⇒ Shift



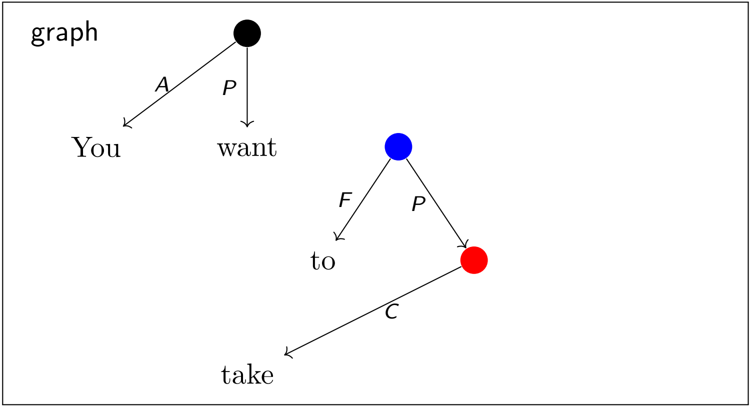
Transition-Based UCCA Parsing

⇒ RIGHT-EDGE$_{P}$



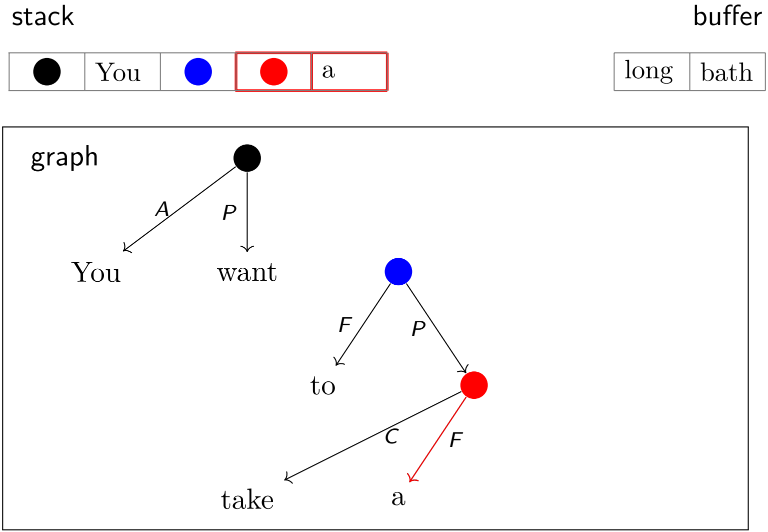
Transition-Based UCCA Parsing

$\Rightarrow$ SHIFT



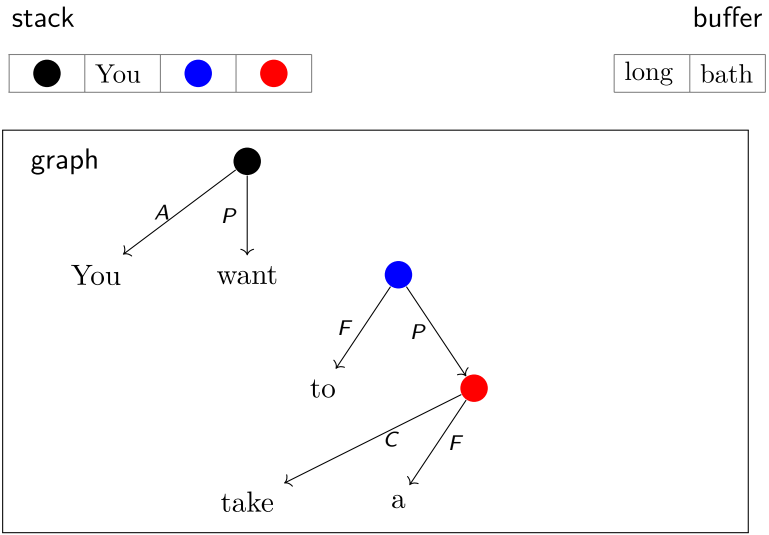
Transition-Based UCCA Parsing

⇒ RIGHT-EDGE$_F$



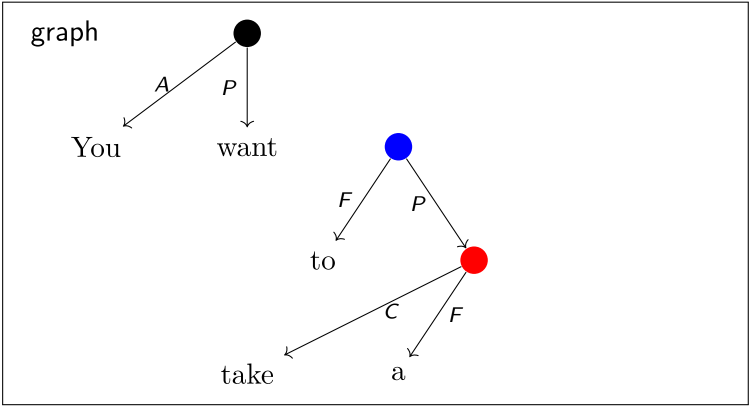
Transition-Based UCCA Parsing

$\Rightarrow$ REDUCE



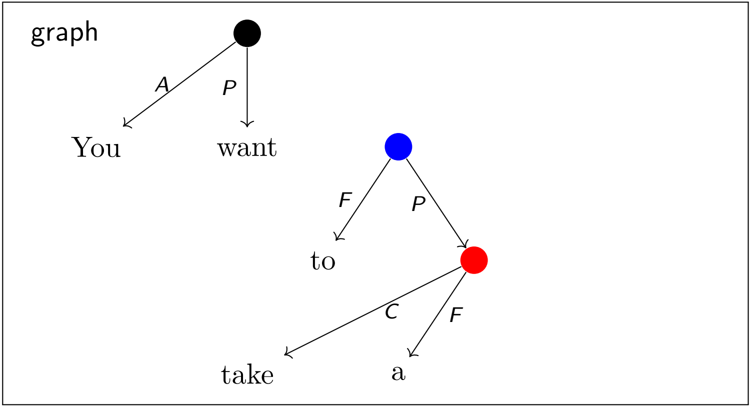
Transition-Based UCCA Parsing

⇒ SHIFT



Transition-Based UCCA Parsing

⇒ SWAP

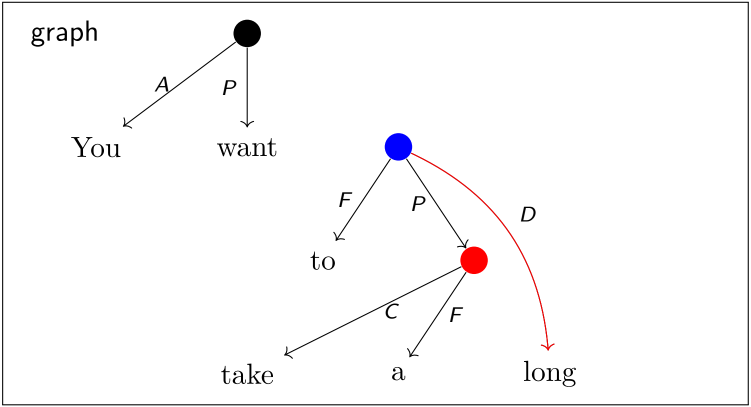


Transition-Based UCCA Parsing

⇒ RIGHT-EDGE$_{\text{D}}$

stack

buffer



Transition-Based UCCA Parsing

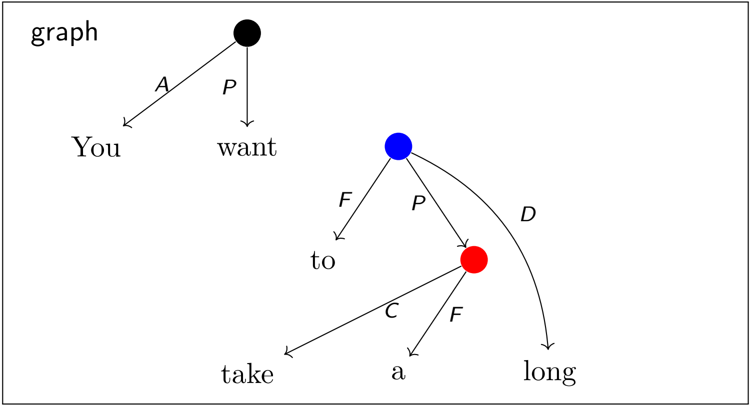
⇒ REDUCE

stack

●	You	●

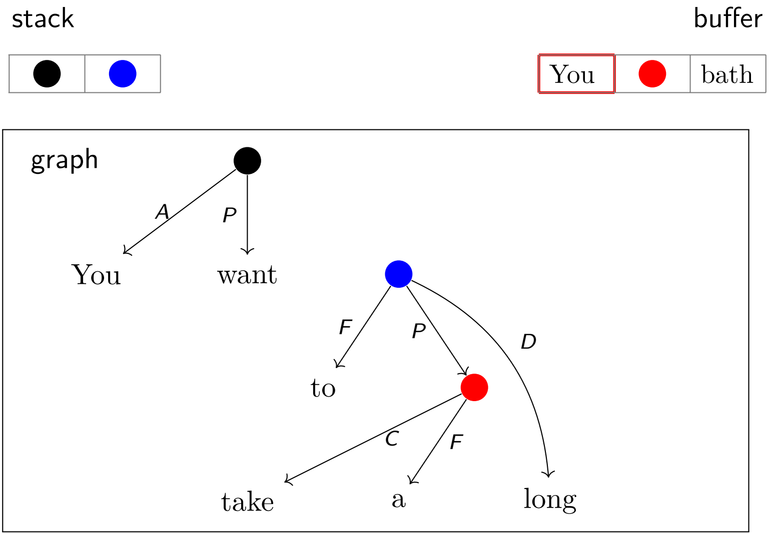
buffer

●	bath



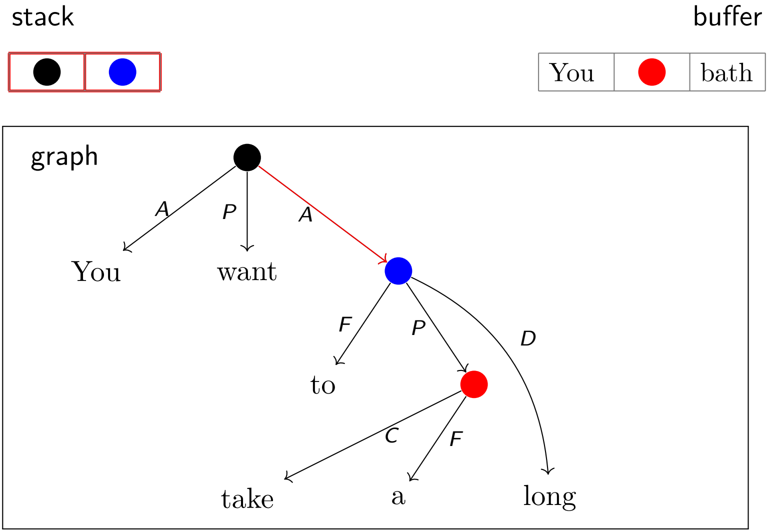
Transition-Based UCCA Parsing

$\Rightarrow$ SWAP



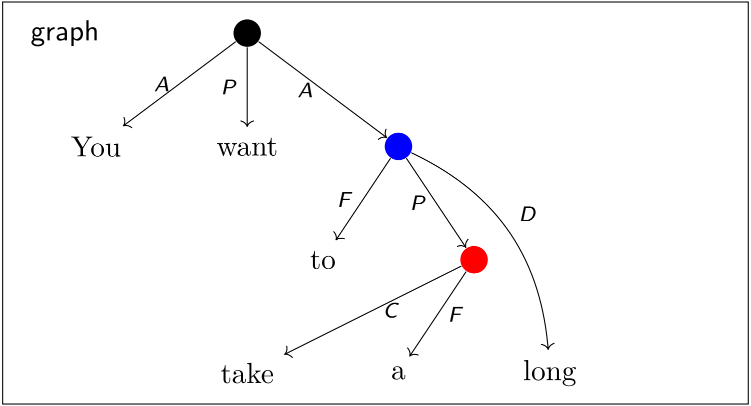
Transition-Based UCCA Parsing

$\Rightarrow$ RIGHT-EDGE$_{\text{A}}$



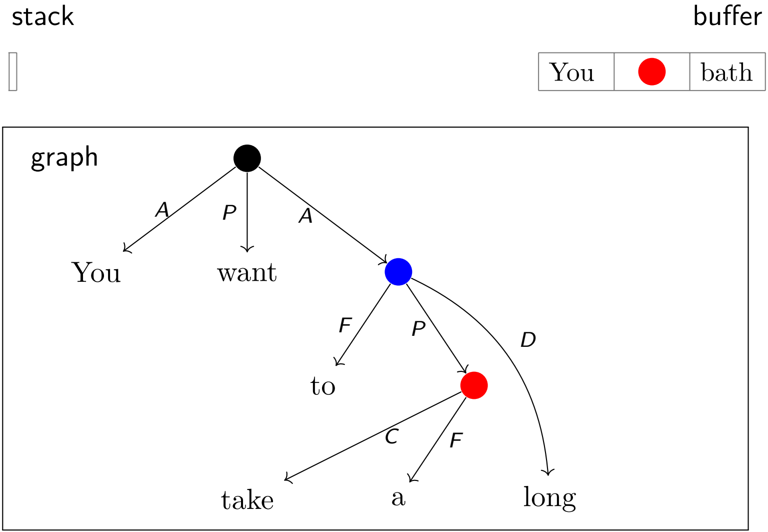
Transition-Based UCCA Parsing

⇒ REDUCE



Transition-Based UCCA Parsing

⇒ Reduce



Transition-Based UCCA Parsing

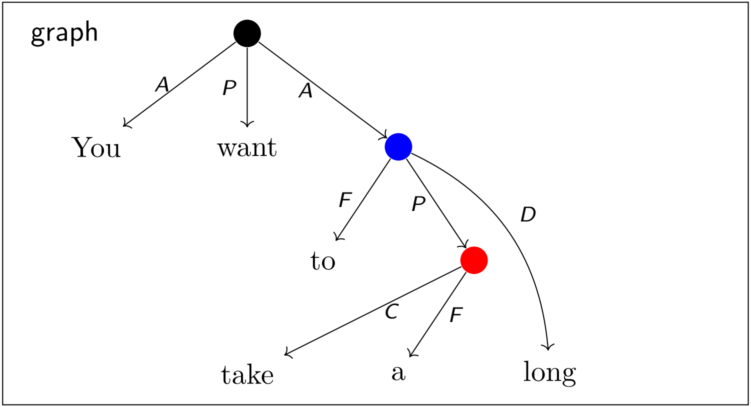
⇒ SHIFT

stack

You

buffer

bath



Transition-Based UCCA Parsing

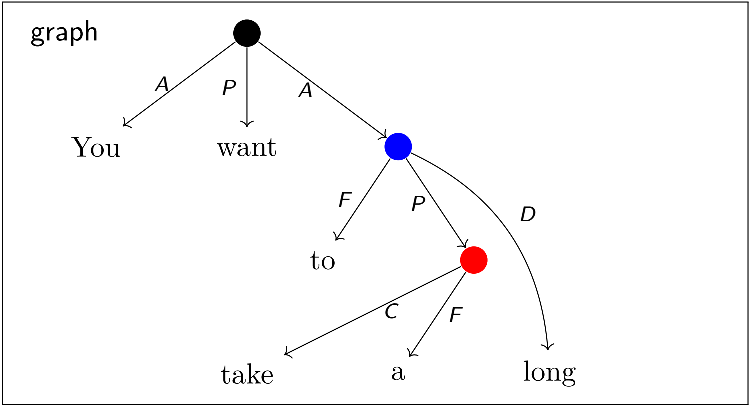
⇒ SHIFT

stack

You	●

buffer

bath



Transition-Based UCCA Parsing

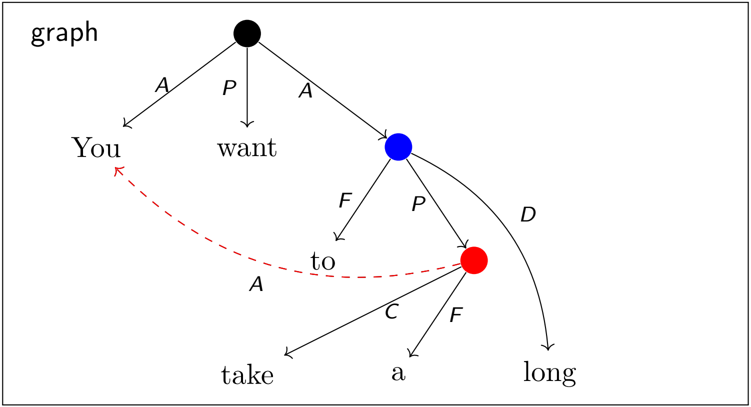
⇒ LEFT-REMOTE$_A$

stack

You	

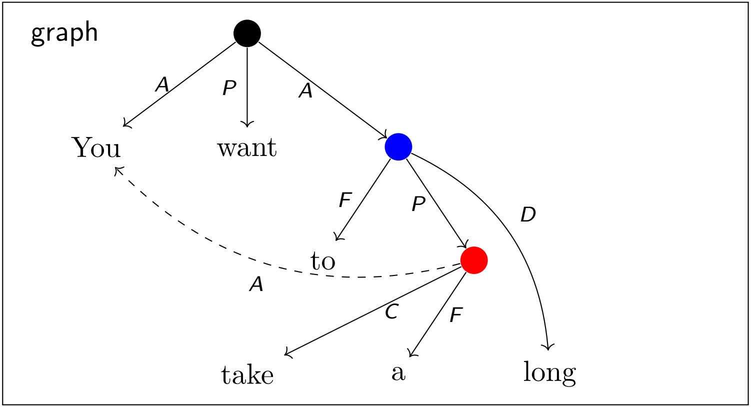
buffer

bath



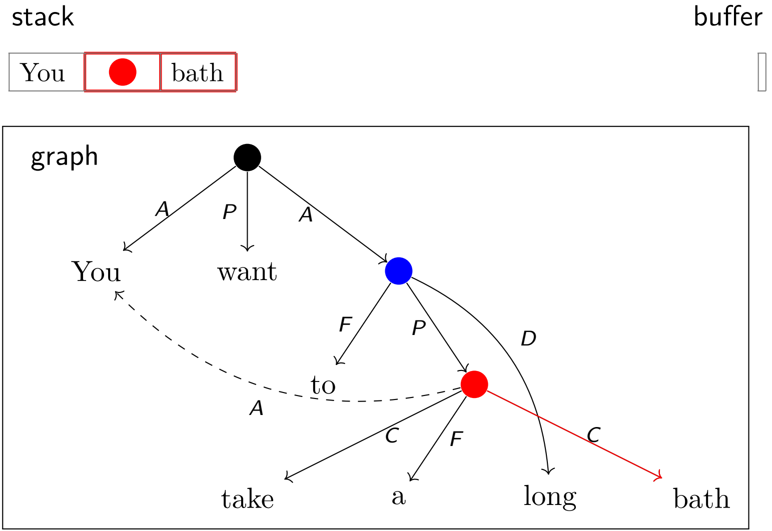
Transition-Based UCCA Parsing

⇒ Shift



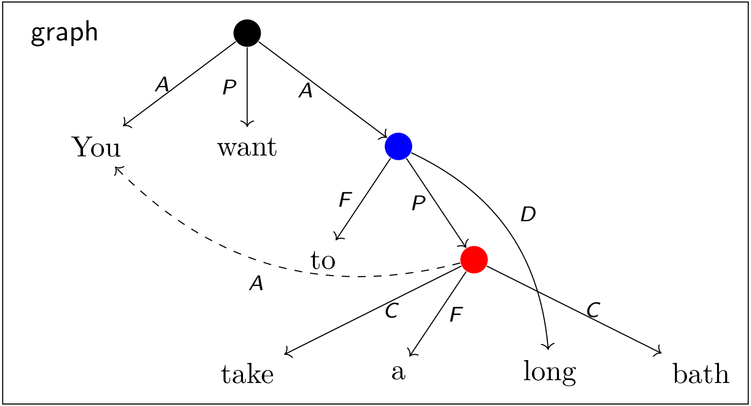
Transition-Based UCCA Parsing

⇒ RIGHT-EDGE$_C$



Transition-Based UCCA Parsing

⇒ FINISH

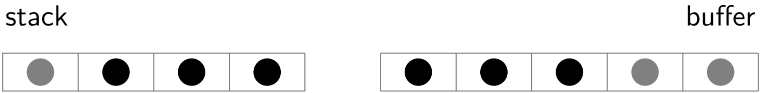


TUPA Model

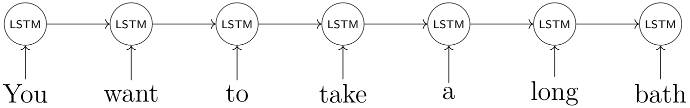
Greedy parsing, experimenting with three classifiers:

Sparse	Perceptron with sparse features.
MLP	Embeddings + feedforward NN classifier.
BiLSTM	Embeddings + deep bidirectional LSTM + MLP.

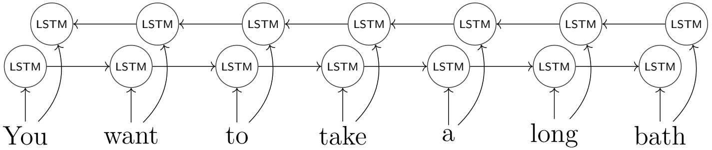
Features: words, POS, syntactic dependencies, existing edge labels from the stack and buffer + parents, children, grandchildren; ordinal features (height, number of parents and children)



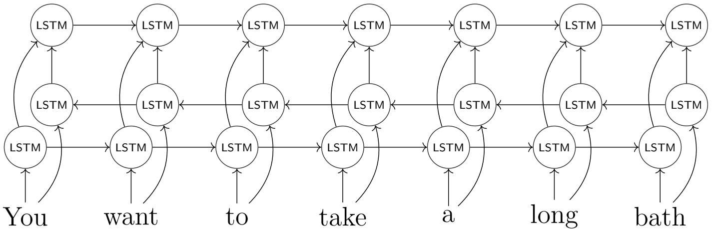
BiLSTM


LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
You
want
to
take
a
long
bath

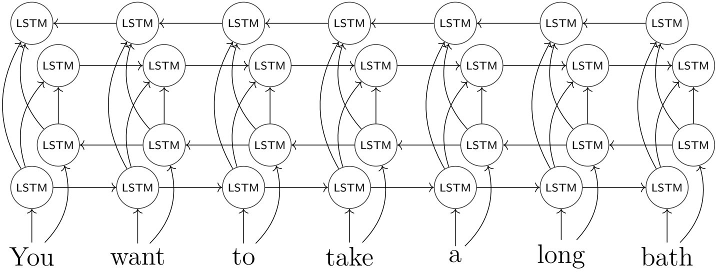
BiLSTM

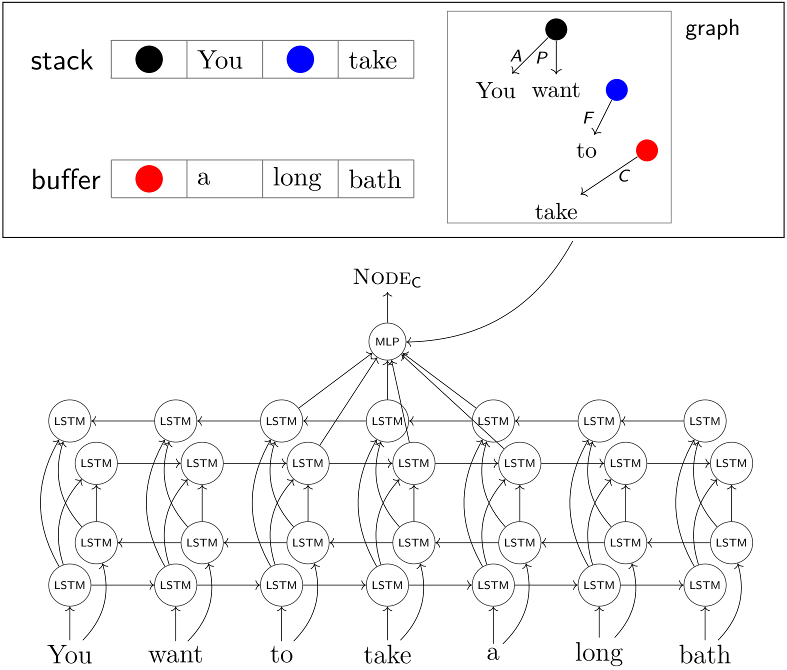


BiLSTM



BiLSTM

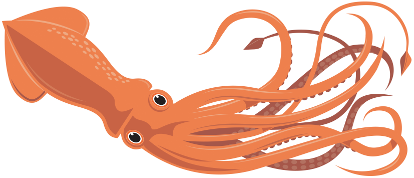




stack
You
take
buffer
a
long
bath
graph
A
P
You
want
F
to
C
take
NODE$_C$
MLP
LSTM
You
want
to
take
a
long
bath

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).



Results

	Primary			Remote		
	LP	**LR**	**LF**	**LP**	**LR**	**LF**
Sparse	64.5	63.7	64.1	19.8	13.4	16
MLP	65.2	64.6	64.9	23.7	13.2	16.9
BiLSTM	74.4	72.7	**73.5**	47.4	51.6	**49.4**

Results on the Wiki test set.

Results

	Primary			Remote		
	LP	**LR**	**LF**	**LP**	**LR**	**LF**
Sparse	64.5	63.7	64.1	19.8	13.4	16
MLP	65.2	64.6	64.9	23.7	13.2	16.9
BiLSTM	74.4	72.7	**73.5**	47.4	51.6	**49.4**

Results on the Wiki test set.

	Primary			Remote		
	LP	**LR**	**LF**	**LP**	**LR**	**LF**
Sparse	59.6	59.9	59.8	22.2	7.7	11.5
MLP	62.3	62.6	62.5	20.9	6.3	9.7
BiLSTM	68.7	68.5	**68.6**	38.6	18.8	**25.3**

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

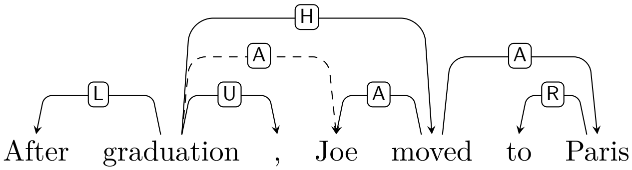
Baselines

Bilexical DAG parsers:

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



Bilexical DAG approximation.

Results

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

	Primary			Remote		
	LP	**LR**	**LF**	**LP**	**LR**	**LF**
$\text{TUPA}_{\text{Sparse}}$	64.5	63.7	64.1	19.8	13.4	16
TUPA_{MLP}	65.2	64.6	64.9	23.7	13.2	16.9
$\text{TUPA}_{\text{BiLSTM}}$	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	–	–	–
LSTM Parser	73.2	66.9	69.9	–	–	–
Tree			(100)			–
UPARSE	60.9	61.2	61.1	–	–	–

Results on the Wiki test set.

Results

Similar on out-of-domain test set:

	Primary			Remote		
	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	–	–	–
LSTM Parser	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 challenging structural properties.
- TUPA is a transition-based parser suitable for UCCA, achieving high accuracy with BiLSTM model.
- Outperforms conversion-based parsing with a variety of strong bilexical DAG and tree baselines.

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

Conclusion

- UCCA's semantic distinctions require a graph structure including challenging structural properties.
- TUPA is a transition-based parser suitable for UCCA, achieving high accuracy with BiLSTM model.
- Outperforms conversion-based parsing with a variety of strong bilexical DAG and tree baselines.

Future Work:

- More languages (German corpus construction is underway).
- Parsing other schemes, such as AMR.
- Application to text simplification and other tasks.

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

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.

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.

References II

Goldberg, Y. (2016).
A primer on neural network models for natural language processing.

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

Nivre, J. (2005).
Dependency grammar and dependency parsing.

Nivre, J. (2009).
Non-projective dependency parsing in expected linear time.
In *Proc. of ACL*, pages 351–359.

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., Hajič, J., and Urešová, Z. (2015).
SemEval 2015 task 18: Broad-coverage semantic dependency parsing.
In *Proc. of SemEval*, pages 915–926.

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.

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.

Tesniére, L. (1959).
Elements de syntaxe structuralle.
Klincksieck, Paris, 2 edition.

UCCA Corpora

	Wiki			20K
	Train	Dev	Test	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:

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

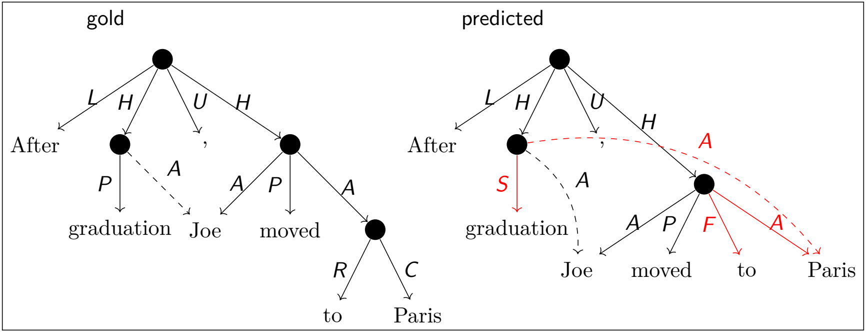
$$\mathrm{LF} = \frac{2 \cdot \mathrm{LP} \cdot \mathrm{LR}}{\mathrm{LP} + \mathrm{LR}}.$$

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

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%