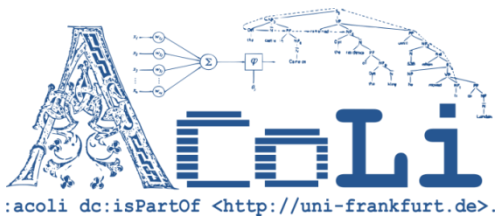
Graph-based Annotation Engineering
Towards a gold corpus for Role and Reference Grammar

Christian Chiarcos & Christian Fäth

Applied Computational Linguistics (ACoLi)

Goethe-Universität Frankfurt am Main



Annotation Engineering: Create gold data for machine learning

- low resource languages are on the rise
  - NLP: projection and induction techniques
    - UD, UniMorph
  - linguistics & philologies
    - quantitative methods and digital turn
- however, SOTA parsing paradigms grew out of European traditions
  - grammar frameworks actually *designed for* the analysis of low resource languages have a low level of technical support
  - here: Role and Reference Grammar (RRG)

Annotation Engineering with CoNLL-RDF

https://github.com/acoli-repo/conll-rdf

„RDF done in an NLP-friendly fashion“
(Chiarcos & Fäth@LDK-2017)

- CoNLL/TSV
  - one word per line, empty line between sentences
  - one column per annotation (except for certain semantic phenomena)
    - dialects/tools differ wrt. arrangement of columns
    - existing applications are mostly dialect-specific

⇒ idea: use RDF to

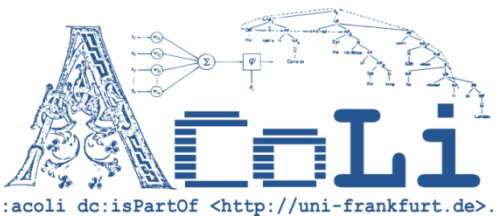
disentangle format logic (table)
from transformation logic (graph)

1. generic parser into a shallow RDF representation
2. manipulations with order-invariant graph transformation: SPARQL Update

Role and Reference Grammar (RRG)

Towards an implementation of

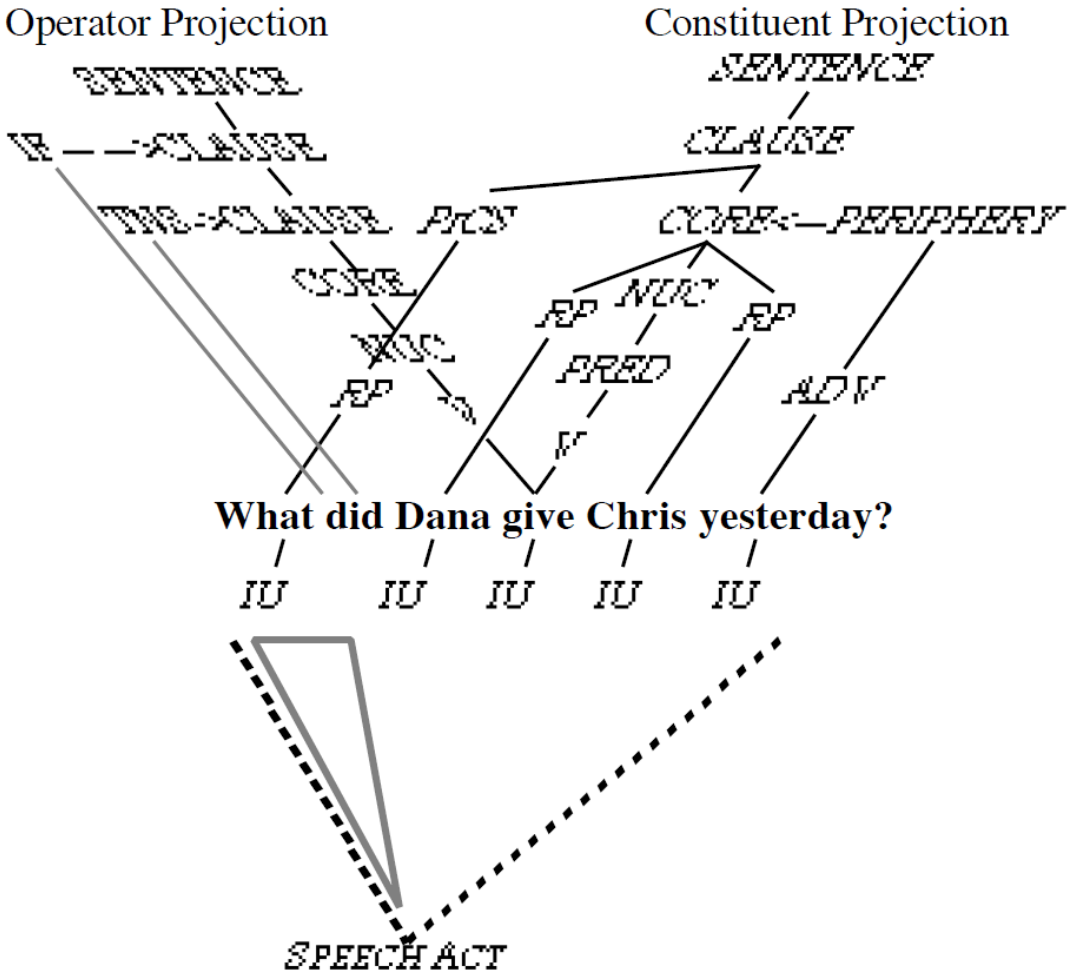
Van Valin and Lapolla (1997)



Why RRG ?

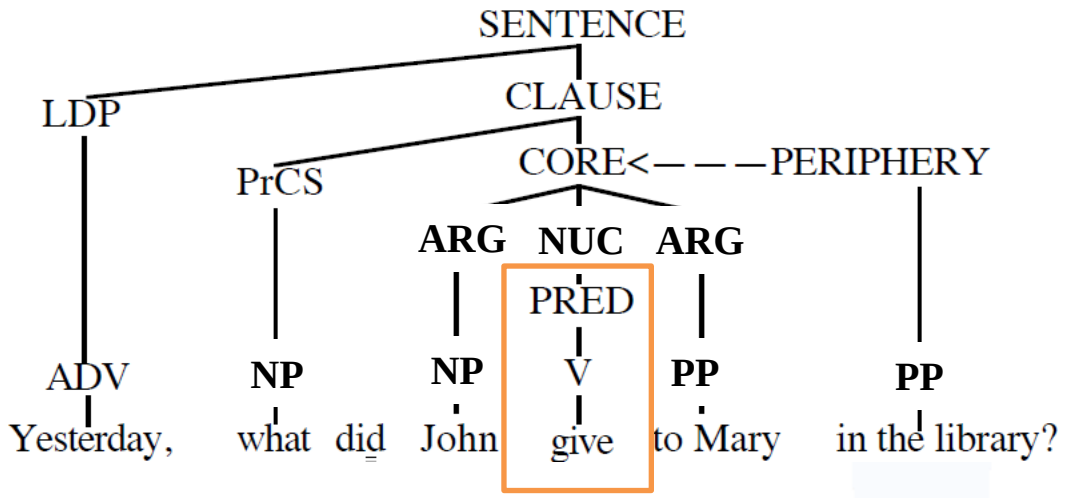
- wide use in typology and language documentation
  - no European bias
- semantics-based rather than syntax-focused
- not generative, but descriptive
  - flexibly deals with free word order
- modular
  - avoids overload of syntactic structures
- no data available (except for text book examples)
  - some rule-based parser prototypes

Syntax on multiple dimensions



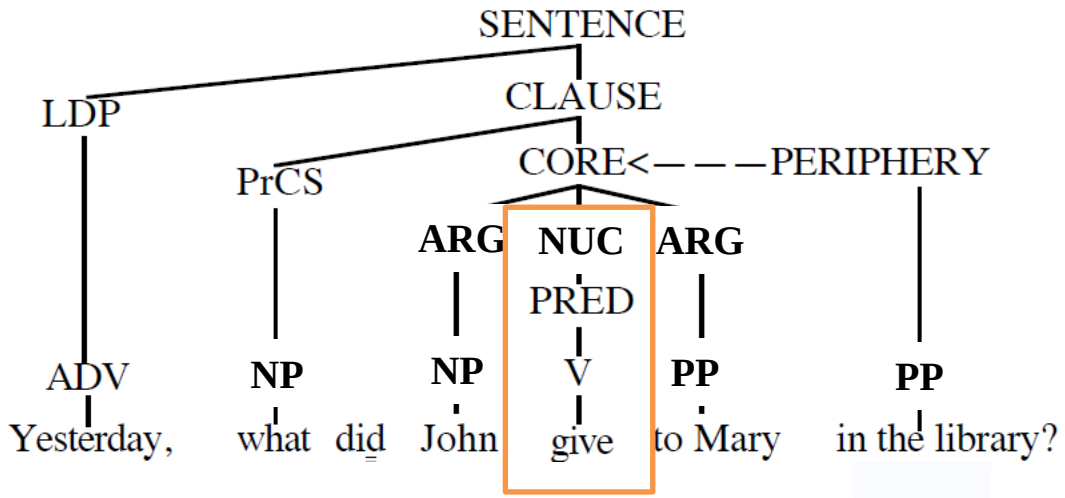
Focus Structure Projection

Constituent projection



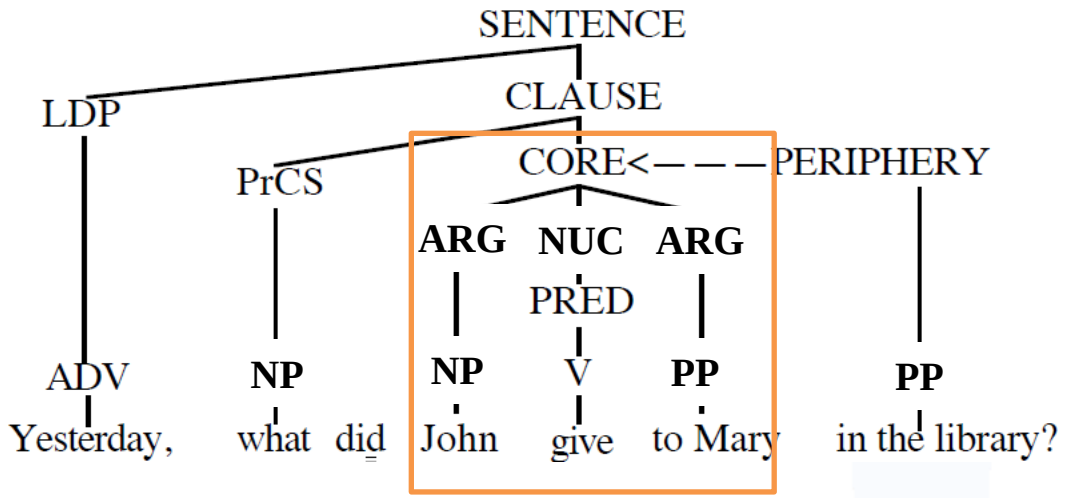
- PRED(icate)
  - is the semantic nucleus of the clause

Constituent projection



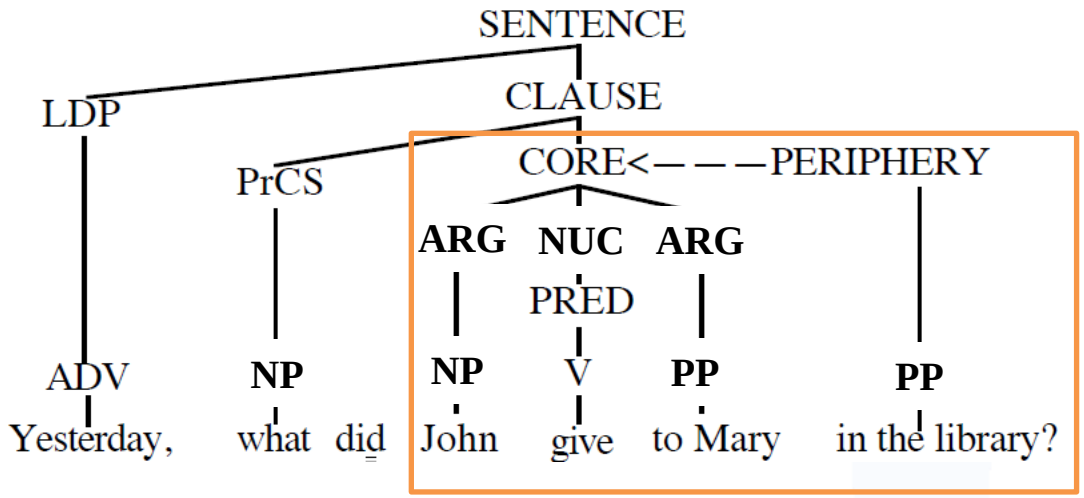
- NUC(leus)
  - contains the predicate(s)
    - roughly corresponding to a frame instance
    - can carry operators, e.g., aspect

Constituent projection



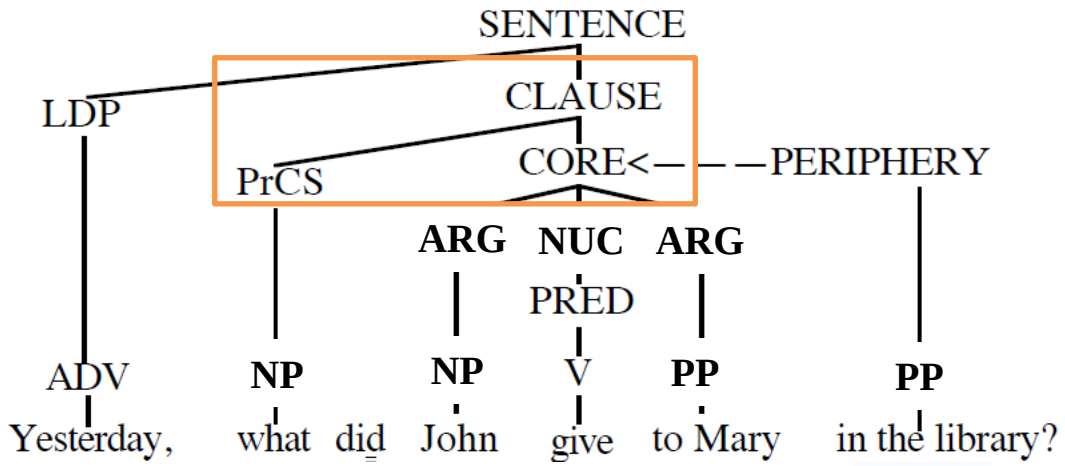
- CORE
  - contains the *semantic* arguments
    - cf. PropBank A0..A4
  - can carry operators, e.g., modality

Constituent projection



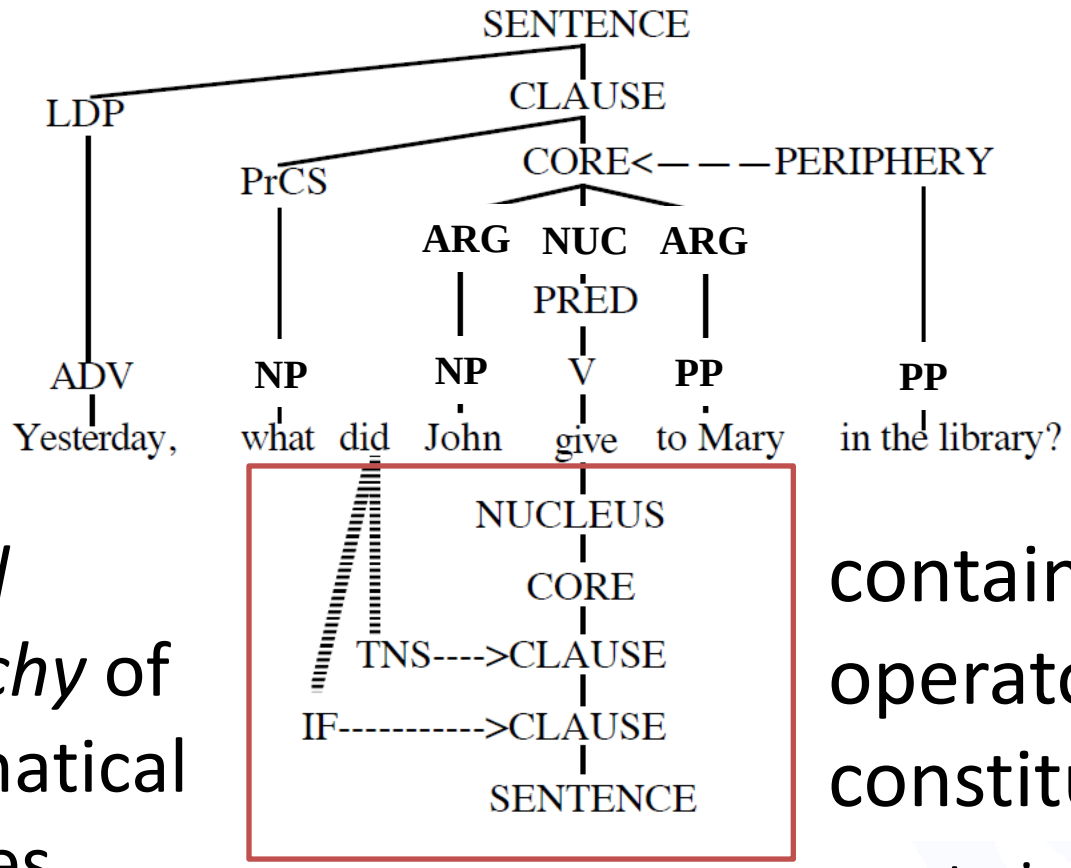
- CORE
  - contains the *semantic* arguments
  - PERIPHERY
    - contains *semantic* modifiers (cf. PropBank: ARG-M)

Constituent projection



- CLAUSE
  - contains CORE and optional peripheral positions
  - can carry operators, e.g., tense

Operator projection



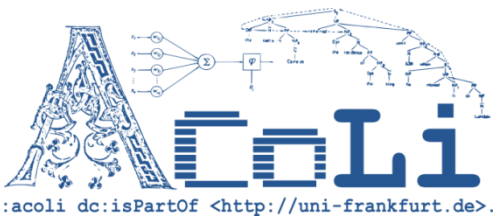
Nested hierarchy of grammatical features

containing the operators that constituents carry

constrain co(sub)ordination

Towards an RRG Treebank

According to Van Valin and Lapolla (1997)

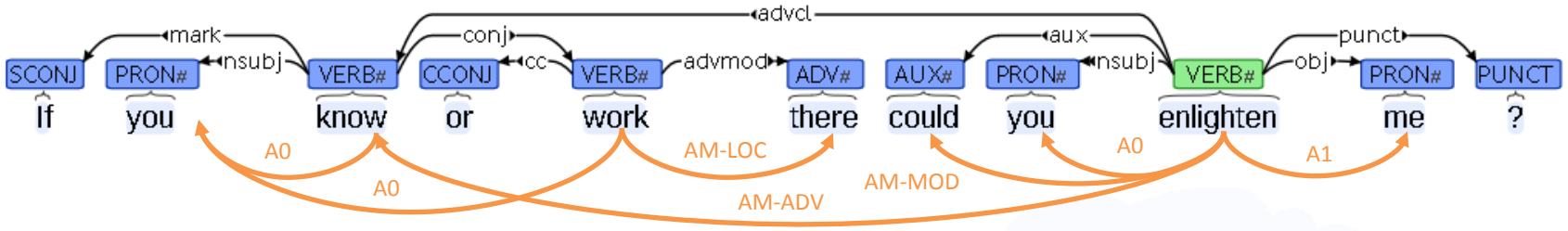


Treebanking by annotation engineering

- the ideal source corpus combines manual annotations for syntax with manual annotations for semantic roles
  - English Web Treebank (EWT)
    - ⇒ PTB + UD: CoNLL-Merge (Chiarcos & Schenk@LDK-2019)
- development data
  - 453 examples from textbooks (automatically parsed)
  - EWT corpus dev/answers

0. Preprocessing

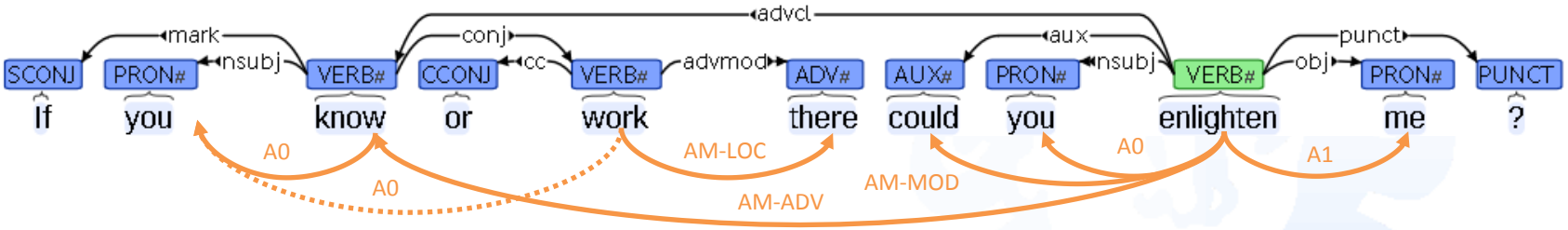
Merge PropBank and Universal Dependencies Annotations



If you know or work there could you enlighten me ?

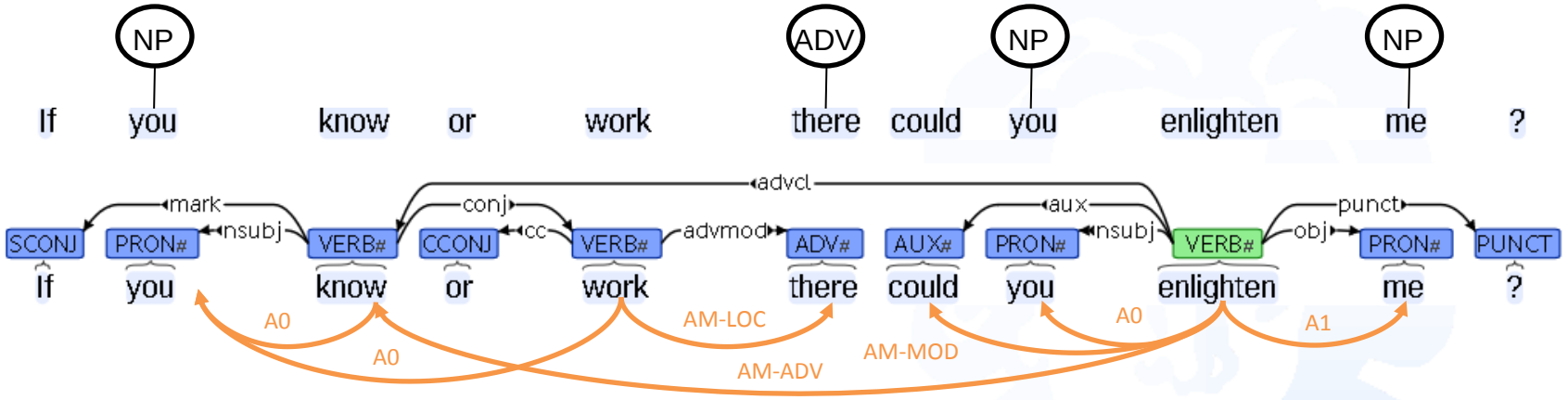
0. Preprocessing

reduce to local arguments, indirect arguments (dashed) and arguments of head (dashed)



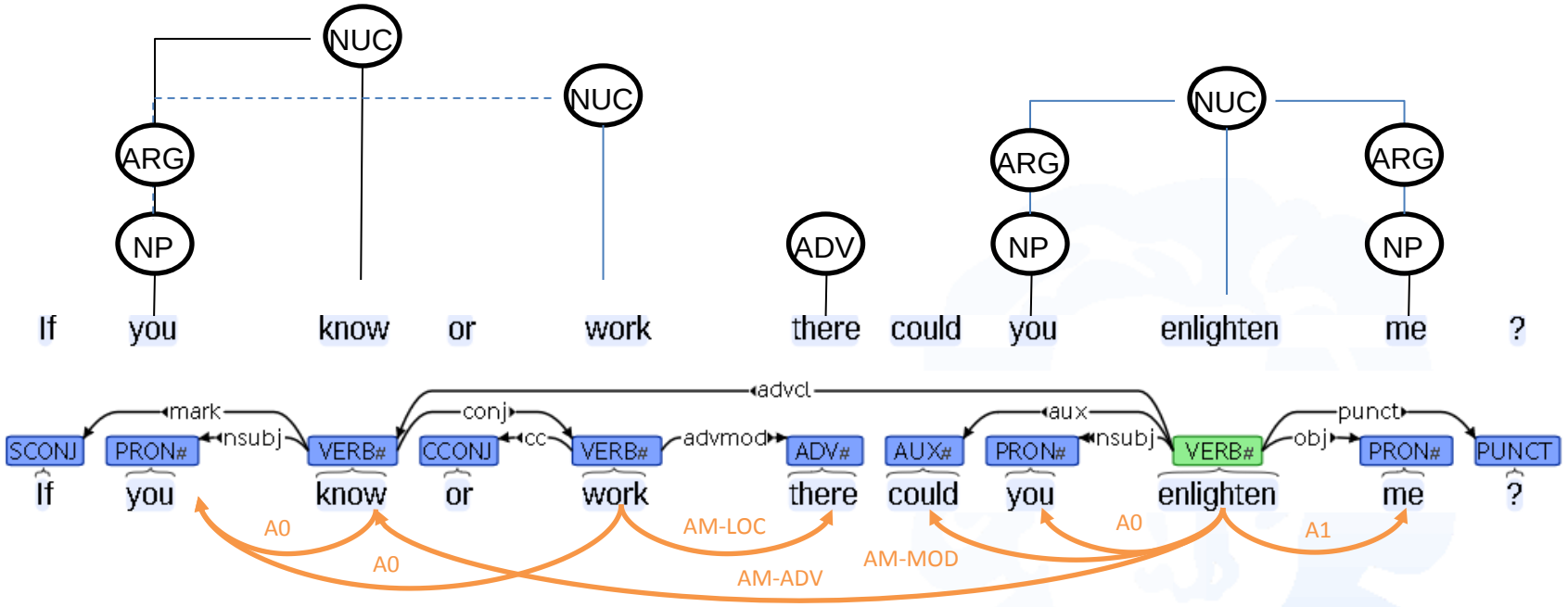
I. NP/PP/ADV chunks

based on UD (dependencies)

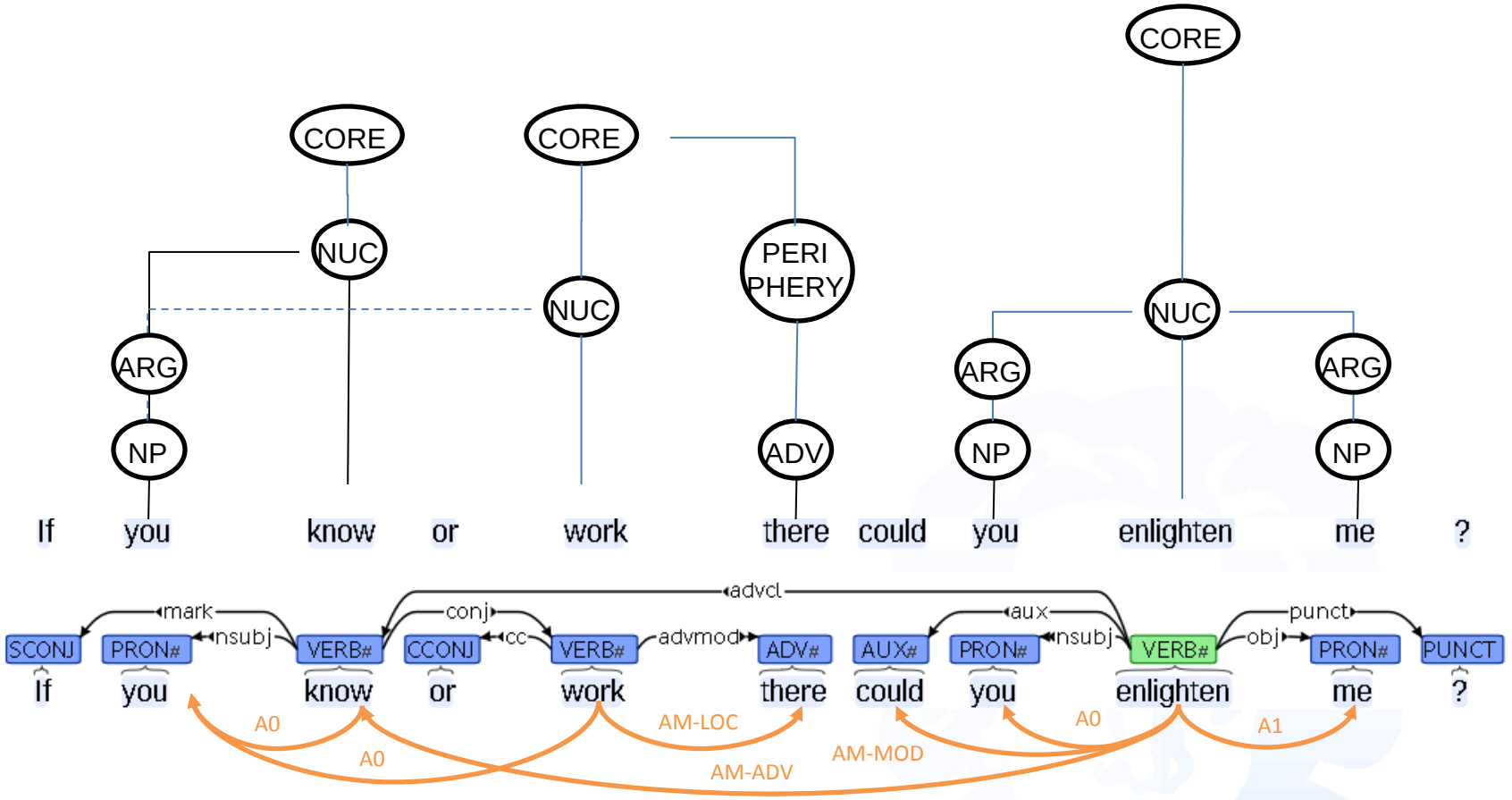


2. NUC / ARG: CORE arguments

based on PropBank Core roles

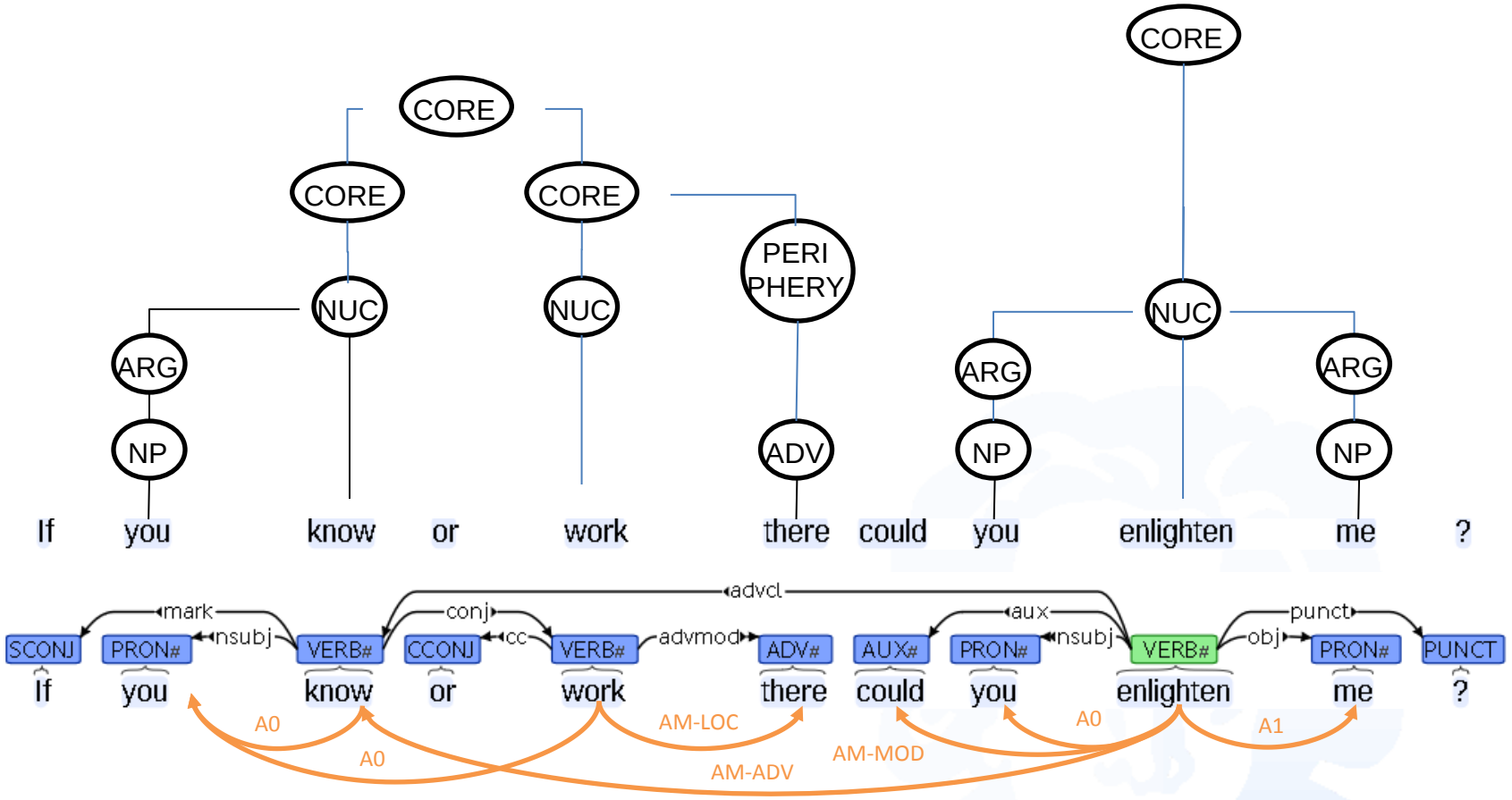


2. NUC / ARG / PERIPHERY



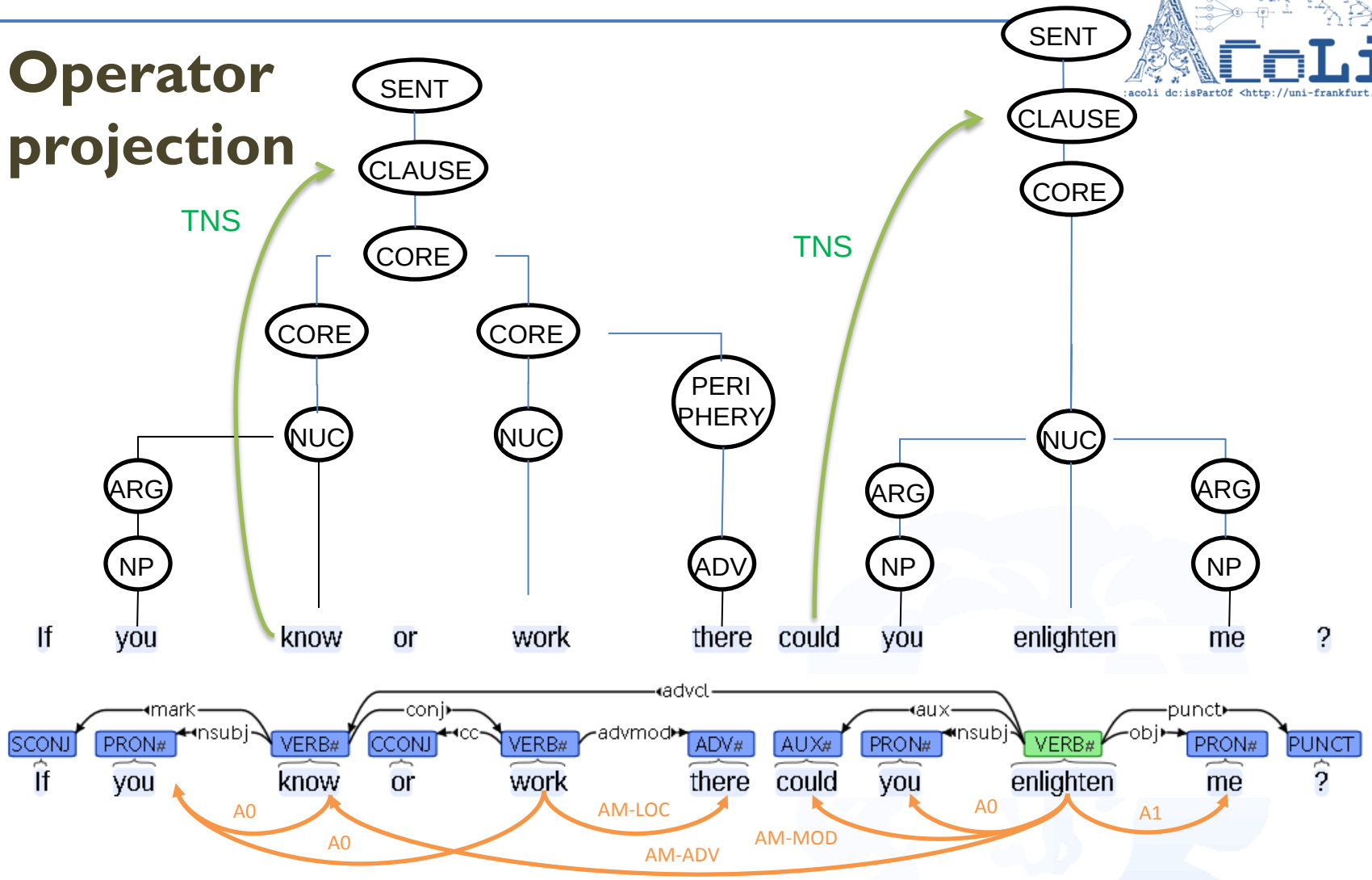
based on PropBank non-Core roles

2. CORE cosubordination



shared PropBank arguments

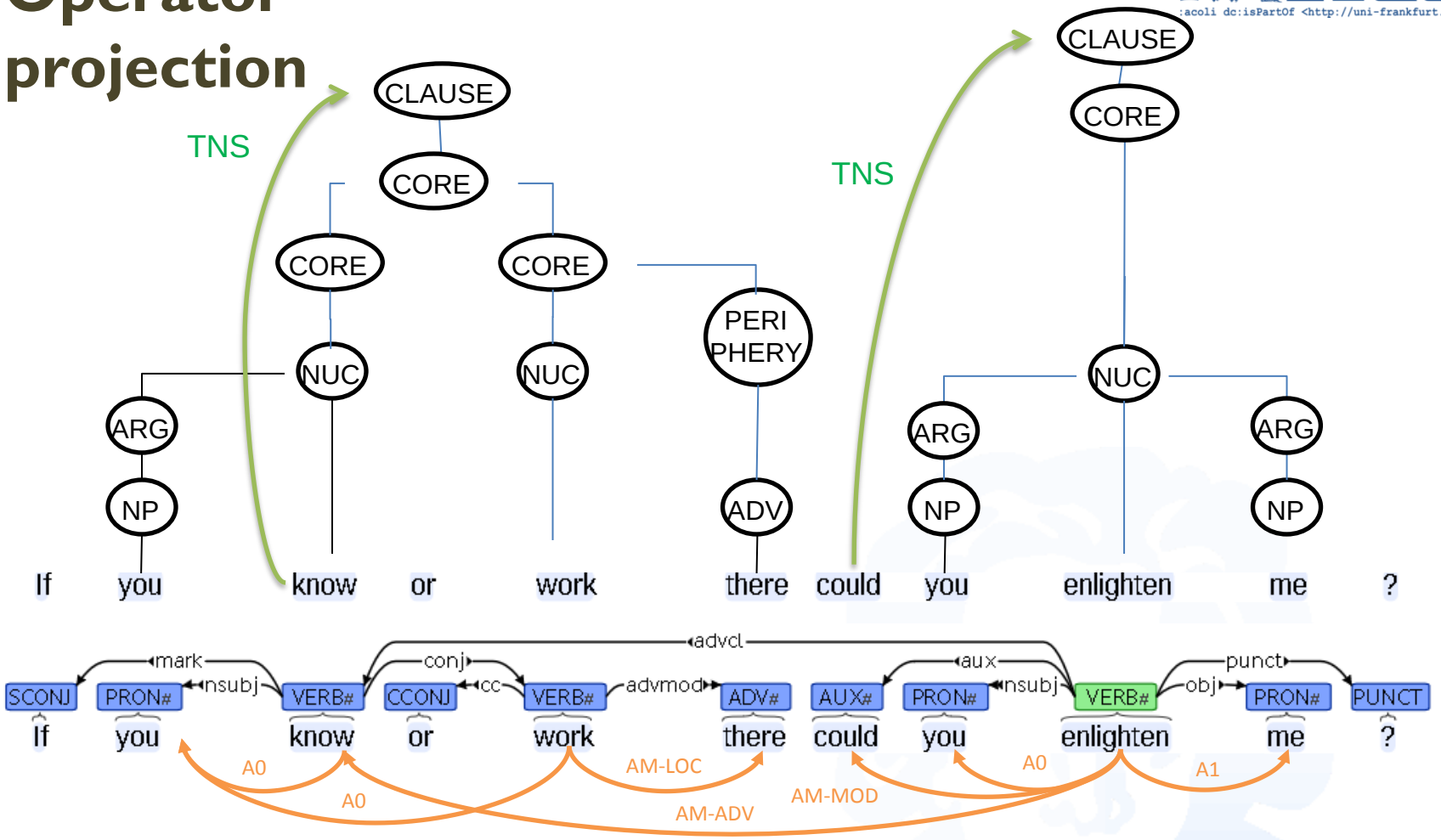
3. Operator projection



(a) create potential parent nodes of clausal nodes

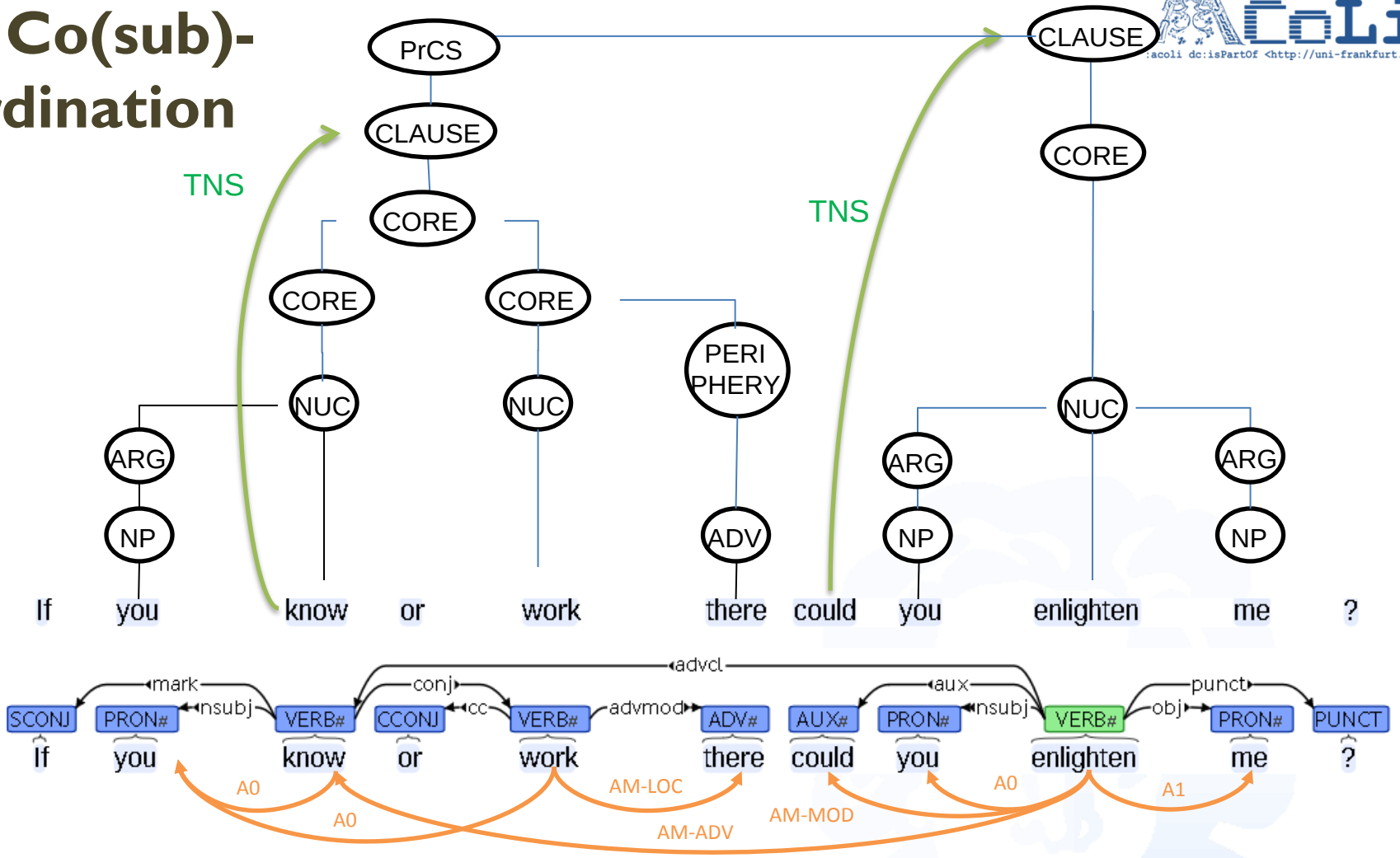
(b) add operators (from FEATS)

3. Operator projection

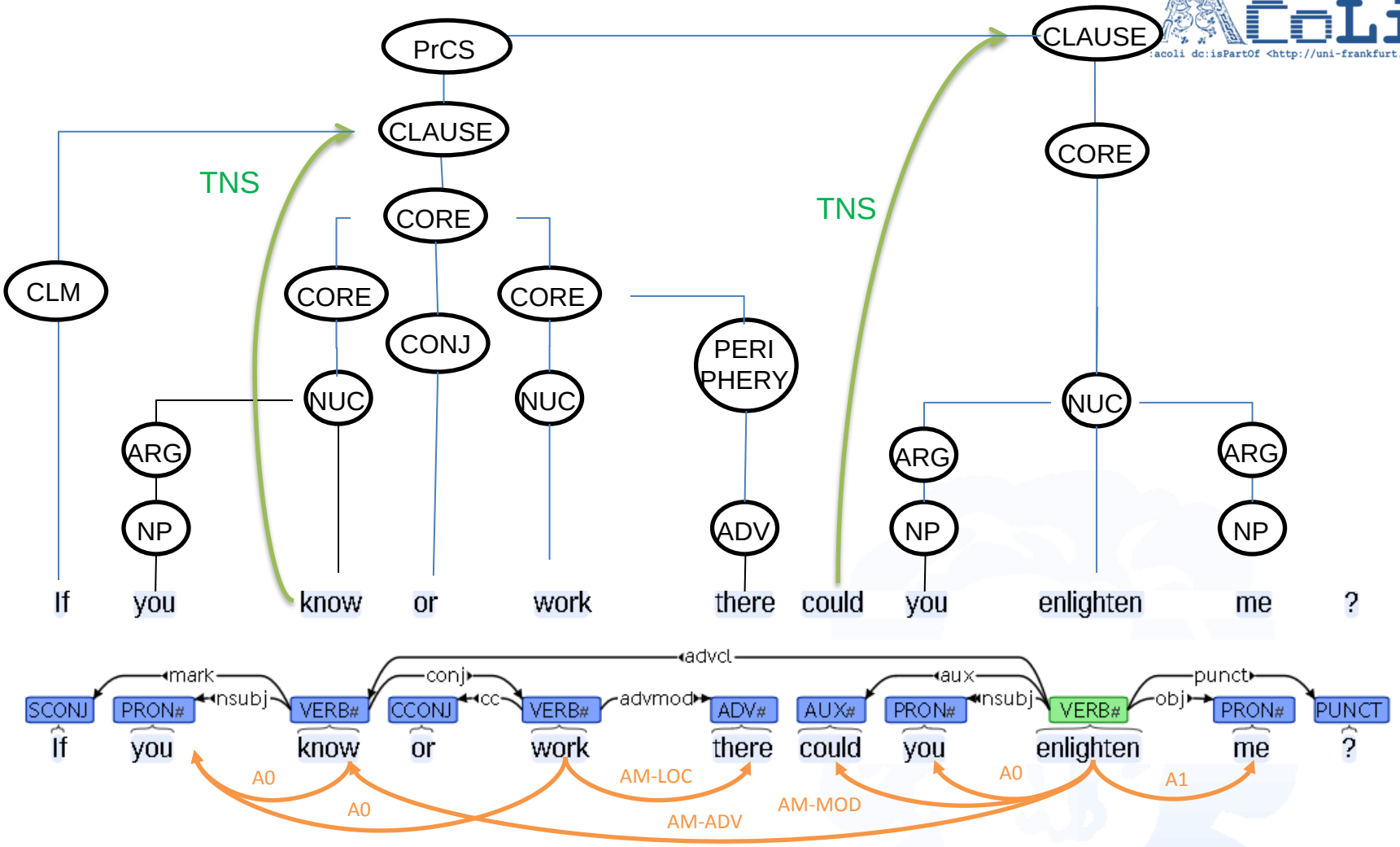


(c) prune tree (remove non-branching top-level nodes)

4. Co(sub)-ordination

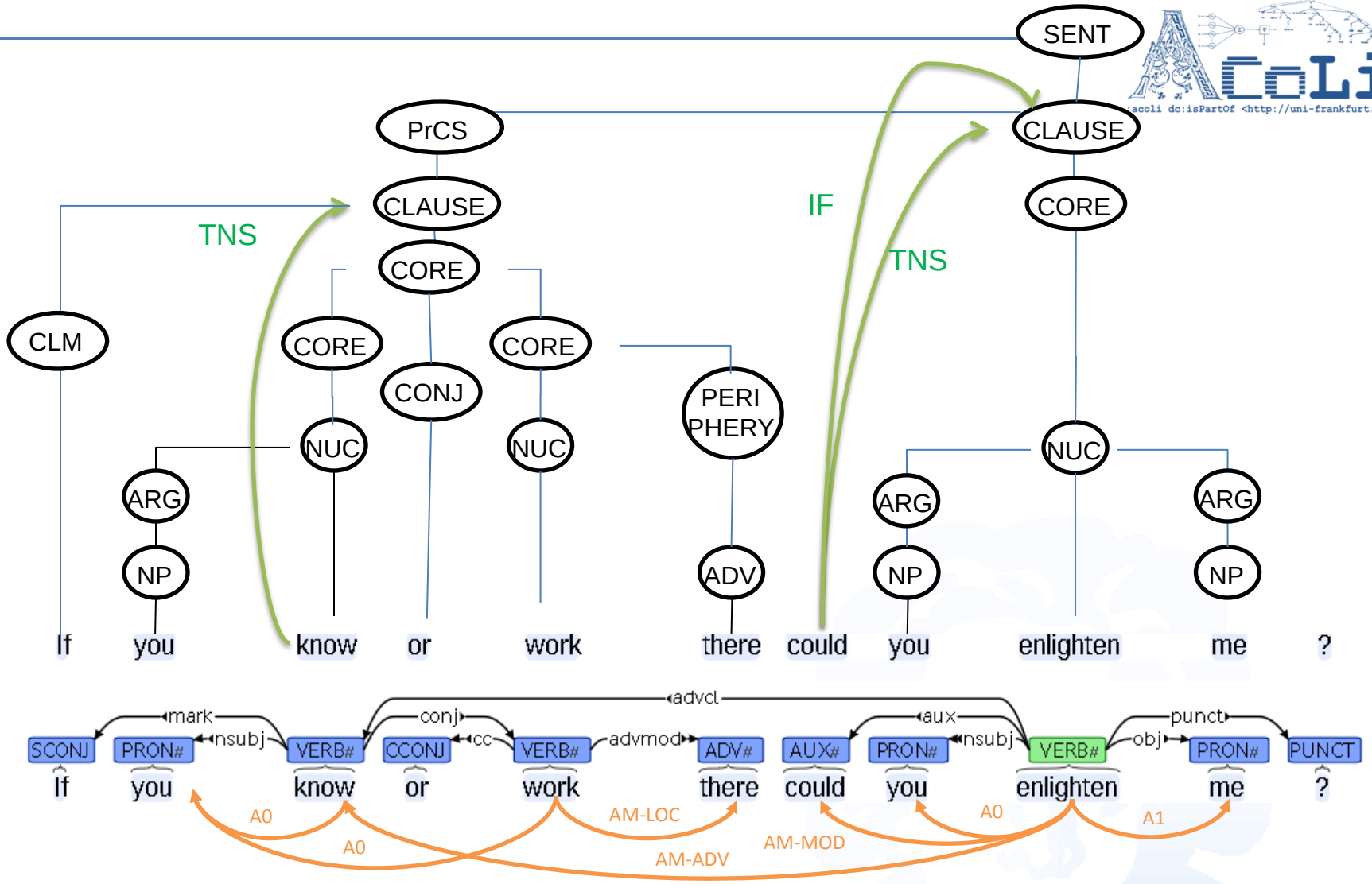


(a) attach clausal arguments (PropBank)



(a) attach clausal arguments (PropBank)

(b) clause linkage markers (UD)



5. complete sentential structure

IF (illocutionary force) operator

SENTENCE node

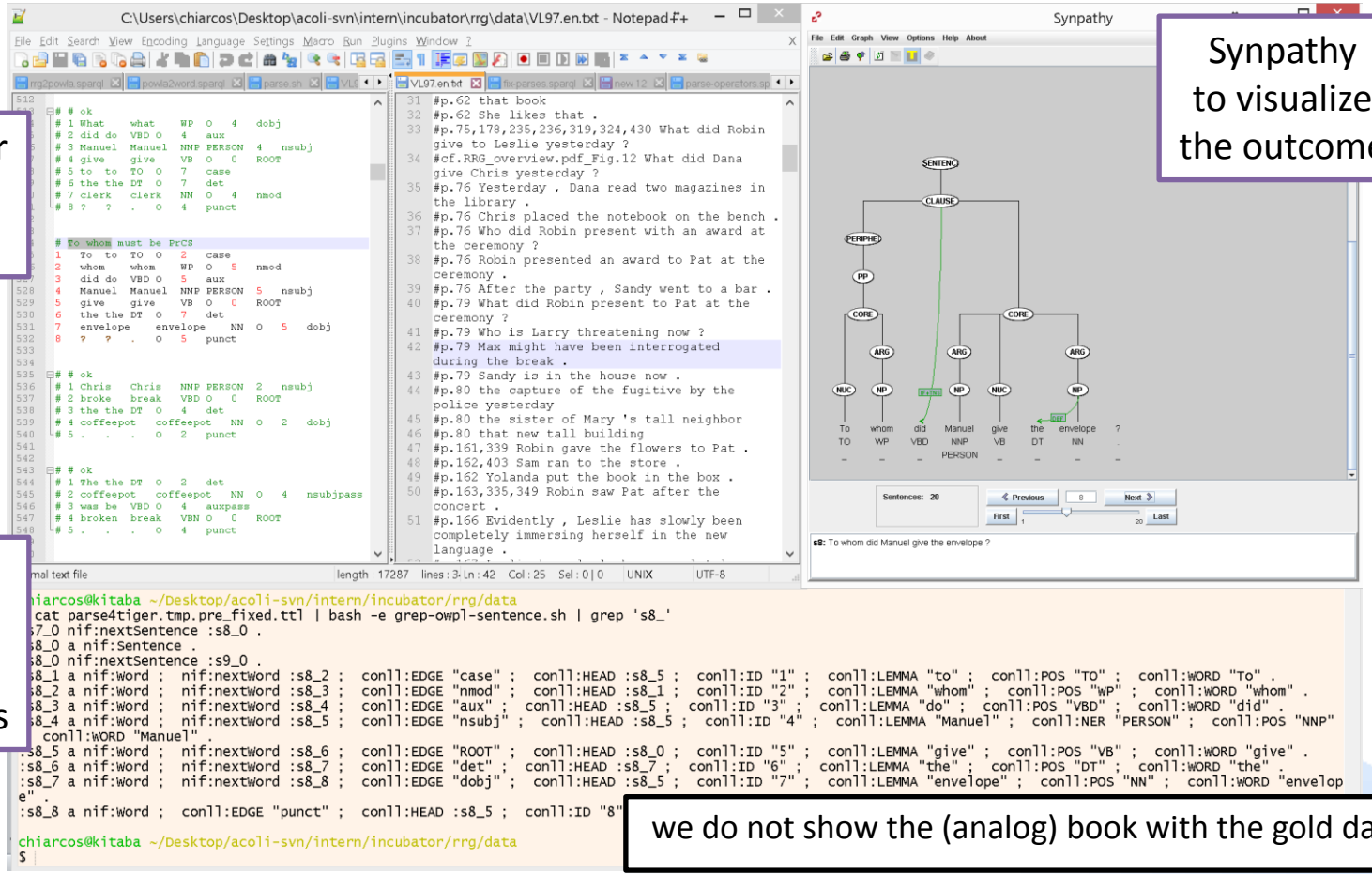
Simulating an IDE

- ensemble of tools to address different stages of the workflow
  - a second screen does help

text editor for SPARQL and source data

Synpathy to visualize the outcome

Un*x shell for conversion and checking intermediate results



we do not show the (analog) book with the gold data ;)

Evaluation: English Web Treebank

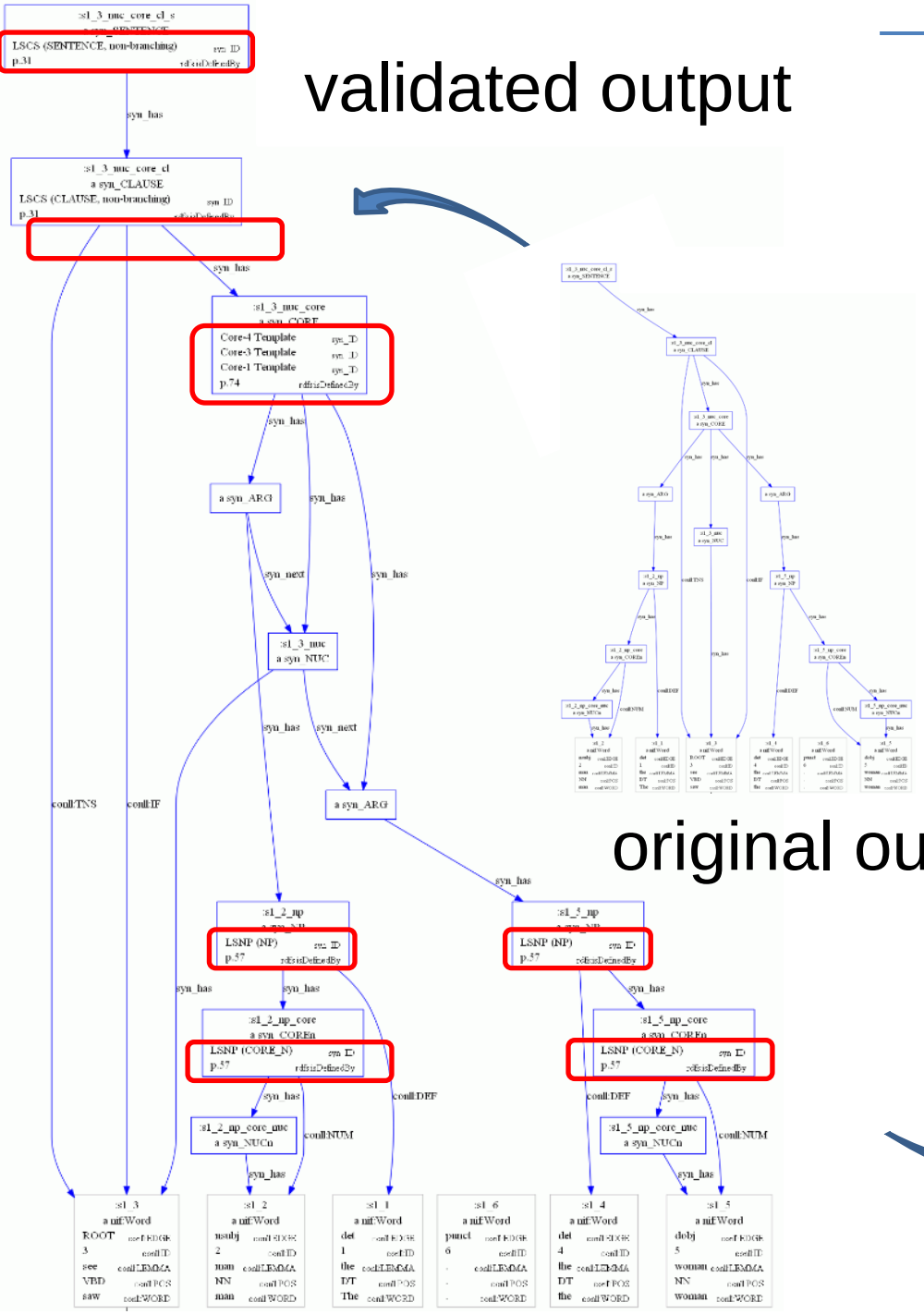
- SPARQL Update postprocessing step
- Structural evaluation („FRAG“)
  - fix inconsistent trees by inserting CYCLE/MPARENT nodes (=> FRAG)
- Pattern validation („INVALID“): matching against a generation rule

	dev	test	train
sentences	1974	2061	11364
INVALID	25,1%	26,1%	35,1%
<= FRAG	7,4%	8,8%	11,7%
<= MPARENT	4,6%	5,9%	6,7%
<= CYCLE	0,0%	0,0%	0,0%

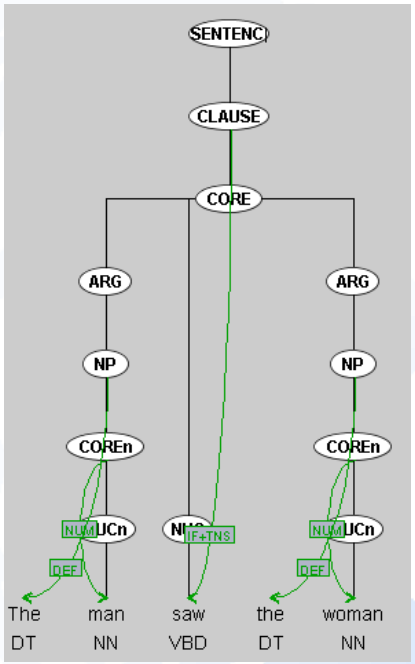
Template validation

- match pattern
- add reference to pattern definition
- do not export

validated output



original output



Future work

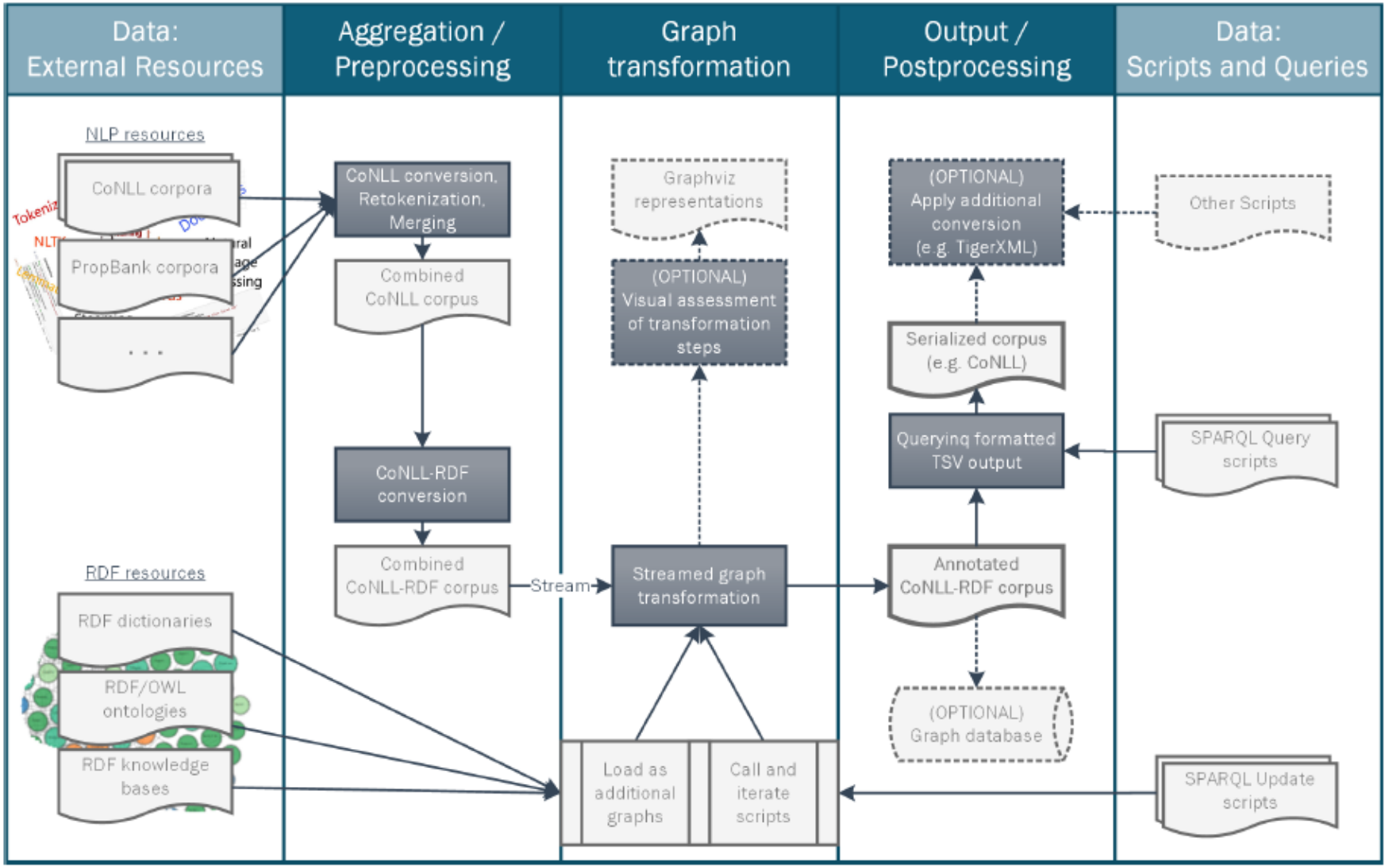
- integrate external knowledge sources
  - disambiguate complementizers and clausal prepositions
    - frequency dictionaries
      - no suitable RDF vocabulary => Ontolex-frac proposal

        https://acoli-repo.github.io/ontolex-frac/
  - disambiguate core and peripheral arguments
    - currently via PropBank
    - for UD-only data (e.g., textbook), could be disambiguated using PDEV

      http://pdev.org.uk/pdevlemonfiles/pdevlemon.ttl.tar.gz

Thank you!

scripts and data to be released soon: https://github.com/acoli-repo/RRG



Annotation Engineering with SPARQL Update

```
# nominal head of a PP must be an NP
INSERT {
    ?np a syn:NP;
        syn:head ?n .
} WHERE {
    ?n conll:POS ?pos; conll:EDGE ?e.
    FILTER(strstarts(?pos,"N"))
    ?n conll:HEAD/conll:EDGE "case" # PP
};
```

SPARQL

```
s8_1    . . / nsubj-------- Susan      ID 1 LEMMA Susan NER PERSON POS NNP
s8_2    . . / aux---------- does       ID 2 LEMMA do POS VBZ
s8_3    . . / neg---------- n't        ID 3 LEMMA not POS RB
s8_4    . \ ROOT            want       ID 4 LEMMA want POS VB
s8_5    . . . / mark------- to         ID 5 LEMMA to POS TO
s8_6    . . . / auxpass---- be         ID 6 LEMMA be POS VB
s8_7    . . \ xcomp-------- arrested   ID 7 LEMMA arrest POS VBN
s8_8    . . . \ case------- by         ID 8 LEMMA by POS IN
s8_9    . . . . . \ det---- the        ID 9 LEMMA the POS DT
s8_10   . . . . \ nmod----- police     ID 10 LEMMA police POS NN
s8_11   . . / punct-------- .          ID 11 LEMMA . POS .
```

conll:HEAD of ?n

?n

CoNLL-RDF dependency visualization