

# An Agda Formalization of Nonassociative Lambek Calculus

Niccolò Veltri   Cheng-Syuan Wan

Tallinn University of Technology

TABLEAUX, Reykjavík, 2025

# Associative Lambek calculus

Formulae of associative Lambek calculus are inductively generated by the grammar:

$$A, B ::= X \mid A \Rightarrow B \mid B \Leftarrow A \mid A \otimes B$$

Sequents in associative Lambek calculus are of the form

$$\Gamma \vdash C$$

with list of formulae as antecedents.

# Associative Lambek calculus

Formulae of associative Lambek calculus are inductively generated by the grammar:

$$A, B ::= X \mid A \Rightarrow B \mid B \Leftarrow A \mid A \otimes B$$

Sequents in associative Lambek calculus are of the form

$$\Gamma \vdash C$$

with list of formulae as antecedents.

Derivations are generated inductively by the following rules:

$$\frac{}{A \vdash A} \text{ ax}$$

$$\frac{A, \Gamma \vdash B}{\Gamma \vdash A \Rightarrow B} \Rightarrow R \qquad \frac{\Delta \vdash A \quad \Gamma_0, B, \Gamma_1 \vdash C}{\Gamma_0, \Delta, A \Rightarrow B, \Gamma_1 \vdash C} \Rightarrow L$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash B \Leftarrow A} \Leftarrow R \qquad \frac{\Delta \vdash A \quad \Gamma_0, B, \Gamma_1 \vdash C}{\Gamma_0, B \Leftarrow A, \Delta, \Gamma_1 \vdash C} \Leftarrow L$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \otimes R \qquad \frac{\Gamma_0, A, B, \Gamma_1 \vdash C}{\Gamma_0, A \otimes B, \Gamma_1 \vdash C} \otimes L$$

# Cut admissibility for associative Lambek calculus

$$\frac{\Delta \vdash A \quad \Gamma_0, A, \Gamma_1 \vdash C}{\Gamma_0, \Delta, \Gamma_1 \vdash C} \text{ cut}$$

Proof proceeds by induction on height of derivations and complexity of cut formulae. In some cases, we have to check the relative positions of the cut formula and the principal formula of a left rule.

# Cut admissibility for associative Lambek calculus

$$\frac{\Delta \vdash A \quad \Gamma_0, A, \Gamma_1 \vdash C}{\Gamma_0, \Delta, \Gamma_1 \vdash C} \text{ cut}$$

Proof proceeds by induction on height of derivations and complexity of cut formulae. In some cases, we have to check the relative positions of the cut formula and the principal formula of a left rule.

For example, consider the derivation:

$$\frac{\Delta \vdash A \quad \frac{\frac{\Lambda_1 \vdash A' \quad \Lambda_0, B', \Lambda_1 \vdash C}{\Lambda_0, \Lambda_1, A' \Rightarrow B', \Lambda_2 \vdash C} \Rightarrow L}{\Gamma_0, \Delta, \Gamma_1 \vdash C} \text{ cut}$$

which tells us an equation of two lists:  $\Gamma_0, A, \Gamma_1 = \Lambda_0, \Lambda_1, A' \Rightarrow B', \Lambda_2$ . There are four possibilities of the equation,

1.  $A = A' \Rightarrow B'$ ,
2.  $A$  is in  $\Lambda_0$ ,
3.  $A$  is in  $\Lambda_1$ , or
4.  $A$  is in  $\Lambda_2$ .

# Nonassociative Lambek calculus (NL)

In NL, sequents are of the form

$$T \vdash C$$

with binary trees as antecedents.

# Nonassociative Lambek calculus (NL)

In NL, sequents are of the form

$$T \vdash C$$

with binary trees as antecedents.

Trees are defined inductively by the grammar  $T, U ::= A \mid (T, U)$  where  $A$  is a single formula.

# Nonassociative Lambek calculus (NL)

In NL, sequents are of the form

$$T \vdash C$$

with binary trees as antecedents.

Trees are defined inductively by the grammar  $T, U ::= A \mid (T, U)$  where  $A$  is a single formula.

Contexts are trees with a hole, inductively specified by the grammar  $\mathcal{C} ::= [\bullet] \mid (\mathcal{C}, T) \mid (T, \mathcal{C})$ .



# Nonassociative Lambek calculus (NL)

In NL, sequents are of the form

$$T \vdash C$$

with binary trees as antecedents.

Trees are defined inductively by the grammar  $T, U ::= A \mid (T, U)$  where  $A$  is a single formula.

Contexts are trees with a hole, inductively specified by the grammar  $\mathcal{C} ::= [\bullet] \mid (\mathcal{C}, T) \mid (T, \mathcal{C})$ .

Substitution  $\mathcal{C}[U]$  of a tree  $U$  into a hole of a context  $\mathcal{C}$  is defined by structural recursion on  $\mathcal{C}$ :

$$\begin{aligned} [\bullet][U] &= U \\ (\mathcal{C}, V)[U] &= (\mathcal{C}[U], V) \\ (V, \mathcal{C})[U] &= (V, \mathcal{C}[U]) \end{aligned}$$

# Nonassociative Lambek calculus (NL)

Derivations are generated inductively by the following rules:

$$\frac{}{A \vdash A} \text{ax}$$

$$\frac{A, T \vdash B}{T \vdash A \Rightarrow B} \Rightarrow R \qquad \frac{U \vdash A \quad C[B] \vdash C}{C[U, A \Rightarrow B] \vdash C} \Rightarrow L$$

$$\frac{T, A \vdash B}{T \vdash B \Leftarrow A} \Leftarrow R \qquad \frac{U \vdash A \quad C[B] \vdash C}{C[B \Leftarrow A, U] \vdash C} \Leftarrow L$$

$$\frac{T \vdash A \quad U \vdash B}{T, U \vdash A \otimes B} \otimes R \qquad \frac{C[A, B] \vdash C}{C[A \otimes B] \vdash C} \otimes L$$

## Cut admissibility for NL

$$\frac{U \vdash A \quad \mathcal{C}[A] \vdash C}{\mathcal{C}[U] \vdash C} \text{ cut}$$

Proof proceeds similarly to the case of associative Lambek calculus.

## Cut admissibility for NL

$$\frac{U \vdash A \quad \mathcal{C}[A] \vdash C}{\mathcal{C}[U] \vdash C} \text{ cut}$$

Proof proceeds similarly to the case of associative Lambek calculus.  
For instance, consider the following derivation:

$$\frac{U \vdash A \quad \frac{V \overset{g}{\vdash} A' \quad C' \overset{h}{\vdash} B'}{C'[V, A' \Rightarrow B'] \vdash C} \Rightarrow L}{\mathcal{C}[U] \vdash C} \text{ cut}$$

which tells us an equation  $\mathcal{C}[A] = \mathcal{C}'[V, A' \Rightarrow B']$  and we need to figure out all possibilities of the equation.

# Cases of the equality of trees

In general, we have to distinguish all possible cases of  $\mathcal{C}_1[U_1] = \mathcal{C}_2[U_2]$ .

# Cases of the equality of trees

In general, we have to distinguish all possible cases of  $\mathcal{C}_1[U_1] = \mathcal{C}_2[U_2]$ .  
The cases split into 3 groups and in total 7 cases.

- ▶  $U_1 = U_2$  and  $\mathcal{C}_1 = \mathcal{C}_2$  (1).
- ▶ One tree contains another (4).
- ▶ Two trees are disjoint (2).

The proof of cut admissibility for NL is achievable on pen and paper. However, when dealing with more complicated properties, e.g. the associativity and commutativity of cut:

The proof of cut admissibility for NL is achievable on pen and paper. However, when dealing with more complicated properties, e.g. the associativity and commutativity of cut:

$$\frac{\frac{V \vdash^f D \quad C'[D] \vdash^g E}{C'[V] \vdash E} \text{ cut} \quad C[E] \vdash^h C}{C[C'[V]] \vdash C} \text{ cut} = \frac{V \vdash^f D \quad \frac{C'[D] \vdash^g E \quad C[E] \vdash^h C}{C[C'[D]] \vdash C} \text{ cut}}{C[C'[V]] \vdash C} \text{ cut}$$

$$\frac{T \vdash^f D \quad \frac{U \vdash^g E \quad C[C_1[D], C_2[E]] \vdash^h C}{C[C_1[D], C_2[U]] \vdash C} \text{ cut}}{C[C_1[T], C_2[U]] \vdash C} \text{ cut} = \frac{U \vdash^g E \quad \frac{T \vdash^f D \quad C[C_1[D], C_2[E]] \vdash^h C}{C[C_1[T], C_2[E]] \vdash C} \text{ cut}}{C[C_1[T], C_2[U]] \vdash C} \text{ cut}$$

The number of cases increases drastically, which is not easy to check on pen and paper.

Therefore, we use the proof assistant Agda to help us do the proofs carefully.



# Base of the formalization

## ► trees

data Tree : Set where

- : Tree
- $\eta$  : Fma  $\rightarrow$  Tree
- $-, -$  : Tree  $\rightarrow$  Tree  $\rightarrow$  Tree

# Base of the formalization

## ► trees

```
data Tree : Set where
  •       : Tree
  η       : Fma → Tree
  -, -    : Tree → Tree → Tree
```

## ► path in a tree

```
data Path : Tree → Set where
  •       : Path •
  _◀_     : ∀ { T } (p : Path T) U → Path (T , U)
  _▶_     : ∀ T { U } (p : Path U) → Path (T , U)
```

# Base of the formalization

## ► trees

data Tree : Set where  
    • : Tree  
     $\eta$  : Fma  $\rightarrow$  Tree  
     $-, -$  : Tree  $\rightarrow$  Tree  $\rightarrow$  Tree

## ► path in a tree

data Path : Tree  $\rightarrow$  Set where  
    • : Path •  
     $-\blacktriangleleft-$  :  $\forall \{T\} (p : \text{Path } T) U \rightarrow \text{Path } (T, U)$   
     $-\blacktriangleright-$  :  $\forall T \{U\} (p : \text{Path } U) \rightarrow \text{Path } (T, U)$

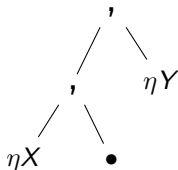
## ► substitution

sub :  $\forall \{T\} \rightarrow \text{Path } T \rightarrow \text{Tree} \rightarrow \text{Tree}$   
sub •  $U$  =  $U$   
sub ( $p \blacktriangleleft V$ )  $U$  = sub  $p$   $U, V$   
sub ( $V \blacktriangleright p$ )  $U$  =  $V, \text{sub } p U$

# Example

## Example

Consider the tree  $T = (\eta X, \bullet), \eta Y$ , which contains a single hole. The path to the hole is  $p = (\eta X \blacktriangleright \bullet) \blacktriangleleft \eta Y$ , which indicates that, starting from the root node, we take one step to the left followed by one step to the right, after which we reach the hole.



# Formalizing the case distinction

In Agda,  $\mathcal{C}_1[U_1] = \mathcal{C}_2[U_2]$  is written as  $\text{sub } p_1 \ U_1 \equiv \text{sub } p_2 \ U_2$ . The 7 cases are formalized as the following datatype:

```
data SubEq (p1 : Path C1) (p2 : Path C2) (U1 U2 : Tree) : Set where
  case1  : Same p1 p2 U1 U2    → SubEq p2 p1 U1 U2
  case2  : ∈Left p1 p2 U1 U2    → SubEq p2 p1 U1 U2
  case3  : ∈Right p1 p2 U1 U2   → SubEq p2 p1 U1 U2
  case4  : ∈Left p2 p1 U2 U1    → SubEq p2 p1 U1 U2
  case5  : ∈Right p2 p1 U2 U1   → SubEq p2 p1 U1 U2
  case6  : Disj p1 p2 U1 U2     → SubEq p2 p1 U1 U2
  case7  : Disj p2 p1 U2 U1     → SubEq p2 p1 U1 U2
```

This datatype is correct because for any  $U_1, U_2$  and  $p_1, p_2$ ,  $\text{SubEq } p_1 \ p_2 \ U_1 \ U_2$  is equivalent to  $\text{sub } p_1 \ U_1 \equiv \text{sub } p_2 \ U_2$ .

I will discuss two cases:

- ▶ case<sub>1</sub> : Same  $p_1$   $p_2$   $U_1$   $U_2$  and
- ▶ case<sub>2</sub> :  $\in \text{Left } p_1$   $p_2$   $U_1$   $U_2$ .

case<sub>1</sub> : Same  $p_1$   $p_2$   $U_1$   $U_2$

$U_1$  is equal to  $U_2$

In this case,  $C_1$  must be equal to  $C_2$  and  $p_1$  and  $p_2$  must be equal as well. We collect this information in the record type Same  $p_1$   $p_2$   $U_1$   $U_2$ .

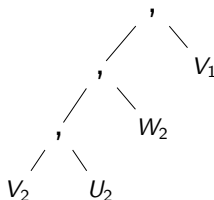
```
record Same (p1 : Path C1) (p2 : Path C2) (U1 U2 : Tree) : Set where
  field
    eqC   : C1 ≡ C2
    eqU   : U1 ≡ U2
    eqp   : subst Path eqT p1 ≡ p2
```

Terms of this type are triples consisting of three equalities about the outer trees, the inner trees used for substitution into holes, and paths, respectively.

case<sub>2</sub> :  $\in \text{Left } p_1 \ p_2 \ U_1 \ U_2$

$U_1$  contains  $U_2$  in its left subtree

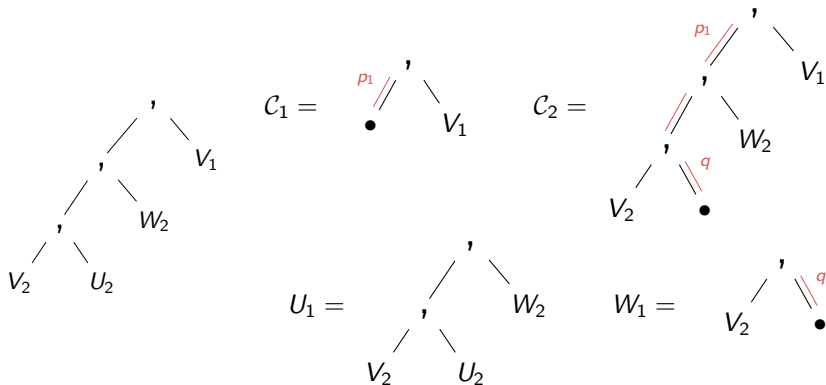
In this case, there exist trees  $W_1$  and  $W_2$  and a path  $q$  in  $W_1$  such that  $U_1$  is equal to the tree (sub  $q \ U_2$  ,  $W_2$ ). Moreover,  $p_2$  is equal to the concatenation of  $p_1$  with  $q \blacktriangleleft W_2$ .





case<sub>2</sub> :  $\in \text{Left } p_1 \ p_2 \ U_1 \ U_2$

$U_1$  contains  $U_2$  in its left subtree



$\text{case}_2 : \in \text{Left } p_1 \ p_2 \ U_1 \ U_2$

This information is collected in the record type  $\in \text{Left } p_1 \ p_2 \ U_1 \ U_2$ :

record  $\in \text{Left } (p_1 : \text{Path } \mathcal{C}_1) \ (p_2 : \text{Path } \mathcal{C}_2) \ (U_1 \ U_2 : \text{Tree}) : \text{Set}$  where  
field

$\{W_1 \ W_2\}$	: Tree
$q$	: Path $W_1$
$\text{eq}\mathcal{C}$	: $\mathcal{C}_2 \equiv \text{sub } p_1 \ (W_1, W_2)$
$\text{eq}U$	: $U_1 \equiv \text{sub } (q \blacktriangleleft W_2) \ U_2$
$\text{eq}p$	: $\text{subst Path eq}T \ p_2 \equiv p_1 ++ (q \blacktriangleleft W_2)$

Terms of this type are tuples consisting of trees  $W_1$  and  $W_2$  (which are implicit), as well as a path  $q : \text{Path } W_1$ , which indicates how to extend  $p_1$ . Additionally, there are three equalities characterizing  $\mathcal{C}_2$ ,  $U_1$ , and  $p_2$ .

# Cut admissibility in Agda

The cut rule in NL takes the form

$$\frac{U \vdash D \quad \mathcal{C}[D] \vdash C}{\mathcal{C}[U] \vdash C} \text{ cut}$$

which in Agda becomes

```
cut : {p : Path C} (f : U ⊢ D) (g : W ⊢ C) (eq : W ≡ sub p (η D))  
  → sub p U ⊢ C
```

# Cut admissibility in Agda

The construction of the function `cut` proceeds by pattern-matching on the second premise.

# Cut admissibility in Agda

The construction of the function cut proceeds by pattern-matching on the second premise.

Recall the following derivation

$$\frac{U \vdash A \quad \frac{V \vdash A' \quad C'[B'] \vdash C}{C'[V, A' \Rightarrow B'] \vdash C} \Rightarrow L}{C[U] \vdash C} \text{ cut}$$

which tells us an equation  $C[A] = C'[V, A' \Rightarrow B']$ .

# Cut admissibility in Agda

The construction of the function cut proceeds by pattern-matching on the second premise.

Recall the following derivation

$$\frac{U \vdash A \quad \frac{V \vdash A' \quad C'[B'] \vdash C}{C'[V, A' \Rightarrow B'] \vdash C} \Rightarrow L}{C[U] \vdash C} \text{ cut}$$

which tells us an equation  $C[A] = C'[V, A' \Rightarrow B']$ .

In Agda, SubEq helps to distinguish four possible cases:

1.  $A = A' \Rightarrow B'$  ( $\in \text{Right}$ ),
2.  $A$  is in  $V$  ( $\in \text{Left}$ ),
3.  $A$  and  $(V, A' \Rightarrow B')$  are disjoint and  $A$  is at the left of  $(V, A' \Rightarrow B')$  ( $\text{Disj}$ ),
4. the dual disjoint case ( $\text{Disj}$ ).

# Equivalence of derivations

Representative examples of three classes of permutative conversions: (i) left rules permute with right rules; (ii) sequential application of left rules; (iii) parallel application of left rules.

$$\frac{\frac{A', C[A, B] \vdash B'}{C[A, B] \vdash A' \Rightarrow B'} \Rightarrow R}{C[A \otimes B] \vdash A' \Rightarrow B'} \otimes L \quad \doteq \quad \frac{\frac{A', C[A, B] \vdash B'}{A', C[A \otimes B] \vdash B'} \otimes L}{C[A \otimes B] \vdash A' \Rightarrow B'} \Rightarrow R$$

$$\frac{\frac{C[A', B'] \vdash A}{C'[C[A', B'], A \Rightarrow B] \vdash C} \Rightarrow L \quad \frac{C'[B] \vdash C}{C'[C[A' \otimes B'], A \Rightarrow B] \vdash C} \otimes L}{C'[C[A' \otimes B'], A \Rightarrow B] \vdash C} \otimes L \quad \doteq \quad \frac{\frac{C[A', B'] \vdash A}{C[A' \otimes B'] \vdash A} \otimes L \quad \frac{C'[B] \vdash C}{C'[C[A' \otimes B'], A \Rightarrow B] \vdash C} \Rightarrow L}{C'[C[A' \otimes B'], A \Rightarrow B] \vdash C} \Rightarrow L$$

$$\frac{\frac{U \vdash A}{C[C_1[B], C_2[V, A' \Rightarrow B']] \vdash C} \Rightarrow L \quad \frac{V \vdash A' \quad C[C_1[B], C_2[B']] \vdash C}{C[C_1[B], C_2[V, A' \Rightarrow B']] \vdash C} \Rightarrow L}{C[C_1[U, A \Rightarrow B], C_2[V, A' \Rightarrow B']] \vdash C} \Rightarrow L \quad \doteq \quad \frac{\frac{U \vdash A' \quad C[C_1[B], C_2[B']] \vdash C}{C[C_1[U, A \Rightarrow B], C_2[B']] \vdash C} \Rightarrow L}{C[C_1[U, A \Rightarrow B], C_2[V, A' \Rightarrow B']] \vdash C} \Rightarrow L$$

# Cut and $\doteq$

We consider the equational theory of derivations, therefore all admissible rules (e.g. cut) and constructions on derivations must be well-defined wrt. to  $\doteq$ . This means that the cut procedure must produce equivalent outputs when applied to equivalent inputs. In Agda, this statement is formalized as follows.

$$\begin{aligned} \text{cut}\doteq_1 &: \{f \ f' : U \vdash D\} \{g : W \vdash C\} (eq_1 : W \equiv \text{sub } p \ \eta \ D) (eq_2 : f \doteq f') \\ &\rightarrow \text{cut } f \ g \ eq_1 \doteq \text{cut } f' \ g \ eq_1 \\ \text{cut}\doteq_2 &: \{f : U \vdash D\} \{g \ g' : W \vdash C\} (eq_1 : W \equiv \text{sub } p \ \eta \ D) (eq_2 : g \doteq g') \\ &\rightarrow \text{cut } f \ g \ eq_1 \doteq \text{cut } f \ g' \ eq_1 \end{aligned}$$



# More examples

We further proved the following properties related to Maehara interpolation with the help of SubEq.

- ▶ Maehara interpolation (MIP):
  - given a derivation  $f : \mathcal{C}[U] \vdash C$ , there exist a formula  $D$  and two derivations  $g : \mathcal{C}[D] \vdash C$  and  $h : U \vdash D$  such that  $\text{var}(D) \subseteq \text{var}(U) \cap \text{var}(\mathcal{C}, C)$ .

# More examples

We further proved the following properties related to Maehara interpolation with the help of SubEq.

- ▶ Maehara interpolation (MIP):
  - given a derivation  $f : \mathcal{C}[U] \vdash C$ , there exist a formula  $D$  and two derivations  $g : \mathcal{C}[D] \vdash C$  and  $h : U \vdash D$  such that  $\text{var}(D) \subseteq \text{var}(U) \cap \text{var}(\mathcal{C}, C)$ .
- ▶ Well-definedness of MIP:
  - for any two derivations  $f, f' : \mathcal{C}[U] \vdash C$ , if  $f \doteq f'$ , then for their interpolant triples,  $(D, g, h)$  and  $(D', g', h')$ ,  $D = D'$ ,  $g \doteq g'$ , and  $h \doteq h'$ .

# More examples

We further proved the following properties related to Maehara interpolation with the help of SubEq.

- ▶ Maehara interpolation (MIP):
  - given a derivation  $f : \mathcal{C}[U] \vdash C$ , there exist a formula  $D$  and two derivations  $g : \mathcal{C}[D] \vdash C$  and  $h : U \vdash D$  such that  $\text{var}(D) \subseteq \text{var}(U) \cap \text{var}(\mathcal{C}, C)$ .
- ▶ Well-definedness of MIP:
  - for any two derivations  $f, f' : \mathcal{C}[U] \vdash C$ , if  $f \doteq f'$ , then for their interpolant triples,  $(D, g, h)$  and  $(D', g', h')$ ,  $D = D'$ ,  $g \doteq g'$ , and  $h \doteq h'$ .
- ▶ Proof-relevant interpolation:
  - if  $g : \mathcal{C}[D] \vdash C$  and  $h : U \vdash D$  are derivations obtained by applying the interpolation procedure on a derivation  $f : \mathcal{C}[U] \vdash C$ , then  $\text{cut}(h, g) \doteq f$ .

## Concluding remarks

- ▶ We have presented an Agda formalization of NL, showcasing the proofs of cut admissibility and various properties related to cut and Maehara interpolation.

# Concluding remarks

- ▶ We have presented an Agda formalization of NL, showcasing the proofs of cut admissibility and various properties related to cut and Maehara interpolation.
- ▶ The formal characterization for case distinction on equality of substituted trees should serve as the bedrock for future formalization of other properties.

# Concluding remarks

- ▶ We have presented an Agda formalization of NL, showcasing the proofs of cut admissibility and various properties related to cut and Maehara interpolation.
- ▶ The formal characterization for case distinction on equality of substituted trees should serve as the bedrock for future formalization of other properties.
- ▶ Future work
  - To formalize soundness and completeness wrt. the Hilbert-style calculus.
  - To strengthen the variable condition to include the multiplicity and the polarity of occurrences of atomic formulae.
  - To generalize well-definedness of the interpolation procedure wrt.  $\equiv$  to richer substructural logics.

# Concluding remarks

- ▶ We have presented an Agda formalization of NL, showcasing the proofs of cut admissibility and various properties related to cut and Maehara interpolation.
- ▶ The formal characterization for case distinction on equality of substituted trees should serve as the bedrock for future formalization of other properties.
- ▶ Future work
  - To formalize soundness and completeness wrt. the Hilbert-style calculus.
  - To strengthen the variable condition to include the multiplicity and the polarity of occurrences of atomic formulae.
  - To generalize well-definedness of the interpolation procedure wrt.  $\equiv$  to richer substructural logics.
- ▶ The Agda formalization is freely available online at:  
<https://github.com/cswphilo/nonassociative-Lambek/tree/main/code>.