A Polynomial Time Algorithm for the Lambek Calculus with Brackets of Bounded Order

Max Kanovich, Stepan Kuznetsov, Glyn Morrill, Andre Scedrov

The Lambek Calculus $\mathbf{L}^*$

[Lambek 1958, 1961]

$$A \to A$$

$$\frac{A, \Pi \to B}{\Pi \to A \setminus B} \qquad \frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma, \Pi, (A \setminus B), \Delta \to C}$$

$$\frac{\Pi, A \to B}{\Pi \to B \,/\, A} \qquad \frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma, (B \,/\, A), \Pi, \Delta \to C}$$

$$\frac{\Gamma \to A \quad \Delta \to B}{\Gamma, \Delta \to A \cdot B} \qquad \frac{\Gamma, A, B, \Delta \to C}{\Gamma, A \cdot B, \Delta \to C}$$

$$\frac{\Pi \to A \quad \Gamma, A, \Delta \to C}{\Gamma, \Pi, \Delta \to C}\ (\mathrm{cut})$$

The Lambek Calculus $\mathbf{L}^*$

[Lambek 1958, 1961]

$$A \to A$$

$$\frac{A, \Pi \to B}{\Pi \to A \backslash B} \qquad \frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma, \Pi, (A \backslash B), \Delta \to C}$$

$$\frac{\Pi, A \to B}{\Pi \to B \,/\, A} \qquad \frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma, (B \,/\, A), \Pi, \Delta \to C}$$

$$\frac{\Gamma \to A \quad \Delta \to B}{\Gamma, \Delta \to A \cdot B} \qquad \frac{\Gamma, A, B, \Delta \to C}{\Gamma, A \cdot B, \Delta \to C}$$

$$\frac{\Pi \to A \quad \Gamma, A, \Delta \to C}{\Gamma, \Pi, \Delta \to C}\ (\mathrm{cut})$$

- A fragment of non-commutative intuitionistic linear logic.

The Lambek Calculus $\mathbf{L}^*$

[Lambek 1958, 1961]

$$A \to A$$

$$\frac{A, \Pi \to B}{\Pi \to A \backslash B} \qquad \frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma, \Pi, (A \backslash B), \Delta \to C}$$

$$\frac{\Pi, A \to B}{\Pi \to B \,/\, A} \qquad \frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma, (B \,/\, A), \Pi, \Delta \to C}$$

$$\frac{\Gamma \to A \quad \Delta \to B}{\Gamma, \Delta \to A \cdot B} \qquad \frac{\Gamma, A, B, \Delta \to C}{\Gamma, A \cdot B, \Delta \to C}$$

$$\frac{\Pi \to A \quad \Gamma, A, \Delta \to C}{\Gamma, \Pi, \Delta \to C}\ (\mathrm{cut})$$

- A fragment of non-commutative intuitionistic linear logic.
- We consider the variant of the Lambek calculus that allows empty antecedents.

Natural Language Syntax Analysis with Lambek Grammar

Natural Language Syntax Analysis with Lambek Grammar

- Mary likes John.
  $N, (N \setminus S) / N, N \to S$

Natural Language Syntax Analysis with Lambek Grammar

- Mary likes John.
  $N, (N \setminus S) / N, N \to S$

Natural Language Syntax Analysis with Lambek Grammar

- Mary likes John.
  $N, (N \setminus S) \,/\, N, N \to S$
- man who Mary likes
  $CN, (CN \setminus CN) \,/ (S \,/\, N), N, (N \setminus S) \,/\, N \to CN$

Natural Language Syntax Analysis with Lambek Grammar

- Mary likes John.
  $N, (N \setminus S) / N, N \to S$
- man who Mary likes
  $CN, (CN \setminus CN) / (S / N), N, (N \setminus S) / N \to CN$

Natural Language Syntax Analysis with Lambek Grammar

- Mary likes John.
  $N, (N \setminus S) \,/\, N, N \to S$
- man who Mary likes
  $CN, (CN \setminus CN) \,/ (S \,/\, N), N, (N \setminus S) \,/\, N \to CN$
- man who John knows Mary likes
  man who Mary knows John knows Mary likes
  ...

Natural Language Syntax Analysis with Lambek Grammar

- Mary likes John.
  $N, (N \setminus S) / N, N \to S$
- man who Mary likes
  $CN, (CN \setminus CN) / (S / N), N, (N \setminus S) / N \to CN$
- man who John knows Mary likes
  man who Mary knows John knows Mary likes
  ...
- *book which John laughed without reading
- *girl who John likes Mary and Pete likes

Natural Language Syntax Analysis with Lambek Grammar

- Mary likes John.
  $N, (N \setminus S) / N, N \to S$
- man who Mary likes
  $CN, (CN \setminus CN) / (S / N), N, (N \setminus S) / N \to CN$
- man who John knows Mary likes
  man who Mary knows John knows Mary likes
  ...
- *book which John laughed without reading
  $CN, (CN \setminus CN) / (S / CN), N, N \setminus S, ((N \setminus S) \setminus (N \setminus S)) / (N \setminus S), (N \setminus S) / N \to S$
- *girl who John likes Mary and Pete likes
  $CN, (CN \setminus CN) / (S / CN), N, (N \setminus S) / N, N, (S \setminus S) / S, N, (N \setminus S) / N \to S$

The Lambek Calculus with Brackets ($\mathbf{Lb}^*$)

[Morrill 1992, Moortgat 1995]

$$A \to A$$

$$\frac{\Pi \to A \quad \Delta(B) \to C}{\Delta(\Pi, A \setminus B) \to C} \qquad \frac{A, \Pi \to B}{\Pi \to A \setminus B} \qquad \frac{\Gamma(A, B) \to C}{\Gamma(A \cdot B) \to C}$$

$$\frac{\Pi \to A \quad \Delta(B) \to C}{\Delta(B \,/\, A, \Pi) \to C} \qquad \frac{\Pi, A \to B}{\Pi \to B \,/\, A} \qquad \frac{\Gamma \to A \quad \Delta \to B}{\Gamma, \Delta \to A \cdot B}$$

$$\frac{\Delta([A]) \to C}{\Delta(\langle\rangle A) \to C} \qquad \frac{\Pi \to A}{[\Pi] \to \langle\rangle A} \qquad \frac{\Delta(A) \to C}{\Delta([[]^{-1} A]) \to C} \qquad \frac{[\Pi] \to A}{\Pi \to []^{-1} A}$$

The Lambek Calculus with Brackets ($\mathbf{Lb}^*$)

[Morrill 1992, Moortgat 1995]

$$A \to A$$

$$\frac{\Pi \to A \quad \Delta(B) \to C}{\Delta(\Pi, A \setminus B) \to C} \qquad \frac{A, \Pi \to B}{\Pi \to A \setminus B} \qquad \frac{\Gamma(A, B) \to C}{\Gamma(A \cdot B) \to C}$$

$$\frac{\Pi \to A \quad \Delta(B) \to C}{\Delta(B \,/\, A, \Pi) \to C} \qquad \frac{\Pi, A \to B}{\Pi \to B \,/\, A} \qquad \frac{\Gamma \to A \quad \Delta \to B}{\Gamma, \Delta \to A \cdot B}$$

$$\frac{\Delta([A]) \to C}{\Delta(\langle\rangle A) \to C} \qquad \frac{\Pi \to A}{[\Pi] \to \langle\rangle A} \qquad \frac{\Delta(A) \to C}{\Delta([[]^{-1}A]) \to C} \qquad \frac{[\Pi] \to A}{\Pi \to []^{-1}A}$$

- Brackets introduce *controlled non-associativity.*

The Lambek Calculus with Brackets (**Lb***)

[Morrill 1992, Moortgat 1995]

$$A \to A$$

$$\frac{\Pi \to A \quad \Delta(B) \to C}{\Delta(\Pi, A \setminus B) \to C} \qquad \frac{A, \Pi \to B}{\Pi \to A \setminus B} \qquad \frac{\Gamma(A, B) \to C}{\Gamma(A \cdot B) \to C}$$

$$\frac{\Pi \to A \quad \Delta(B) \to C}{\Delta(B \,/\, A, \Pi) \to C} \qquad \frac{\Pi, A \to B}{\Pi \to B \,/\, A} \qquad \frac{\Gamma \to A \quad \Delta \to B}{\Gamma, \Delta \to A \cdot B}$$

$$\frac{\Delta([A]) \to C}{\Delta(\langle\rangle A) \to C} \qquad \frac{\Pi \to A}{[\Pi] \to \langle\rangle A} \qquad \frac{\Delta(A) \to C}{\Delta([[]^{-1} A]) \to C} \qquad \frac{[\Pi] \to A}{\Pi \to []^{-1} A}$$

- Brackets introduce *controlled non-associativity.*
- Cut elimination proved by Moortgat [1996].

Islands: Blocking Unwanted Derivations Using Brackets

Islands: Blocking Unwanted Derivations Using Brackets

- book which John laughed without reading

Islands: Blocking Unwanted Derivations Using Brackets

- book which John laughed [without reading]

Islands: Blocking Unwanted Derivations Using Brackets

- book which John laughed [without reading]

$$CN, (CN \backslash CN) / (S / CN), N, N \backslash S, [[]^{-1}((N \backslash S) \backslash (N \backslash S)) / (N \backslash S), (N \backslash S) / N] \to S$$

Islands: Blocking Unwanted Derivations Using Brackets

- book which John laughed [without reading]

$CN, (CN \setminus CN) / (S / CN), N, N \setminus S, [[]^{-1}((N \setminus S) \setminus (N \setminus S)) / (N \setminus S), (N \setminus S) / N] \to S$

This sequent is not derivable in **Lb***.

Islands: Blocking Unwanted Derivations Using Brackets

- book which John laughed [without reading]

$CN, (CN \setminus CN) / (S / CN), N, N \setminus S, [[]^{-1}((N \setminus S) \setminus (N \setminus S)) / (N \setminus S), (N \setminus S) / N] \to S$

  This sequent is not derivable in **Lb***.

- girl who John likes Mary and Pete likes

Islands: Blocking Unwanted Derivations Using Brackets

- book which John laughed [without reading]

$CN, (CN \setminus CN) / (S / CN), N, N \setminus S, [[]^{-1}((N \setminus S) \setminus (N \setminus S)) / (N \setminus S), (N \setminus S) / N] \to S$

  This sequent is not derivable in $\mathbf{Lb}^*$.

- girl who [John likes Mary and Pete likes]

Islands: Blocking Unwanted Derivations Using Brackets

- book which John laughed [without reading]

$$CN, (CN \setminus CN) / (S / CN), N, N \setminus S, [[]^{-1}((N \setminus S) \setminus (N \setminus S)) / (N \setminus S), (N \setminus S) / N] \to S$$

  This sequent is not derivable in **Lb***.

- girl who [John likes Mary and Pete likes]

$$CN, (CN \setminus CN) / (S / CN), [N, (N \setminus S) / N, N, (S \setminus []^{-1} S) / S, N, (N \setminus S) / N] \to S$$

Islands: Blocking Unwanted Derivations Using Brackets

- book which John laughed [without reading]

$CN, (CN \setminus CN) / (S / CN), N, N \setminus S, [[]^{-1}((N \setminus S) \setminus (N \setminus S)) / (N \setminus S), (N \setminus S) / N] \to S$

  This sequent is not derivable in $\mathbf{Lb}^*$.

- girl who [John likes Mary and Pete likes]

$CN, (CN \setminus CN) / (S / CN), [N, (N \setminus S) / N, N, (S \setminus []^{-1} S) / S, N, (N \setminus S) / N] \to S$

  Neither is this one.

Complexity

Complexity

- Derivability problem in $\mathbf{L}^*$ (and therefore in $\mathbf{Lb}^*$) is NP-complete [Pentus 2006].

Complexity

- Derivability problem in $\mathbf{L}^*$ (and therefore in $\mathbf{Lb}^*$) is NP-complete [Pentus 2006].
- Non-associative Lambek calculus (**NL**) is polynomially decidable [Aarts and Trautwein 1995].

Complexity

- Derivability problem in $\mathbf{L}^*$ (and therefore in $\mathbf{Lb}^*$) is NP-complete [Pentus 2006].
- Non-associative Lambek calculus (**NL**) is polynomially decidable [Aarts and Trautwein 1995].
- **NL** is also embedded into $\mathbf{Lb}^*$ [Kurtonina 1995].

Complexity

- Derivability problem in $\mathbf{L}^*$ (and therefore in $\mathbf{Lb}^*$) is NP-complete [Pentus 2006].
- Non-associative Lambek calculus (**NL**) is polynomially decidable [Aarts and Trautwein 1995].
- **NL** is also embedded into $\mathbf{Lb}^*$ [Kurtonina 1995].
- There exists an algorithm for $\mathbf{L}^*$ with running time $\mathrm{poly}(N, 2^R)$ [Pentus 2010, Fowler 2009], where N is the size of the sequent and R is the *order (depth)*.
  In linguistic applications, R is small.

Complexity

- Derivability problem in $\mathbf{L}^*$ (and therefore in $\mathbf{Lb}^*$) is NP-complete [Pentus 2006].
- Non-associative Lambek calculus (**NL**) is polynomially decidable [Aarts and Trautwein 1995].
- **NL** is also embedded into $\mathbf{Lb}^*$ [Kurtonina 1995].
- There exists an algorithm for $\mathbf{L}^*$ with running time $\mathrm{poly}(N, 2^R)$ [Pentus 2010, Fowler 2009], where N is the size of the sequent and R is the *order (depth)*.
  In linguistic applications, R is small.
- Pentus' algorithm extended to $\mathbf{Lb}^*$, running time $\mathrm{poly}(N, 2^R, N^B)$ **[this talk].**
  Here B is the bracket nesting depth.

Complexity Parameters

Complexity Parameters

- $N = \|\Pi \to A\|$, counted as the total number of connectives in the sequent.

Complexity Parameters

- $N = \|\Pi \to A\|$, counted as the total number of connectives in the sequent.
- $R = \mathrm{ord}(\Pi \to A) = \max\{\mathrm{ord}(\Pi) + 1, \mathrm{ord}(A) + \mathrm{prod}(A)\}$, where $\mathrm{prod}(A)$ is 1 if $A = A_1 \cdot A_2$, and 0 otherwise.
  $\mathrm{ord}(p_i) = 0$; $\mathrm{ord}(A \cdot B) = \max\{\mathrm{ord}(A), \mathrm{ord}(B)\}$;
  $\mathrm{ord}(A \setminus B) = \mathrm{ord}(B \,/\, A) = \max\{\mathrm{ord}(A) + 1, \mathrm{ord}(B) + \mathrm{prod}(B)\}$;
  $\mathrm{ord}(\Lambda) = 0$; $\mathrm{ord}(\Gamma, \Delta) = \max\{\mathrm{ord}(\Gamma), \mathrm{ord}(\Delta)\}$.

Complexity Parameters

- $N = \|\Pi \to A\|$, counted as the total number of connectives in the sequent.
- $R = \mathrm{ord}(\Pi \to A) = \max\{\mathrm{ord}(\Pi) + 1, \mathrm{ord}(A) + \mathrm{prod}(A)\}$, where $\mathrm{prod}(A)$ is 1 if $A = A_1 \cdot A_2$, and 0 otherwise.
  $\mathrm{ord}(p_i) = 0$; $\mathrm{ord}(A \cdot B) = \max\{\mathrm{ord}(A), \mathrm{ord}(B)\}$;
  $\mathrm{ord}(A \setminus B) = \mathrm{ord}(B \,/\, A) = \max\{\mathrm{ord}(A) + 1, \mathrm{ord}(B) + \mathrm{prod}(B)\}$;
  $\mathrm{ord}(\Lambda) = 0$; $\mathrm{ord}(\Gamma, \Delta) = \max\{\mathrm{ord}(\Gamma), \mathrm{ord}(\Delta)\}$.
  If we translate Lambek formulae into linear logic, R is the maximal alternation depth of $⅋$'s and $\otimes$'s.

Complexity Parameters

- $N = \|\Pi \to A\|$, counted as the total number of connectives (incl. bracket modalities) and brackets in the sequent.
- $R = \mathrm{ord}(\Pi \to A) = \max\{\mathrm{ord}(\Pi) + 1, \mathrm{ord}(A) + \mathrm{prod}(A)\}$, where $\mathrm{prod}(A)$ is 1 if $A = A_1 \cdot A_2$ or $\langle\rangle A'$, and 0 otherwise.
  $\mathrm{ord}(p_i) = 0$; $\mathrm{ord}(A \cdot B) = \max\{\mathrm{ord}(A), \mathrm{ord}(B)\}$;
  $\mathrm{ord}(A \setminus B) = \mathrm{ord}(B \,/\, A) = \max\{\mathrm{ord}(A) + 1, \mathrm{ord}(B) + \mathrm{prod}(B)\}$;
  $\mathrm{ord}(\langle\rangle A) = \mathrm{ord}(A)$; $\mathrm{ord}([]^{-1} A) = \max\{\mathrm{ord}(A) + \mathrm{prod}(A), 1\}$;
  $\mathrm{ord}(\Lambda) = 0$; $\mathrm{ord}(\Gamma, \Delta) = \max\{\mathrm{ord}(\Gamma), \mathrm{ord}(\Delta)\}$.

Complexity Parameters

- $N = \|\Pi \to A\|$, counted as the total number of connectives (incl. bracket modalities) and brackets in the sequent.
- $R = \mathrm{ord}(\Pi \to A) = \max\{\mathrm{ord}(\Pi) + 1, \mathrm{ord}(A) + \mathrm{prod}(A)\}$, where $\mathrm{prod}(A)$ is 1 if $A = A_1 \cdot A_2$ or $\langle\rangle A'$, and 0 otherwise. $\mathrm{ord}(p_i) = 0$; $\mathrm{ord}(A \cdot B) = \max\{\mathrm{ord}(A), \mathrm{ord}(B)\}$; $\mathrm{ord}(A \setminus B) = \mathrm{ord}(B \mathbin{/} A) = \max\{\mathrm{ord}(A) + 1, \mathrm{ord}(B) + \mathrm{prod}(B)\}$; $\mathrm{ord}(\langle\rangle A) = \mathrm{ord}(A)$; $\mathrm{ord}([]^{-1} A) = \max\{\mathrm{ord}(A) + \mathrm{prod}(A), 1\}$; $\mathrm{ord}(\Lambda) = 0$; $\mathrm{ord}(\Gamma, \Delta) = \max\{\mathrm{ord}(\Gamma), \mathrm{ord}(\Delta)\}$.
- B is the bracket and bracket modalities nesting depth.

So... Why This Talk?

So... Why This Talk?

- Practical motivation: aim to optimise parsers for type-logical grammar (CatLog by Morrill et al.)

So... Why This Talk?

- Practical motivation: aim to optimise parsers for type-logical grammar (CatLog by Morrill et al.)
- Theoretical interest: a combination of proof net and finite automata techniques

Proof Nets for **Lb***

Mary danced before singing

Proof Nets for **Lb***

[Mary] danced [before singing]

Proof Nets for **Lb***

[Mary] danced [before singing]

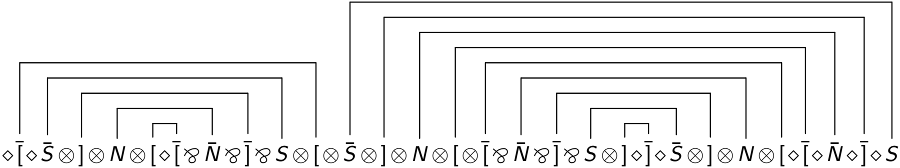
$$[N], \langle\rangle N \backslash S, [[]^{-1}((\langle\rangle N \backslash S) \backslash (\langle\rangle N \backslash S)) / (\langle\rangle N \backslash S), \langle\rangle N \backslash S] \to S$$

Proof Nets for **Lb***

[Mary] danced [before singing]

$$[N], \langle\rangle N \backslash S, [[]^{-1}((\langle\rangle N \backslash S) \backslash (\langle\rangle N \backslash S)) / (\langle\rangle N \backslash S), \langle\rangle N \backslash S] \rightarrow S$$

Axiom links:



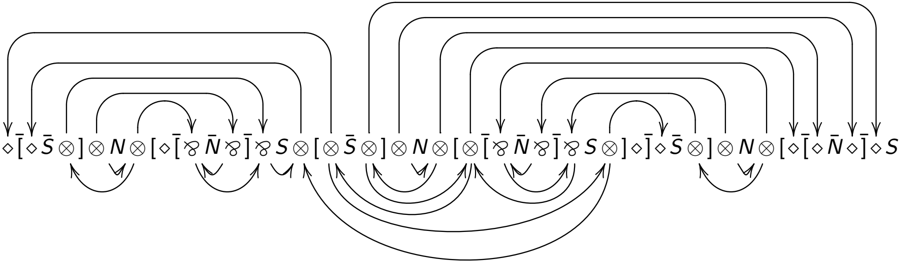
$$\diamond \bar{[} \diamond \bar{S} \otimes] \otimes N \otimes [\diamond \bar{[} \parr \bar{N} \parr \bar{]} \parr S \otimes [\otimes \bar{S} \otimes] \otimes N \otimes [\otimes \bar{[} \parr \bar{N} \parr \bar{]} \parr S \otimes] \diamond \bar{]} \diamond \bar{S} \otimes] \otimes N \otimes [\diamond \bar{[} \diamond \bar{N} \diamond \bar{]} \diamond S$$

Proof Nets for **Lb***

[Mary] danced [before singing]

$$[N], \langle\rangle N \setminus S, [[]^{-1}((\langle\rangle N \setminus S) \setminus (\langle\rangle N \setminus S)) / (\langle\rangle N \setminus S), \langle\rangle N \setminus S] \rightarrow S$$

Acyclicity [Pentus 1998]:

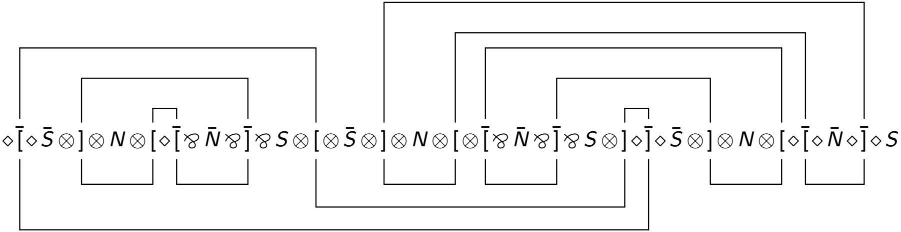


Proof Nets for **Lb***

[Mary] danced [before singing]

$$[N], \langle\rangle N \backslash S, [[]^{-1}((\langle\rangle N \backslash S) \backslash (\langle\rangle N \backslash S)) / (\langle\rangle N \backslash S), \langle\rangle N \backslash S] \rightarrow S$$

Sisterhood [Fadda and Morrill 2005]:



Proof Nets for **Lb***

- Extending Pentus-style proof nets for cyclic multiplicative linear logic.

Proof Nets for **Lb***

- Extending Pentus-style proof nets for cyclic multiplicative linear logic.
- Axiom links ($\mathcal{E}$) connect atoms (p_i and $\bar{p}_i$).

Proof Nets for **Lb***

- Extending Pentus-style proof nets for cyclic multiplicative linear logic.
- Axiom links ($\mathcal{E}$) connect atoms (p_i and $\bar{p}_i$).
- Brackets are considered as a special kind of atoms: $[$, $]$, $\bar{[}$, $\bar{]}$.

Proof Nets for **Lb***

- Extending Pentus-style proof nets for cyclic multiplicative linear logic.
- Axiom links ($\mathcal{E}$) connect atoms (p_i and $\bar{p}_i$).
- Brackets are considered as a special kind of atoms: $[$, $]$, $\bar{[}$, $\bar{]}$.
- Acyclicity condition: $\prec$ is the syntactic forest relation, $\mathcal{A}$ connects each $\otimes$ to the (unique) $\parr$ in the same $\mathcal{E}$-region; $\mathcal{A} \cup \prec$ should be acyclic.

Proof Nets for **Lb***

- Extending Pentus-style proof nets for cyclic multiplicative linear logic.
- Axiom links ($\mathcal{E}$) connect atoms (p_i and $\bar{p}_i$).
- Brackets are considered as a special kind of atoms: $[$, $]$, $\bar{[}$, $\bar{]}$.
- Acyclicity condition: $\prec$ is the syntactic forest relation, $\mathcal{A}$ connects each $\otimes$ to the (unique) $\parr$ in the same $\mathcal{E}$-region; $\mathcal{A} \cup \prec$ should be acyclic.
- Sisterhood condition: $\mathcal{E}$ should respect pairing of brackets.

Importance of the Sisterhood Condition

— Could one just treat brackets as atoms? (cf. [Versmissen 1996])

Importance of the Sisterhood Condition

— Could one just treat brackets as atoms? (cf. [Versmissen 1996])
— **No.**

Importance of the Sisterhood Condition

— Could one just treat brackets as atoms? (cf. [Versmissen 1996])
— **No.**

Counter-example [Fadda and Morrill 2005]:

$$[[]^{-1}p], [[]^{-1}q] \rightarrow \langle\rangle[]^{-1}(p \cdot q)$$

Importance of the Sisterhood Condition

— Could one just treat brackets as atoms? (cf. [Versmissen 1996])
— **No.**

Counter-example [Fadda and Morrill 2005]:

$$[[]^{-1}p], [[]^{-1}q] \to \langle\rangle[]^{-1}(p \cdot q)$$

- This sequent is not derivable in $\mathbf{Lb}^*$.

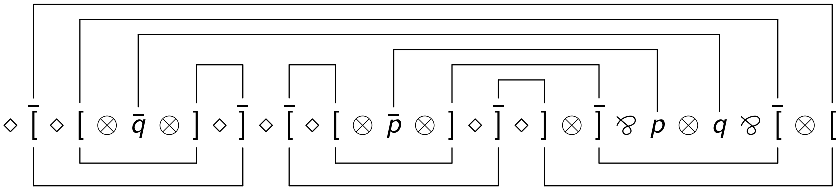
Importance of the Sisterhood Condition

— Could one just treat brackets as atoms? (cf. [Versmissen 1996])
— **No.**

Counter-example [Fadda and Morrill 2005]:

$$[[]^{-1}p], [[]^{-1}q] \rightarrow \langle\rangle[]^{-1}(p \cdot q)$$

- This sequent is not derivable in $\mathbf{Lb}^*$.
- The only possible proof net violates sisterhood:



Encoding Proof Nets

- $\mathcal{E} \rightsquigarrow c(\mathcal{E}) \in \{c_1, \ldots, c_n\}^n$:
  if the i-th and the j-th atoms are connected by an axiom link, then the i-th letter of $c(\mathcal{E})$ is e_j and the j-th letter is e_i.

Encoding Proof Nets

- $\mathcal{E} \rightsquigarrow c(\mathcal{E}) \in \{c_1, \ldots, c_n\}^n$:
  if the i-th and the j-th atoms are connected by an axiom link, then the i-th letter of $c(\mathcal{E})$ is e_j and the j-th letter is e_i.
- $P_1 = \{c(\mathcal{E}) \mid \mathcal{E} \text{ is a proof net, but maybe violating sisterhood}\}$

Encoding Proof Nets

- $\mathcal{E} \rightsquigarrow c(\mathcal{E}) \in \{c_1, \ldots, c_n\}^n$:
  if the i-th and the j-th atoms are connected by an axiom link, then the i-th letter of $c(\mathcal{E})$ is e_j and the j-th letter is e_i.
- $P_1 = \{c(\mathcal{E}) \mid \mathcal{E} \text{ is a proof net, but maybe violating sisterhood}\}$
  There exists, and can be effectively constructed, a $\mathrm{poly}(n, 2^R)$-size context free grammar for P_1 [Pentus 2010].

Encoding Proof Nets

- $\mathcal{E} \rightsquigarrow c(\mathcal{E}) \in \{c_1, \dots, c_n\}^n$:
  if the i-th and the j-th atoms are connected by an axiom link, then the i-th letter of $c(\mathcal{E})$ is e_j and the j-th letter is e_i.
- $P_1 = \{c(\mathcal{E}) \mid \mathcal{E}$ is a proof net, but maybe violating sisterhood$\}$
  There exists, and can be effectively constructed, a $\mathrm{poly}(n, 2^R)$-size context free grammar for P_1 [Pentus 2010].
- For a language P_2, such that $c(\mathcal{E}) \in P_2$ iff $\mathcal{E}$ respects sisterhood, there exists, and can be effectively constructed, a $\mathrm{poly}(n, n^B)$-size finite automaton for P_2.

Encoding Proof Nets

- $\mathcal{E} \rightsquigarrow c(\mathcal{E}) \in \{c_1, \ldots, c_n\}^n$:
  if the i-th and the j-th atoms are connected by an axiom link, then the i-th letter of $c(\mathcal{E})$ is e_j and the j-th letter is e_i.
- $P_1 = \{c(\mathcal{E}) \mid \mathcal{E}$ is a proof net, but maybe violating sisterhood$\}$
  There exists, and can be effectively constructed, a $\mathrm{poly}(n, 2^R)$-size context free grammar for P_1 [Pentus 2010].
- For a language P_2, such that $c(\mathcal{E}) \in P_2$ iff $\mathcal{E}$ respects sisterhood, there exists, and can be effectively constructed, a $\mathrm{poly}(n, n^B)$-size finite automaton for P_2.
- There exists, and can be effectively constructed, a polynomial-size context free grammar for $P_1 \cap P_2$ (see [Ginsburg 1966]).

Encoding Proof Nets

- $\mathcal{E} \rightsquigarrow c(\mathcal{E}) \in \{c_1, \ldots, c_n\}^n$:
  if the i-th and the j-th atoms are connected by an axiom link, then the i-th letter of $c(\mathcal{E})$ is e_j and the j-th letter is e_i.
- $P_1 = \{c(\mathcal{E}) \mid \mathcal{E}$ is a proof net, but maybe violating sisterhood$\}$
  There exists, and can be effectively constructed, a $\mathrm{poly}(n, 2^R)$-size context free grammar for P_1 [Pentus 2010].
- For a language P_2, such that $c(\mathcal{E}) \in P_2$ iff $\mathcal{E}$ respects sisterhood, there exists, and can be effectively constructed, a $\mathrm{poly}(n, n^B)$-size finite automaton for P_2.
- There exists, and can be effectively constructed, a polynomial-size context free grammar for $P_1 \cap P_2$ (see [Ginsburg 1966]).
- The non-emptiness of $P_1 \cap P_2$ (our goal) is checked in polynomial time.

Encoding Proof Nets

- $\mathcal{E} \rightsquigarrow c(\mathcal{E}) \in \{c_1, \ldots, c_n\}^n$:
  if the i-th and the j-th atoms are connected by an axiom link, then the i-th letter of $c(\mathcal{E})$ is e_j and the j-th letter is e_i.
- $P_1 = \{c(\mathcal{E}) \mid \mathcal{E}$ is a proof net, but maybe violating sisterhood$\}$
  There exists, and can be effectively constructed, a $\mathrm{poly}(n, 2^R)$-size context free grammar for P_1 [Pentus 2010].
- For a language P_2, such that $c(\mathcal{E}) \in P_2$ iff $\mathcal{E}$ respects sisterhood, there exists, and can be effectively constructed, a $\mathrm{poly}(n, n^B)$-size finite automaton for P_2.
- There exists, and can be effectively constructed, a polynomial-size context free grammar for $P_1 \cap P_2$ (see [Ginsburg 1966]).
- The non-emptiness of $P_1 \cap P_2$ (our goal) is checked in polynomial time.
- Notice that P_1 and P_2 are *finite,* so $P_1 \cap P_2$ is trivially context free. The real achievement is *polynomiality* of the grammar.

Future Work

Future Work

- Develop an efficient parsing procedure for $\mathbf{Lb}^*$-*grammars* (cf. [Pentus 2010] for $\mathbf{L}^*$).

Future Work

- Develop an efficient parsing procedure for $\mathbf{Lb}^*$-*grammars* (cf. [Pentus 2010] for $\mathbf{L}^*$).
  In a Lambek grammar, several types can be assigned to one lexeme: an extra level of non-determinism.

Future Work

- Develop an efficient parsing procedure for $\mathbf{Lb}^*$-*grammars* (cf. [Pentus 2010] for $\mathbf{L}^*$).
  In a Lambek grammar, several types can be assigned to one lexeme: an extra level of non-determinism.
- The problem whether $\mathbf{Lb}^*$-grammars are context free is still open: Jäger [2003] uses the incorrect Versmissen's lemma. Cf. $\mathbf{L}^*$-grammars are context free [Pentus 1993].

Future Work

- Develop an efficient parsing procedure for $\mathbf{Lb}^*$-*grammars* (cf. [Pentus 2010] for $\mathbf{L}^*$).
  In a Lambek grammar, several types can be assigned to one lexeme: an extra level of non-determinism.
- The problem whether $\mathbf{Lb}^*$-grammars are context free is still open: Jäger [2003] uses the incorrect Versmissen's lemma. Cf. $\mathbf{L}^*$-grammars are context free [Pentus 1993].
- **Lb**: the Lambek calculus with brackets and non-empty antecedents.

Future Work

- Develop an efficient parsing procedure for $\mathbf{Lb}^*$-*grammars* (cf. [Pentus 2010] for $\mathbf{L}^*$).
  In a Lambek grammar, several types can be assigned to one lexeme: an extra level of non-determinism.
- The problem whether $\mathbf{Lb}^*$-grammars are context free is still open: Jäger [2003] uses the incorrect Versmissen's lemma. Cf. $\mathbf{L}^*$-grammars are context free [Pentus 1993].
- **Lb**: the Lambek calculus with brackets and non-empty antecedents.
- Feasible fragments of other enrichments of the Lambek calculus (even generally undecidable).

Thank you!