Chart-based RRG parsing using an automatically extracted RRG grammar with features

David Arps & Laura Kallmeyer & Tatiana Bladier

TreeGraSP Workshop
15 May 2019
Heinrich-Heine-Universität Düsseldorf

Agenda

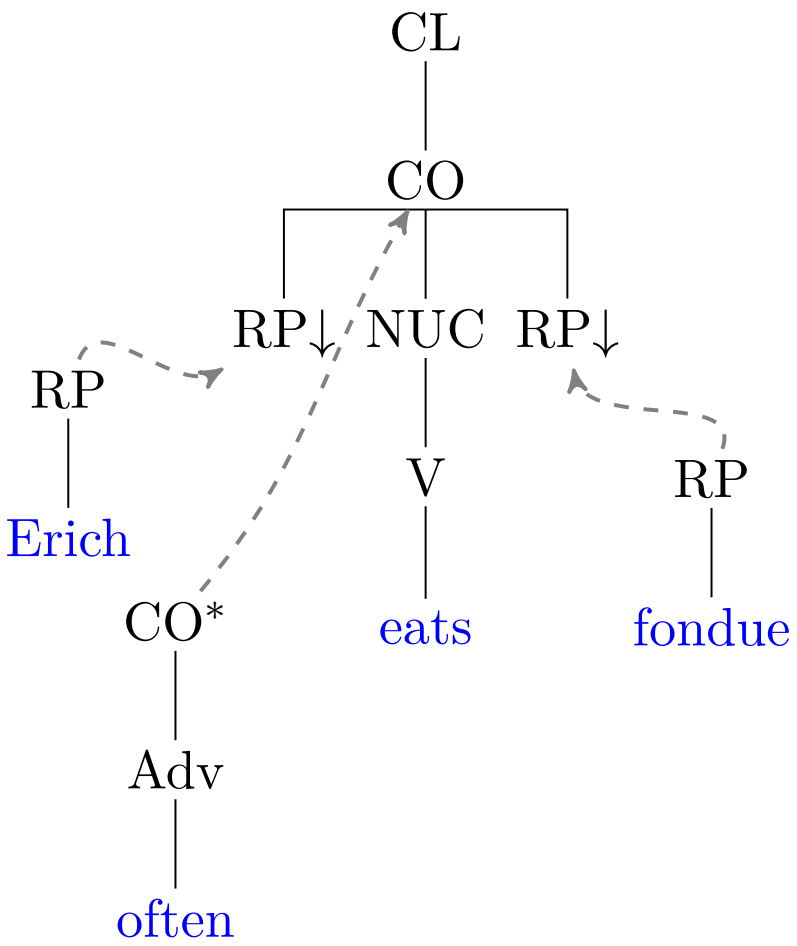
- ⋆ Chart-based bottom-up parser in CYK-style for RRG grammars
  → hand-crafted & automatically extracted RRG grammars;
  → features on elementary trees are used to improve the performance of the parser.

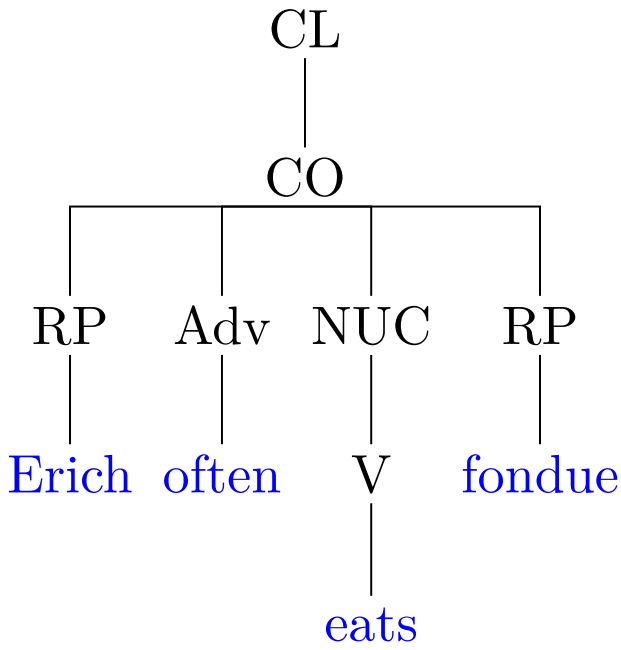
- ⋆ **Today we are presenting:**
  → Design of RRG grammars.
  → Elementary trees with features and combination operations.
  → Automatic extraction of the elementary trees (≈ supertags) from RRGbank.
  → Parsing experiments with the automatically extracted RRG grammar.

Elementary trees in RRG Grammars

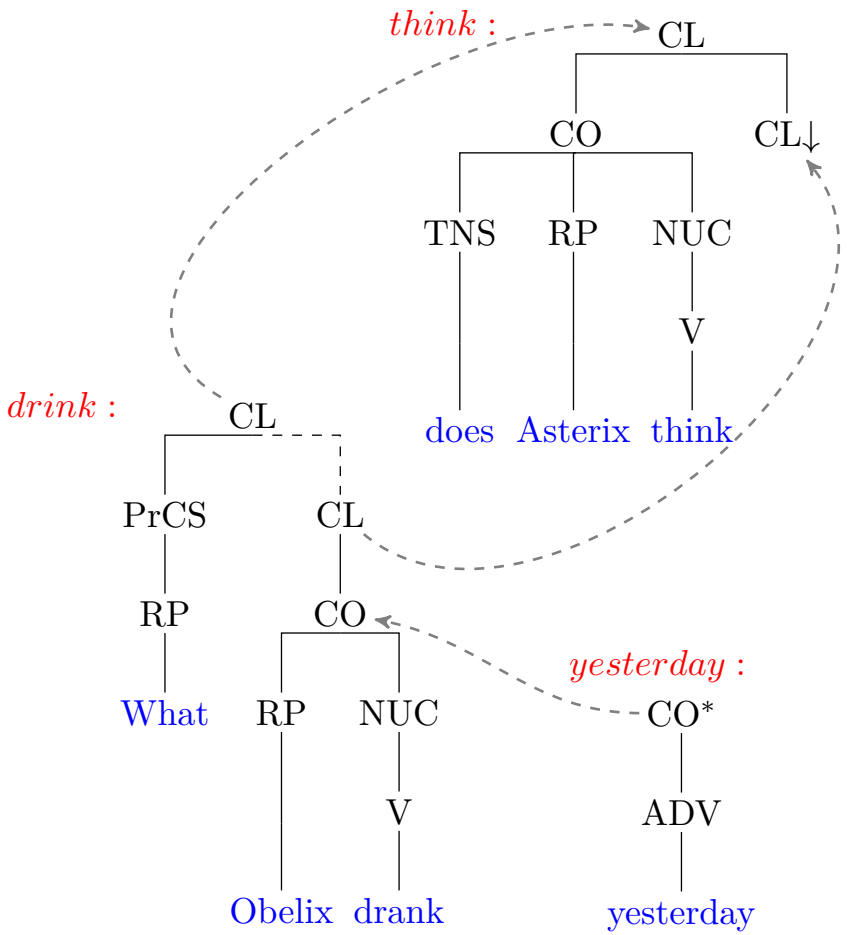
- We follow Kallmeyer et al. (2013); Osswald and Kallmeyer (2018) in design of the elementary trees in our grammar.
- Three tree composition operations:
  → *substitution* (for argument slot filling)
  → *wrapping substitution* (for argument slot filling combined with "extraction")
  → *sister adjunction* (for adding operators and periphery elements among others).

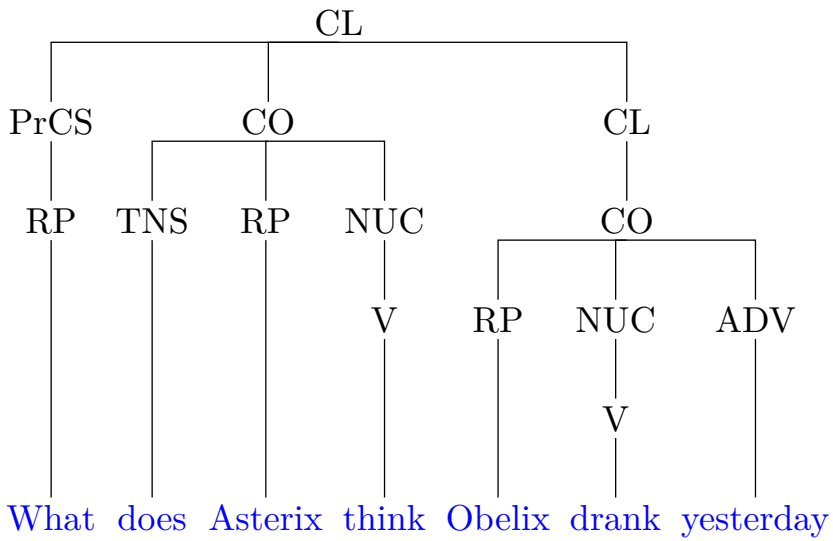
Combination operations: Substitution and sister adjunction





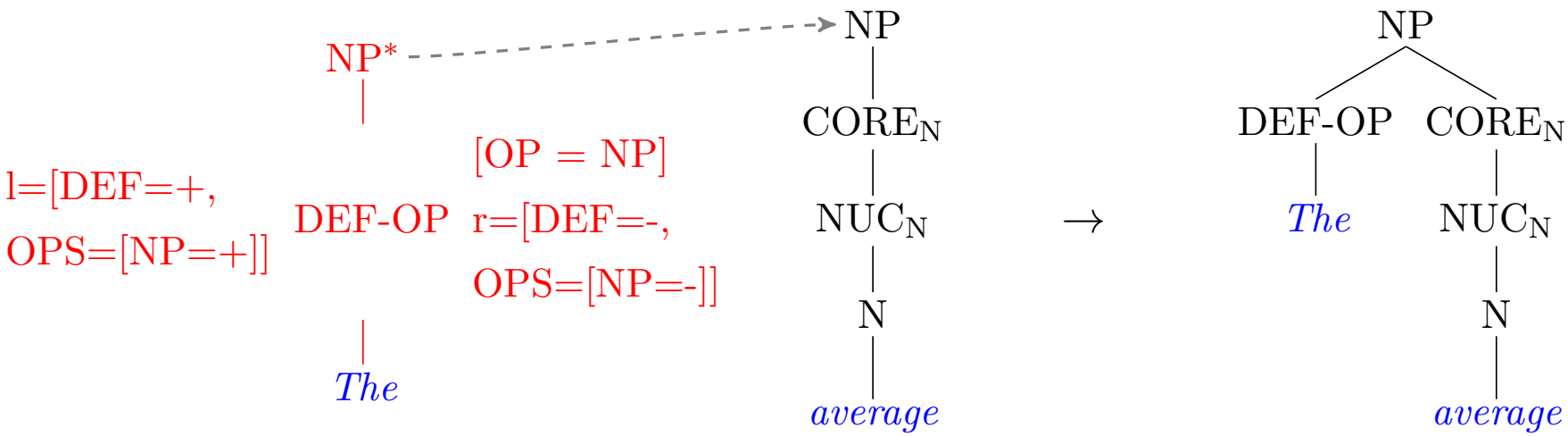
Combination operations: Wrapping substitution





Elementary trees with features

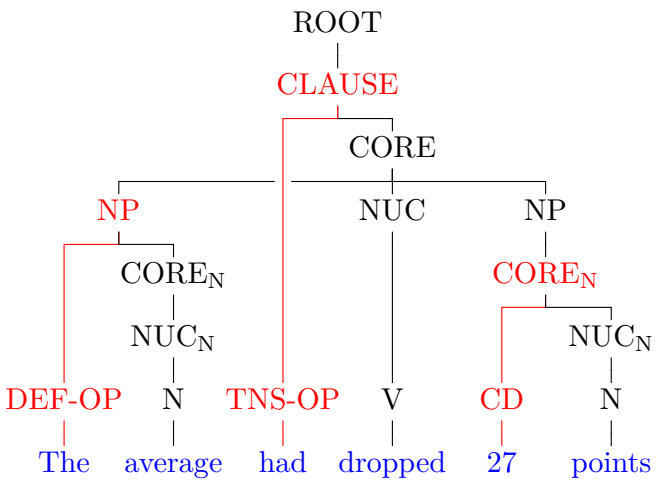
- ⋆ **Edge features:** Left and right edge features are used to model adjunction constraints during parsing.
- ⋆ **Node features:** Nodes have just a single feature structure. Node features are used to store and share syntactic or semantic information.
- ⋆ Such RRG grammars capture long-distance dependencies.

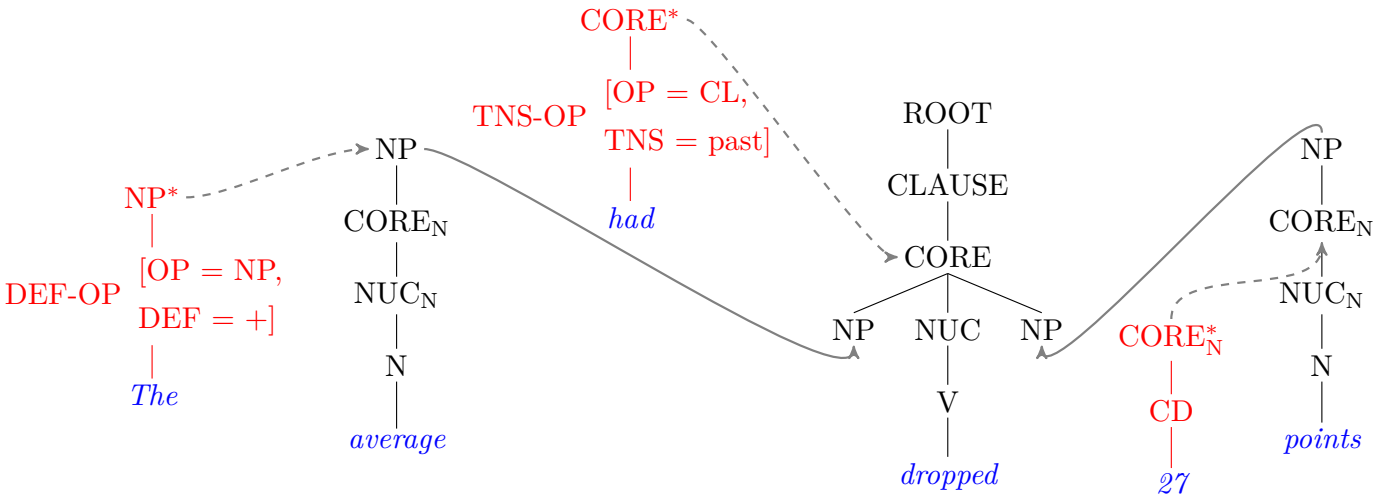


RRG Grammar extraction algorithm (1)

- ⋆ Elementary tree extraction inspired by Xia (1999) algorithm for LTAG induction.
- ⋆ Top-down extraction using heuristics from head-modifier percolation tables.
- ⋆ We use RRG structures from RRGbank for grammar induction
  → corpus of derivation structures automatically converted to RRG from the Penn Treebank (Bladier et al., 2018);
  → `rrgbank.phil.hhu.de`.
- ⋆ We decided to reattach some nodes in the RRG structures before extraction to avoid crossing branches
  → we use a node feature which indicates the correct placement of these nodes.

RRG Grammar extraction algorithm (2)





The grammar I

- source: 300 sentences from RRGBank
- ≈ 2500 lexicalized elementary trees for 1911 lexical items
- 1150 auxiliary trees
  - ADV, CLM, AUX, some nouns
- one sentence with wrapping
- removed punctuation

The grammar II

- 2 versions:
  1. no features
  2. edge features for operators model adjunction constraints
- Why extract features? all the information is in RRGBank!

- Do features improve performance?
  - How much?
  - How?

Parsing experiments I

- Parse 240 of 300 sentences we extracted the elementary trees from
- features decrease number of results by $\approx 40\%$
- w/o features: $>$ 20000 parses for some sentences (average 866)
- w features: 8200 at most (average 550)
  $\rightarrow$ still too much

Parsing experiments II

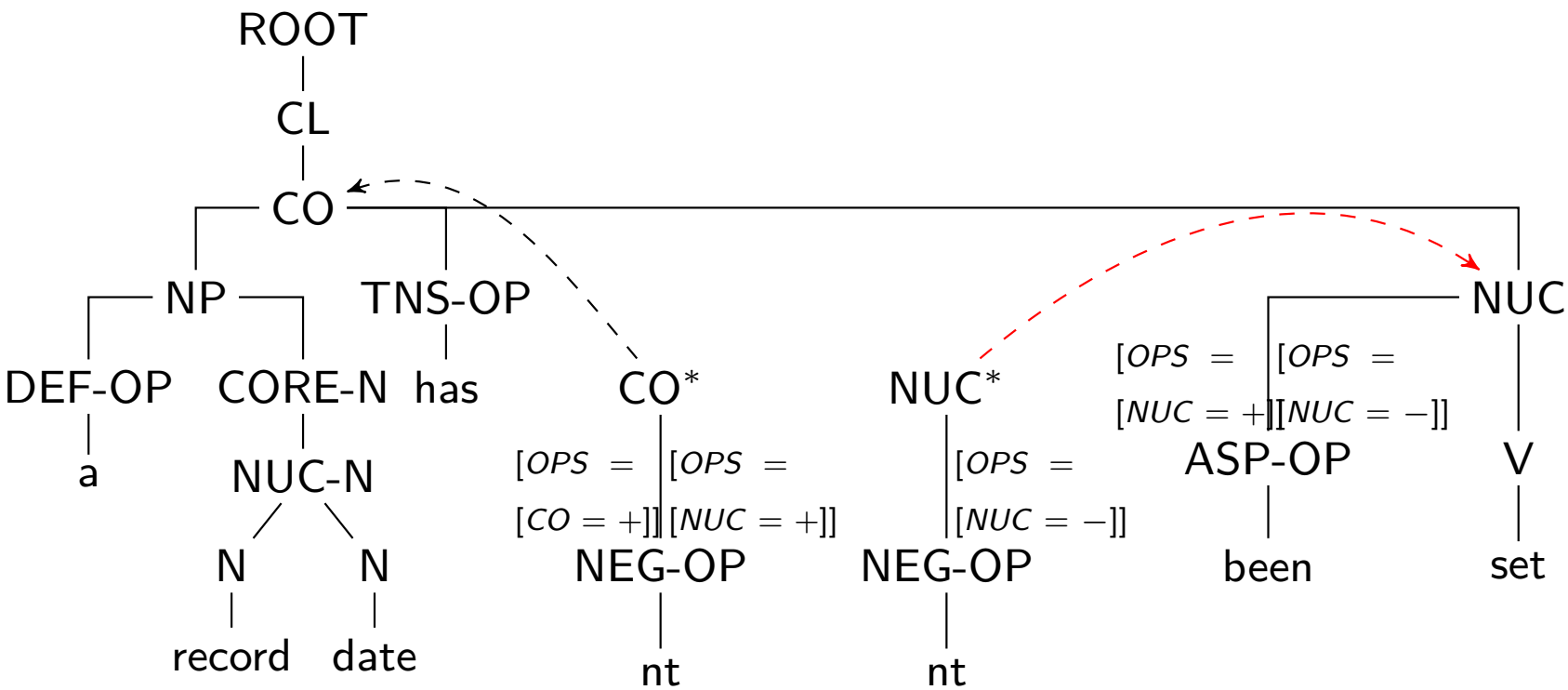
What's the problem?

- too many trees for frequent lexical items
- prepositions, articles and CLMs adjoin at different layers
- about 10 trees each for prepositions like *to*, *for*, *in*

Can this be solved by features? Sometimes

What is currently done I

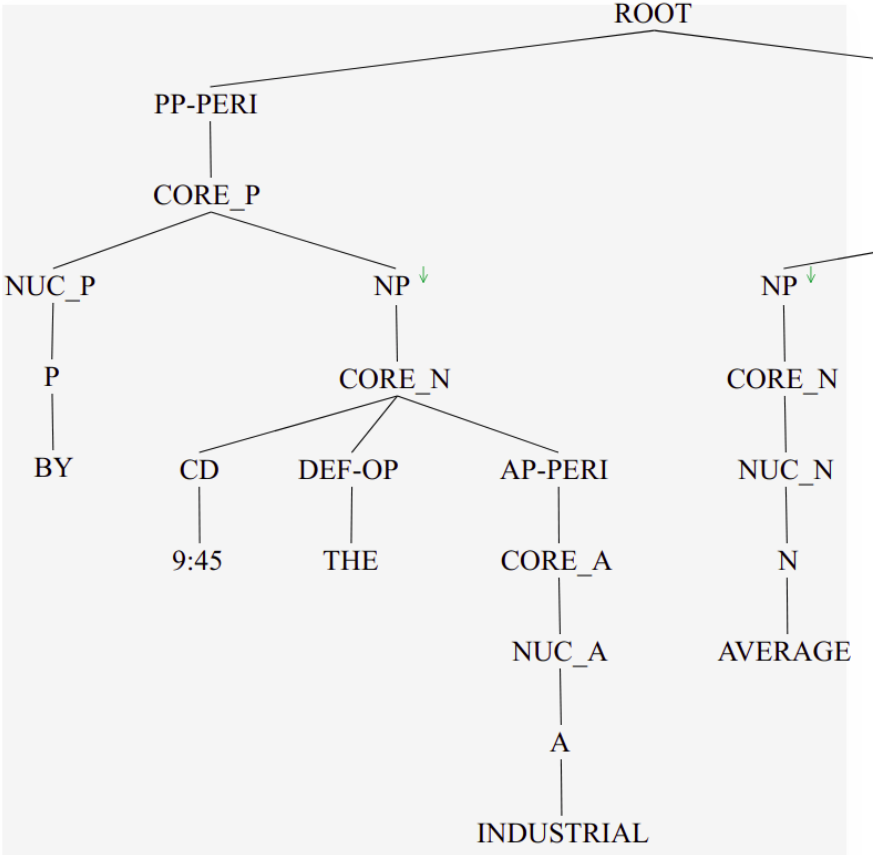
A record date hasn't been set



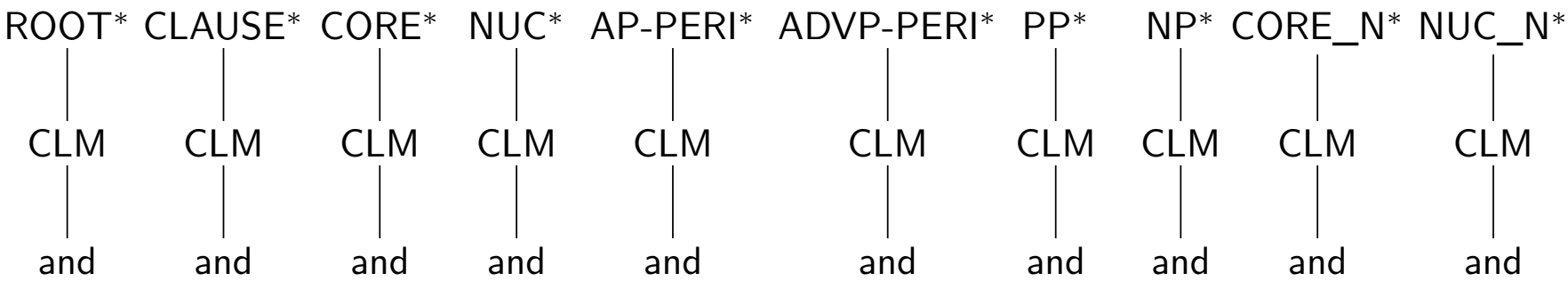
What can (and will) be done

By 9:45, the industrial average had dropped 27 points.

- Some operator features at anchors are still missing
- peripheral adjunction constraints

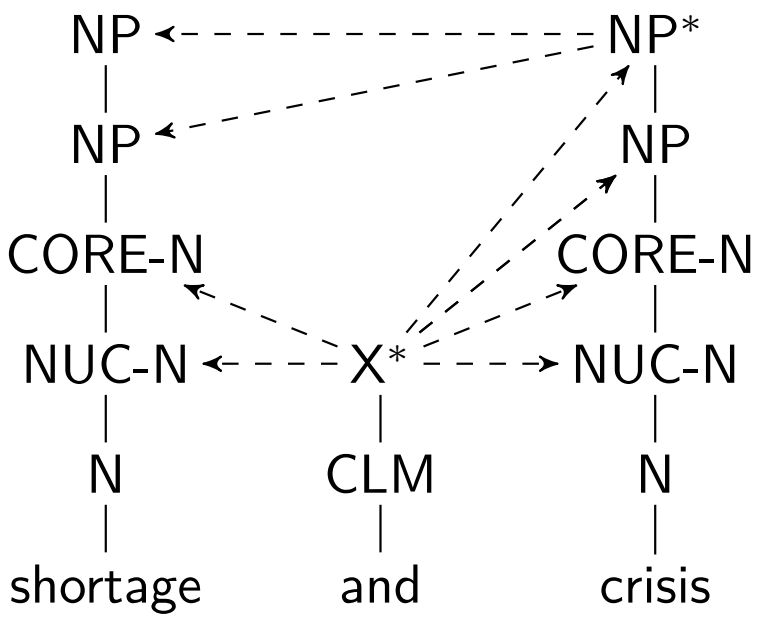


What maybe can not be done I



syntactic information needed that might not be in RRGBank

What maybe can not be done II



→ 13 results

Summary

- automatically extracted elementary trees from hand-crafted RRGBank
- experiments with exhaustive parsing of 240 sentences
- parsing w/o edge features $\rightarrow$ too many results
- some edge features already rule out bad results
- plan for the future: more edge features = better results

Performance

Necessity of probabilistic parsing

- 100s of results → not satisfying
- ambiguity and a few annotation/extraction mistakes have bad consequences
- Possible solution: probabilistic grammar and parsing

Outlook TuLiPA & RRG

- edge feature unification happens as last step of processing
  $\rightarrow$ expensive
- integration in the chart parsing algo will help

- layered lexicon and anchoring
- (Web)GUI?
- A* parsing

Thank you!

THANK YOU VERY MUCH FOR YOUR ATTENTION!

References I

Bladier, T., van Cranenburgh, A., Evang, K., Kallmeyer, L., Möllemann, R., and Osswald, R. (2018). RRGbank: a Role and Reference Grammar Corpus of Syntactic Structures Extracted from the Penn Treebank. In *Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories (TLT 2018), December 13–14, 2018, Oslo University, Norway*, number 155, pages 5–16. Linköping University Electronic Press.

Kallmeyer, L., Osswald, R., and Van Valin, Jr., R. D. (2013). Tree Wrapping for Role and Reference Grammar. In Morrill, G. and Nederhof, M.-J., editors, *Formal Grammar 2012/2013*, volume 8036 of *LNCS*, pages 175–190. Springer.

Osswald, R. and Kallmeyer, L. (2018). Towards a formalization of role and reference grammar. In Kailuweit, R., Künkel, L., and Staudinger, E., editors, *Applying and Expanding Role and Reference Grammar.*, pages 355–378. Albert-Ludwigs-Universität, Universitätsbibliothek. [NIHIN studies], Freiburg.

Xia, F. (1999). Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, pages 398–403.