A Natural Proof System for Natural Language

NPS4NL-4: Wide-Coverage Theorem Prover for Natural Language

Lasha Abzianidze Reinhard Muskens

university of groningen

Tilburg University

ESSLLI 2019 in Rīga, Latvija

Today:

- Obtain LLFs from wide-coverage text;
- Automatize the proof procedure;
- Employ a knowledge base;
- Collect a relevant set of tableau rules;
- The theorem prover for natural logic and language

LLFs and Categorial Grammar

LLFs are similar to formal derivations studied in Categorial Grammars (CGs) [Ajdukiewicz, 1935a, Hillel, 1953].

- CGs treat each lexical item as a function;
- Categorial Type-Transparency principle links syntactic types to semantic ones.

Combinatory Categorial Grammar (CCG) [Steedman, 2000] is the only CG that is scaled up for wide-coverage text processing:

- CCG is well-studied from linguistic perspectives;
- There exists robust and accurate wide-coverage parsers for CCG, e.g., C&C parser [Clark and Curran, 2007] and EasyCCG [Lewis and Steedman, 2014].

Categorial Grammar

Categorial Grammar is a lexicalized formalism: less combining rules and more information at lexical units.

Each lexical unit gets a syntactic category:

- Primitive category: N, NP, PP, and S (usually);
- Function category: Y/X or $Y\backslash X$, where X and Y are categories.

For semantic representation, each lexical unit gets usually a lambda term as lexical semantics:

$$\texttt{Token}:\texttt{Category}:\lambda\texttt{term}$$

Combining rules:

$$Y/X:\texttt{f} \quad X:\texttt{a} \Rightarrow Y:\texttt{fa} \qquad \textit{Forward functional application} \qquad (>)$$

$$X:\texttt{a} \quad Y\backslash X:\texttt{f} \Rightarrow Y:\texttt{fa} \qquad \textit{Backward functional application} \qquad (<)$$

The simplest CG, called AB-grammar [Ajdukiewicz, 1935b, Hillel, 1953]

Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) employs additional combinatory rules (an incomplete list below):

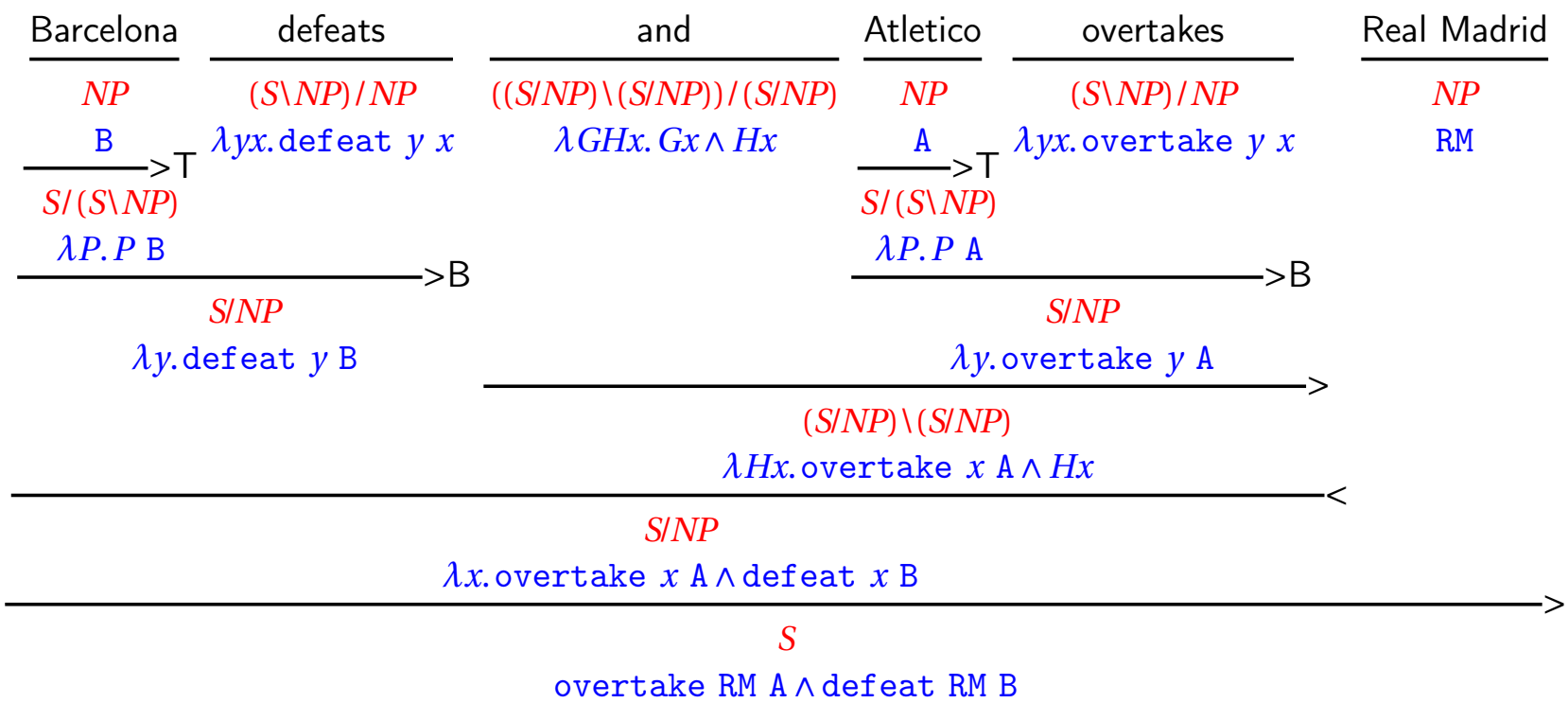
$$X : \mathtt{a} \;\Rightarrow\; T/(T\backslash X) : \lambda x.\, x\mathtt{a} \qquad \textit{Forward type-raising} \qquad (>\mathsf{T})$$

$$X : \mathtt{a} \;\Rightarrow\; T\backslash(T/X) : \lambda x.\, x\mathtt{a} \qquad \textit{Backward type-raising} \qquad (<\mathsf{T})$$

$$Z/Y : \mathtt{f} \quad Y/X : \mathtt{g} \;\Rightarrow\; Z/X : \lambda x.\, \mathtt{f}(\mathtt{g}x) \qquad \textit{Forward functional comp.} \qquad (>\mathsf{B})$$

$$Y/X : \mathtt{g} \quad Z\backslash Y : \mathtt{f} \;\Rightarrow\; Z/X : \lambda x.\, \mathtt{f}(\mathtt{g}x) \qquad \textit{Backw. crossing fun. comp.} \qquad (<\mathsf{B}_{\times})$$

CCG derivation

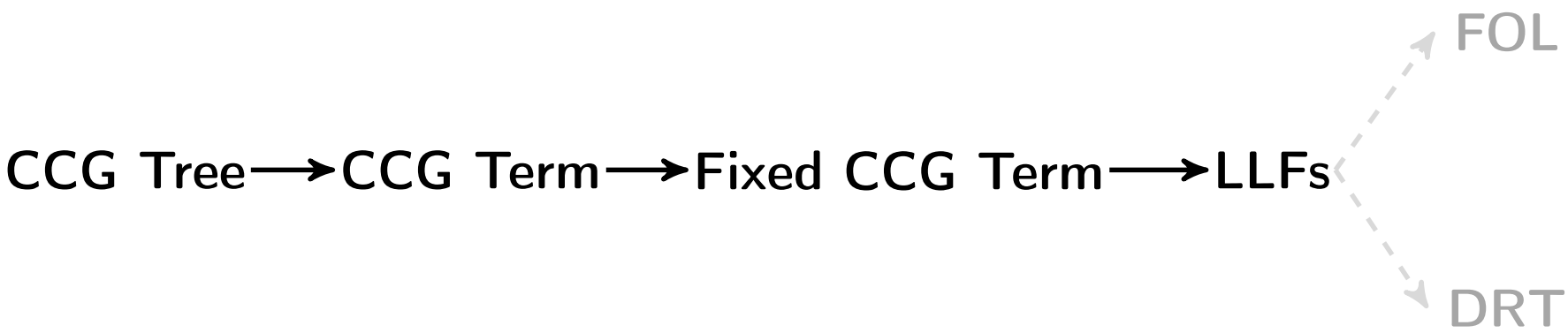


$$X : \mathtt{a} \Rightarrow T/(T\backslash X) : \lambda x.x\mathtt{a} \qquad \textit{Forward type-raising} \qquad (>\mathsf{T})$$

$$Z/Y : \mathtt{f} \quad Y/X : \mathtt{g} \Rightarrow Z/X : \lambda x.\mathtt{f}(\mathtt{g}x) \qquad \textit{Forward functional comp.} \qquad (>\mathsf{B})$$

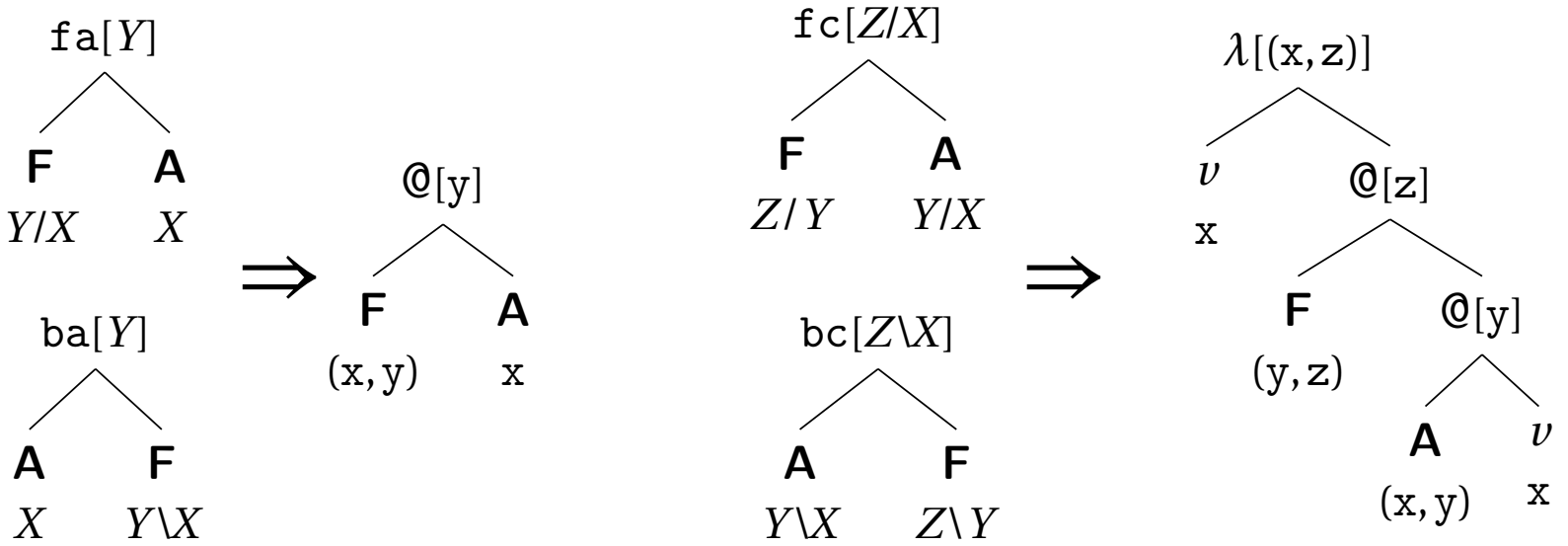
From CCG Trees to LLFs

Producing an LLF from a CCG derivation consists of several steps:

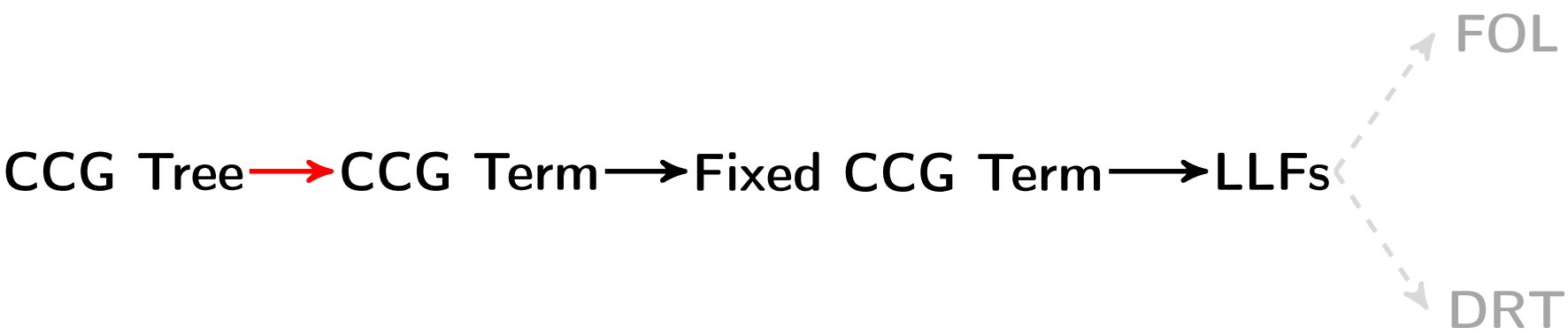


CCG Tree → CCG term

CCG Tree → CCG Term → Fixed CCG Term → LLFs → FOL, DRT



CCG Tree → CCG term



Removing directionality:

$$Y\backslash X \text{ and } Y/X \rightsquigarrow (\mathrm{x}, \mathrm{y})$$

$$\mathtt{ba}(A_X, F_{Y\backslash X}) \rightsquigarrow FA$$

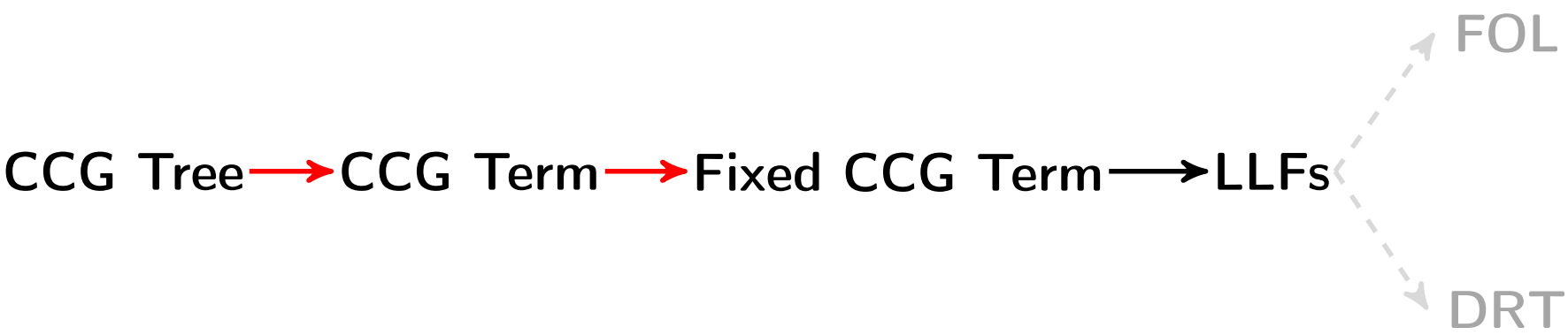
$$\mathtt{fxc}(F_{Z/Y}, A_{Y\backslash X}) \rightsquigarrow \lambda x.\, F(Ax)$$

$$\mathtt{tr}(T/(T\backslash X), A_X) \rightsquigarrow \lambda x.\, xA$$

$$\mathtt{lx}(Y, A_X) \rightsquigarrow [A_X]_Y$$

$$\mathtt{conj}(C_{conj}, A_X) \rightsquigarrow C_{X,X,X}A$$

CCG term → fixed CCG term



Correcting CCG terms:

$$[\mathtt{Dow}^{\mathtt{PER}}_{\mathtt{n,n}}\ \mathtt{Jones}^{\mathtt{PER}}_{\mathtt{n}}]_{\mathtt{np}} \rightsquigarrow \mathtt{Dow\_Jones}_{\mathtt{np}}$$

$$\mathtt{nobody}_{\mathtt{np}} \rightsquigarrow \mathtt{no}_{\mathtt{n,np}}\ \mathtt{person}_{\mathtt{n}}$$

$$[\mathtt{ice}_{\mathtt{n}}]_{\mathtt{np}} \rightsquigarrow \mathtt{a}_{\mathtt{n,np}}\ \mathtt{ice}_{\mathtt{n}}$$

$$[\mathtt{two}_{\mathtt{n,n}}\ \mathtt{dogs}_{\mathtt{n}}]_{\mathtt{np}} \rightsquigarrow \mathtt{two}_{\mathtt{n,np}}\ \mathtt{dogs}_{\mathtt{n}}$$

$$[\mathtt{working}_{\mathtt{np,s}}]_{\mathtt{n,n}} \rightsquigarrow \mathtt{who}_{\mathtt{(np,s),n,n}}\ \mathtt{working}$$

$$\mathtt{who}\ V(Q_{\mathtt{n,np}}N) \rightsquigarrow Q_{\mathtt{n,np}}(\mathtt{who}'\ VN)$$

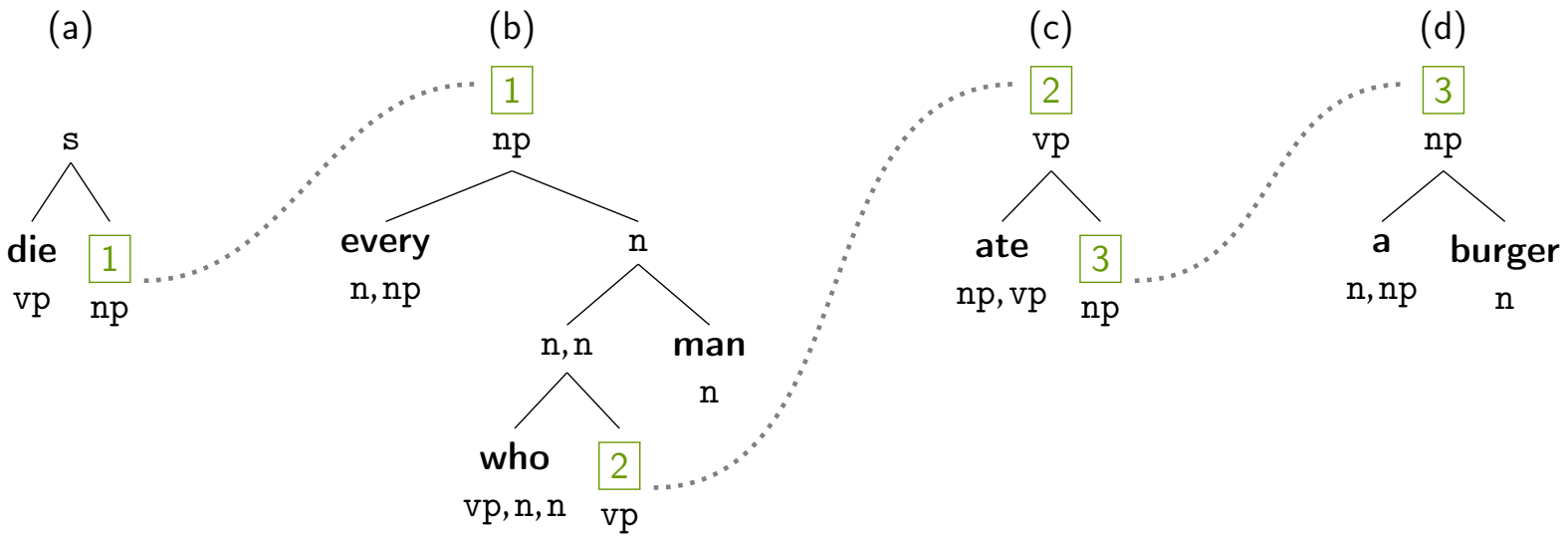
fixed CCG term → LLFs

Every man who ate a burger died

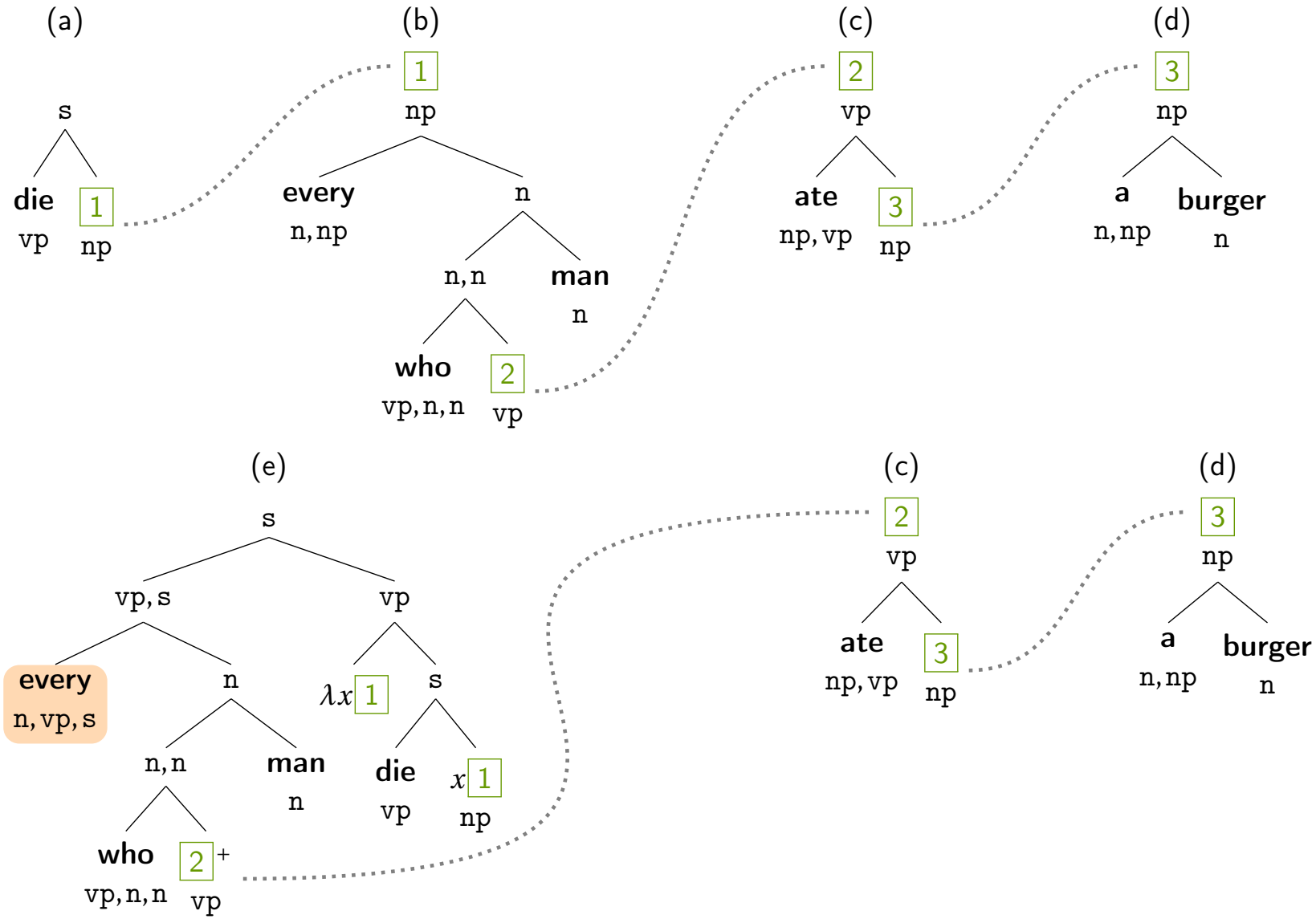
$$\mathbf{die}_{\mathrm{vp}}\ (\mathbf{every}_{\mathrm{n,np}}(\mathbf{who}_{\mathrm{vp,n,n}}\ (\mathbf{eat}_{\mathrm{np,vp}}\ (\mathbf{a}_{\mathrm{n,np}}\ \mathbf{burger}_{\mathrm{n}}))\ \mathbf{man}_{\mathrm{n}}))\ \rightsquigarrow$$

$$\mathbf{EVERY}_{\mathrm{n,vp,s}}(\mathbf{who}\ (\lambda x.\mathbf{A}_{\mathrm{n,vp,s}}\ \mathbf{burger}\ (\lambda y.\mathbf{eat}\ y_{\mathrm{np}}\ x_{\mathrm{np}}))\mathbf{man})\mathbf{die}$$

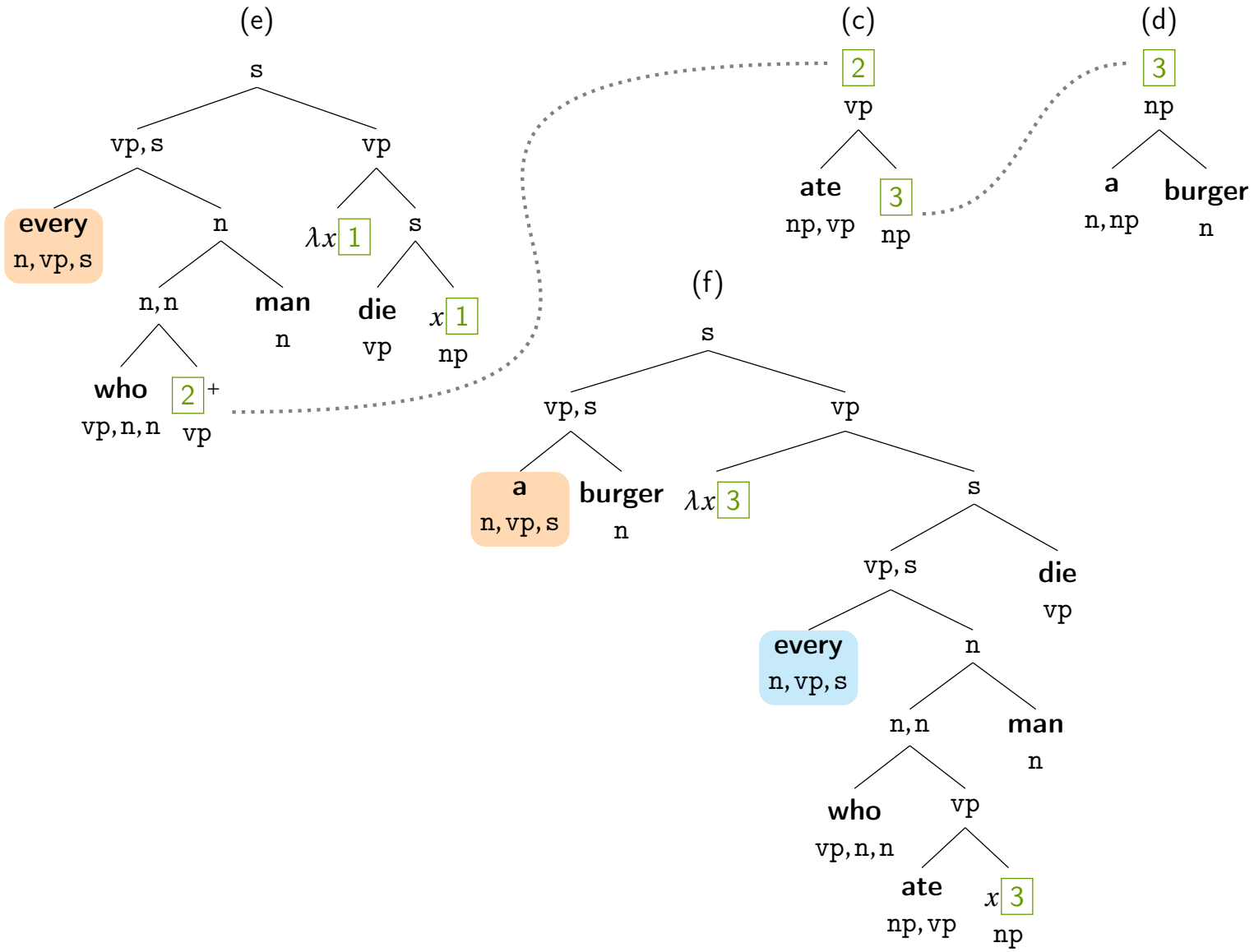
$$\mathbf{A}_{\mathrm{n,vp,s}}\ \mathbf{burger}(\lambda y.\mathbf{EVERY}_{\mathrm{n,vp,s}}(\mathbf{who}\ (\lambda x.\mathbf{eat}\ y_{\mathrm{np}}\ x_{\mathrm{np}})\mathbf{man})\mathbf{die})$$



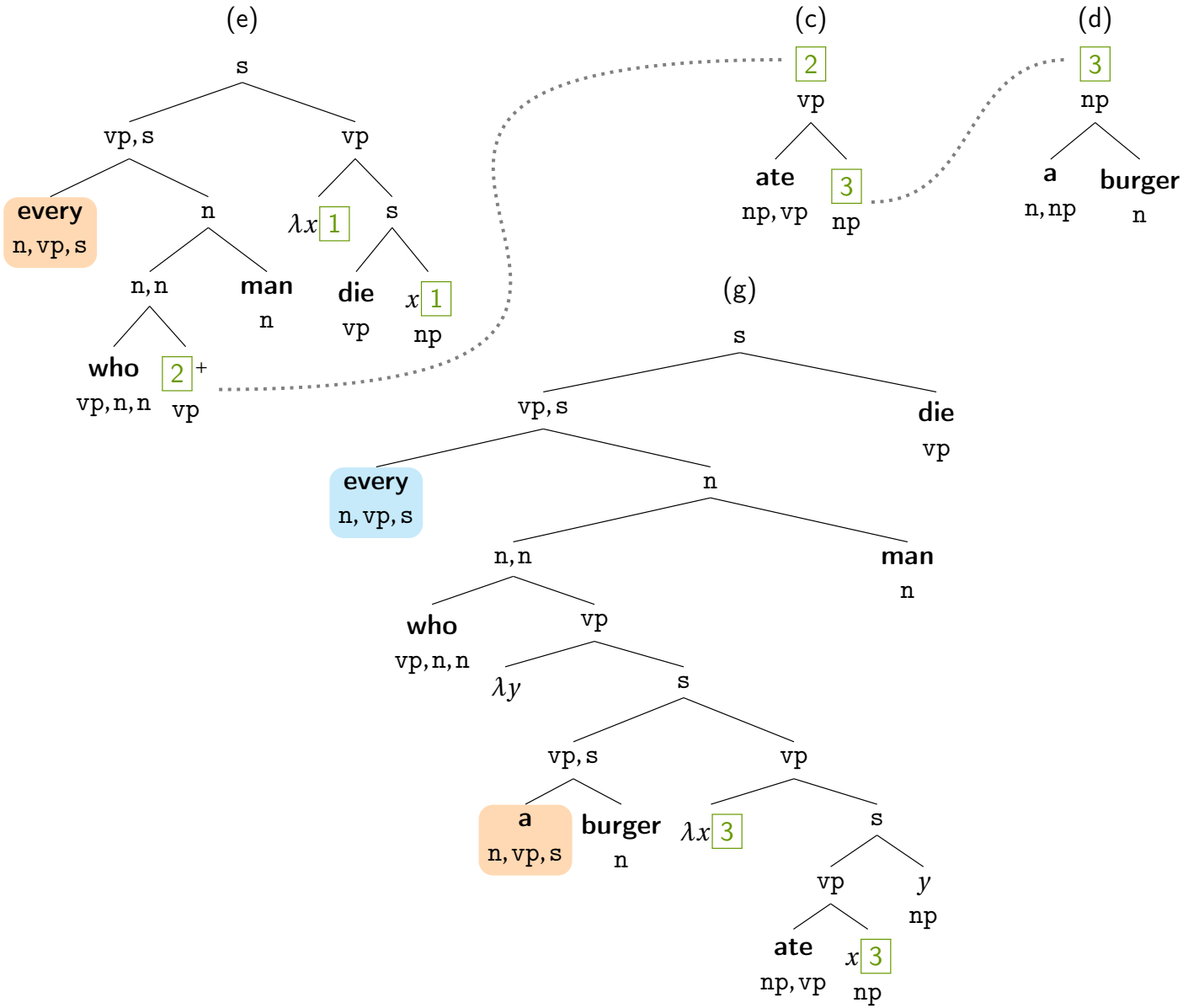
Gluing 1


(a)
s
die
vp
1
np
(b)
1
np
every
n, np
n
n, n
man
n
who
vp, n, n
2
vp
(c)
2
vp
ate
np, vp
3
np
(d)
3
np
a
n, np
burger
n
(e)
s
vp, s
vp
every
n, vp, s
n
λx 1
s
n, n
man
n
die
vp
x 1
np
who
vp, n, n
2+
vp
(c)
2
vp
ate
np, vp
3
np
(d)
3
np
a
n, np
burger
n

Gluing 2.1: a > every



Gluing 2.2: **every > a**



Sentence coordination

For sentence coordination, the quantified NP raising algorithm avoids scope interaction across the sentences.

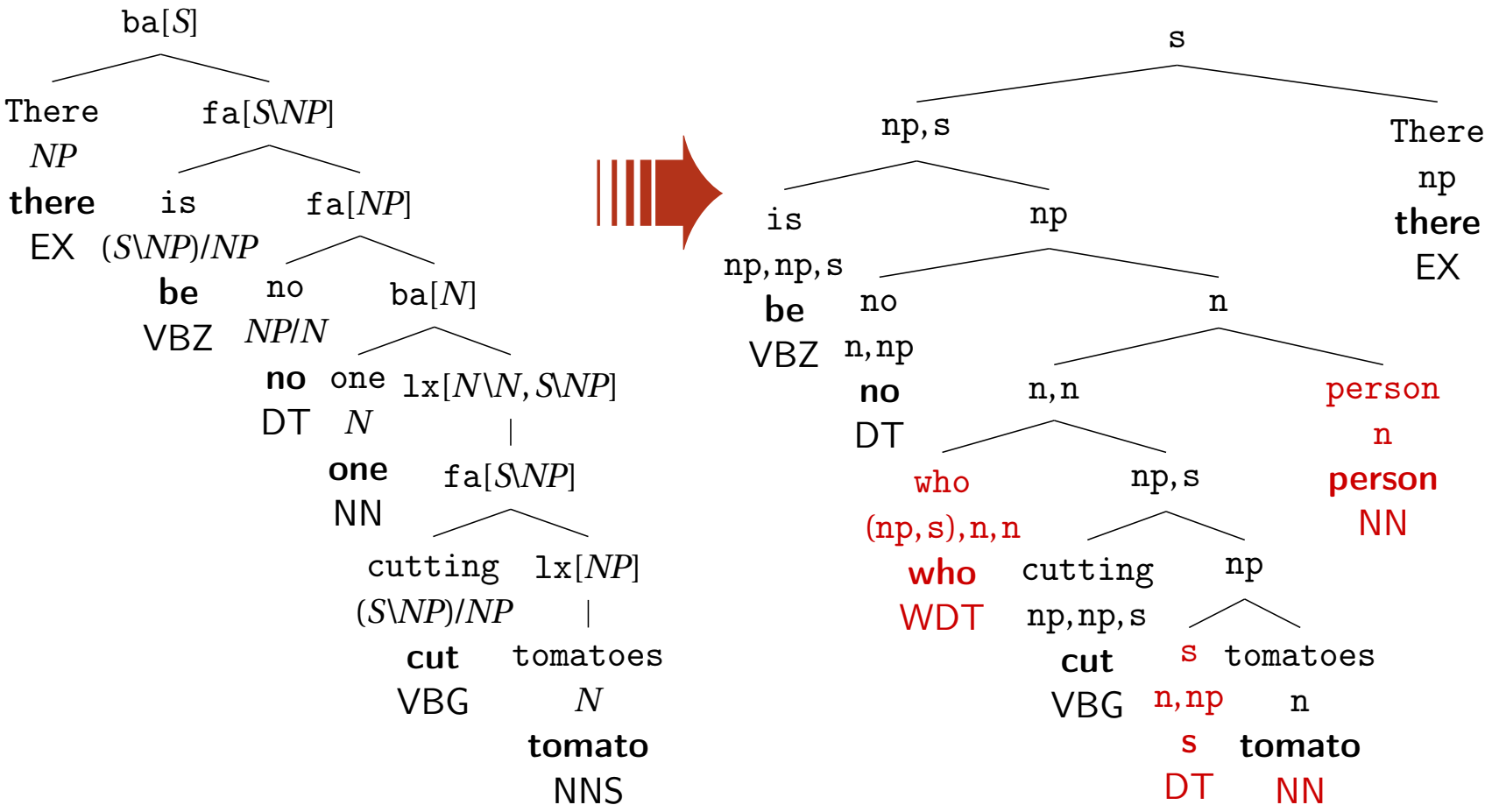
Every man sleeps and some woman worries

$$\mathbf{and}_{s,s,s}\left(\mathbf{sleep}_{vp}\left(\mathbf{every}_{n,np}\ \mathbf{man}_{n}\right)\right)\left(\mathbf{worry}_{vp}\left(\mathbf{some}_{n,np}\ \mathbf{woman}_{n}\right)\right)$$

$$\mathbf{and}\ \left(\mathbf{EVERY}_{n,vp,s}\ \mathbf{man}\ \mathbf{sleep}\right)\left(\mathbf{SOME}_{n,vp,s}\ \mathbf{woman}\ \mathbf{worry}\right)$$

$$\mathbf{SOME}_{n,vp,s}\,\mathbf{woman}\left(\lambda y.\,\mathbf{EVERY}_{n,vp,s}\,\mathbf{man}\left(\lambda x.\,\mathbf{and}\left(\mathbf{sleep}\,x\right)\left(\mathbf{worry}\ y\right)\right)\right)$$
$$\mathbf{EVERY}_{n,vp,s}\,\mathbf{man}\left(\lambda x.\,\mathbf{SOME}_{n,vp,s}\,\mathbf{woman}\left(\lambda y.\,\mathbf{and}\left(\mathbf{sleep}\,x\right)\left(\mathbf{worry}\ y\right)\right)\right)$$

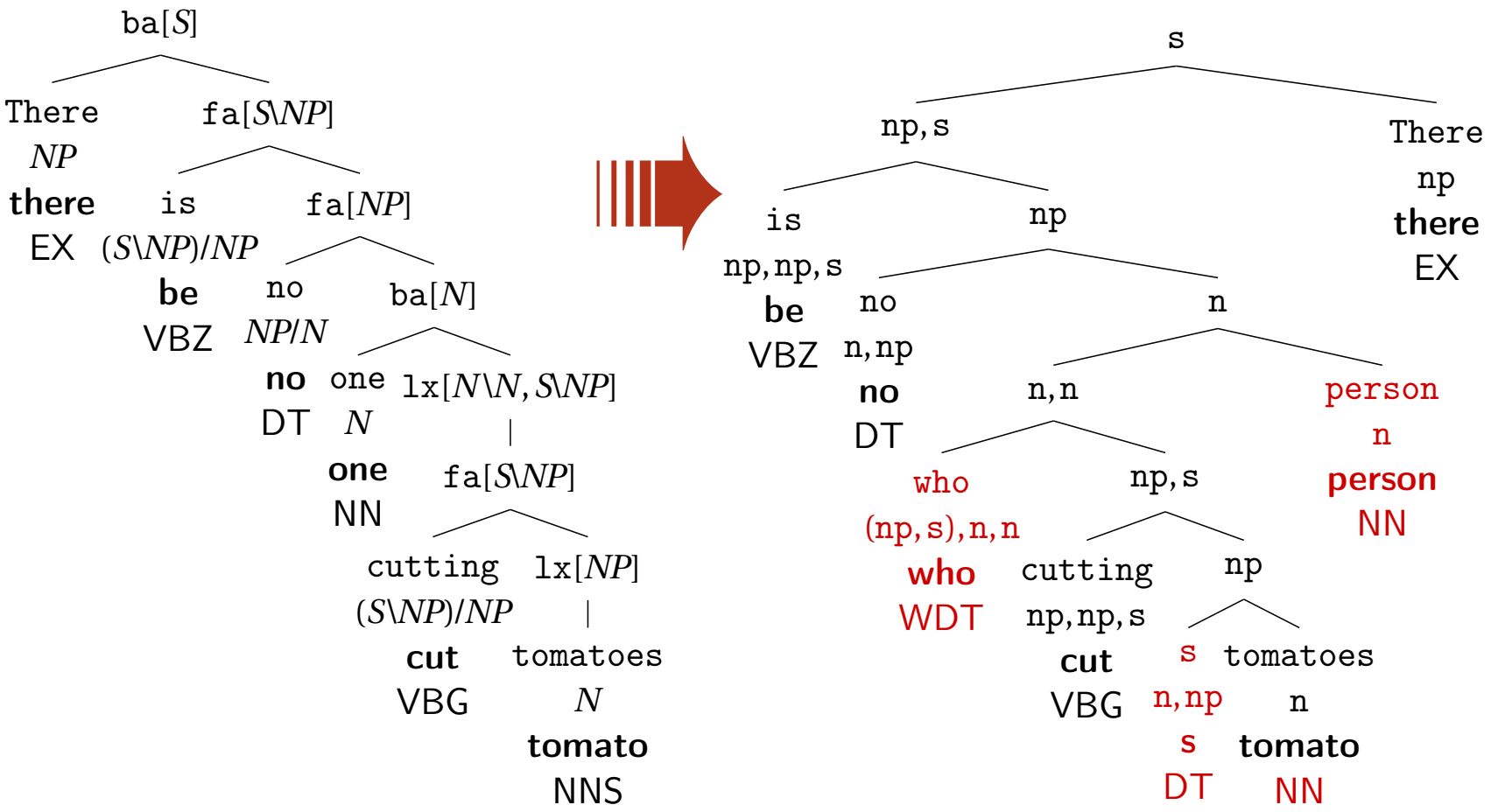
LLFgen: CCG tree → fixed CCG term → LLFs



There is no one cutting tomatoes ⇝

be(no(who(cut(s tomato))person))there

LLFgen: CCG tree → fixed CCG term → LLFs



$$\textbf{be}(\textbf{no}(\textbf{who}(\textbf{cut}(\textbf{s tomato}))\textbf{person}))\textbf{there} \rightsquigarrow$$
$$\textbf{no}\,(\textbf{who}\ (\lambda x.\ \textbf{s tomato}\ (\lambda y.\ \textbf{cut}\,y\,x))\ \textbf{person})\,(\lambda z.\textbf{be}\ z\ \textbf{there})$$
$$\textbf{s tomato}\ (\lambda y.\textbf{no}\,(\textbf{who}\ (\textbf{cut}\ y)\ \textbf{person})\,(\lambda z.\textbf{be}\ z\ \textbf{there}))$$

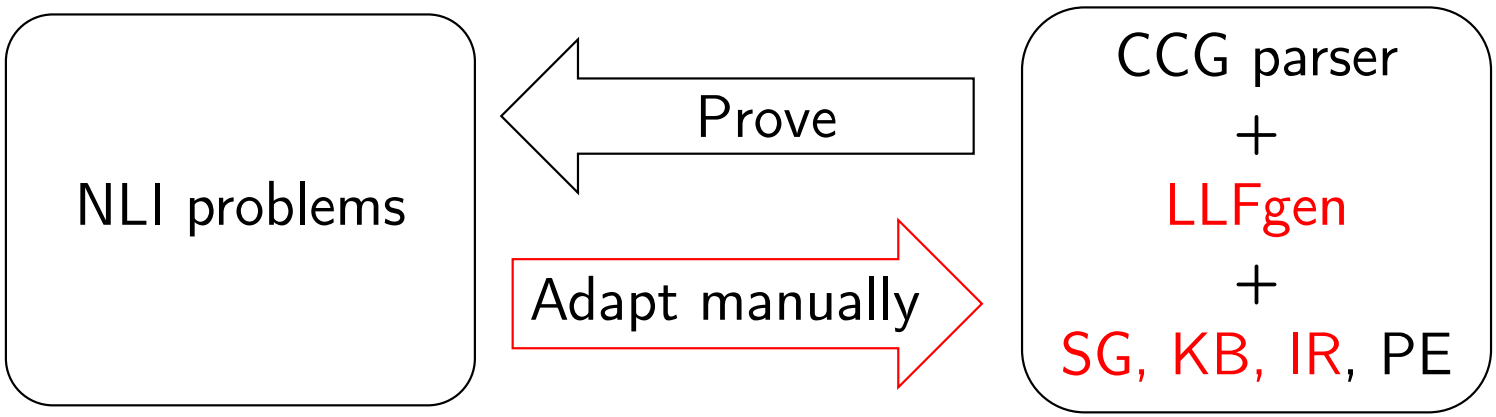
Linguistic Rules

Linguistic rules, in contrast to the algebraic rules, account for a certain syntactic constructions.

After having automatically generated LLFs, we can design specific tableau rules for common syntactic constructions.

We will also include the rules that *remedy* errors introduced in the syntactic derivation trees.

Most of the linguistic rules are collected in a data-driven fashion:



Rules for prepositions

$$\frac{p^{\mathtt{IN}}_{\mathrm{np,n,n}}\, d\, N : [c] : \mathbb{T}}{N : [c] : \mathbb{T} \quad p_{\mathrm{np,pp}} : [d, c] : \mathbb{T}}\ \mathrm{PP@N}_{\mathbb{T}}$$

$$\mathsf{with}_{\mathrm{np,n,n}}\ g_e\ \mathsf{bicycle}_{\mathrm{n}} : [c_e] : \mathbb{T}$$

$$\mathsf{bicycle} : [c] : \mathbb{T}$$

$$\mathsf{with}_{\mathrm{np,pp}} : [g, c] : \mathbb{T}$$

$$\frac{p^{\mathtt{IN}}_{\mathrm{np,n,n}}\, d\, N : [c] : \mathbb{F}}{N : [c] : \mathbb{F} \quad p_{\mathrm{np,pp}} : [g, c] : \mathbb{F}}\ \mathrm{PP@N}_{\mathbb{F}}$$

$$\mathsf{with}_{\mathrm{np,n,n}}\ g_e\ \mathsf{bicycle}_{\mathrm{n}} : [c_e] : \mathbb{F}$$

$$\mathsf{bicycle} : [c] : \mathbb{F} \qquad \mathsf{with}_{\mathrm{np,pp}} : [g, c] : \mathbb{F}$$

Problem of PP attachment

PP attachment is characterised with an attachment site and a nature of the attachment:

$$\text{John } [[\text{ate a roll}]_{VP/PP} [\text{with his hands}]_{PP}]_{VP} \quad (1)$$

$$\text{John } [[\text{ate a roll}]_{VP} [\text{with Sam}]_{VP\backslash VP}]_{VP} \quad (2)$$

$$\text{John ate a } [\text{roll}_{N} [\text{with eel}]_{N\backslash N}]_{N} \quad (3)$$

$$\text{John ate a } [\text{roll}_{N/PP} [\text{of sushi}]_{PP}]_{N} \quad (4)$$

Parsers can introduce errors in the PP attachments:

$$\text{John ate a } [[\text{roll}]_{N} [\text{with his hands}]_{N\backslash N}]_{N} \quad (1a)$$

$$\text{John } [[\text{ate a roll}]_{VP/PP} [\text{with Sam}]_{PP}]_{VP} \quad (2a)$$

$$\text{John } [[\text{ate a roll}]_{VP} [\text{with eel}]_{VP\backslash VP}]_{VP} \quad (3a)$$

Treating a nature of PP attachment

SICK-9069 GOLD: entailment BY: C&C

Two boys are $[[\text{laying}_{VP}\ [\text{in the ocean}]_{VP\backslash VP}]\ [\text{close to the beach}]_{VP\backslash VP}]$

Two boys are $[[\text{laying}_{VP/PP}\ [\text{in the water}]_{PP}]\ [\text{close to the beach}]_{VP\backslash VP}]$

V@PP

$V_{\text{pp},\alpha}\ (p^{\text{IN}}_{\text{np},\text{pp}}\ D) : [\vec{C}] : \mathbb{X}$

$p^{\text{IN}}_{\text{np},\alpha,\alpha}\ D\ V_{\alpha} : [\vec{C}] : \mathbb{X}$

$\alpha = (\text{np}^{*}, \text{vp})$

$\textbf{lie}_{\text{pp},\text{vp}}\ (\textbf{in}_{\text{np},\text{pp}}\ o_e) : [c] : \mathbb{F}$

$\textbf{in}_{\text{np},\text{vp},\text{vp}}\ o_e\ \textbf{lie}_{\text{vp}} : [c] : \mathbb{F}$

SICK-340 GOLD: entailment BY: C&C

$[\text{schoolgirl}_{N/PP}\ [\text{with a black bag}]_{PP}]$ is on a crowded train

$[\text{girl}_{N}\ [\text{with a black bag}]_{N\backslash N}]$ is on a crowded train

N@PP$_{\mathbb{T}}$

$N_{\text{pp},\text{n}}\ P_{\text{pp}} : [c_e] : \mathbb{T}$

$N_{\text{n}} : [c] : \mathbb{T}$

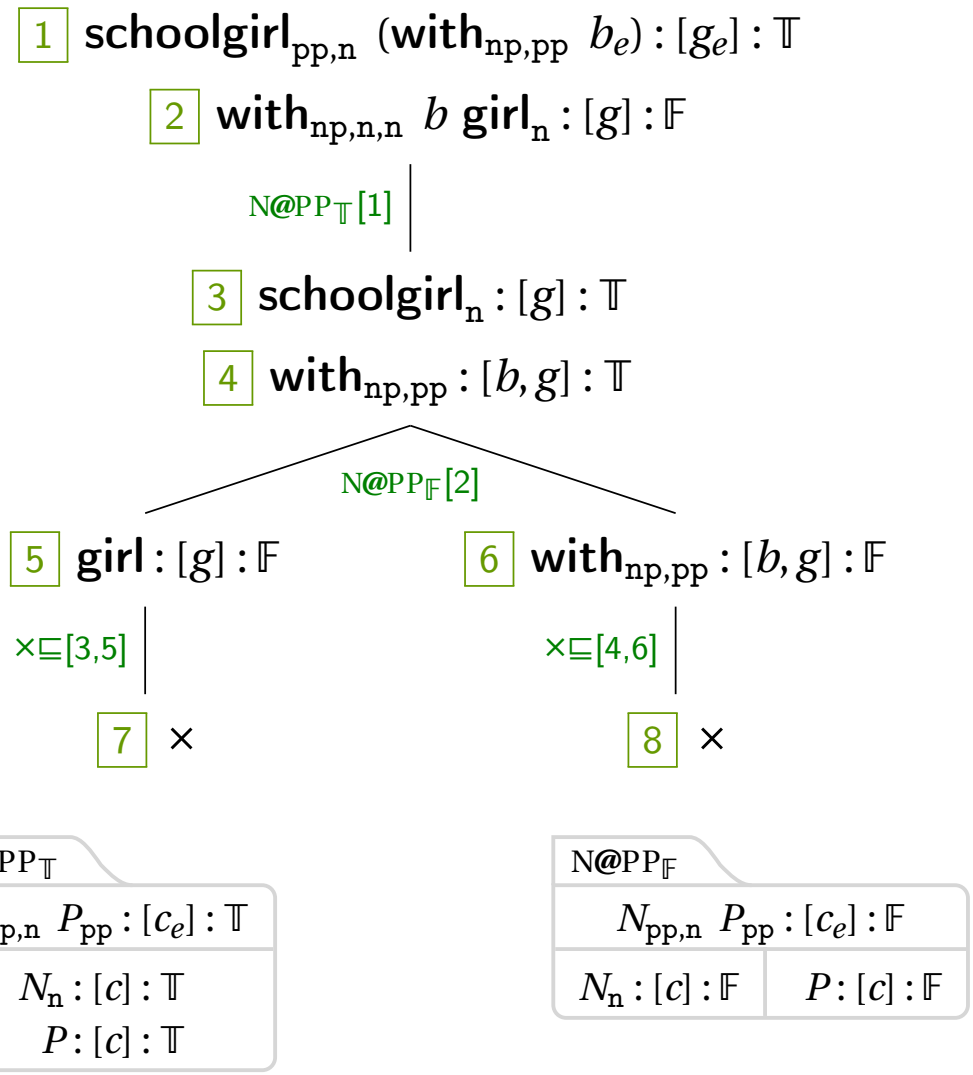
$P : [c] : \mathbb{T}$

N@PP$_{\mathbb{F}}$

$N_{\text{pp},\text{n}}\ P_{\text{pp}} : [c_e] : \mathbb{F}$

$N_{\text{n}} : [c] : \mathbb{F}$ | $P : [c] : \mathbb{F}$

Treatment case



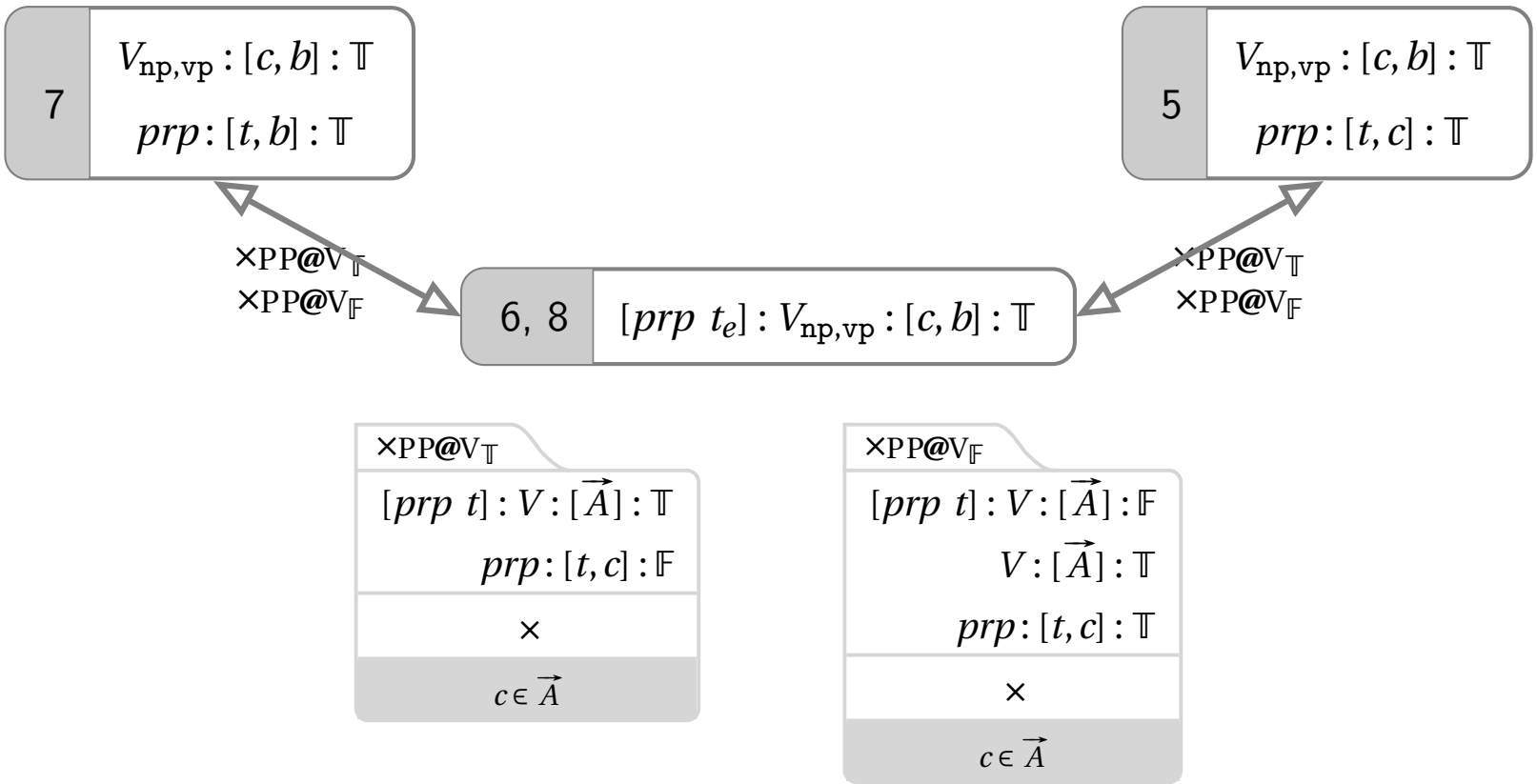
Treating a site of PP attachment

A boy saw a $[\text{criminal}_N\ [\text{with a telescope}]_{N\backslash N}]_N$ (5)

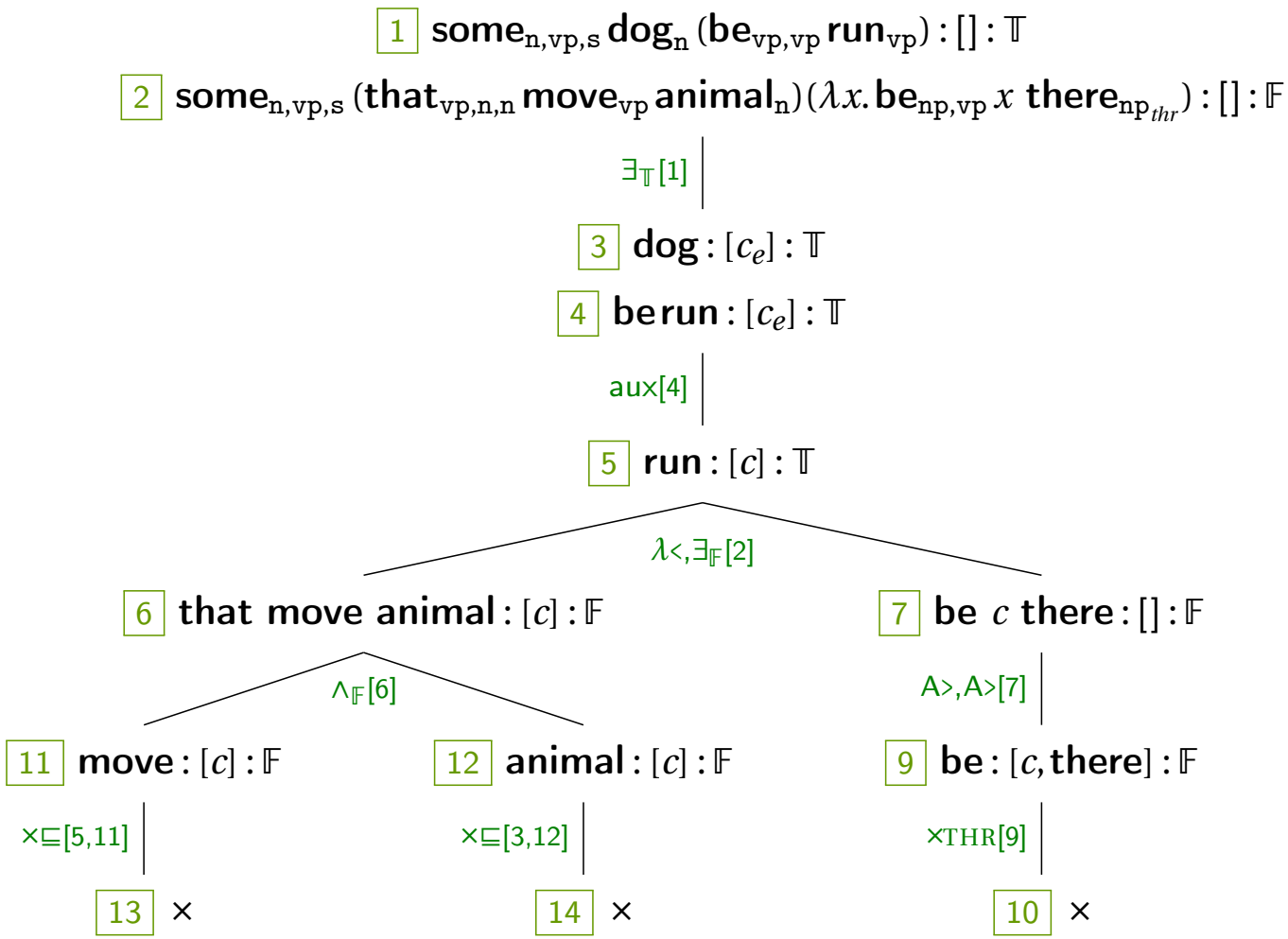
A boy $[[\text{saw a criminal}]_{VP}\ [\text{with a telescope}]_{VP\backslash VP}]_{VP}$ (6)

A criminal was seen by a $[\text{boy}_N\ [\text{with a telescope}]_{N\backslash N}]_N$ (7)

A criminal $[[\text{was seen by a boy}]_{VP}\ [\text{with a telescope}]_{VP\backslash VP}]_{VP}$ (8)



Expletive *there*



×THR

$\mathbf{be}_{\mathrm{np,vp}} : [C, D] : \mathbb{F}$

×

$\mathbf{there}_{\mathrm{np}_{thr}} \in \{C, D\}$

Other closure rules

Open compound nouns:

×CPN

$$N_{\mathrm{n}} : [d] : \mathbb{T}$$
$$H_{\mathrm{pp,n}}(prp\ d) : [c] : \overline{\mathbb{X}}$$
$$A_{\mathrm{n,n}}\ H_{\mathrm{n}} : [c] : \mathbb{X}$$
$$\times$$

$N \approx A$ or $N \approx_d A$

$$\frac{\mathsf{protection}_{\mathrm{n}} : [d_e] : \mathbb{T} \quad \mathsf{gear}_{\mathrm{pp,n}}(\mathsf{for}_{\mathrm{np,pp}}\ d_e) : [c_e] : \mathbb{F} \quad \mathsf{protective}_{\mathrm{n,n}}\ \mathsf{gear}_{\mathrm{n}} : [c_e] : \mathbb{T}}{\times}\,(\times\mathrm{CPN}^*)$$

Light verb constructions:

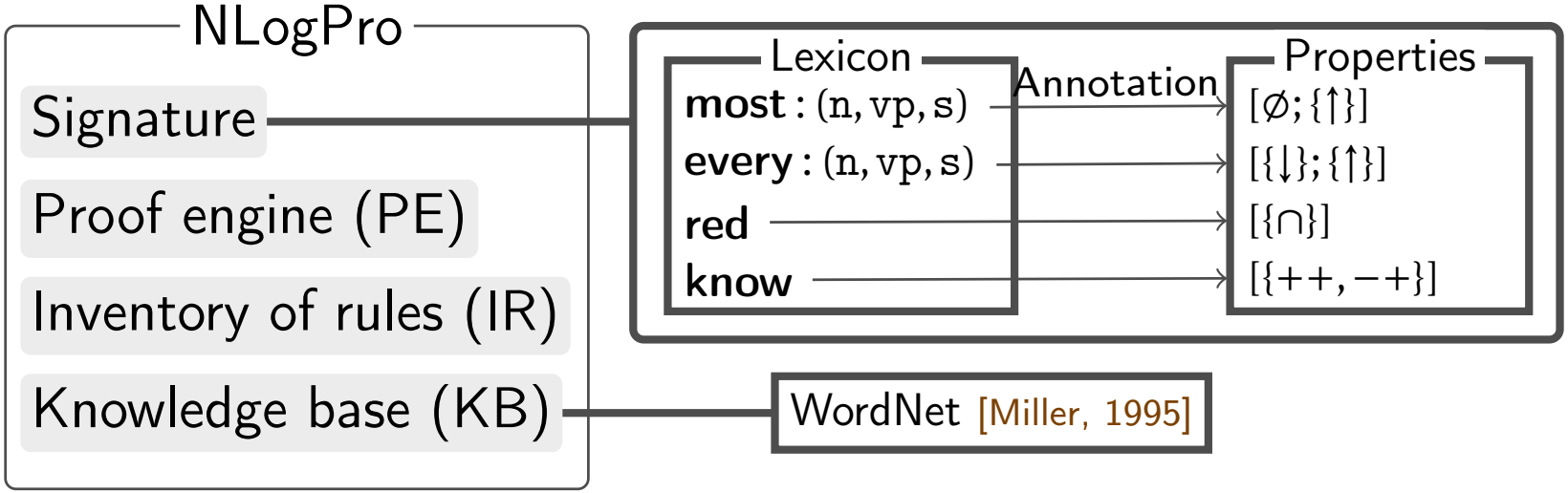
×LVC

$$l_{\vec{\alpha},\mathrm{vp}} : [c, \vec{D}] : \mathbb{X}$$
$$u_{\mathrm{n}} : [c] : \mathbb{T}$$
$$v_{\vec{\alpha},\mathrm{s}} : [D] : \overline{\mathbb{X}}$$
$$\times$$

$l \in \{\mathsf{do}, \mathsf{get}, \mathsf{give}, \mathsf{have}, \mathsf{make}, \mathsf{take}\}$, $\vec{\alpha}$ is formed by np and pp, and $u \approx_d v$

$$\frac{\mathsf{do}_{\mathrm{np,vp}} : [d_e, h_e] : \mathbb{T} \quad \mathsf{dance}_{\mathrm{n}} : [d_e] : \mathbb{T} \quad \mathsf{dance}_{\mathrm{vp}} : [h_e] : \mathbb{F}}{\times}\,(\times\mathrm{LVC}^*)$$

Natural logic theorem prover (NLogPro)



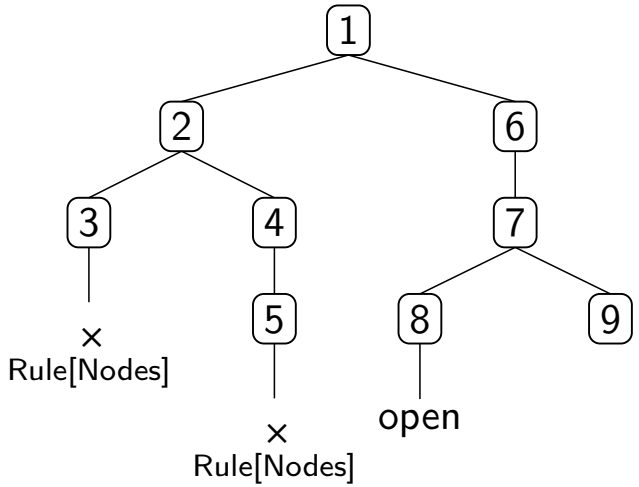
KB uses 4 relations from WordNet 3.0:

- derivation
- similarity
- hyponymy/hypernymy
- antonymy

⚠ No word sense disambiguation system is used.

Two data structures

The proof engine builds both a tree and a list structures:



$Br_1 : \langle History_1, Entities_1 \rangle$ 1 — 2 — 3

$Br_2 : \langle History_2, Entities_2 \rangle$ 1 — 2 — 4 — 5

$Br_3 : \langle History_3, Entities_3 \rangle$ 1 — 6 — 7 — 8

$Br_4 : \langle History_4, Entities_4 \rangle$ 1 — 6 — 7 — 9

Some derivable rules

Derivable rules are shortcuts for several rule applications.

$\exists^n_{\mathbb{F}}$

$q_{\mathrm{n,vp,s}}\,N\,V : [] : \mathbb{F}$
$N : [c_e] : \mathbb{T}$

$V : [c] : \mathbb{F}$

$q \in \{\mathsf{a}, \mathsf{some}, \mathsf{the}, \mathsf{s}\}$

$\forall^n_{\mathbb{T}}$

$q_{\mathrm{n,vp,s}}\,N\,V : [] : \mathbb{T}$
$N : [c_e] : \mathbb{T}$

$V : [c] : \mathbb{T}$

$q \in \{\mathsf{every}, \mathsf{the}\}$

$\mathrm{NO}^n_{\mathbb{T}}$

$\mathbf{no}_{\mathrm{n,vp,s}}\,N\,V : [] : \mathbb{T}$
$N : [c_e] : \mathbb{T}$

$V : [c] : \mathbb{F}$

$\exists^v_{\mathbb{F}}$

$q_{\mathrm{n,vp,s}}\,N\,V : [] : \mathbb{F}$
$V : [c_e] : \mathbb{T}$

$N : [c] : \mathbb{F}$

$q \in \{\mathsf{a}, \mathsf{some}, \mathsf{the}, \mathsf{s}\}$

$\forall^v_{\mathbb{T}}$

$q_{\mathrm{n,vp,s}}\,N\,V : [] : \mathbb{T}$
$V : [c_e] : \mathbb{F}$

$N : [c] : \mathbb{F}$

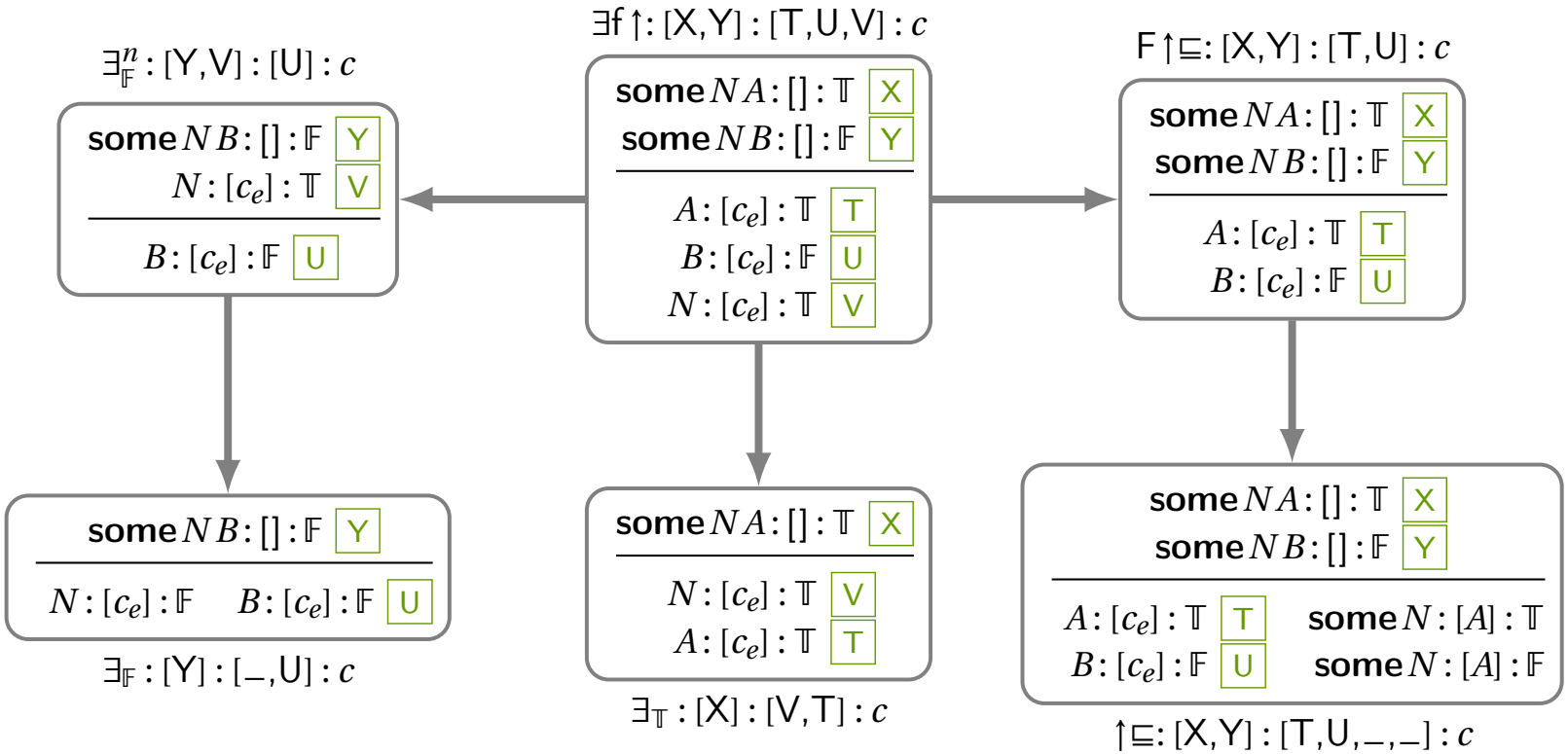
$q \in \{\mathsf{every}, \mathsf{the}\}$

$\mathrm{NO}^v_{\mathbb{T}}$

$\mathbf{no}_{\mathrm{n,vp,s}}\,N\,V : [] : \mathbb{T}$
$V : [c_e] : \mathbb{T}$

$N : [c] : \mathbb{F}$

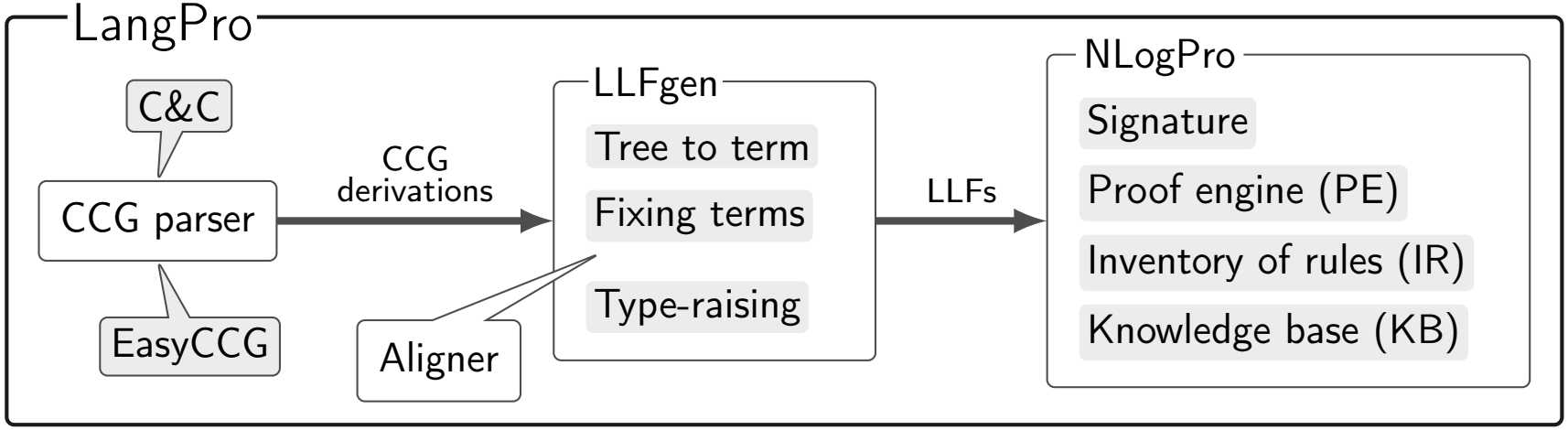
Rule application subsumption



$$\exists^n_{\mathbb{F}} : [Y,V] : [U] : c \Rightarrow \exists_{\mathbb{F}} : [Y] : [\_,U] : c$$

Natural language theorem prover (LangPro)

Chaining a CCG parser, the LLF generator and NLogPro results in a theorem prover for natural language.

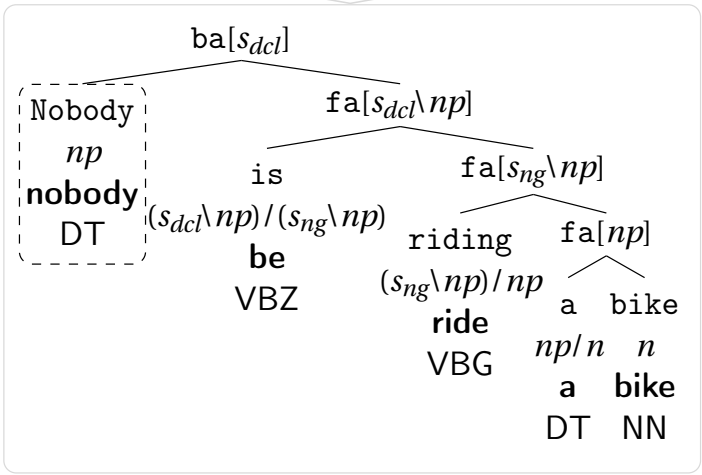


Online demo at: http://naturallogic.pro

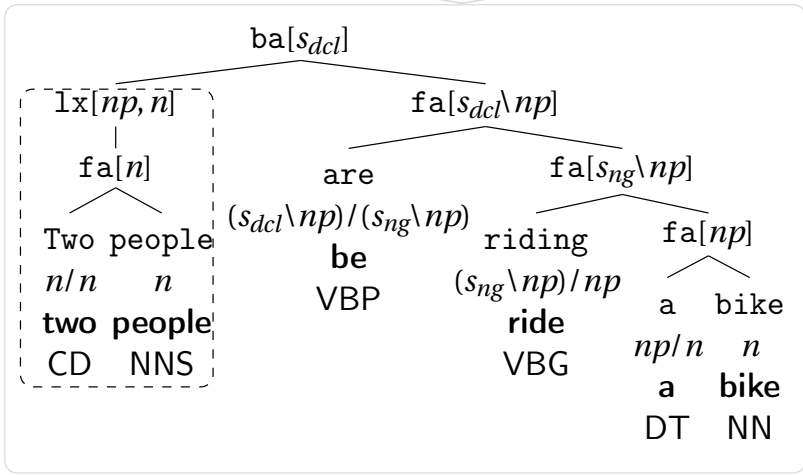
LangPro in action

SICK-2865: `Nobody is riding a bike` ⇏ `Two people are riding a bike`

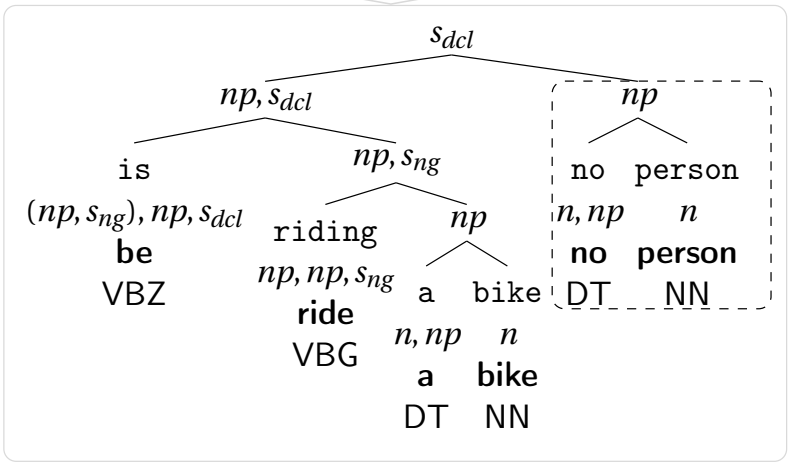
the C&C parser



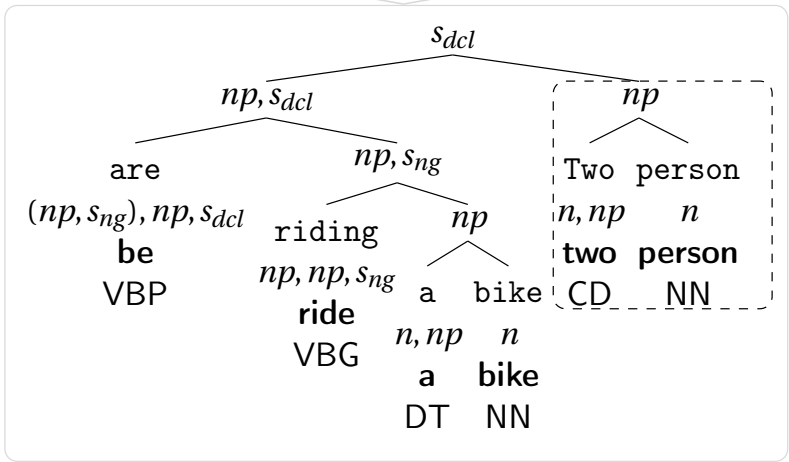
the C&C parser



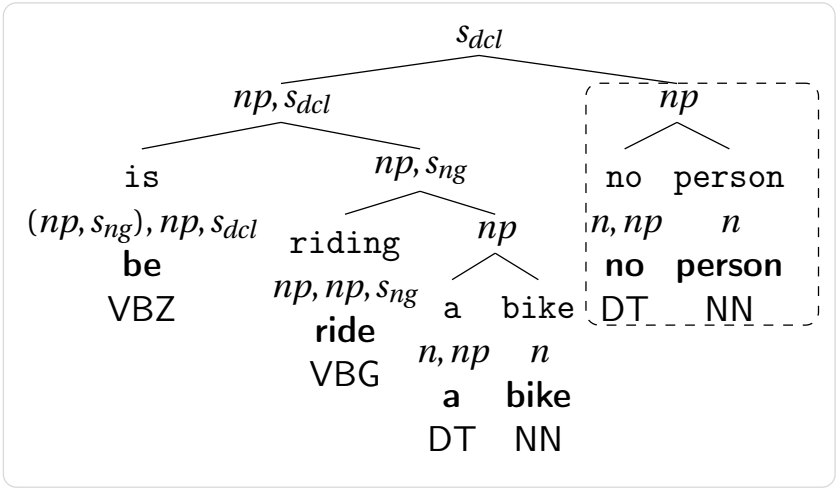
Fixing

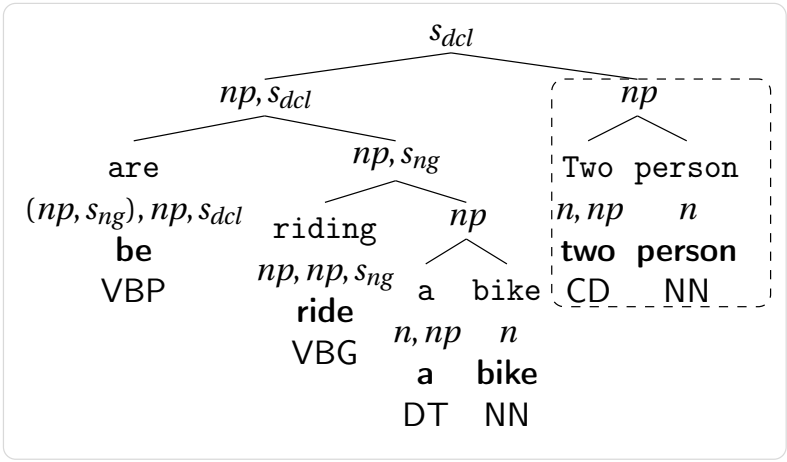


Fixing



LangPro in action (2)





Type-raising

no person (be ($\lambda x.$ (**a bike**) ($\lambda y.$ **ride** y x)))
a bike ($\lambda x.$ **no person** (be (ride x)))

Type-raising

two person (be ($\lambda x.$ (**a bike**) ($\lambda y.$ **ride** y x)))
a bike ($\lambda x.$ **two person** (be (ride x)))

Proving by PE using IR & KB

`intial nodes for entailment checking:`
no person (be ($\lambda x.$ (**a bike**) ($\lambda y.$ **ride** y x))): [] : $\mathbb{T}$
two person (be ($\lambda x.$ (**a bike**) ($\lambda y.$ **ride** y x))): [] : $\mathbb{F}$

`intial nodes for contradiction checking:`
no person (be ($\lambda x.$ (**a bike**) ($\lambda y.$ **ride** y x))): [] : $\mathbb{T}$
two person (be ($\lambda x.$ (**a bike**) ($\lambda y.$ **ride** y x))): [] : $\mathbb{T}$

LangPro in action (3)

1 **no person** (be($\lambda x.$ (**a bike**) ($\lambda y.$ **ride** y x))): [] : $\mathbb{T}$

2 **two person** (be ($\lambda x.$ (**a bike**) ($\lambda y.$ **ride** y x))): [] : $\mathbb{T}$

$\exists_{\mathbb{T}}$[2]

3 **person**: $[c] : \mathbb{T}$

4 be($\lambda x.$ (**a bike**) ($\lambda y.$ **ride** y x)): $[c] : \mathbb{T}$

$\text{no}^{n}_{\mathbb{T}}$[1,4]

5 **person**: $[c] : \mathbb{F}$

6 $\times$

$$\frac{\textbf{no}\ A\ B : [] : \mathbb{T} \qquad A : [c] : \mathbb{T}}{B : [c] : \mathbb{F}}\ \text{no}^{n}_{\mathbb{T}} \qquad \frac{N^{\text{CD}}\ A\ B : [] : \mathbb{T}}{A : [c] : \mathbb{T} \quad B : [c] : \mathbb{T}}\ \exists_{\mathbb{T}}$$

Conclusion

- Automatically generating LLFs from CCG derivations;
- Introduce new, linguistically motivated rules;
- The theorem prover NLogPro for natural logic;
- The theorem prover for natural language is a pipeline: CCG parser + LLFgen + NLogPro + WordNet;
- Play with it: http://naturallogic.pro
- Clone or fork it: https://github.com/kovvalsky/LangPro

References I


Ajdukiewicz, K. (1935a). Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27. English translation "Syntactic Connexion" by H. Weber in McCall, S. (Ed.) *Polish Logic*, pp. 207–231, Oxford University Press, Oxford, 1967.


Ajdukiewicz, K. (1935b). Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27. English translation "Syntactic Connexion" by H. Weber in McCall, S. (Ed.) *Polish Logic*, pp. 207–231, Oxford University Press, Oxford, 1967.


Clark, S. and Curran, J. R. (2007). Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33.


Hillel, Y. B. (1953). A Quasi-Arithmetical Notation for Syntactic Description. *Language*, 29(1):47–58.


Lewis, M. and Steedman, M. (2014). A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, Doha, Qatar. Association for Computational Linguistics.


Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.


Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA, USA.