IN2110: Methods in Language Technology

Grammatical Structure Wrap-Up

Stephan Oepen

Language Technology Group (LTG)

May 7, 2019

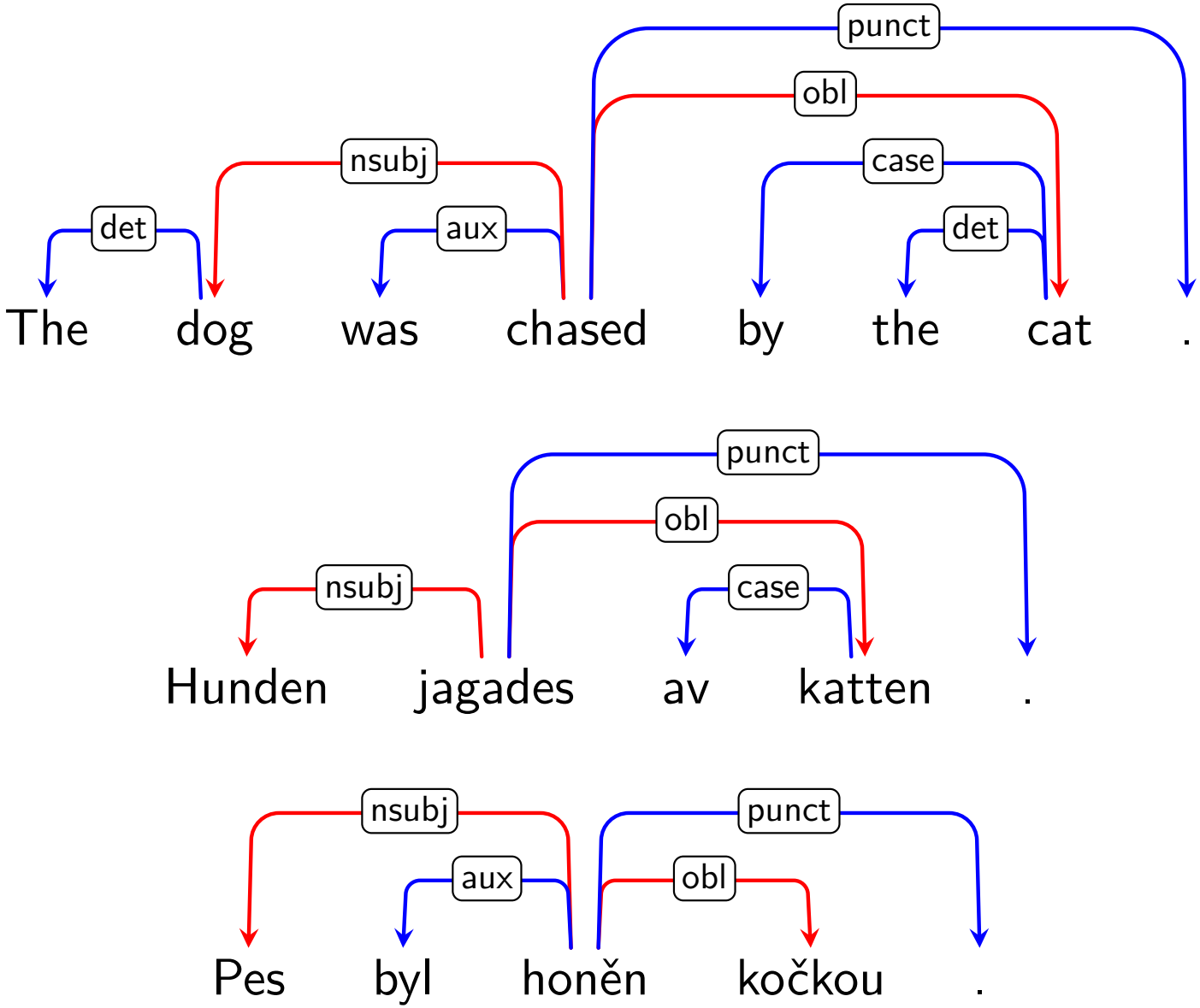
Topics for Today

- Short recap:
  - Universal Dependencies
  - Transition-based dependency parsing
- Dependency Parser Evaluation
- Variants on data-driven dependency parsing
  - Graph-based dependency parsing
  - Arc-standard transition system
  - Semantic dependency graphs
- Syntactic structure in negation resolution
- Sample exam questions

Recap: 'Universal' Dependency Types

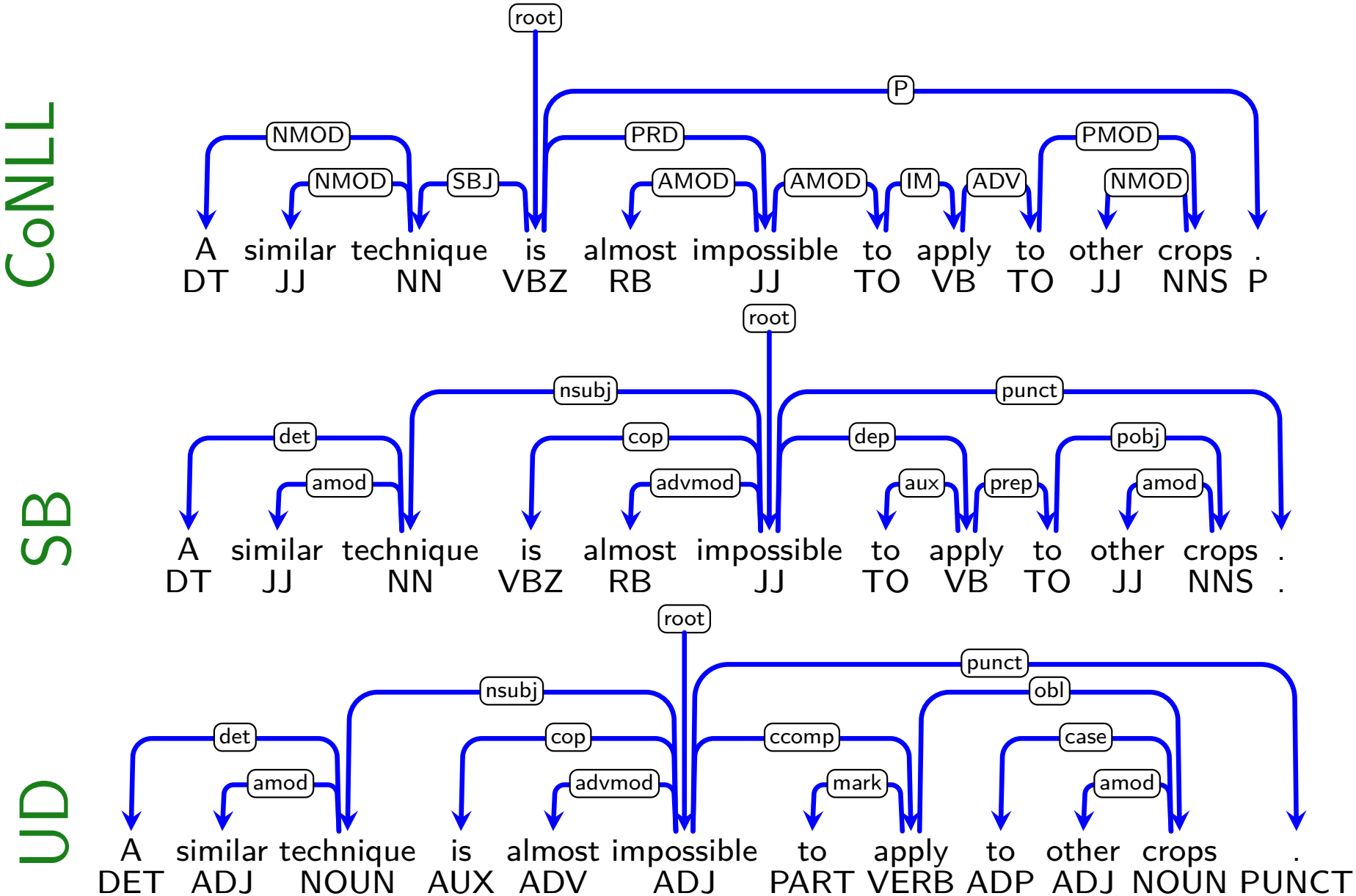
nsubj	nominal subject	She arrived.
csubj	clausal subject	That she arrived surprised me.
obj	(direct) object	My mother called me.
iobj	indirect object	She teaches my daughter maths.
ccomp	clausal complement	She knew that she arrived.
xcomp	open clausal complement	She promised to sing.
obl	oblique nominal	She arrived on Monday
obl	oblique nominal	She depends on me.
nmod	nominal modifier	the office of the chair is empty.
amod	adjectival modifier	the fierce dog barks.
acl	adjectival clause	the dog that barks arrived.
conj	conjunct	Kim and Sandy arrived.
cc	coordinating conjunction	Kim and Sandy arrived.

Recap: Cross-Linguistic Consistency

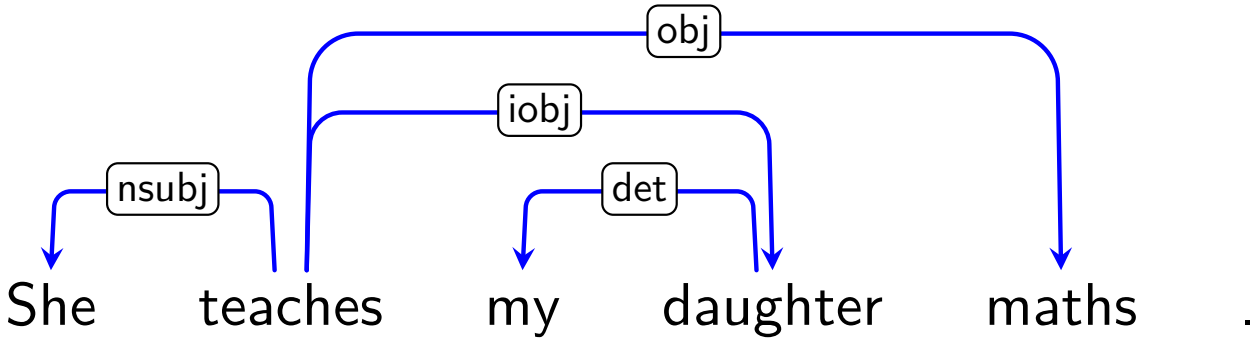


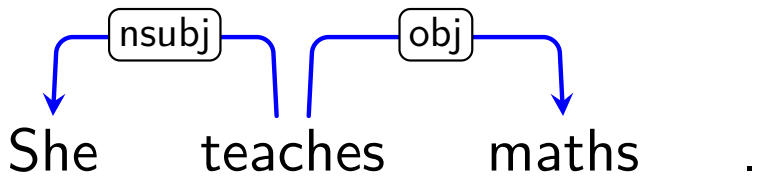
- Capitalize on content words, e.g. demote case-marking prepositions.

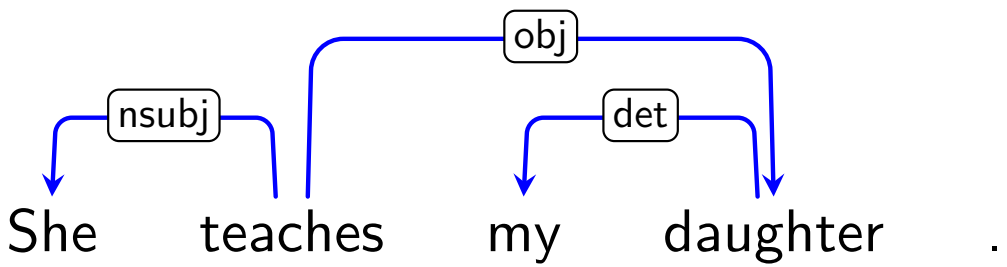
Functional vs. Content Heads



Consistency Can be Evasive







- UD guidelines: [...] if there is just one object, it should be labeled `obj`.

UD: The Big Picture

	Nominals	Clauses	Modifier words	Function Words
Core arguments	nsubj obj iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod* discourse	aux cop mark
Nominal dependents	nmod appos nummod	acl	amod	det clf case
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj cc	fixed flat compound	list parataxis	orphan goeswith reparandum	punct root dep

Dependency Evaluation Metrics

General Ideas

- Similar to ParsEval, want to award partial credit: granular evaluation.
- Fixed number of tokens: per-token accuracy scores (like in tagging).
- Can consider just tree topology or topology plus dependency types.
- Punctuation tokens (e.g. by Unicode property) are often excluded.

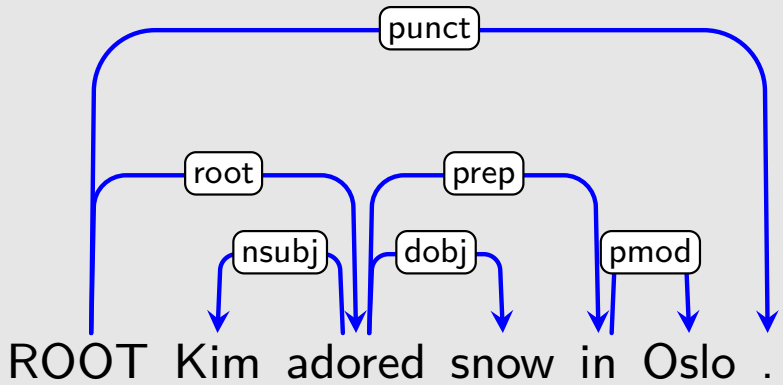
UAS: Unlabeled Attachment Score

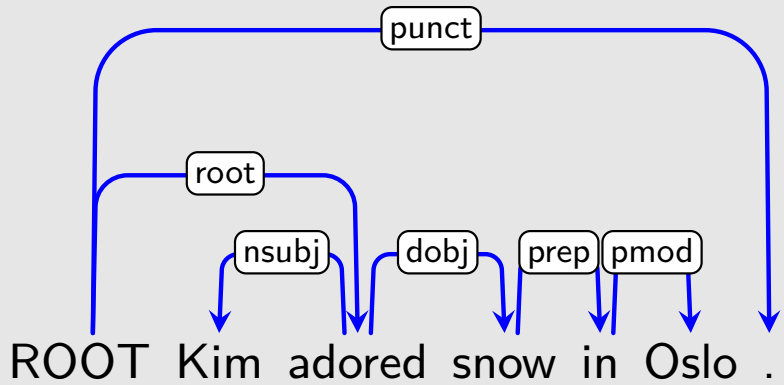
- For each token, does it have correct head (source of incoming edge)?

LAS: Labeled Attachment Score

- In addition to the head, is the dependency type (edge label) correct?

Exercise (5): Dependency Evaluation

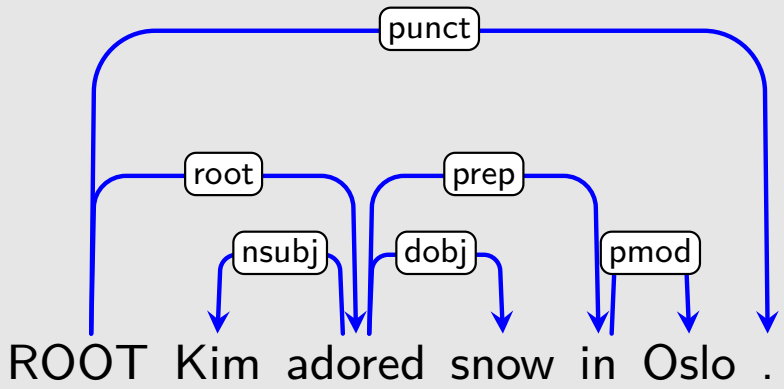


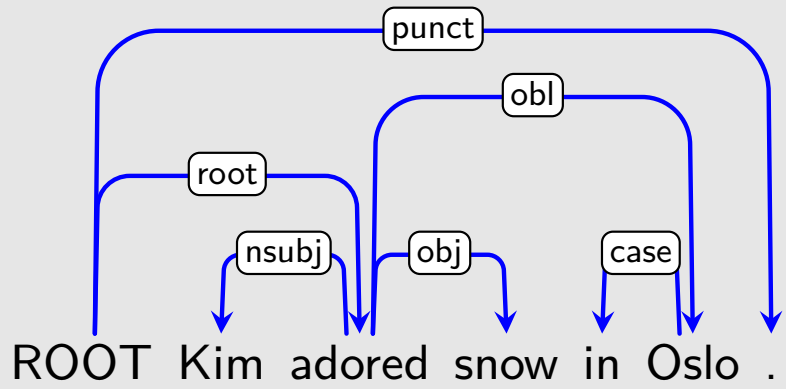


(5) What are the LAS and UAS scores for the two trees?

Gold standard on the left, system prediction on the right.

Exercise (6): More Dependency Evaluation





(6) What are the LAS and UAS scores for the two trees?

Recap: Dependency Parsing Summary

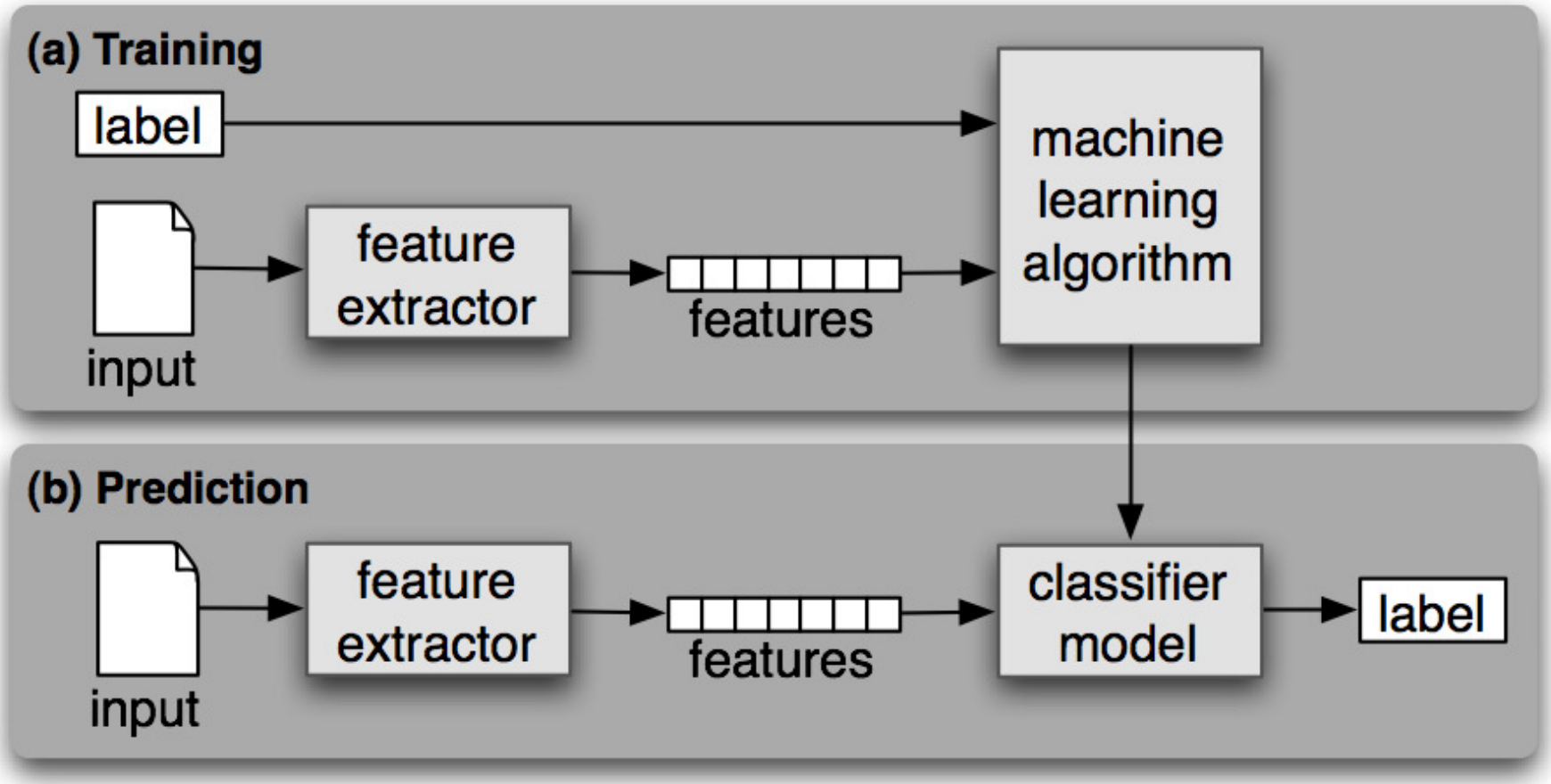
Transition-Based Dependency Parsing

- Transition system ensures formal wellformedness of dependency trees;
- A specific sequence of transitions determines the final parsing result.
- Much room for experimentation: Feature models and types of classifiers;
- decent results with Maximum Entropy or Support Vector Machines.

Variants on Data-Driven Dependency Parsing

- Other transition systems (e.g. arc-standard; like 'classic' shift-reduce);
- different techniques for non-projective trees; e.g. swap transitions;
- can relax transition system further, to output general, non-tree graphs.
- Beam search: exploring the top-n transitions out of each configuration.
- So-called graph-based dependency parsing: somewhat similar to CKY.
- Multi-stratal (multi-layer) representations: MTT, FGD, enhanced UD.

The Transition Oracle as a Classifier



Recap: Adapting Shift–Reduce Parsing

- Originally developed for non-ambiguous languages: deterministic.
- Shift ('read') tokens from input buffer, one at a time, left-to-right;
- compare top n symbols on stack against rule RHS: reduce to LHS.
- Dependencies: create arcs between top of stack and front of buffer.

SHIFT	move from front of buffer to top of stack
REDUCE	pop the top of stack (requires existing head)
LEFT-ARC(K)	leftward dependency of type k; reduce
RIGHT-ARC(K)	rightward dependency of type k; shift

- At REDUCE, token should be fully processed (head and dependents).
- LEFT-ARC must respect single-head constraint and unique root node.

Recent Advances in Dependency Parsing

Tutorial, EACL, April 27th, 2014

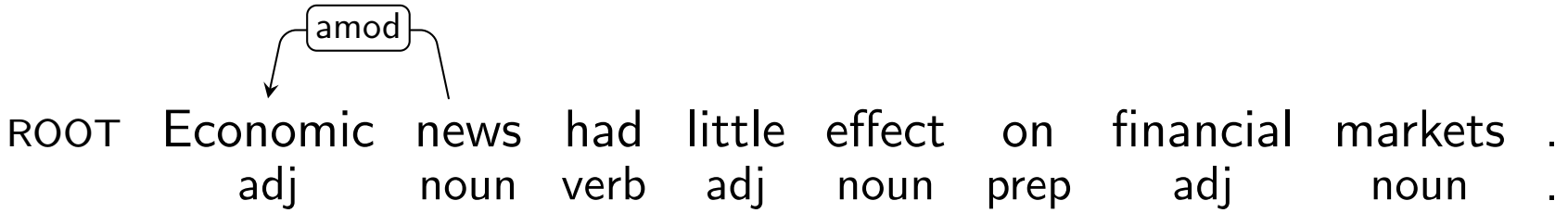
Ryan McDonald[1] Joakim Nivre[2]

[1]Google Inc., USA/UK
E-mail: ryanmcd@google.com

[2]Uppsala University, Sweden
E-mail: joakim.nivre@lingfil.uu.se

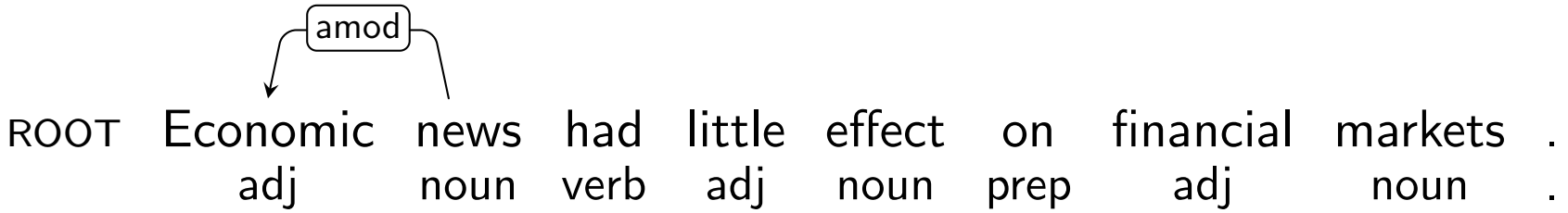
Example Transition Sequence

[ROOT]$_S$ [news, had, little, effect, on, financial, markets, .]$_B$



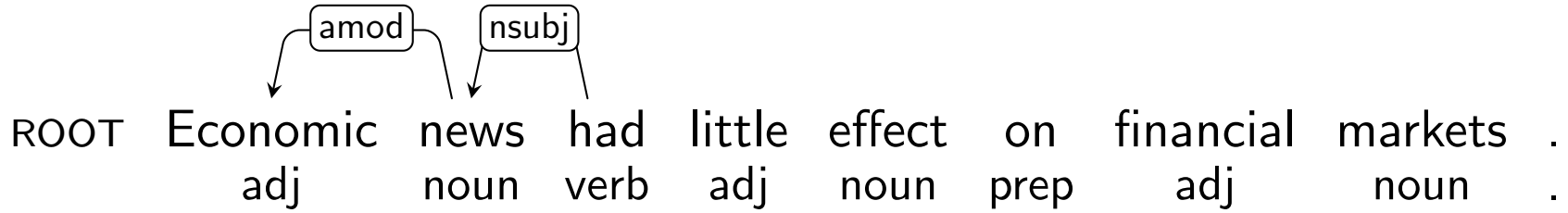
Example Transition Sequence

[ROOT, news]$_S$ [had, little, effect, on, financial, markets, .]$_B$



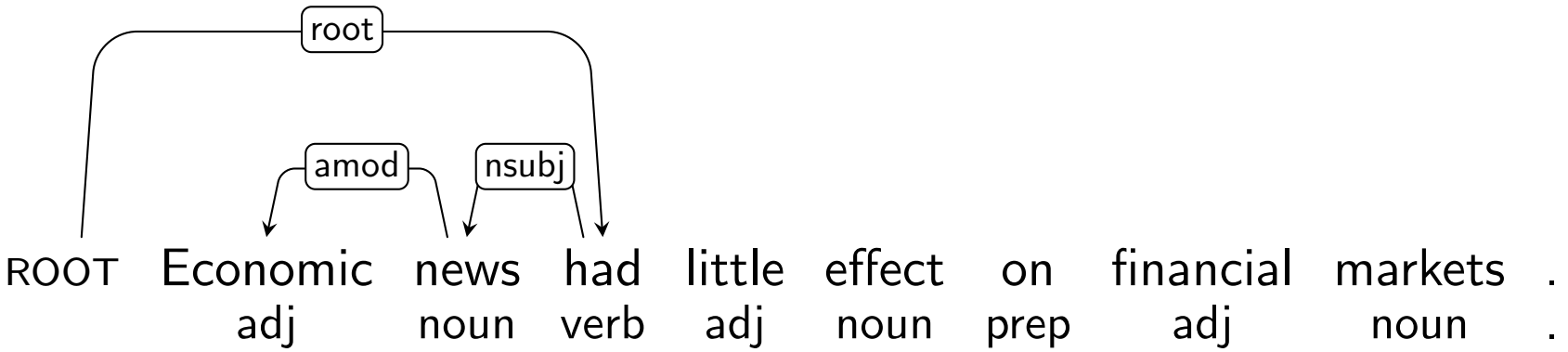
Example Transition Sequence

[ROOT]$_S$ [had, little, effect, on, financial, markets, .]$_B$



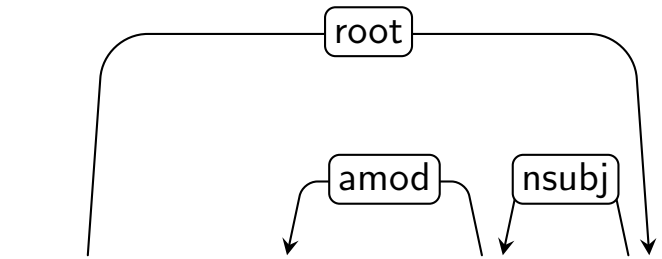
Example Transition Sequence

[ROOT, had]$_S$ [little, effect, on, financial, markets, .]$_B$



Example Transition Sequence

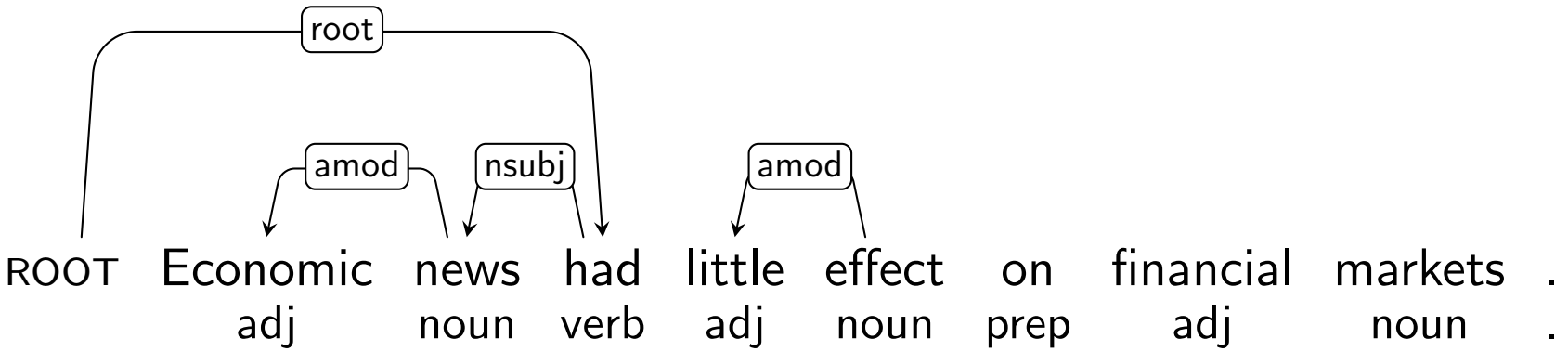
[ROOT, had, little]$_S$ [effect, on, financial, markets, .]$_B$



ROOT	Economic	news	had	little	effect	on	financial	markets	.
	adj	noun	verb	adj	noun	prep	adj	noun	.

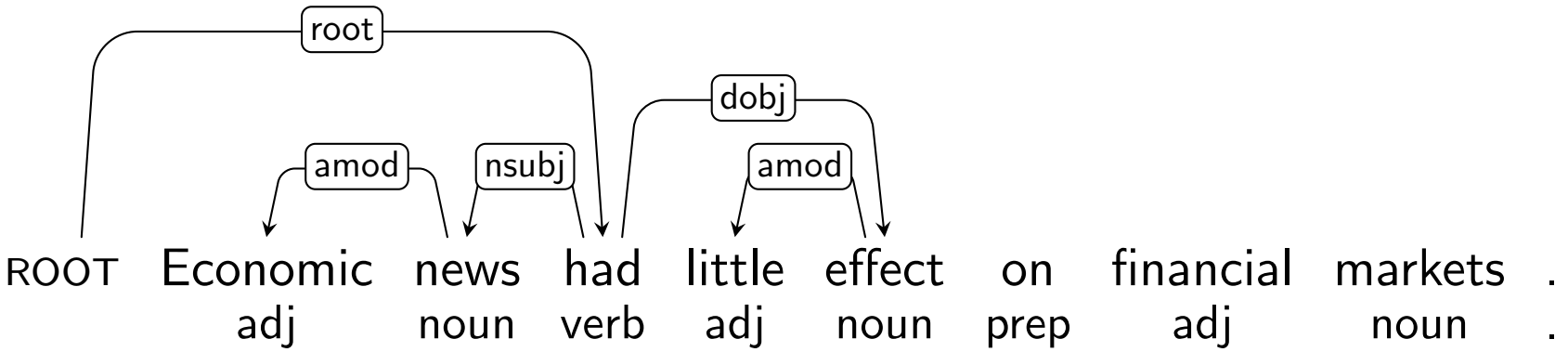
Example Transition Sequence

[ROOT, had]$_S$ [effect, on, financial, markets, .]$_B$



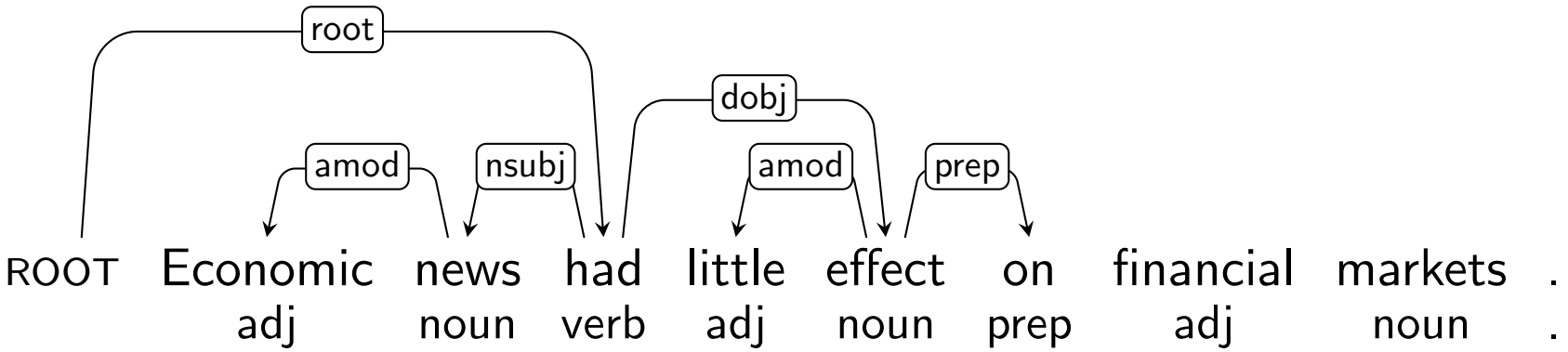
Example Transition Sequence

[ROOT, had, effect]$_S$ [on, financial, markets, .]$_B$



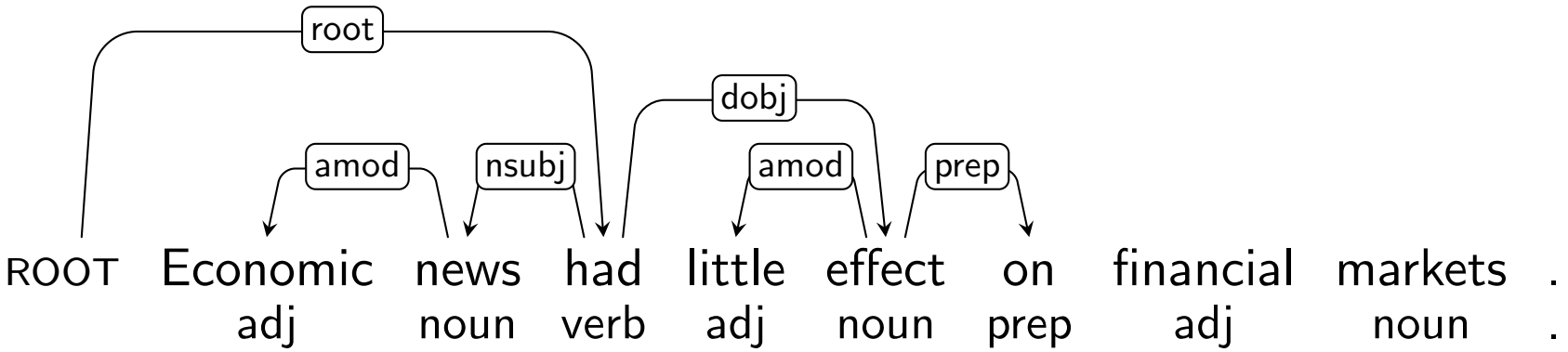
Example Transition Sequence

[ROOT, had, effect, on]$_S$ [financial, markets, .]$_B$



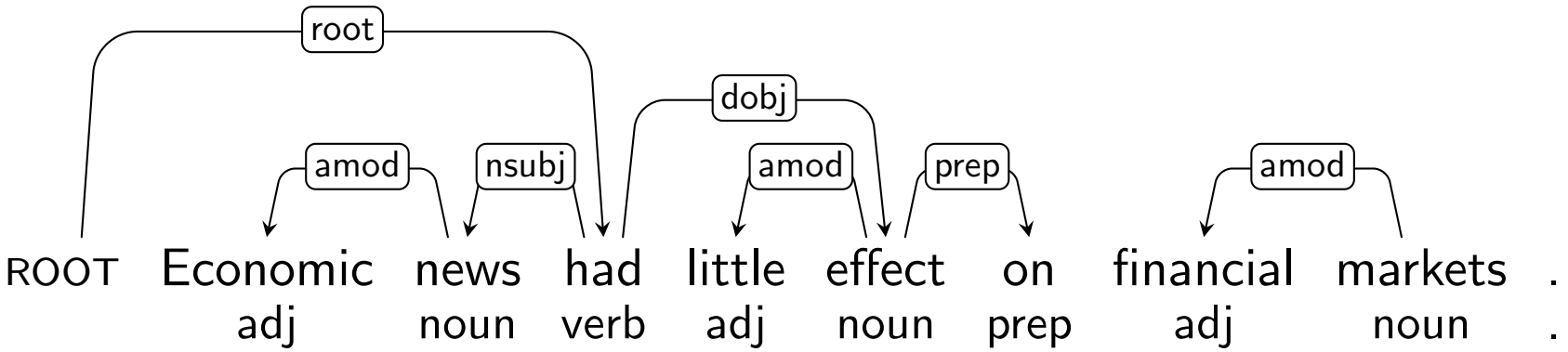
Example Transition Sequence

[ROOT, had, effect, on, financial]$_S$ [markets, .]$_B$



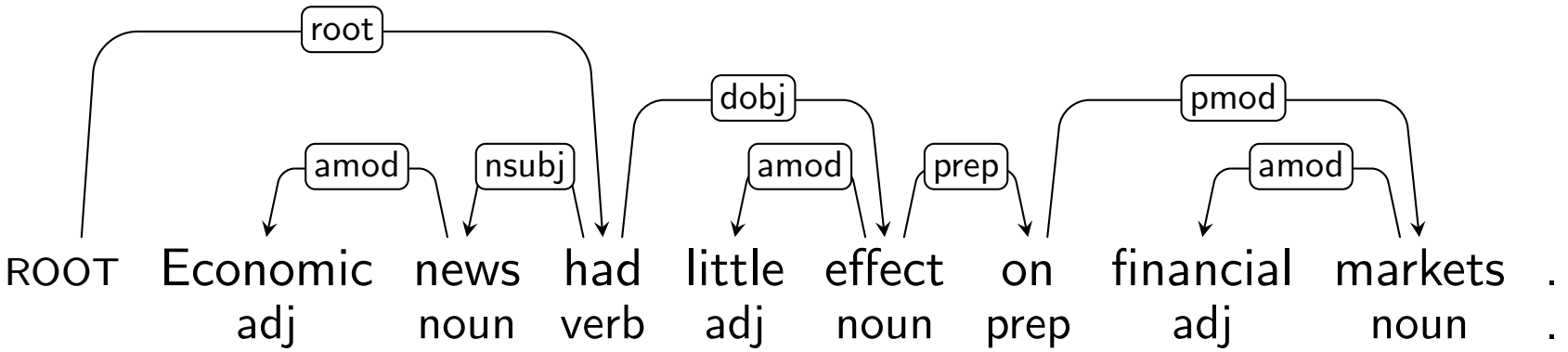
Example Transition Sequence

[ROOT, had, effect, on]$_S$ [markets, .]$_B$



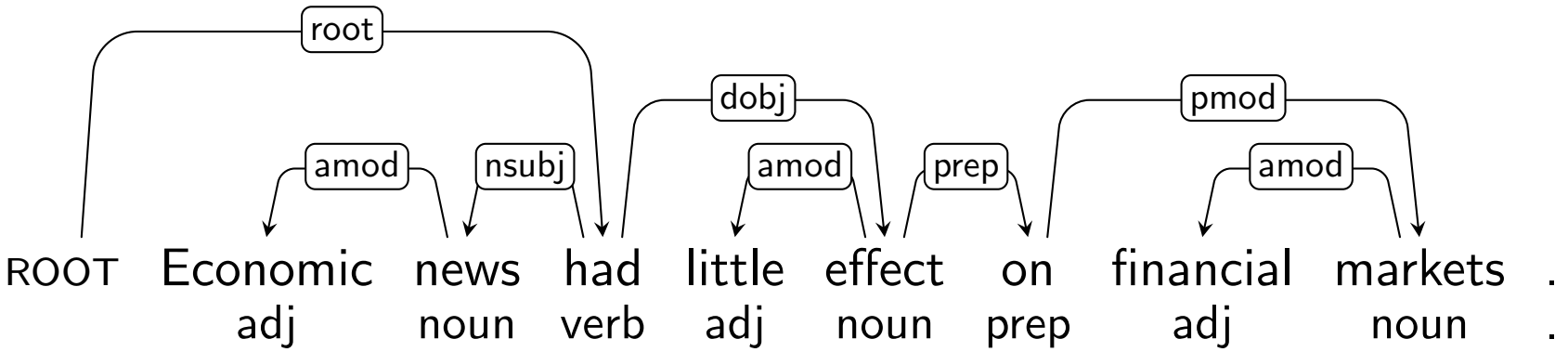
Example Transition Sequence

[ROOT, had, effect, on, markets]$_S$ [.]$_B$



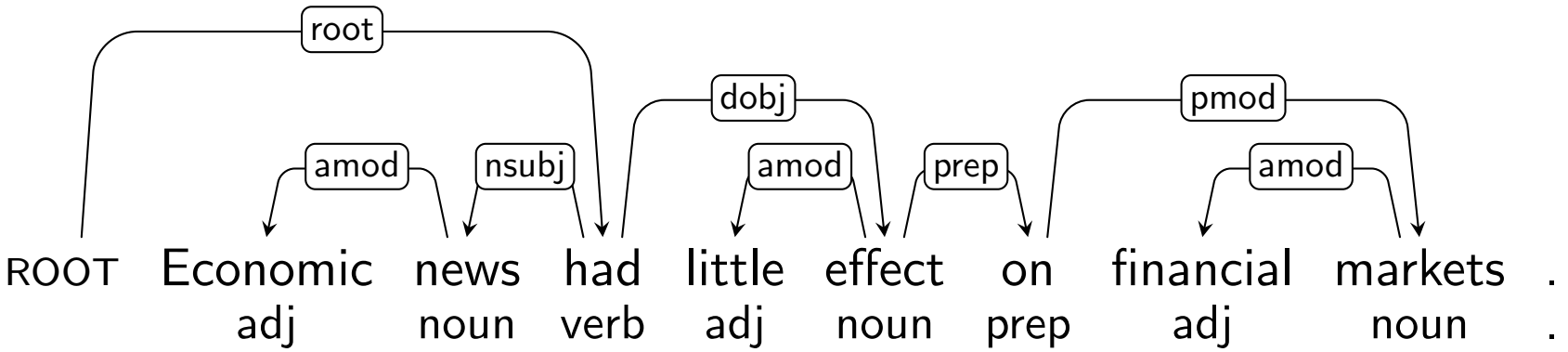
Example Transition Sequence

[ROOT, had, effect, on]$_S$ [.]$_B$



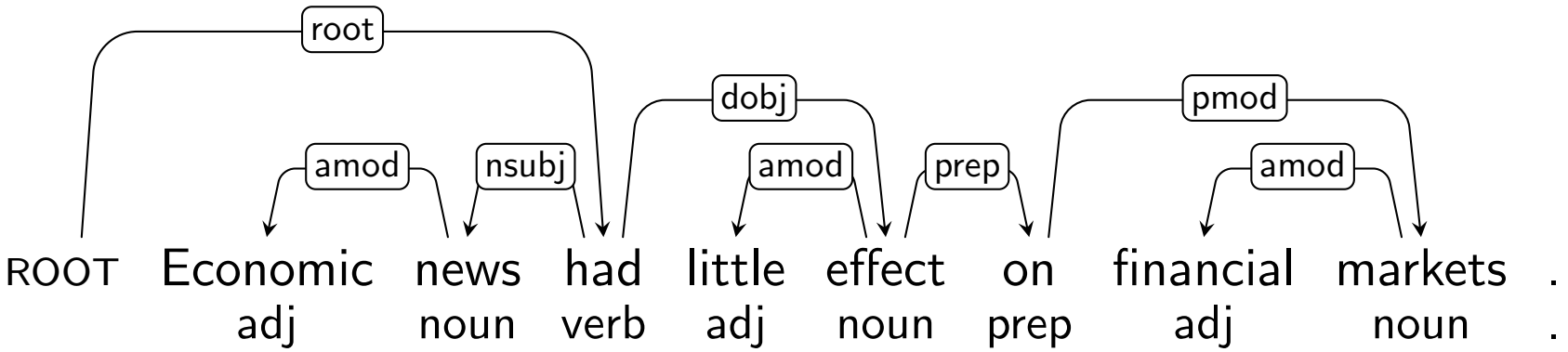
Example Transition Sequence

[ROOT, had, effect]$_S$ [.]$_B$



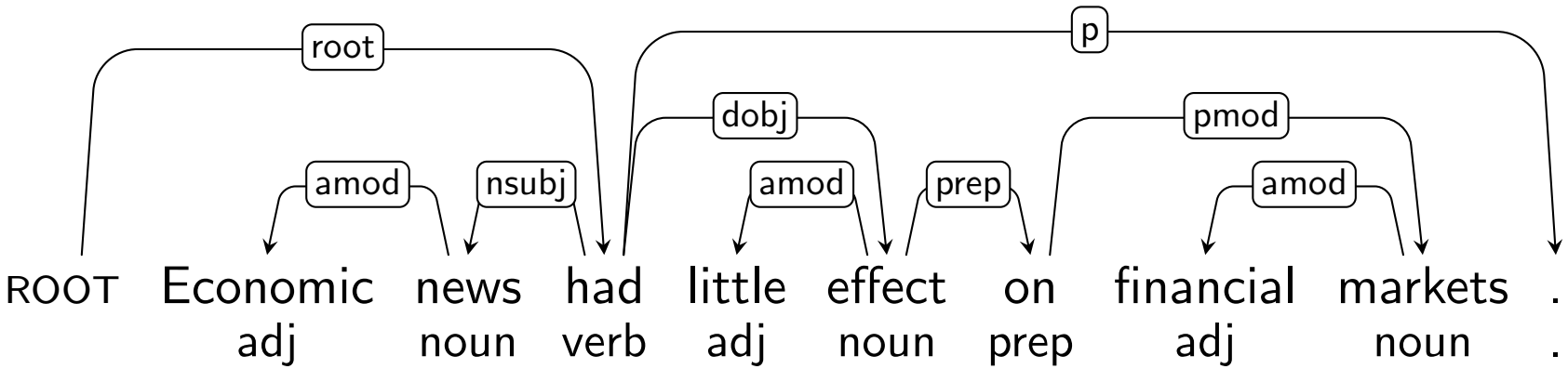
Example Transition Sequence

[ROOT, had]$_S$ [.]$_B$



Example Transition Sequence

[ROOT, had, .]$_S$ [ ]$_B$



Arc-Standard Transition System [Nivre 2004]

Configuration: (S, B, A) $[S = \text{Stack},\ B = \text{Buffer},\ A = \text{Arcs}]$

Initial: $([\,], [0, 1, \ldots, n], \{\,\})$

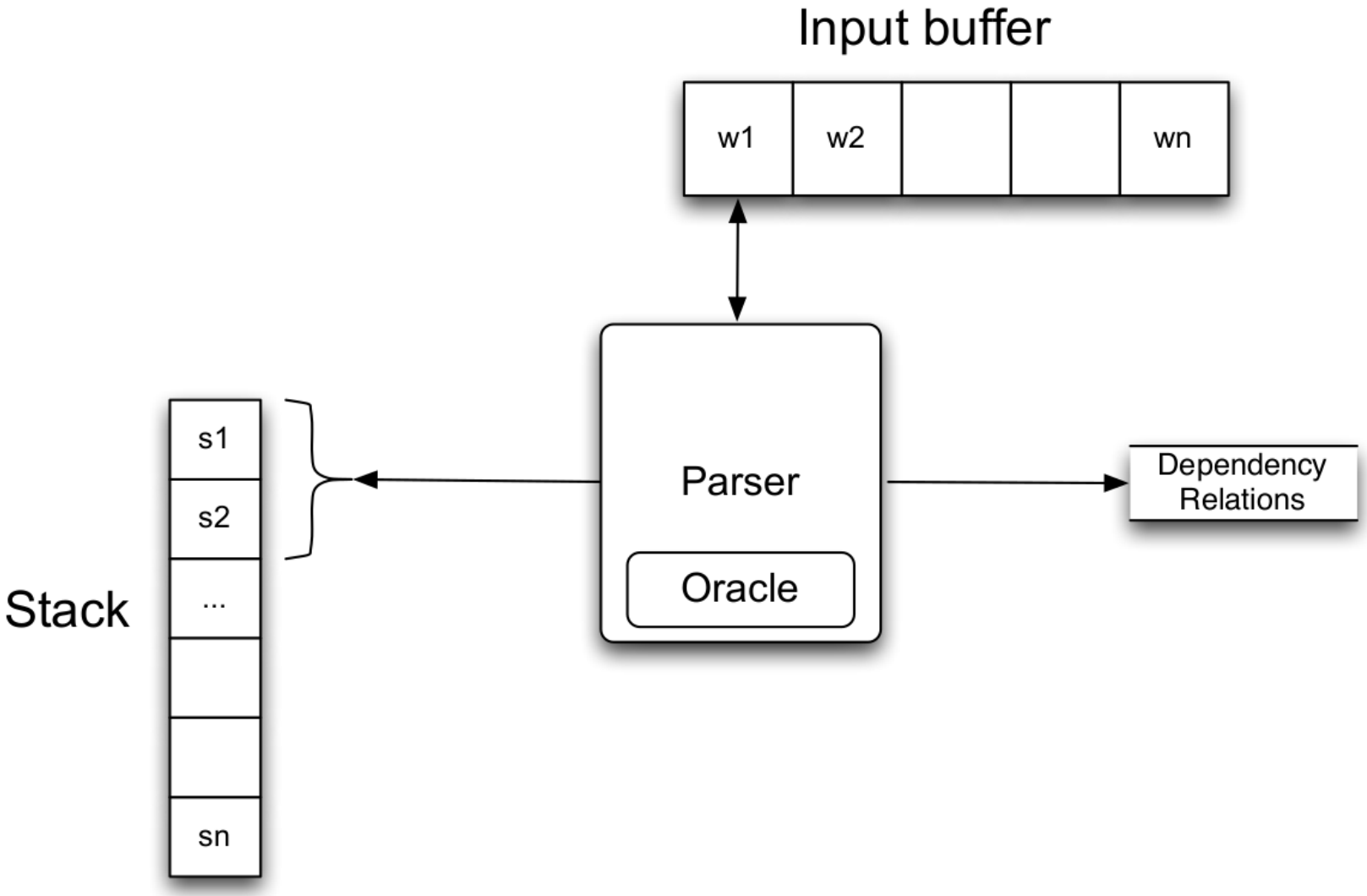
Terminal: $([0], [\,], A)$

Shift: $(S, i|B, A) \Rightarrow (S|i, B, A)$

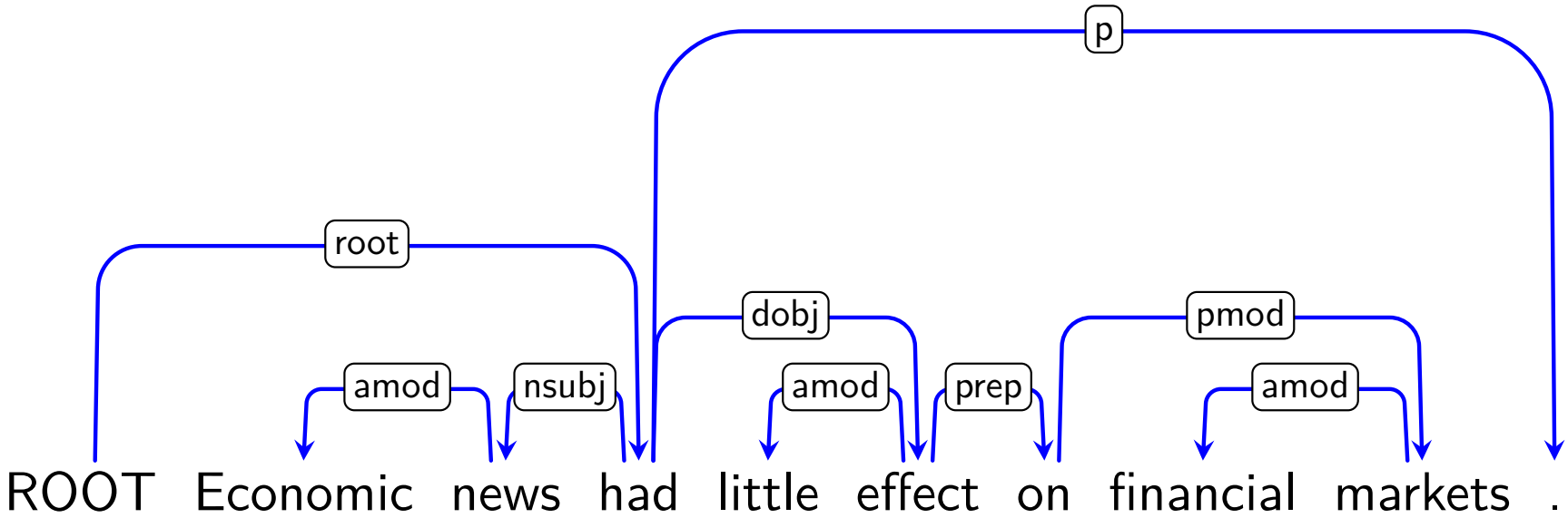
Right-Arc(k): $(S|i|j, B, A) \Rightarrow (S|i, B, A \cup \{(i, j, k)\})$

Left-Arc(k): $(S|i|j, B, A) \Rightarrow (S|j, B, A \cup \{(j, i, k)\})$ $\quad i \neq 0$

Arc-Standard More Like 'Classic' Shift–Reduce



Using the Arc-Standard Transition System



In Comparison

SHIFT SHIFT SHIFT

LEFT-ARC(AMOD)

SHIFT LEFT-ARC(NSUBJ)

SHIFT SHIFT LEFT-ARC(AMOD)

SHIFT SHIFT SHIFT

LEFT-ARC(AMOD)

RIGHT-ARC(PMOD)

RIGHT-ARC(PREP)

RIGHT-ARC(DOBJ)

SHIFT RIGHT-ARC(P)

RIGHT-ARC(ROOT)

SHIFT SHIFT LEFT-ARC(AMOD)

SHIFT LEFT-ARC(NSUBJ)

RIGHT-ARC(ROOT)

SHIFT LEFT-ARC(AMOD)

RIGHT-ARC(DOBJ)

RIGHT-ARC(PREP)

SHIFT LEFT-ARC(AMOD)

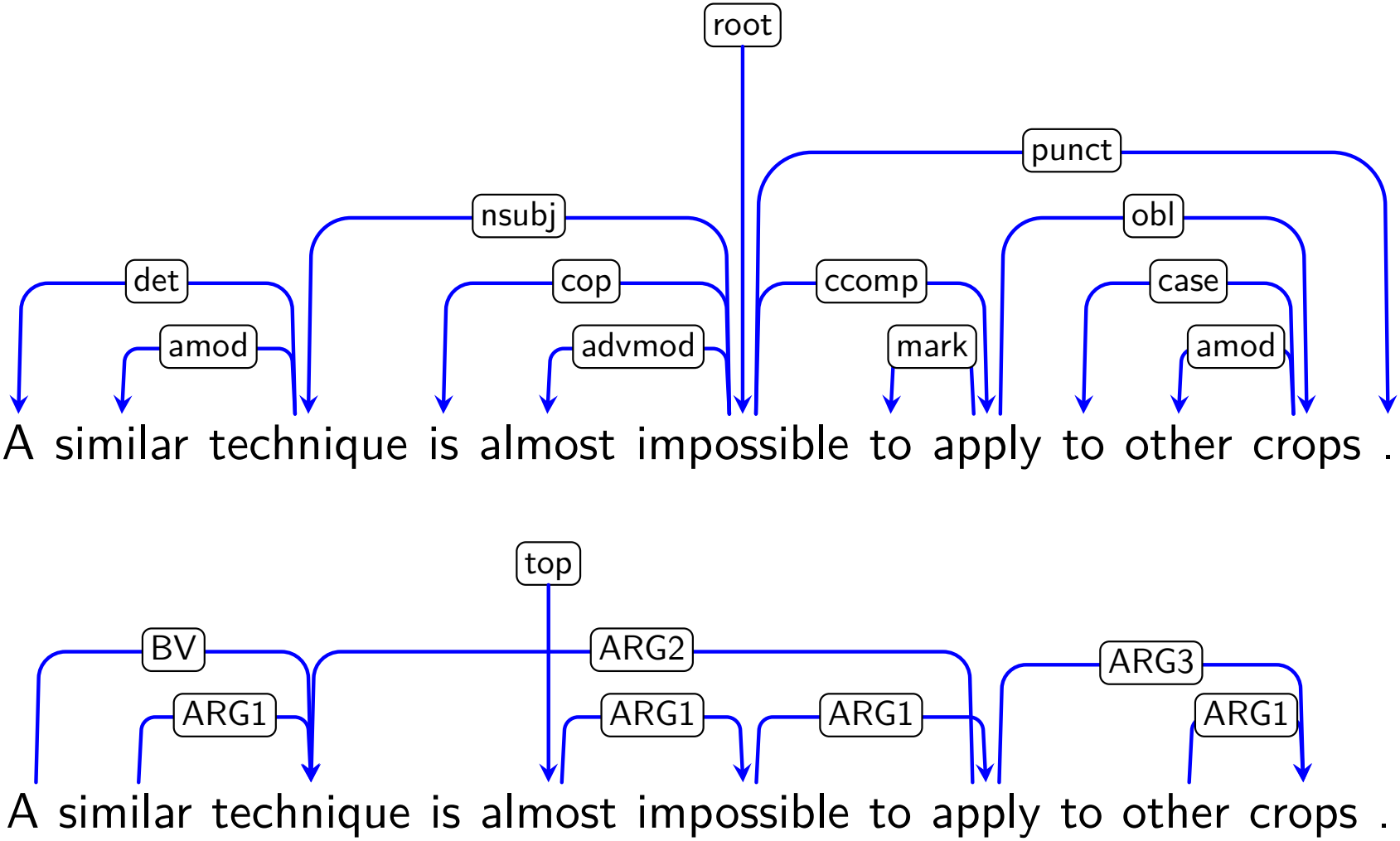
RIGHT-ARC(PMOD)

REDUCE REDUCE REDUCE

RIGHT-ARC(P)

REDUCE REDUCE

Outlook: Semantics as Dependency Graphs



- DELPH-IN MRS Dependencies: General graph (beyond rooted trees).
- Argument sharing requires re-entrancy, e.g. control or relative clauses.

Looking Back: How We had Motivated Syntactic Structure

Formal grammars describe a language, providing key notions of:

Wellformedness

- *Kim was happy because* _____ *passed the exam.*
- *Kim was happy because* _____ *final grade was an A.*
- *Kim was happy when she saw* _____ *on television.*

Meaning

- *Kim gave Sandy the book.*
- *Kim gave the book to Sandy.*
- *Sandy was given the book by Kim.*

Ambiguity

- *Kim ate sushi with chopsticks.*
- *Have her report on my desk by Friday!*