

Ling 566

Dec 5, 2024

Dependency Parsing, Final Preview

Overview

- Dependency grammar: High level overview
- Variation in dependency annotations
- Redwoods treebank
- Exporting dependencies from HPSG analyses
- MRP shared tasks
- Reading questions
- Final preview

Grammatical Dependencies

- Relate words in the sentence to each other
- A labeled with the type of dependency
- Are typically represented as graphs (sometimes trees)
 - Where each node is a word in the sentence
 - Where word in the sentence is (usually) a node

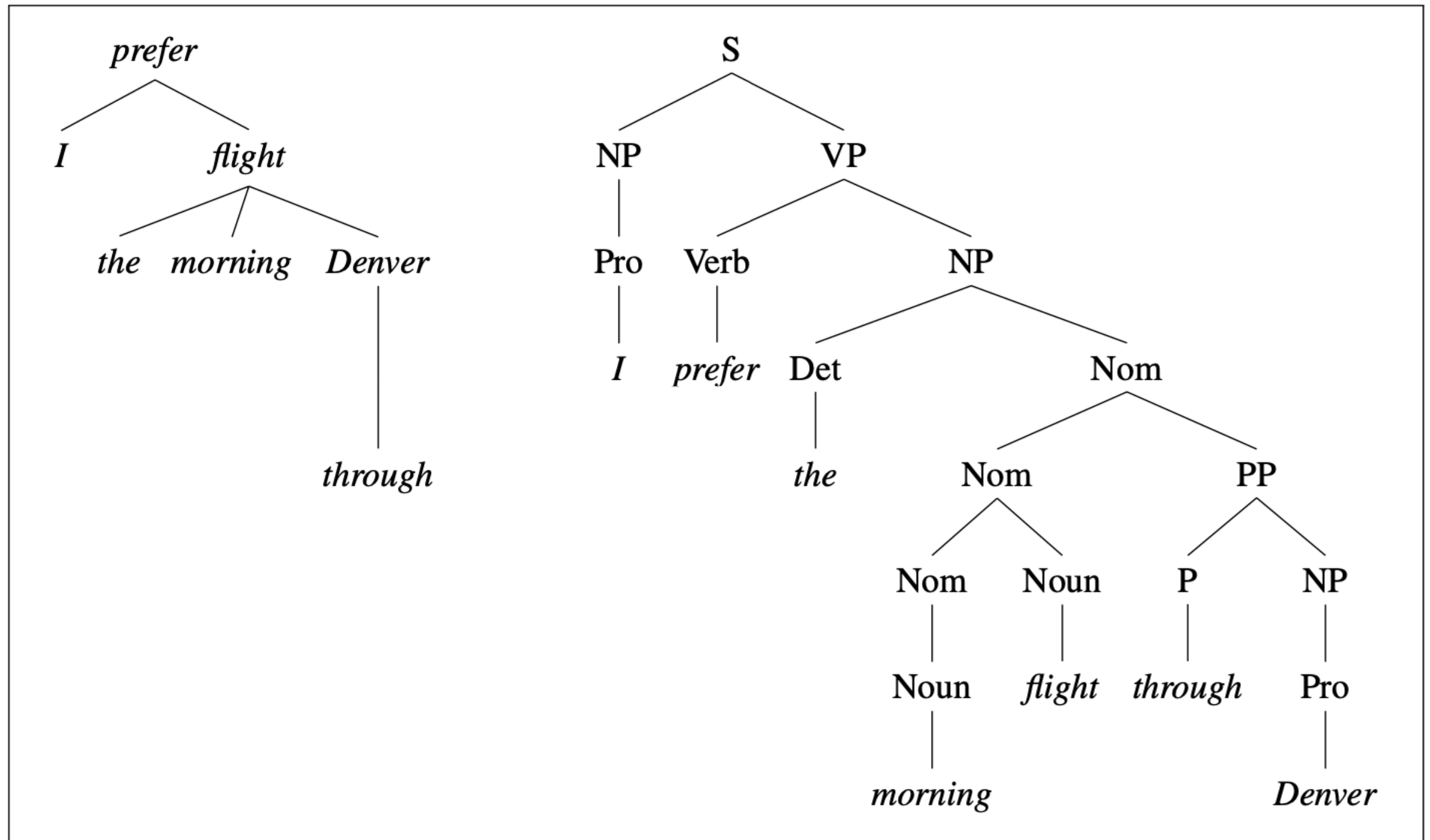
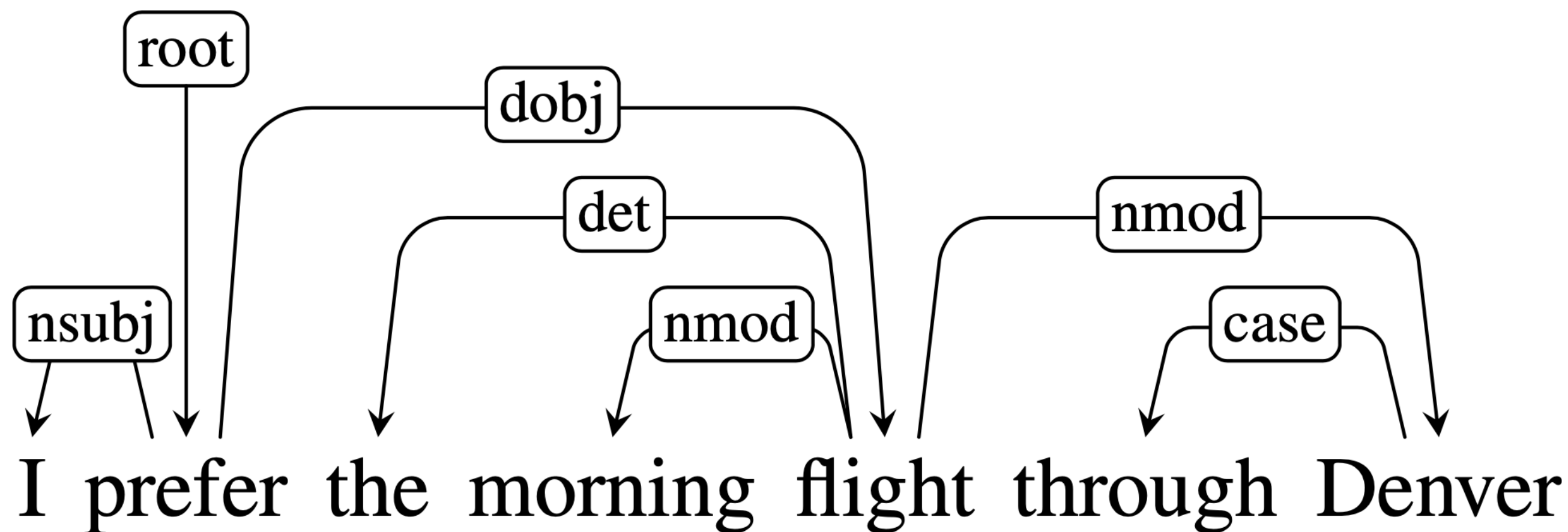


Figure 15.1 A dependency-style parse alongside the corresponding constituent-based analysis for *I prefer the morning flight through Denver*.



Dependency Grammar

- Theoretical foundations: Tesnière 1959, Mel'čuk 1988, Hudson 1984, Sgall et al 1986
- Focus not on grammaticality (“What’s a possible sentence?”) but on grammatical structure, given a string

Dependency parsing, popularity of

- Very fast algorithms
- Dependency trees tend to be closer to semantics and therefore more useful in applications than phrase structure trees
- UD project produces really appealing datasets

Dependency Treebanks: Universal Dependencies

- <https://universaldependencies.org/>
- Builds on:
 - Stanford dependencies (LFG-inspired transformation of CFG representations for English from the Stanford parser)
 - Theoretical work on dependency grammar
 - “Universal” POS tagset developed initially for cross-linguistic error analysis (McDonald and Nivre 2007)

What is needed for UD to be successful?

(from universaldependencies.org/introduction.html)

- The secret to understanding the design and current success of UD is to realize that the design is a very subtle compromise between approximately 6 things:
 - UD needs to be satisfactory on linguistic analysis grounds for individual languages.
 - UD needs to be good for linguistic typology, i.e., providing a suitable basis for bringing out cross-linguistic parallelism across languages and language families.
 - UD must be suitable for rapid, consistent annotation by a human annotator.
 - UD must be suitable for computer parsing with high accuracy.
 - UD must be easily comprehended and used by a non-linguist, whether a language learner or an engineer with prosaic needs for language processing. We refer to this as seeking a habitable design, and it leads us to favor traditional grammar notions and terminology.
 - UD must support well downstream language understanding tasks (relation extraction, reading comprehension, machine translation, ...).
- It's easy to come up with a proposal that improves UD on one of these dimensions. The interesting and difficult part is to improve UD while remaining sensitive to all these dimensions.

“Typical” dependency structures (quoted from Ivanova et al 2012, p.3)

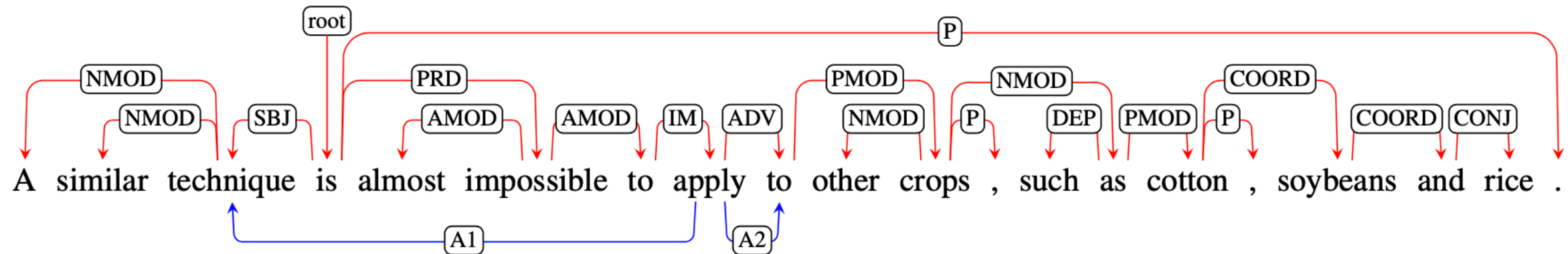
- Dependency structures are directed trees: labeled, directed graphs, where the word tokens in a sentence constitute the nodes, and
 - (i) every token in the sentence is a node in the graph (combined with a designated root node, conventionally numbered as 0),
 - (ii) the graph is (weakly) connected,
 - (iii) every node in the graph has at most one head, and
 - (iv) the graph is acyclic (Nivre et al., 2007).

Variation in dependency structure standards

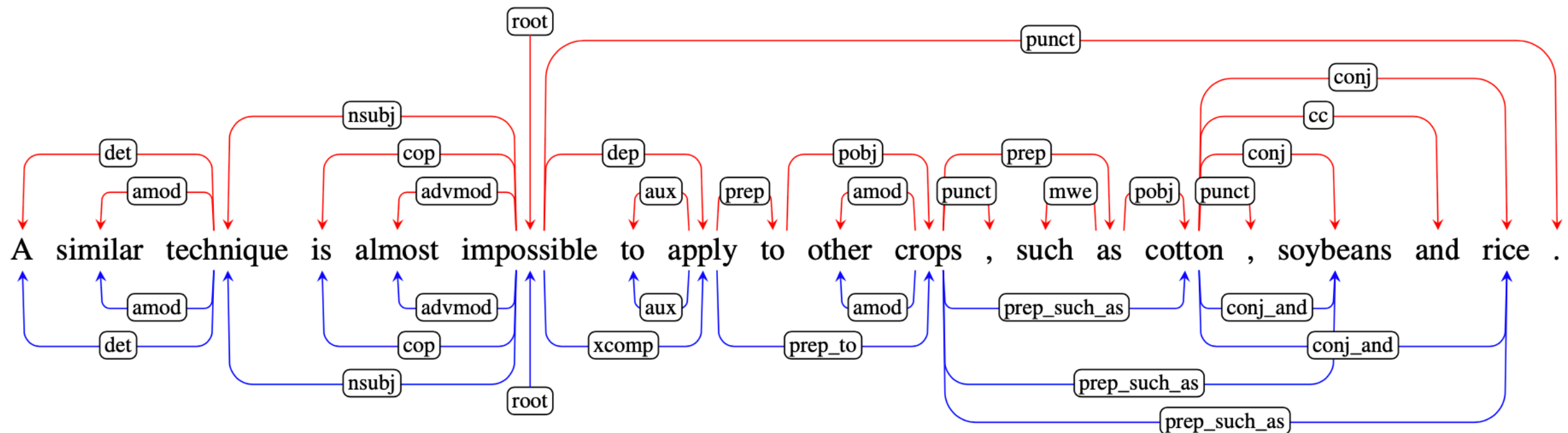
- Head selection (functional v. substantive)
- Dependency inventory
- Formal constraints on admissible graphs
- Bilexical dependencies v. allowing abstract nodes
- Are all words in the string accounted for?

Variation in dependency structure standards

(Ivanova et al 2012, Fig 3)

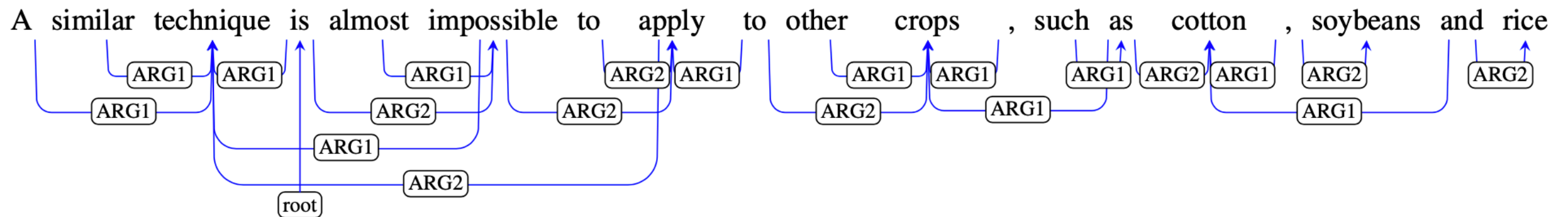


(a) CoNLL 2008 *syntactic dependencies* (CD; top) and *propositional semantics* (CP; bottom).

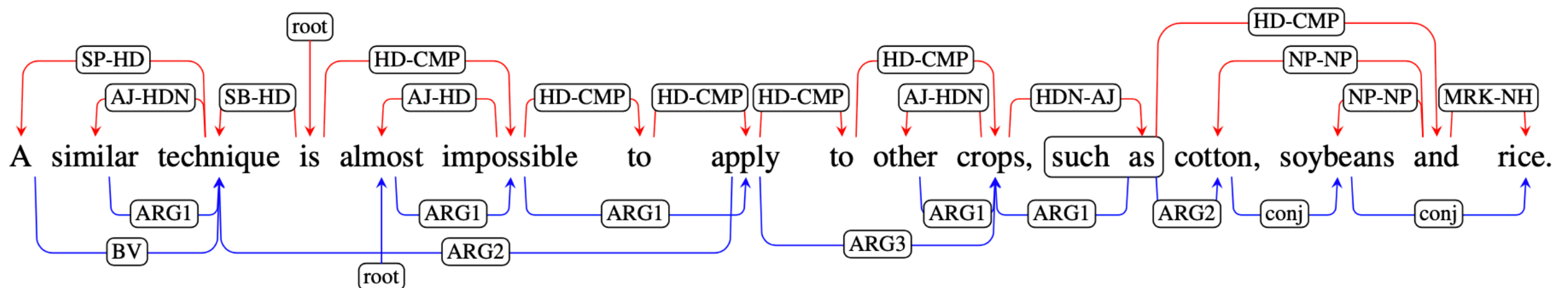


(b) Stanford Dependencies, in the so-called *basic* (SB; top) and *collapsed & propagated* (SD; bottom) variants.

Variation in dependency structure standards (Ivanova et al 2012, Fig 3)



(c) Enju *predicate-argument structures* (EP).



(d) DELPH-IN *syntactic derivation tree* (DT; top) and *Minimal Recursion Semantics* (DM; bottom).

Figure 3: Dependency representations in (a) CoNLL, (b) Stanford, (c) Enju, and (d) DELPH-IN formats.

RQ: Why do so many schemes put the root at *impossible*? Or *almost*?

Overview

- Dependency grammar: High level overview
- Variation in dependency annotations
- Redwoods treebank
- Exporting dependencies from HPSG analyses
- MRP shared tasks
- Reading questions
- Final preview

Flickinger et al 2017: Central claims

- Developing complex linguistic annotations calls for an approach which allows for the incremental improvement of existing annotations by encoding all manual effort in such a way that its value is preserved and enhanced even as the resource is improved over time
- Manual effort:
 - Annotation design => Encode in a grammar
 - Disambiguation => Store disambiguation decisions in a treebank

Redwoods Treebank (Oepen et al 2004)

- Under development since 2001
- As of 'ninth growth', 1.5 million tokens
- Initial motivation: train parse ranking models
- Also quite useful for grammar maintenance and development

Redwoods: Contents

- Rich syntactico-semantic structures, from which different ‘views’ can be projected.
- As illustrated in the following figures from Flickinger et al 2017

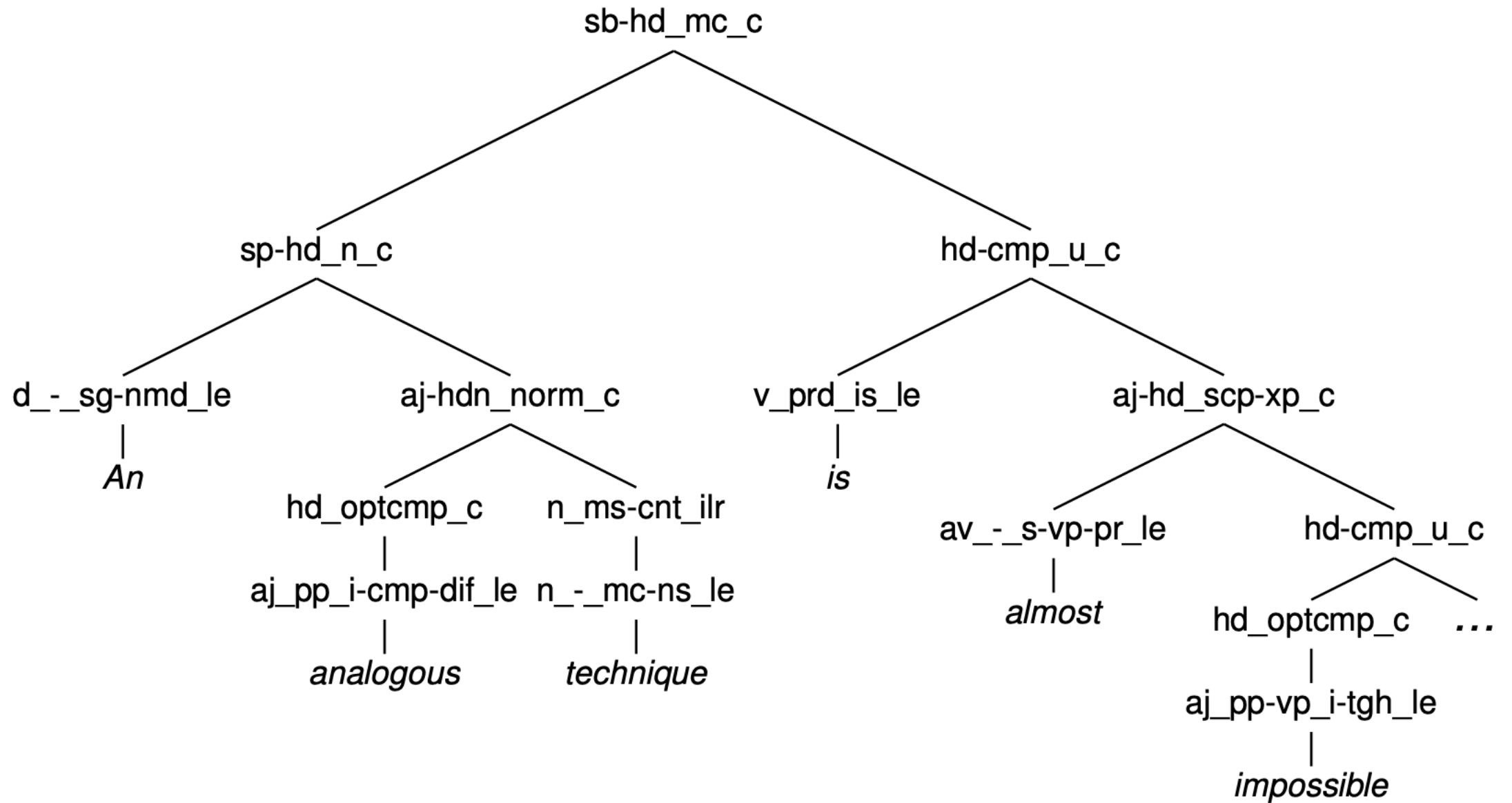
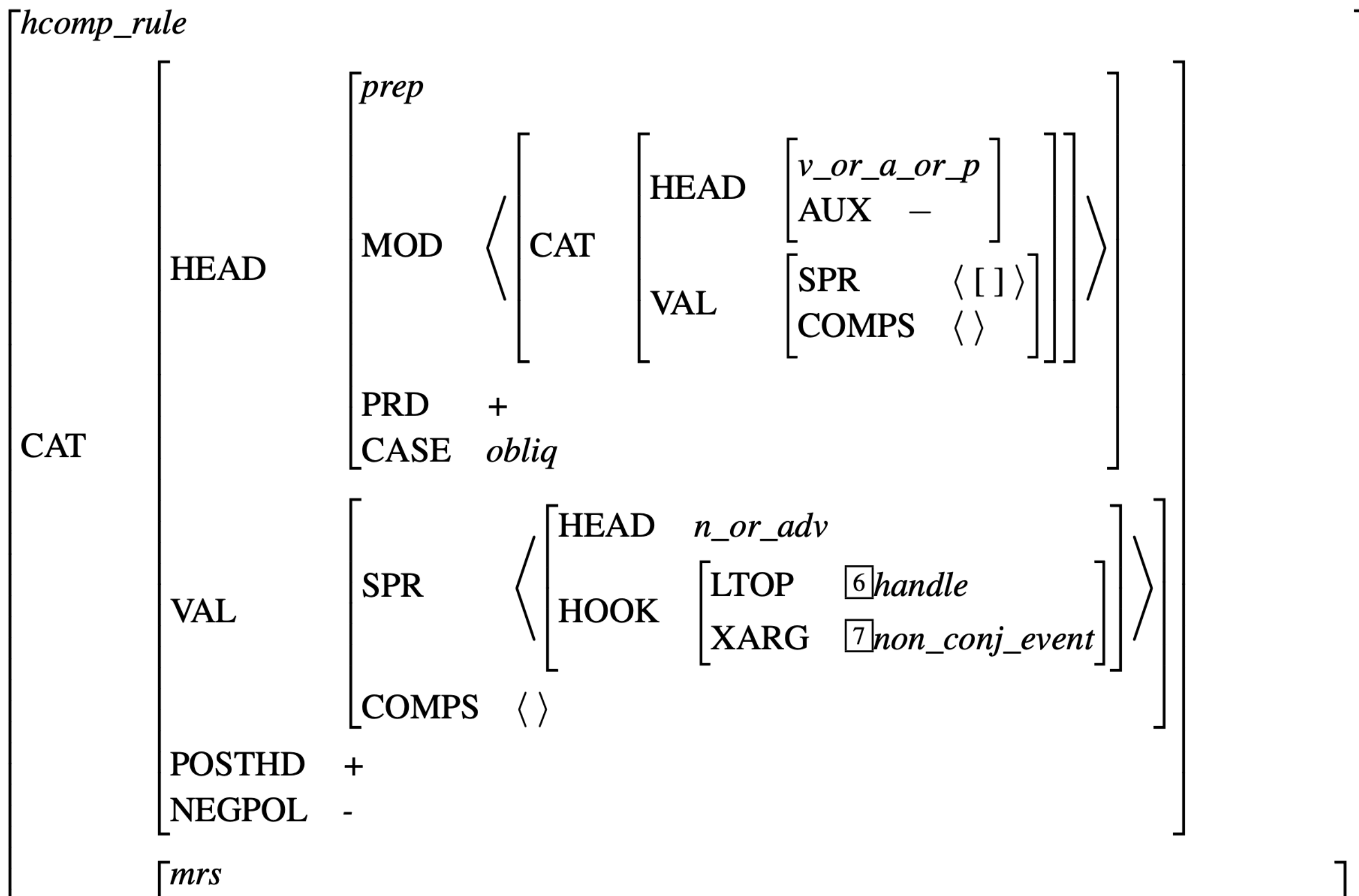


Fig. 1 ERG derivation tree for example (1).

This snippet from the ERG has non-branching nodes. Aren't we trying to avoid this?



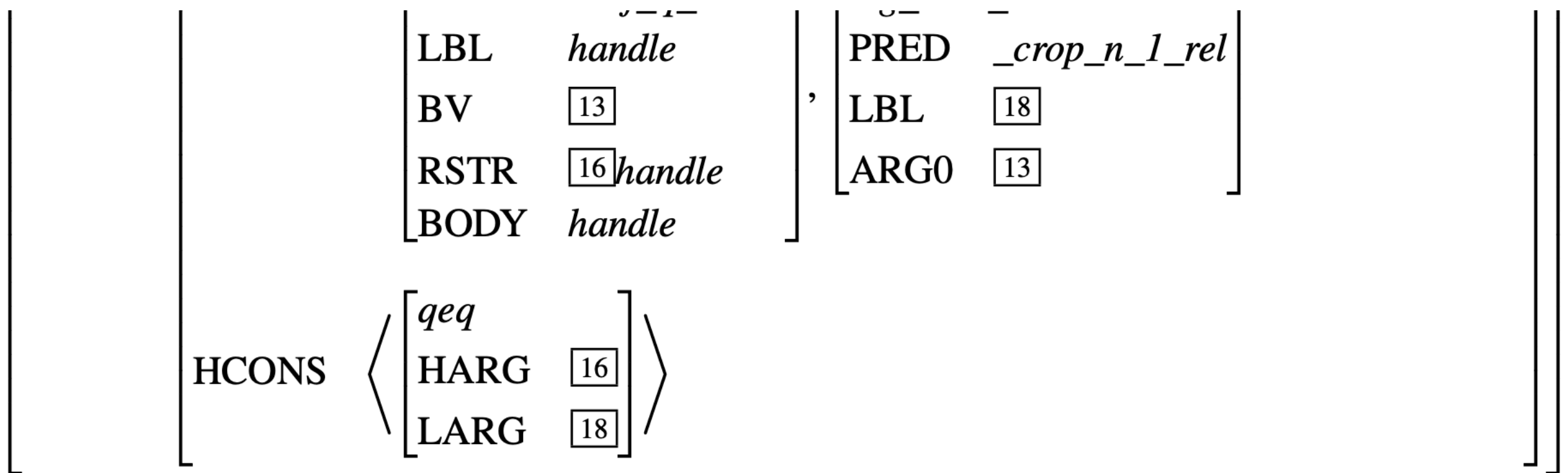


Fig. 2 Partial feature structure for PP *to other crops*

$$\begin{aligned}
&\langle h_1, \\
&\quad h_4: _a_q(\text{BV } x_6, \text{RSTR } h_7, \text{BODY } h_5), \\
&\quad h_8: _analogous_a_to(\text{ARG0 } e_9, \text{ARG1 } x_6), h_8: \text{comp}(\text{ARG0 } e_{11}, \text{ARG1 } e_9, \text{ARG2 } _), \\
&\quad h_8: _technique_n_1(\text{ARG0 } x_6), \\
&\quad h_2: _almost_a_1(\text{ARG0 } e_{12}, \text{ARG1 } h_{13}), h_{14}: _impossible_a_for(\text{ARG0 } e_3, \text{ARG1 } h_{15}, \text{ARG2 } _), \\
&\quad h_{17}: _apply_v_to(\text{ARG0 } e_{18}, \text{ARG1 } _, \text{ARG2 } x_6, \text{ARG3 } x_{20}), \\
&\quad h_{21}: _undef_q(\text{BV } x_{20}, \text{RSTR } h_{22}, \text{BODY } h_{23}), h_{24}: _other_a_1(\text{ARG0 } e_{25}, \text{ARG1 } x_{20}), \\
&\quad h_{24}: _crop_n_1(\text{ARG0 } x_{20}) \\
&\quad \{ h_1 =_q h_2, h_7 =_q h_8, h_{13} =_q h_{14}, h_{15} =_q h_{17}, h_{22} =_q h_{24} \} \rangle
\end{aligned}$$

Fig. 3 Minimal Recursion Semantics for example (1).

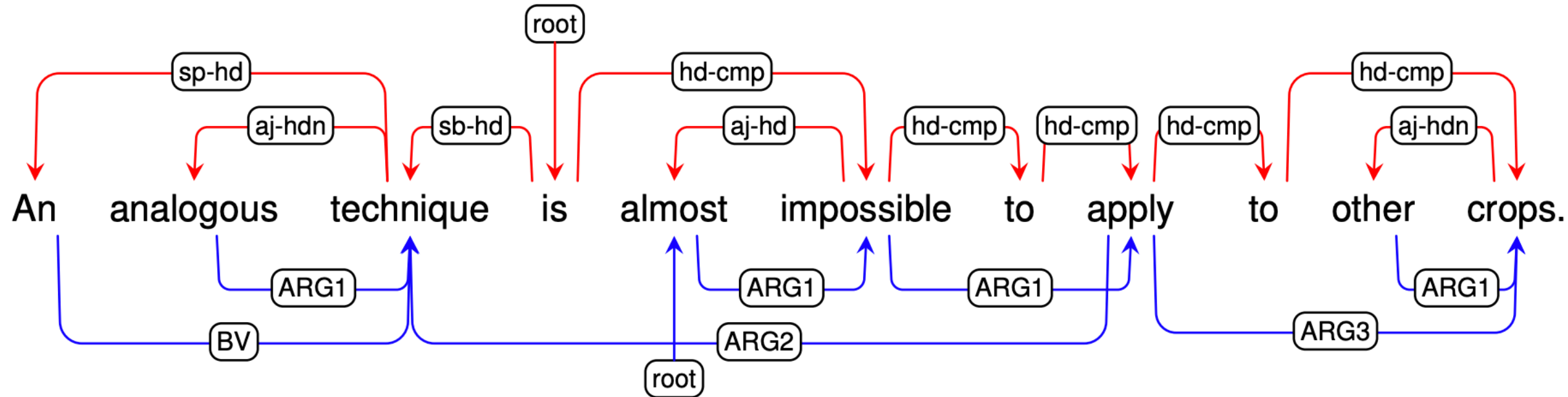


Fig. 4 Bi-lexical syntactic and semantic dependencies for (1).

Redwoods: Methodology

- Parse input corpus
- Calculate ‘discriminants’: properties shared by only a subset of the trees in parse forest (Carter 1997)
 - Picking one tree from among thousands or millions would be infeasible
 - Drawing trees with that level of detail would be infeasible
 - Picking discriminants is quite doable!
- Store both resulting tree & discriminants chosen (and inferred)
 - Maximum value out of all human annotator time

Very high inter-annotator agreement
=> very consistent annotation

- From Bender et al 2015, over 150 sentences from *The Little Prince*

Metric	Annotator Comparison			
	A vs. B	A vs. C	B vs. C	Average
Exact Match	0.73	0.65	0.70	0.70
EDM _a	0.93	0.92	0.94	0.93
EDM _{na}	0.94	0.94	0.95	0.94

Table 1: Exact match ERS and Elementary Dependency Match across three annotators.

- Comparable metric for AMR over the same data is 0.71 “SMATCH” (comparable to EDM) (Banarescu et al 2013)

Dynamic treebanking

- Dynamic *refinement* of the treebank
 - Parse corpus with new grammar (better coverage, improved representations)
 - Rerun discriminants chosen in previous annotation rounds
 - Address remaining added ambiguity / newly parsed sentences
- Dynamic *extension* of the treebank
 - Linguistic analysis encoded as a grammar (as opposed to annotation guidelines) can be automatically deployed to new text

Automatic conversion: Syntactic dependencies (Ivanova et al 2012)

- Remove (collapse) non-branching rules
- Head daughter is marked in (most?) ERG rules
- Dependencies labels come from valence features

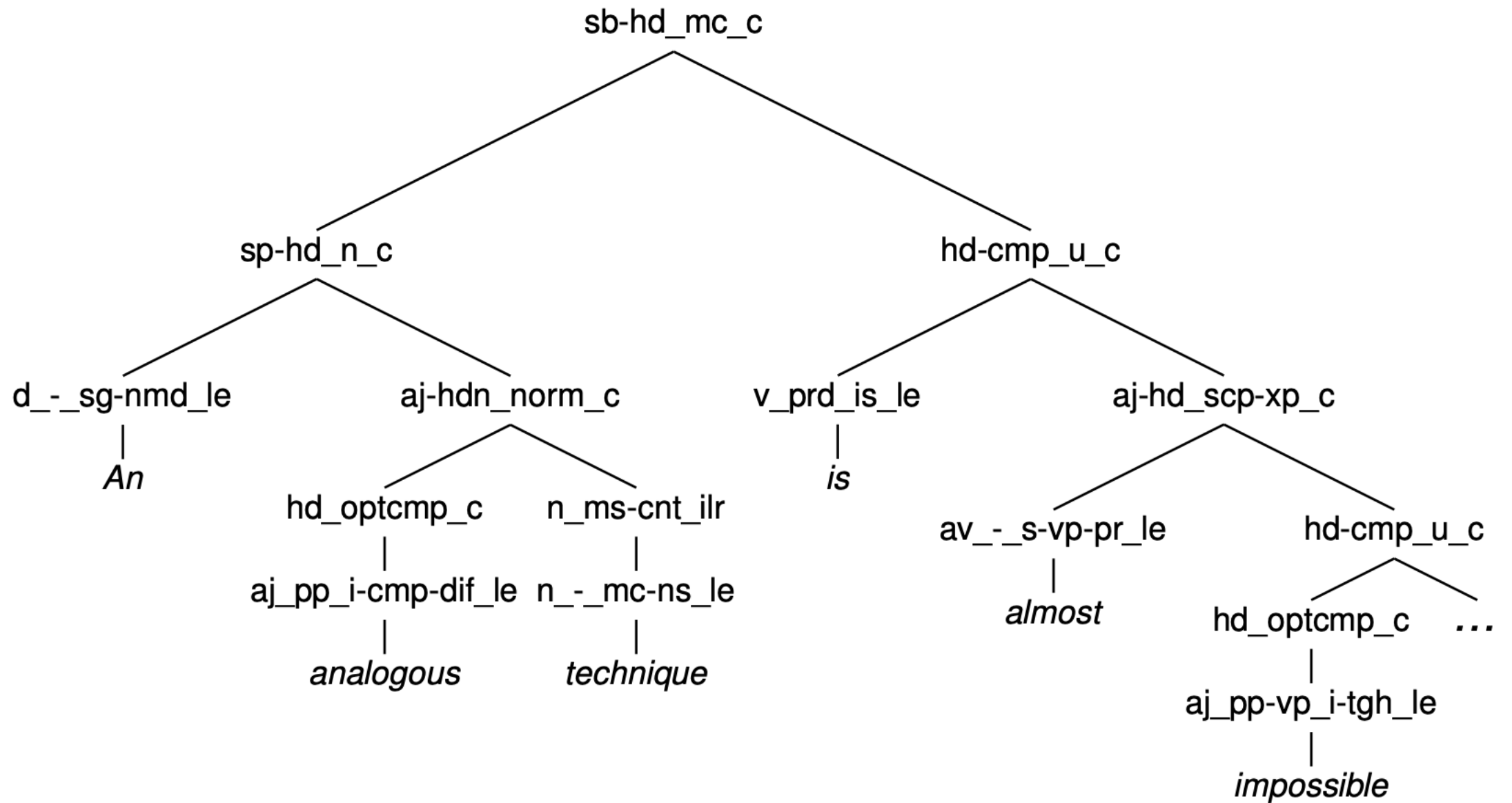


Fig. 1 ERG derivation tree for example (1).

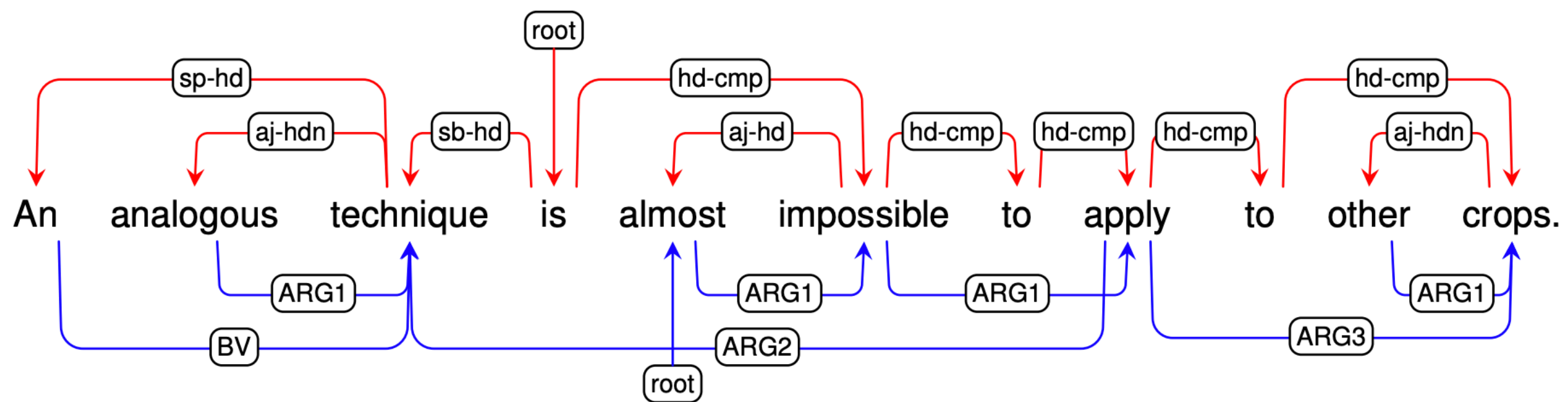


Fig. 4 Bi-lexical syntactic and semantic dependencies for (1).

Automatic conversion: Semantic dependencies (Ivanova et al 2012)

- Lossy conversion of MRS to EDS
(elementary dependency structures,
Oepen & Lønning 2006)
- Regular predicates: ARGn
dependency between tokens
- Transparent and redundant
predicates: remove
- Relational predicates: predicate is
dependency label

```
{ e12
  _1:_a_q(BV x6)
  e9:_similar_a_to(ARG1 x6)
  x6:_technique_n_1
  e12:_almost_a_1(ARG1 e3)
  e3:_impossible_a_for(ARG1 e18)
  e18:_apply_v_to(ARG2 x6, ARG3 x19)
  _2:undef_q(BV x19)
  e25:_other_a_1(ARG1 x19)
  x19:_crop_n_1
  e26:_such+as_p(ARG1 x19, ARG2 x27)
  _3:undef_q(BV x27)
  _4:undef_q(BV x33)
  x33:_cotton_n_1
  _5:undef_q(BV i38)
  x27:implicit_conj(L-INDEX x33, R-INDEX i38)
  _6:undef_q(BV x43)
  x43:_soybeans/nns_u_unknown
  i38:_and_c(L-INDEX x43, R-INDEX x47)
  _7:undef_q(BV x47)
  x47:_rice_n_1
}
```

Figure 2: ERG Elementary Dependency Structure.

Variation in dependency structure standards (Ivanova et al 2012, Fig 3)

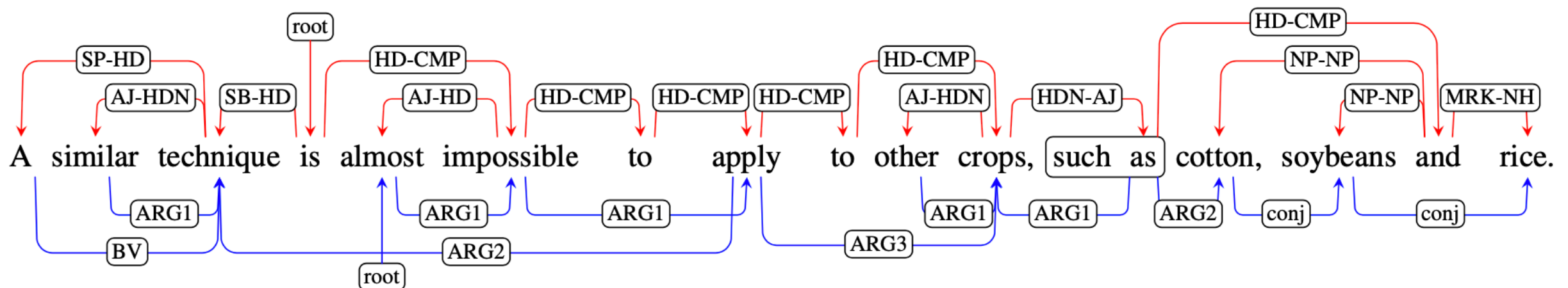
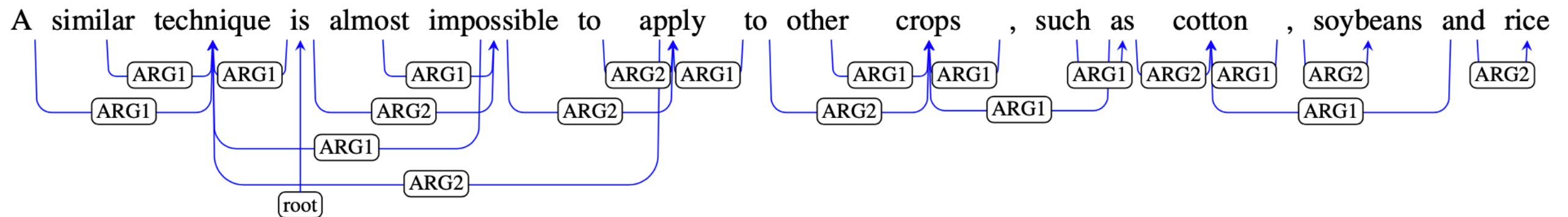


Figure 3: Dependency representations in (a) CoNLL, (b) Stanford, (c) Enju, and (d) DELPH-IN formats.

MRP (meaning representation parsing) shared tasks

- <http://mrp.nlpl.eu/2020/index.php>

RQs: Domain adaptation

- The authors state " the converted data will enable experimentation in domain and genre adaptation for parsers". Would dependency parsers differ across domains and genres?

RQs: Evaluation

- The paper presented an automatic conversion procedure, which converts the HPSG representations in the treebanks to a set of syntactic and semantic dependencies. However, it doesn't present any way to evaluate its accuracy or provide any metrics that can show us how effective the procedure is.
- I'm curious about whether the HPSG method will outperform dependency methods in real world data. If this is the case, is there any established baseline metrics that we can access as a reference?

RQs: Cross-framework comparison

- If I'm understanding the nature of the dataset correctly, the conclusions in the paper are drawn based off of ten sentences. Is this a large enough sample to generalize well? (I'm thinking back to our discussion last class about the wide variety of phenomena and the amount of data/rules we would need to capture everything, so the low number struck me as odd.) Also, is there any evidence that these conclusions would generalize cross-linguistically, beyond the fact that some of these frameworks have been applied to other languages?
- What kind of interoperability is being discussed here, and what are the specific benefits of interoperability between frameworks? What makes Jaccard similarity a good proxy for interoperability (if that's the case)?

RQs: Applications

- Are there applications where dependencies vs. HPSG are superior to the other?
- How does the availability of this automated conversion method help strengthen the argument for using HPSG? Furthermore, the authors mention in the last section that their conversion software could “enable various follow-up activities.” What are some examples of these activities/applications?
- How might preprocessing input data with syntactic analysis techniques (e.g., parse trees, dependency structures) enhance the training efficiency and performance of (LLMs), given the implicit syntactic understanding they already develop during training?

RQs: Why read this paper?

- I am not sure I got out of the reading what I was supposed to get out of it - it just looks like the authors wanted more people to use the resources they created. I wonder if I am missing something. If I were to read the paper again, is there a specific section or insight in this paper that I should focus on or try to understand?

Overview

- Dependency grammar: High level overview
- Variation in dependency annotations
- Redwoods treebank
- Exporting dependencies from HPSG analyses
- MRP shared tasks
- Reading questions
- Final preview