

A Natural Proof System for Natural Language

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

Lasha Abzianidze

Reinhard Muskens



university of
 groningen



ESLLI 2019 in Rīga, Latvija

Today:

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

LLFs and Categorical Grammar

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

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

LLFs and Categorical Grammar

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

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

Combinatory Categorical 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.

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.

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 $Y \setminus X$, where X and Y are categories.

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

Token : Category : λ term

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:

Token : Category : λ term

Combining rules:

$Y/X : f \quad X : a \Rightarrow Y : fa$ *Forward functional application* ($>$)

$X : a \quad Y \backslash X : f \Rightarrow Y : fa$ *Backward functional application* ($<$)

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 $Y \setminus X$, where X and Y are categories.

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

Token : Category : λ term

Combining rules:

$Y/X : f \quad X : a \Rightarrow Y : fa$ *Forward functional application* ($>$)

$X : a \quad Y \setminus X : f \Rightarrow Y : fa$ *Backward functional application* ($<$)

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

Combinatory Categorical Grammar

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

$X : a \Rightarrow T / (T \backslash X) : \lambda x. xa$ *Forward type-raising* ($>T$)

$X : a \Rightarrow T \backslash (T / X) : \lambda x. xa$ *Backward type-raising* ($<T$)

Combinatory Categorical Grammar

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

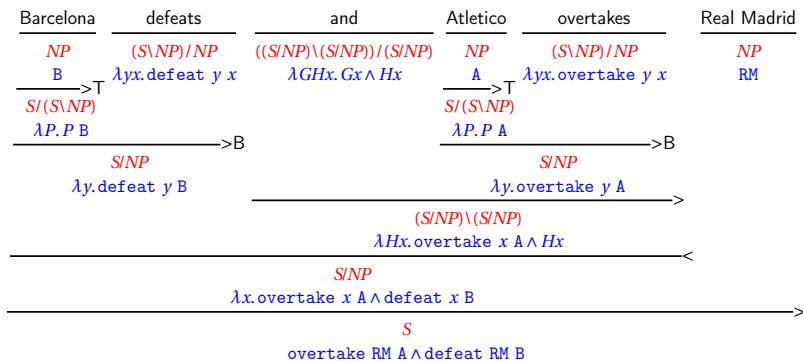
$$X : a \Rightarrow T / (T \backslash X) : \lambda x. xa \quad \text{Forward type-raising} \quad (>T)$$

$$X : a \Rightarrow T \backslash (T / X) : \lambda x. xa \quad \text{Backward type-raising} \quad (<T)$$

$$Z / Y : f \quad Y / X : g \Rightarrow Z / X : \lambda x. f(gx) \quad \text{Forward functional comp.} \quad (>B)$$

$$Y / X : g \quad Z \backslash Y : f \Rightarrow Z / X : \lambda x. f(gx) \quad \text{Backw. crossing fun. comp.} \quad (<B_x)$$

CCG derivation

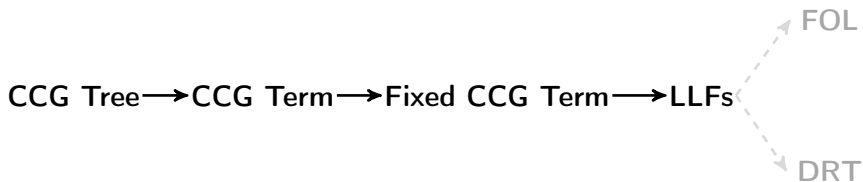


$$X : a \Rightarrow T/(T \setminus X) : \lambda x. xa \quad \text{Forward type-raising} \quad (>T)$$

$$Z/Y : f \quad Y/X : g \Rightarrow Z/X : \lambda x. f(gx) \quad \text{Forward functional comp.} \quad (>B)$$

From CCG Trees to LLFs

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

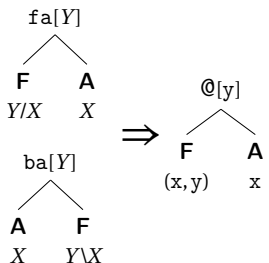


CCG Tree \rightarrow CCG term

CCG Tree \rightarrow CCG Term \rightarrow Fixed CCG Term \rightarrow LLFs

FOL

DRT

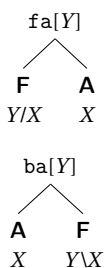
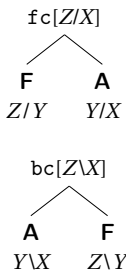
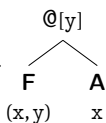
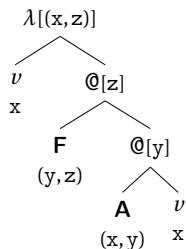


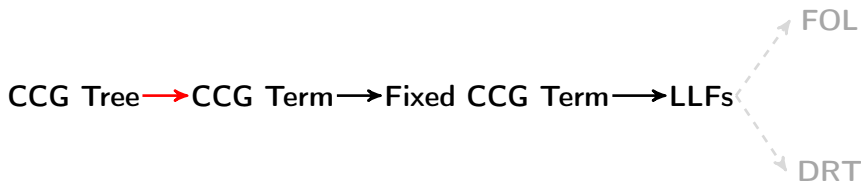
CCG Tree \rightarrow CCG term

CCG Tree \rightarrow CCG Term \rightarrow Fixed CCG Term \rightarrow LLFs

FOL

DRT

 \Rightarrow  \Rightarrow 

CCG Tree \rightarrow CCG term

Removing directionality:

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

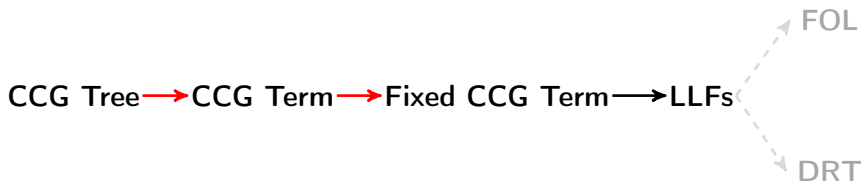
$$\text{ba}(A_X, F_{Y \setminus X}) \rightsquigarrow FA$$

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

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

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

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

CCG term \rightarrow fixed CCG term

Correcting CCG terms:

$$[\text{Dow}_{n,n}^{\text{PER}} \text{Jones}_n^{\text{PER}}]_{np} \rightsquigarrow \text{Dow_Jones}_{np}$$

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

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

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

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

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

fixed CCG term \rightarrow LLFs

Every man who ate a burger died

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

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

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

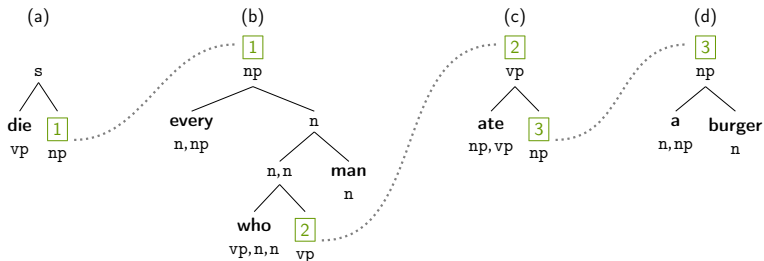
fixed CCG term \rightarrow LLFs

Every man who ate a burger died

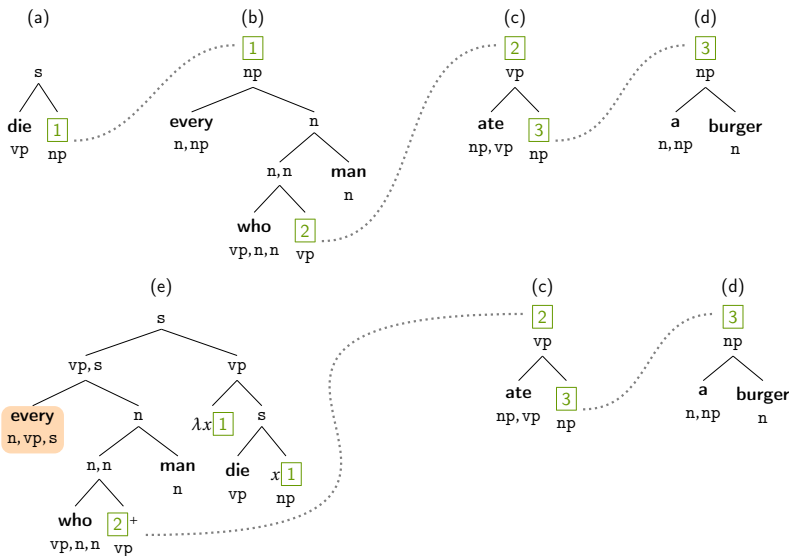
$\text{die}_{vp} (\text{every}_{n,np} (\text{who}_{vp,n,n} (\text{eat}_{np,vp} (\text{a}_{n,np} \text{burger}_n)) \text{man}_n)) \rightsquigarrow$

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

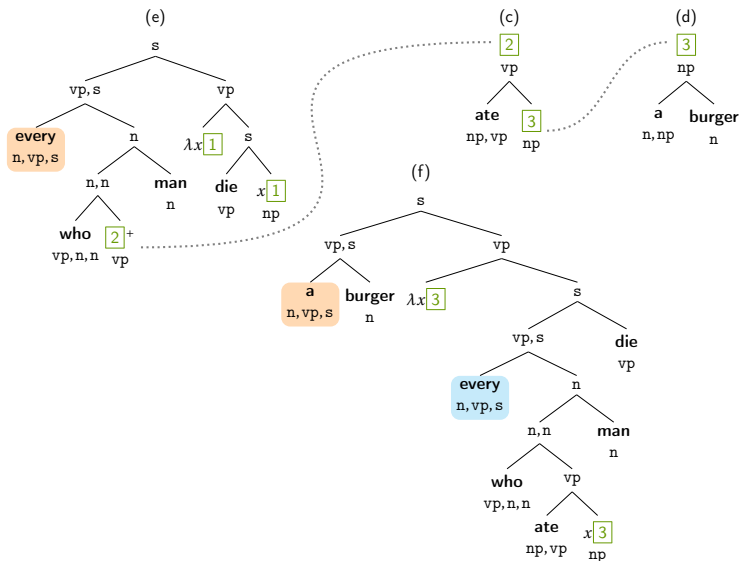
$\text{A}_{n,vp,s} \text{burger} (\lambda y. \text{EVERY}_{n,vp,s} (\text{who} (\lambda x. \text{eat } y_{np} x_{np})) \text{man}) \text{die}$



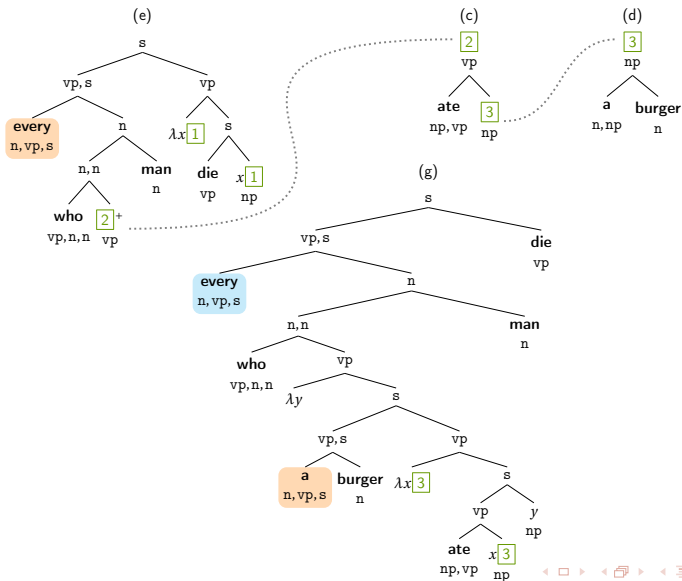
Gluing 1



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

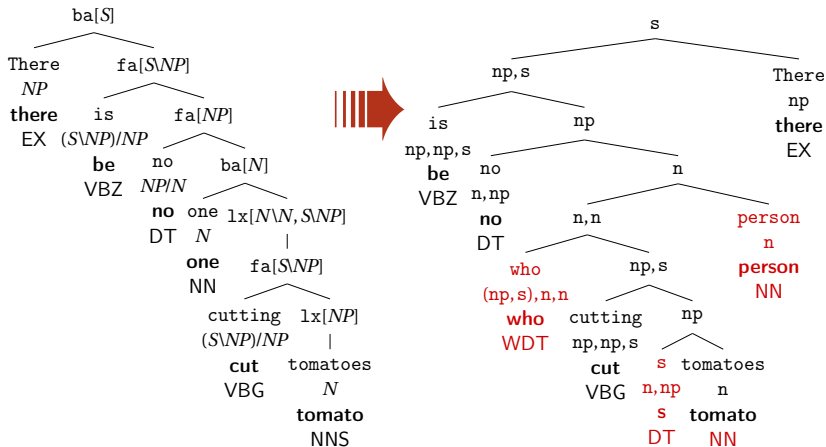
$\text{and}_{s,s,s}(\text{sleep}_{vp}(\text{every}_{n,np} \text{man}_n))(\text{worry}_{vp}(\text{some}_{n,np} \text{woman}_n))$

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

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

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

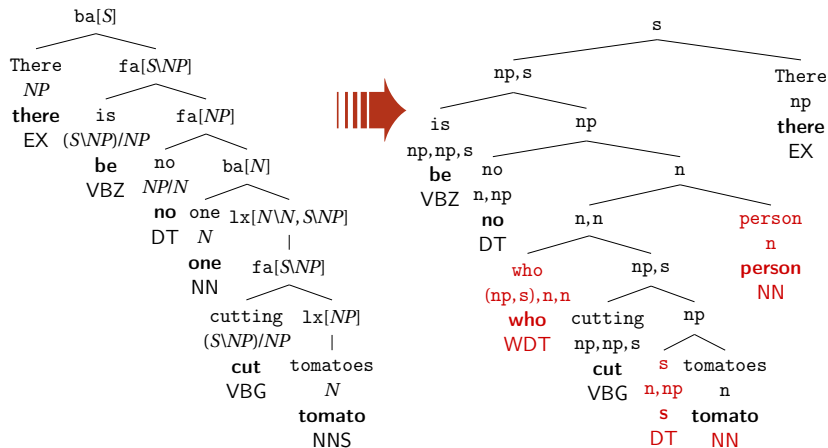
LLFgen: CCG tree \rightarrow fixed CCG term \rightarrow LLFs



There is no one cutting tomatoes \rightsquigarrow

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

LLFgen: CCG tree \rightarrow fixed CCG term \rightarrow LLFs



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

no(who ($\lambda x. s$ tomato ($\lambda y. cut yx$)) person) ($\lambda z. be z$ there)
 s tomato ($\lambda y. no(who (cut y) person) (\lambda z. be z there)$)

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.

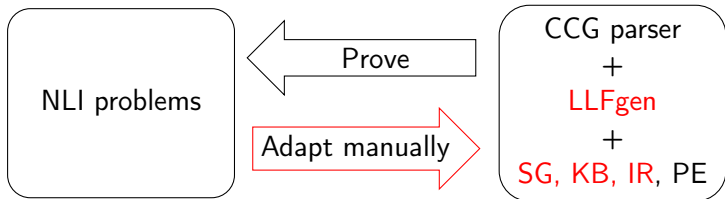
Linguistic Rules

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

 $PP@N_{\top}$
 $p_{np,n,n}^{IN} dN : [c] : \top$
 $N : [c] : \top$
 $p_{np,pp} : [d, c] : \top$
 $with_{np,n,n} g_e \mathbf{bicycle}_n : [c_e] : \top$
 $\mathbf{bicycle} : [c] : \top$
 $with_{np,pp} : [g, c] : \top$

Rules for prepositions

PP@N \top
$p_{np,n,n}^{IN} dN : [c] : \top$
$N : [c] : \top$
$p_{np,pp} : [d, c] : \top$

with_{np,n,n} g_e **bicycle**_n : [c_e] : \top

|
bicycle : [c] : \top

with_{np,pp} : [g, c] : \top

PP@N \mathbb{F}
$p_{np,n,n}^{IN} dN : [c] : \mathbb{F}$
$N : [c] : \mathbb{F}$
$p_{np,pp} : [g, c] : \mathbb{F}$

with_{np,n,n} g_e **bicycle**_n : [c_e] : \mathbb{F}

bicycle : [c] : \mathbb{F}

with_{np,pp} : [g, c] : \mathbb{F}

Problem of PP attachment

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

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

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

John ate a [roll_N [with eel]_{N\N}]_N (3)

John ate a [roll_{N/PP} [of sushi]_{PP}]_N (4)

Problem of PP attachment

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

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

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

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

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

Parsers can introduce errors in the PP attachments:

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

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

John $[[\text{ate a roll}]_{VP} [\text{with eel}]_{VP \setminus VP}]_{VP}$ (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 $[[\text{laying}_{VP/PP} [\text{in the water}]_{PP} [\text{close to the beach}]_{VP\backslash VP}]$

Treating a nature of PP attachment

SICK-9069 GOLD: entailment BY: C&C

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

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

V@PP
$V_{pp,\alpha} (p_{np,pp}^{IN} D) : [\vec{C}] : \mathbb{X}$
$p_{np,\alpha,\alpha}^{IN} DV_{\alpha} : [\vec{C}] : \mathbb{X}$
$\alpha = (np^*, vp)$

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

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

Treating a nature of PP attachment

SICK-9069 GOLD: entailment BY: C&C

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

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

V@PP
$V_{pp,\alpha} (p_{np,pp}^{IN} D) : [\vec{C}] : \mathbb{X}$
$p_{np,\alpha,\alpha}^{IN} DV_{\alpha} : [\vec{C}] : \mathbb{X}$
$\alpha = (np^*, vp)$

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

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

SICK-340 GOLD: entailment BY: C&C

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

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

Treating a nature of PP attachment

SICK-9069 GOLD: entailment BY: C&C

Two boys are [[laying_{VP} [in the ocean]_{VP\VP}] [close to the beach]_{VP\VP}]

Two boys are [[laying_{VP/PP} [in the water]_{PP}] [close to the beach]_{VP\VP}]

V@PP
$V_{pp,\alpha} (p_{np,pp}^{IN} D) : [\vec{C}] : \mathbb{X}$
$p_{np,\alpha,\alpha}^{IN} D V_{\alpha} : [\vec{C}] : \mathbb{X}$
$\alpha = (np^*, vp)$

$lie_{pp,vp} (in_{np,pp} o_e) : [c] : \mathbb{F}$

$in_{np,vp,vp} o_e lie_{vp} : [c] : \mathbb{F}$

SICK-340 GOLD: entailment BY: C&C

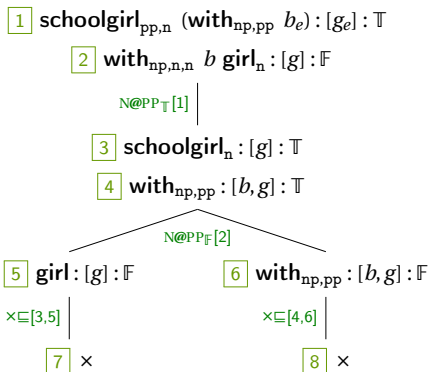
[schoolgirl_{N/PP} [with a black bag]_{PP}] is on a crowded train

[girl_N [with a black bag]_{N\W}] is on a crowded train

N@PP _T
$N_{pp,n} P_{pp} : [c_e] : \mathbb{T}$
$N_n : [c] : \mathbb{T}$
$P : [c] : \mathbb{T}$

N@PP _F
$N_{pp,n} P_{pp} : [c_e] : \mathbb{F}$
$N_n : [c] : \mathbb{F}$
$P : [c] : \mathbb{F}$

Treatment case



$N@PP_T$
$N_{pp,n} P_{pp} : [c_e] : \mathbb{T}$
$N_n : [c] : \mathbb{T}$
$P : [c] : \mathbb{T}$

$N@PP_F$
$N_{pp,n} P_{pp} : [c_e] : \mathbb{F}$
$N_n : [c] : \mathbb{F}$
$P : [c] : \mathbb{F}$

Treating a site of PP attachment

A boy saw a [criminal]_N [with a telescope]_{MN}]_N (5)

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

A criminal was seen by a [boy]_N [with a telescope]_{NW}]_N (7)

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

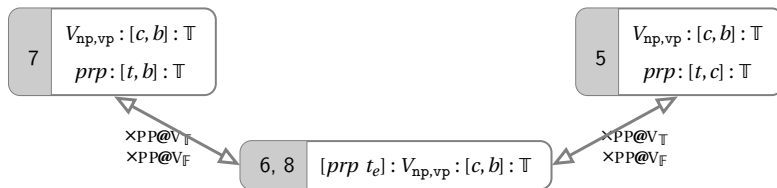
Treating a site of PP attachment

A boy saw a [criminal]_N [with a telescope]_{MN}]_N (5)

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

A criminal was seen by a [boy]_N [with a telescope]_{NW}]_N (7)

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



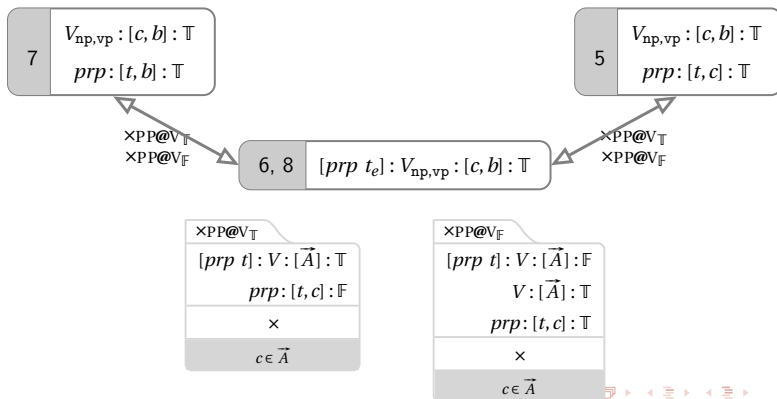
Treating a site of PP attachment

A boy saw a [criminal_N [with a telescope]_{MN}]_N (5)

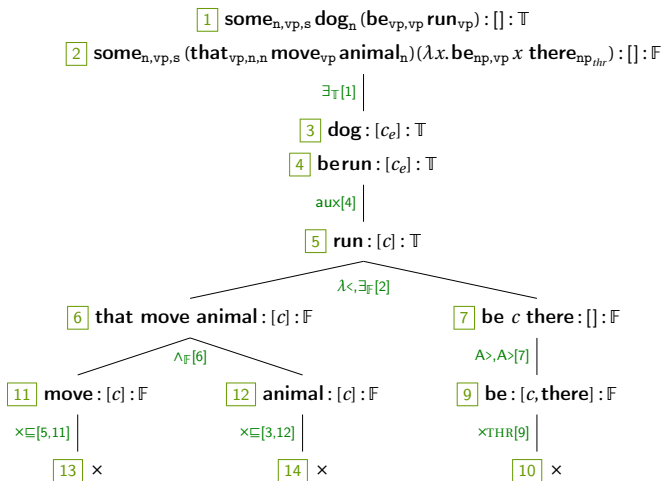
A boy [[saw a criminal]_{VP} [with a telescope]_{VP\VP}]_{VP} (6)

A criminal was seen by a [boy_N [with a telescope]_{NW}]_N (7)

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



Expletive *there*



\times_{THR}
$\text{be}_{np, vp} : [C, D] : \mathbb{F}$
\times
$\text{there}_{np, thr} \in \{C, D\}$

Other closure rules

Open compound nouns:

×CPN
$N_n : [d] : \mathbb{T}$
$H_{pp,n}(prp\ d) : [c] : \overline{\mathbb{X}}$
$A_{n,n} H_n : [c] : \mathbb{X}$
×
$N \approx A$ or $N \approx_d A$

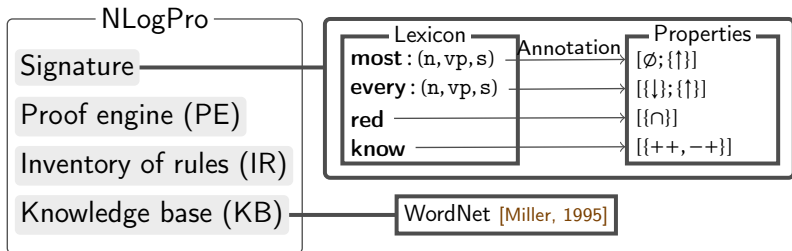
$$\frac{\text{protection}_n : [d_e] : \mathbb{T} \quad \text{gear}_{pp,n}(\text{for}_{np,pp}\ d_e) : [c_e] : \mathbb{F} \quad \text{protective}_{n,n}\ \text{gear}_n : [c_e] : \mathbb{T}}{\times} (\times\text{CPN}^*)$$

Light verb constructions:

×LVC
$l_{\vec{\alpha},vp} : [c, \vec{D}] : \mathbb{X}$
$u_n : [c] : \mathbb{T}$
$v_{\vec{\alpha},s} : [D] : \overline{\mathbb{X}}$
×
$l \in \{\text{do, get, give, have, make, take}\},$ $\vec{\alpha}$ is formed by np and pp, and $u \approx_d v$

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

Natural logic theorem prover (NLogPro)



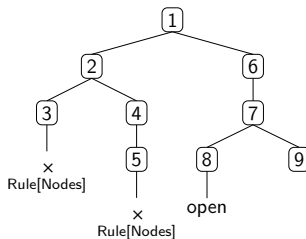
KB uses 4 relations from WordNet 3.0:

- 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:



$Br_1 : \langle \text{History}_1, \text{Entities}_1 \rangle$ ①—②—③

$Br_2 : \langle \text{History}_2, \text{Entities}_2 \rangle$ ①—②—④—⑤

$Br_3 : \langle \text{History}_3, \text{Entities}_3 \rangle$ ①—⑥—⑦—⑧

$Br_4 : \langle \text{History}_4, \text{Entities}_4 \rangle$ ①—⑥—⑦—⑨

Some derivable rules

Derivable rules are shortcuts for several rule applications.

\exists_F^n
$q_{n,vp,s}NV : [] : F$ $N : [c_e] : T$
$V : [c] : F$
$q \in \{a, \text{some}, \text{the}, s\}$

\forall_T^n
$q_{n,vp,s}NV : [] : T$ $N : [c_e] : T$
$V : [c] : T$
$q \in \{\text{every}, \text{the}\}$

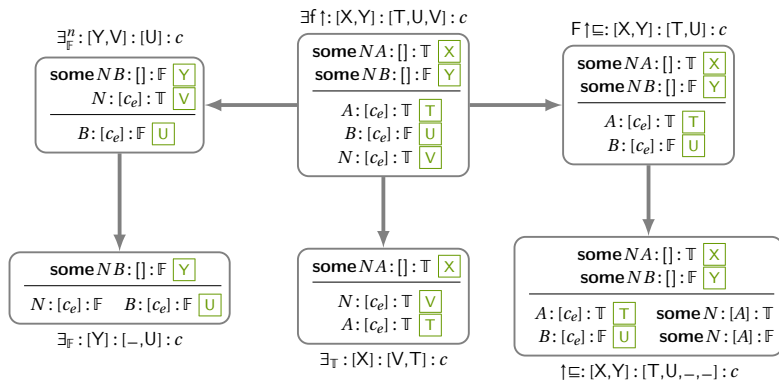
NO_T^n
$\text{no}_{n,vp,s}NV : [] : T$ $N : [c_e] : T$
$V : [c] : F$

\exists_F^v
$q_{n,vp,s}NV : [] : F$ $V : [c_e] : T$
$N : [c] : F$
$q \in \{a, \text{some}, \text{the}, s\}$

\forall_T^v
$q_{n,vp,s}NV : [] : T$ $V : [c_e] : F$
$N : [c] : F$
$q \in \{\text{every}, \text{the}\}$

NO_T^v
$\text{no}_{n,vp,s}NV : [] : T$ $V : [c_e] : T$
$N : [c] : F$

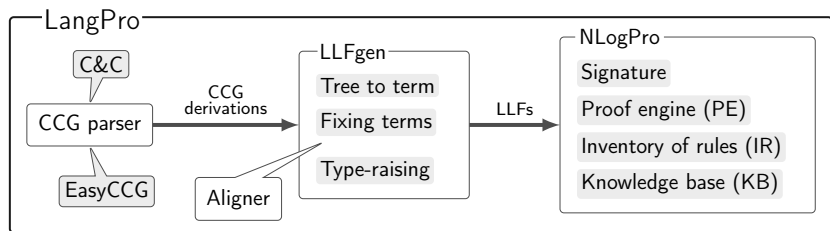
Rule application subsumption



$$\exists F^n: [Y, V]: [U]: c \Rightarrow \exists F: [Y]: [-, U]: c$$

Natural language theorem prover (LangPro)

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

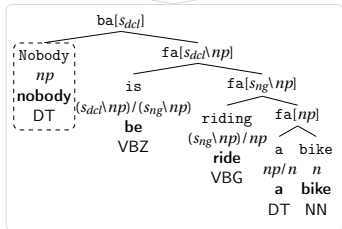


Online demo at: <http://naturallogic.pro>

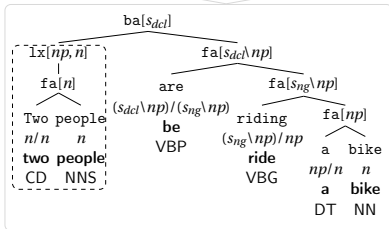
LangPro in action

SICK-2865: Nobody is riding a bike \Leftrightarrow Two people are riding a bike

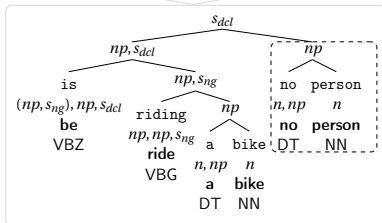
the C&C parser



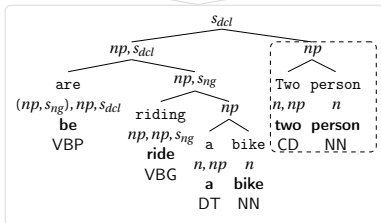
the C&C parser



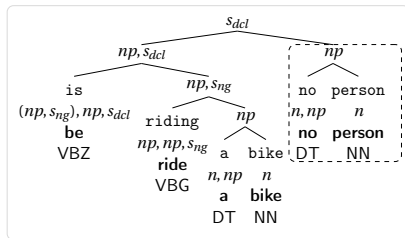
Fixing



Fixing

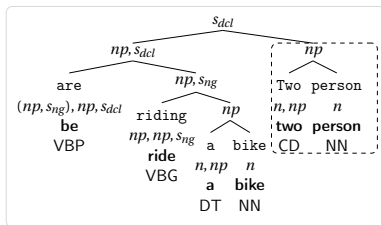


LangPro in action (2)



Type-raising

no person (be (λx . (a bike) (λy . ride $y x$)))
 a bike (λx . no person (be (ride x)))



Type-raising

two person (be (λx . (a bike) (λy . ride $y x$)))
 a bike (λx . two person (be (ride x)))

Proving by PE using IR & KB

intial nodes for entailment checking:
 no person (be (λx . (a bike) (λy . ride $y x$))): []: \top
 two person (be (λx . (a bike) (λy . ride $y x$))): []: \top

intial nodes for contradiction checking:
 no person (be (λx . (a bike) (λy . ride $y x$))): []: \top
 two person (be (λx . (a bike) (λy . ride $y x$))): []: \top

LangPro in action (3)

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

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

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

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

4 be(λx . (a bike) (λy . ride y x)): [c] : \mathbb{T}

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

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

6 \times

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

$$\frac{N^{\text{CD}} \ A \ B: [] : \mathbb{T}}{A: [c] : \mathbb{T} \quad B: [c] : \mathbb{T}} \exists_{\mathbb{T}}$$

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.