Neural Proof Nets

Konstantinos Kogkalidis[◇] Michael Moortgat[◇] Richard Moot[□]

[◇]Utrecht Institute of Linguistics OTS, Utrecht University

[□]LIRMM, Université de Montpellier, CNRS

Overview

tl;dr

A methodology to transcribe raw text to constructive proofs & functional programs

Theory
- Typelogical grammars
- Type Logic: $ILL_{\multimap,\Diamond,\Box}$
- Proof Nets

Practice
- Data
- Supertagging
- Parsing as permutation

Typelogical grammars

Key points

- words are assigned *formulas* of a constructive logic
- parsing is a formal deduction process: parse $\equiv$ *proof*
- Curry-Howard isomorphism: formula $\equiv$ *type*, proof $\equiv$ *program*

$\Longrightarrow$ a syntactic parse becomes the instructions for an executable functional program.

Typelogical grammars

Key points

- words are assigned *formulas* of a constructive logic
- parsing is a formal deduction process: parse $\equiv$ *proof*
- Curry-Howard isomorphism: formula $\equiv$ *type*, proof $\equiv$ *program*

$\implies$ a syntactic parse becomes the instructions for an executable functional program.

The logic

Modal implicational intuitionistic linear logic ($\mathrm{ILL}_{\multimap,\diamondsuit,\Box}$)

$\mathrm{ILL}_{\multimap,\diamondsuit,\Box}$ contains types from the inductive set:

$$\mathcal{T} := A \mid T_1 \multimap T_2 \mid \diamondsuit T \mid \Box T$$

where

- A an *atomic* type, from a finite set $\mathcal{A}$
- $T_1 \multimap T_2$ a *complex* type, denoting the transformation that consumes $T_1 \in \mathcal{T}$ to produce $T_2 \in \mathcal{T}$
- $\diamondsuit$, $\Box$ unary modalities

$\mathrm{ILL}_{\multimap,\diamondsuit,\Box}$ and deep syntax

In our setup:

- $\mathcal{A}$: a set of *syntactic categories*, e.g. $\{n, np, s, pron, \ldots\}$
- parameterized modalities $\diamondsuit^d, \Box^b$ where $d \in \mathcal{D}, b \in \mathcal{B}$:
  - $\mathcal{D}$: a set of *complement markers*, e.g. $\{su, dobj, pobj, predc, \ldots\}$
  - $\mathcal{B}$: a set of *adjunct markers*, e.g. $\{mod, app, det, \ldots\}$

A *lexicon* $\mathcal{L}$ assigns types:

- from $\mathcal{A}$ to autonomous words, e.g.
  blackbirds :: np, berries :: np
- $\Box^b(T_1 \multimap T_2)$ to adjuncts, e.g.
  the :: $\Box^{det}(np \multimap np)$
- $\diamondsuit^d T_1 \multimap T_2$ to phrasal heads, e.g.
  find :: $\diamondsuit^{predc} adj \multimap \diamondsuit^{dobj} np \multimap \diamondsuit^{su} np \multimap s$,
  that :: $\diamondsuit^{body}\left(\diamondsuit^{dobj} np \multimap s\right) \multimap \Box^{mod}(np \multimap np)$

$\mathrm{ILL}_{\multimap,\diamondsuit,\Box}$ and deep syntax

In our setup:

- $\mathcal{A}$: a set of *syntactic categories*, e.g. $\{n, np, s, pron, \ldots\}$
- parameterized modalities $\diamondsuit^d$, $\Box^b$ where $d \in \mathcal{D}$, $b \in \mathcal{B}$:
  - $\mathcal{D}$: a set of *complement markers*, e.g. $\{su, dobj, pobj, predc, \ldots\}$
  - $\mathcal{B}$: a set of *adjunct markers*, e.g. $\{mod, app, det, \ldots\}$

A *lexicon* $\mathcal{L}$ assigns types:

- from $\mathcal{A}$ to autonomous words, e.g.
  blackbirds :: np, berries :: np
- $\Box^b(T_1 \multimap T_2)$ to adjuncts, e.g.
  the :: $\Box^{det}(np \multimap np)$
- $\diamondsuit^d T_1 \multimap T_2$ to phrasal heads, e.g.
  find :: $\diamondsuit^{predc} adj \multimap \diamondsuit^{dobj} np \multimap \diamondsuit^{su} np \multimap s$,
  that :: $\diamondsuit^{body}\left(\diamondsuit^{dobj} np \multimap s\right) \multimap \Box^{mod}(np \multimap np)$

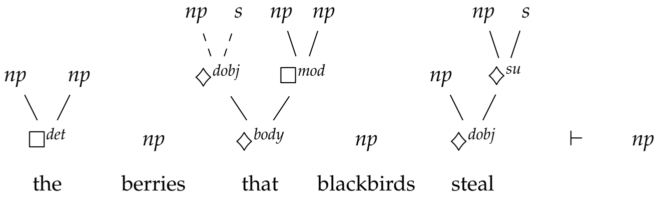
$\mathrm{ILL}_{\multimap,\Diamond,\Box}$ and deep syntax

$\mathrm{ILL}_{\multimap}$ is a *tecto-grammar* logic: it captures function-argument structures, ignoring word order and constituency structure.
$\mathrm{ILL}_{\multimap,\Diamond,\Box}$, further captures *dependency information* and constituency structure under a canonical word order.

Proof nets for $\text{ILL}_{\multimap,\Diamond,\Box}$

Proof frame

A proof frame is a judgement of the form $P_1, \ldots P_n \vdash C$, with *premises* $P_1, \ldots P_n$ and *conclusion* C decomposed intro trees.



Proof nets for $\mathrm{ILL}_{\multimap,\Diamond,\Box}$

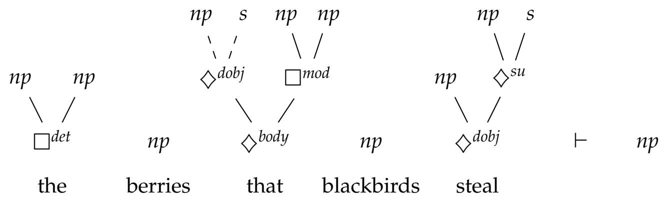
Type polarities

Premise types P_i are *positive*, conclusion type C is *negative*.

Induction:

If $A \multimap B$ positive, A is negative, B is positive.

If $A \multimap B$ negative, A is positive, B is negative.



Proof nets for $\text{ILL}_{\multimap,\Diamond,\Box}$

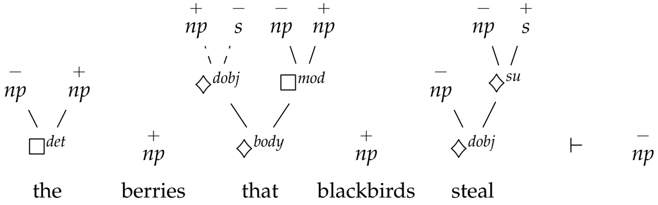
Type polarities

Premise types P_i are *positive*, conclusion type C is *negative*.

Induction:

If $A \multimap B$ positive, A is negative, B is positive.

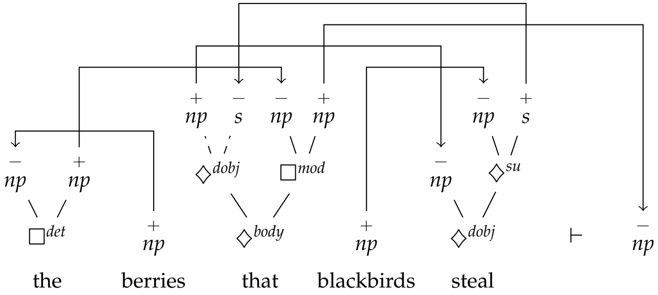
If $A \multimap B$ negative, A is positive, B is negative.



Proof nets for $\text{ILL}_{\multimap,\Diamond,\Box}$

Proof net

A proof net is a proof frame together with *axiom links*, edges from positive to negative atoms.

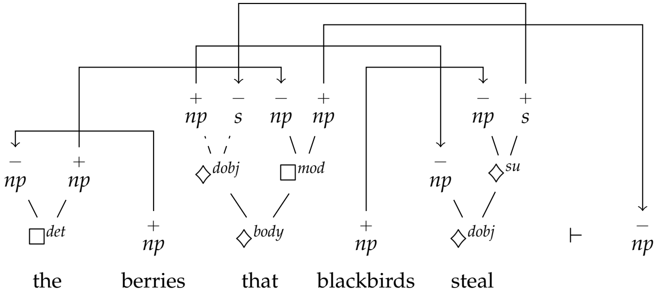


Proof nets for $\text{ILL}_{\multimap,\Diamond,\Box}$

Traversal

Traversal of a $\text{ILL}_{\multimap,\Diamond,\Box}$ proofnet induces a dependency-annotated λ-term, here:

$$\left(that \ \left(\lambda x^{dobj}. (eat\ x) \ \ blackbirds^{su} \right)^{body} \right)_{mod} \ (the_{det}\ berries)$$



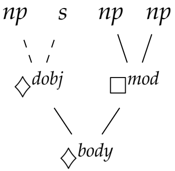
Dataset

We use the Æthel dataset: (abs/1912.12635)

- 72 192 dutch sentences as $\text{ILL}_{\multimap,\diamond,\Box}$ proofs
- 913 404 typed words
- 5 747 unique types, made of
  - 32 syntactic categories ($\mathcal{A}$)
  - 22 dependency labels ($\mathcal{D} \cup \mathcal{B}$)

Supertagging

We flatten type trees to prefix notation:



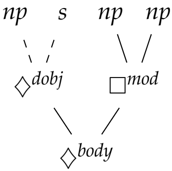
$$\longrightarrow [\Diamond^{body}, \Diamond^{dobj}, np, s, \Box^{mod}, np, np]$$

a proof frame is then the concatenation of flattened types:

$$\Big[\underbrace{\Box^{mod}, np, np,}_{\text{the}} \#, \underbrace{np,}_{\text{berries}} \#, \underbrace{\Diamond^{body}, \Diamond^{dobj}, np, s, \Box^{mod}, np, np,}_{\text{that}} \#, \underbrace{np,}_{\text{blackbirds}} \ldots\Big]$$

Supertagging

We flatten type trees to prefix notation:



$$\longrightarrow [\Diamond^{body}, \Diamond^{dobj}, np, s, \Box^{mod}, np, np]$$

a proof frame is then the concatenation of flattened types:

$$\left[\underbrace{\Box^{mod}, np, np,}_{\text{the}} \#, \underbrace{np,}_{\text{berries}} \#, \underbrace{\Diamond^{body}, \Diamond^{dobj}, np, s, \Box^{mod}, np, np,}_{\text{that}} \#, \underbrace{np,}_{\text{blackbirds}} \ldots \right]$$

Supertagging

seq2seq supertagging

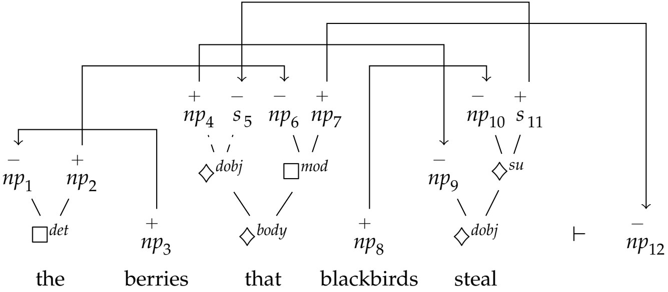
proof frames decoded using output vocabulary $\mathcal{A} \cup \mathcal{D} \cup \mathcal{B} \cup \{\#\}$

- no hard-coded vocabulary (abs/1905/13418)
- reusable representations for primitive symbols

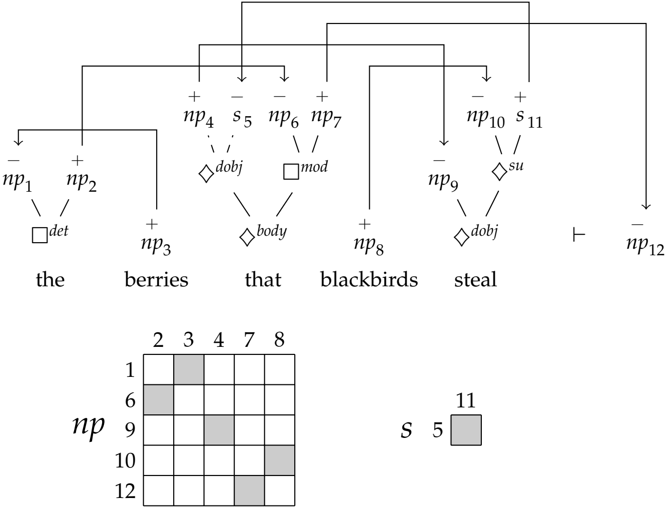
implementation

BERT-encoder, Transformer-decoder, symbols embedded in $\mathbb{C}$

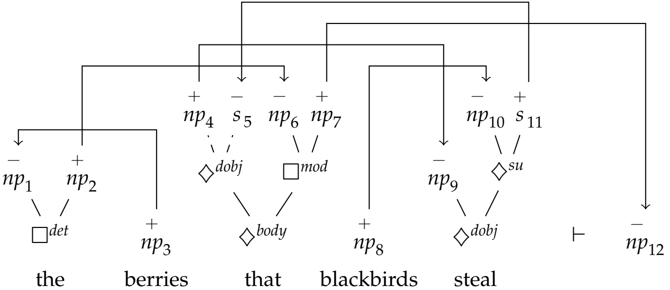
Parsing as permutation

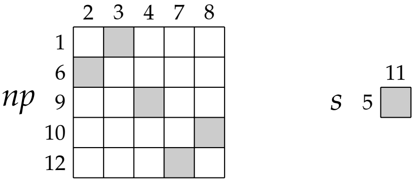


Parsing as permutation



Parsing as permutation





Parsing as permutation

Obtaining permutation matrices

1. parse decoded types to obtain polarity information
2. contextualize atoms \w sentence (bi-modal encoder)
3. for each atomic type A:
   - ▹ index & extract positive and negative vectors A_p, A_n
   - ▹ compute their pair-wise matching as $A_p A_n^\top$
   - ▹ normalize to bistochasticity by iterating the *Sinkhorn operator**

(*) a 2-dimensional, assignment-preserving softmax:

- ▶ structural correctness with no explicit structure
- ▶ easy training with negative log-likelihood
- ▶ sentence- and batch-wide parallelism

Table with numbers

metric	baseline (alpino)	our model greedy	our model 5 beams
type accuracy	56.2	85.5	93.2
frame correct	46.6	57.6	69.6
ILL$_{\multimap}$ correct	45.7	60.0	69.1
ILL$_{\multimap,\diamond,\square}$ correct *	30.4	56.9	67.1
\w type oracle	-	87.4	-

(*) in practical terms:
of sentences correctly converted to *well-typed programs*
(tagged, parsed and dependency-annotated)

paper (preprint): abs/2009.12702
code, model & data publicly available:
`https://github.com/konstantinosKokos/neural-proof-nets`