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:

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

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

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

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

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:

$$\dfrac{\overset{f}{\Delta \vdash A} \quad \dfrac{\overset{g}{\Lambda_1 \vdash A'} \quad \overset{h}{\Lambda_0, B', \Lambda_1 \vdash C}}{\Lambda_0, \Lambda_1, A' \Rightarrow B', \Lambda_2 \vdash C}\ {\Rightarrow}\mathrm{L}}{\Gamma_0, \Delta, \Gamma_1 \vdash C}\ \text{cut}$$

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 Λ_0,
3. A is in Λ_1, or
4. A is in Λ_2.

Nonassociative Lambek calculus ($\mathtt{NL}$)

In $\mathtt{NL}$, sequents are of the form

$$T \vdash C$$

with binary trees as antecedents.

Nonassociative Lambek calculus ($\mathtt{NL}$)

In $\mathtt{NL}$, 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 ($\mathtt{NL}$)

In $\mathtt{NL}$, 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 ($\mathtt{NL}$)

In $\mathtt{NL}$, 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 $\mathcal{C}$:

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

Nonassociative Lambek calculus (NL)

Derivations are generated inductively by the following rules:

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

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

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

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

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

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

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

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

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:

$$\frac{\dfrac{V \overset{f}{\vdash} D \quad \mathcal{C}'[D] \overset{g}{\vdash} E}{\mathcal{C}'[V] \vdash E}\,\text{cut} \qquad \mathcal{C}[E] \overset{h}{\vdash} C}{\mathcal{C}[\mathcal{C}'[V]] \vdash C}\,\text{cut} \quad = \quad \frac{V \overset{f}{\vdash} D \qquad \dfrac{\mathcal{C}'[D] \overset{g}{\vdash} E \quad \mathcal{C}[E] \overset{h}{\vdash} C}{\mathcal{C}[\mathcal{C}'[D]] \vdash C}\,\text{cut}}{\mathcal{C}[\mathcal{C}'[V]] \vdash C}\,\text{cut}$$

$$\frac{T \overset{f}{\vdash} D \qquad \dfrac{U \overset{g}{\vdash} E \quad \mathcal{C}[\mathcal{C}_1[D], \mathcal{C}_2[E]] \overset{h}{\vdash} C}{\mathcal{C}[\mathcal{C}_1[D], \mathcal{C}_2[U]] \vdash C}\,\text{cut}}{\mathcal{C}[\mathcal{C}_1[T], \mathcal{C}_2[U]] \vdash C}\,\text{cut} \quad = \quad \frac{U \overset{g}{\vdash} E \qquad \dfrac{T \overset{f}{\vdash} D \quad \mathcal{C}[\mathcal{C}_1[D], \mathcal{C}_2[E]] \overset{h}{\vdash} C}{\mathcal{C}[\mathcal{C}_1[T], \mathcal{C}_2[E]] \vdash C}\,\text{cut}}{\mathcal{C}[\mathcal{C}_1[T], \mathcal{C}_2[U]] \vdash C}\,\text{cut}$$

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

$$
\begin{array}{rl}
\multicolumn{2}{l}{\mathsf{data}\ \ \mathsf{Tree} : \mathsf{Set}\ \ \mathsf{where}} \\
\bullet & : \mathsf{Tree} \\
\eta & : \mathsf{Fma} \to \mathsf{Tree} \\
\_\mathbf{,}\_ & : \mathsf{Tree} \to \mathsf{Tree} \to \mathsf{Tree}
\end{array}
$$

Base of the formalization

- trees

$$\begin{array}{l} \mathsf{data}\ \ \mathsf{Tree} : \mathsf{Set}\ \ \mathsf{where} \\ \quad \begin{array}{rl} \bullet & : \mathsf{Tree} \\ \eta & : \mathsf{Fma} \to \mathsf{Tree} \\ \_\mathbf{,}\_ & : \mathsf{Tree} \to \mathsf{Tree} \to \mathsf{Tree} \end{array} \end{array}$$

- path in a tree

$$\begin{array}{l} \mathsf{data}\ \ \mathsf{Path} : \mathsf{Tree} \to \mathsf{Set}\ \ \mathsf{where} \\ \quad \begin{array}{rl} \bullet & : \mathsf{Path}\ \bullet \\ \_\blacktriangleleft\_ & : \forall\ \{T\}\ (p : \mathsf{Path}\ \ T)\ U \to \mathsf{Path}\ (T\ \mathbf{,}\ U) \\ \_\blacktriangleright\_ & : \forall\ \ T\ \{U\}\ (p : \mathsf{Path}\ \ U) \to \mathsf{Path}\ (T\ \mathbf{,}\ U) \end{array} \end{array}$$

Base of the formalization

- trees

$$
\begin{array}{l}
\mathsf{data}\ \ \mathsf{Tree} : \mathsf{Set}\ \ \mathsf{where} \\
\quad \begin{array}{ll}
\bullet & : \mathsf{Tree} \\
\eta & : \mathsf{Fma} \to \mathsf{Tree} \\
\_\mathbf{,}\_ & : \mathsf{Tree} \to \mathsf{Tree} \to \mathsf{Tree}
\end{array}
\end{array}
$$

- path in a tree

$$
\begin{array}{l}
\mathsf{data}\ \ \mathsf{Path} : \mathsf{Tree} \to \mathsf{Set}\ \ \mathsf{where} \\
\quad \begin{array}{ll}
\bullet & : \mathsf{Path}\ \bullet \\
\_\blacktriangleleft\_ & : \forall\ \{T\}\ (p : \mathsf{Path}\ \ T)\ U \to \mathsf{Path}\ (T\ \mathbf{,}\ U) \\
\_\blacktriangleright\_ & : \forall\ \ T\ \{U\}\ (p : \mathsf{Path}\ \ U) \to \mathsf{Path}\ (T\ \mathbf{,}\ U)
\end{array}
\end{array}
$$

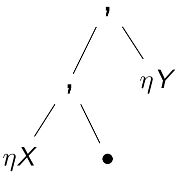
- substitution

$$
\begin{array}{l}
\mathsf{sub} : \forall\{T\} \to \mathsf{Path}\ \ T \to \mathsf{Tree} \to \mathsf{Tree} \\
\begin{array}{ll}
\mathsf{sub}\ \ \bullet\ \ U & = U \\
\mathsf{sub}\ (p \blacktriangleleft V)\ U & = \mathsf{sub}\ p\ U\ \mathbf{,}\ V \\
\mathsf{sub}\ (V \blacktriangleright p)\ U & = V\ \mathbf{,}\ \mathsf{sub}\ p\ U
\end{array}
\end{array}
$$

Example

Example

Consider the tree $T = (\eta X \;\mathbf{,}\; \bullet) \;\mathbf{,}\; \eta Y$, 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 sub $p_1\ U_1 \equiv$ sub $p_2\ U_2$. The 7 cases are formalized as the following datatype:

$$
\begin{array}{lll}
\multicolumn{3}{l}{\mathsf{data}\ \ \mathsf{SubEq}\ (p_1 : \mathsf{Path}\ \mathcal{C}_1)\ (p_2 : \mathsf{Path}\ \mathcal{C}_2)\ (U_1\ \ U_2 : \mathsf{Tree}) : \mathsf{Set}\ \ \mathsf{where}} \\
\ \mathsf{case}_1 & : \mathsf{Same}\ p_1\ \ p_2\ \ U_1\ \ U_2 & \to \mathsf{SubEq}\ p_2\ \ p_1\ \ U_1\ \ U_2 \\
\ \mathsf{case}_2 & : \in\!\mathsf{Left}\ p_1\ \ p_2\ \ U_1\ \ U_2 & \to \mathsf{SubEq}\ p_2\ \ p_1\ \ U_1\ \ U_2 \\
\ \mathsf{case}_3 & : \in\!\mathsf{Right}\ p_1\ \ p_2\ \ U_1\ \ U_2 & \to \mathsf{SubEq}\ p_2\ \ p_1\ \ U_1\ \ U_2 \\
\ \mathsf{case}_4 & : \in\!\mathsf{Left}\ p_2\ \ p_1\ \ U_2\ \ U_1 & \to \mathsf{SubEq}\ p_2\ \ p_1\ \ U_1\ \ U_2 \\
\ \mathsf{case}_5 & : \in\!\mathsf{Right}\ p_2\ \ p_1\ \ U_2\ \ U_1 & \to \mathsf{SubEq}\ p_2\ \ p_1\ \ U_1\ \ U_2 \\
\ \mathsf{case}_6 & : \mathsf{Disj}\ p_1\ \ p_2\ \ U_1\ \ U_2 & \to \mathsf{SubEq}\ p_2\ \ p_1\ \ U_1\ \ U_2 \\
\ \mathsf{case}_7 & : \mathsf{Disj}\ p_2\ \ p_1\ \ U_2\ \ U_1 & \to \mathsf{SubEq}\ p_2\ \ p_1\ \ U_1\ \ U_2
\end{array}
$$

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

I will discuss two cases:

- case_1 : Same p_1 p_2 U_1 U_2 and
- case_2 : $\in$Left p_1 p_2 U_1 U_2.

case_1 : Same p_1 p_2 U_1 U_2

U_1 is equal to U_2

In this case, $\mathcal{C}_1$ must be equal to $\mathcal{C}_2$ 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.

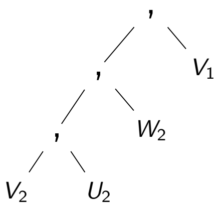
$$
\begin{array}{l}
\text{record Same } (p_1 : \text{Path } \mathcal{C}_1)\ (p_2 : \text{Path } \mathcal{C}_2)\ (U_1\ U_2 : \text{Tree}) : \text{Set where} \\
\quad \text{field} \\
\qquad \begin{array}{rl}
eq\mathcal{C} & : \mathcal{C}_1 \equiv \mathcal{C}_2 \\
eqU & : U_1 \equiv U_2 \\
eqp & : \text{subst Path } eq\mathcal{C}\ p_1 \equiv p_2
\end{array}
\end{array}
$$

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.

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

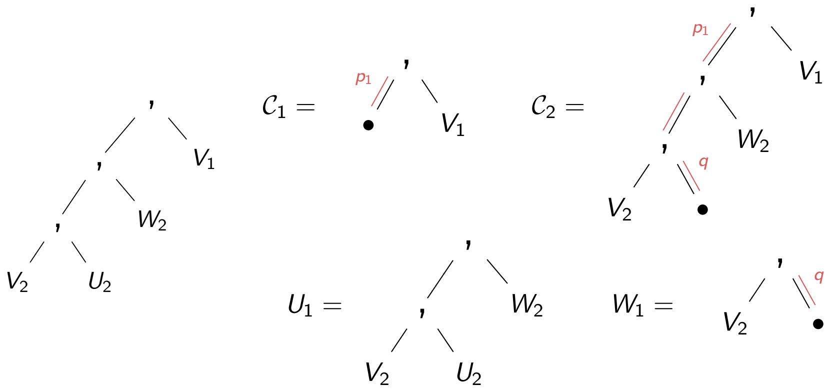
<u>U_1 contains U_2 in its left subtree</u>

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 $q \blacktriangleleft W_2$.



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

<u>U_1 contains U_2 in its left subtree</u>



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

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

$$
\begin{array}{l}
\text{record}\ \in\text{Left}\ (p_1 : \text{Path}\ \mathcal{C}_1)\ (p_2 : \text{Path}\ \mathcal{C}_2)\ (U_1\ U_2 : \text{Tree}) : \text{Set}\ \ \text{where} \\
\quad \text{field} \\
\qquad \begin{array}{ll}
\{W_1\ W_2\} & : \text{Tree} \\
q & : \text{Path}\ W_1 \\
eq\mathcal{C} & : \mathcal{C}_2 \equiv \text{sub}\ p_1\ (W_1\ \mathbf{,}\ W_2) \\
eqU & : U_1 \equiv \text{sub}\ (q \blacktriangleleft W_2)\ U_2 \\
eqp & : \text{subst}\ \text{Path}\ eq\mathcal{C}\ p_2 \equiv p_1 \mathbin{+\!\!+} (q \blacktriangleleft W_2)
\end{array}
\end{array}
$$

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

$$\begin{aligned}\mathsf{cut}\quad &: \{p : \mathsf{Path}\ \mathcal{C}\}\ (f : U \vdash D)\ (g : W \vdash C)\ (eq : W \equiv \mathsf{sub}\ p\ (\eta\ D))\\ &\to \mathsf{sub}\ p\ U \vdash C\end{aligned}$$

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

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

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

Cut admissibility in Agda

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

$$\dfrac{\overset{f}{U \vdash A} \qquad \dfrac{\overset{g}{V \vdash A'} \quad \overset{h}{\mathcal{C}'[B'] \vdash C}}{\mathcal{C}'[V, A' \Rightarrow B'] \vdash C} \Rightarrow\mathsf{L}}{\mathcal{C}[U] \vdash C}\ \mathsf{cut}$$

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

In Agda, SubEq helps to distinguish four possible cases:

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

$$
\dfrac{\dfrac{\overset{f}{A', \mathcal{C}[A,B] \vdash B'}}{\mathcal{C}[A,B] \vdash A' \Rightarrow B'}\Rightarrow\mathrm{R}}{\mathcal{C}[A \otimes B] \vdash A' \Rightarrow B'}\otimes\mathrm{L}
\quad \doteq \quad
\dfrac{\dfrac{\overset{f}{A', \mathcal{C}[A,B] \vdash B'}}{A', \mathcal{C}[A \otimes B] \vdash B'}\otimes\mathrm{L}}{\mathcal{C}[A \otimes B] \vdash A' \Rightarrow B'}\Rightarrow\mathrm{R}
$$

$$
\dfrac{\dfrac{\overset{f}{\mathcal{C}[A', B'] \vdash A} \quad \overset{g}{\mathcal{C}'[B] \vdash C}}{\mathcal{C}'[\mathcal{C}[A', B'], A \Rightarrow B] \vdash C}\Rightarrow\mathrm{L}}{\mathcal{C}'[\mathcal{C}[A' \otimes B'], A \Rightarrow B] \vdash C}\otimes\mathrm{L}
\quad \doteq \quad
\dfrac{\dfrac{\overset{f}{\mathcal{C}[A', B'] \vdash A}}{\mathcal{C}[A' \otimes B'] \vdash A}\otimes\mathrm{L} \quad \overset{g}{\mathcal{C}'[B] \vdash C}}{\mathcal{C}'[\mathcal{C}[A' \otimes B'], A \Rightarrow B] \vdash C}\Rightarrow\mathrm{L}
$$

$$
\dfrac{\overset{f}{U \vdash A} \quad \dfrac{\overset{f'}{V \vdash A'} \quad \overset{g}{\mathcal{C}[\mathcal{C}_1[B], \mathcal{C}_2[B']] \vdash C}}{\mathcal{C}[\mathcal{C}_1[B], \mathcal{C}_2[V, A' \Rightarrow B']] \vdash C}\Rightarrow\mathrm{L}}{\mathcal{C}[\mathcal{C}_1[U, A \Rightarrow B], \mathcal{C}_2[V, A' \Rightarrow B']] \vdash C}\Rightarrow\mathrm{L}
\quad \doteq \quad
\dfrac{\overset{f'}{V \vdash A'} \quad \dfrac{\overset{f}{U \vdash A'} \quad \overset{g}{\mathcal{C}[\mathcal{C}_1[B], \mathcal{C}_2[B']] \vdash C}}{\mathcal{C}[\mathcal{C}_1[U, A \Rightarrow B], \mathcal{C}_2[B']] \vdash C}\Rightarrow\mathrm{L}}{\mathcal{C}[\mathcal{C}_1[U, A \Rightarrow B], \mathcal{C}_2[V, A' \Rightarrow B']] \vdash C}\Rightarrow\mathrm{L}
$$

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.

$$
\begin{aligned}
&\mathsf{cut}{\doteq}_1 : \{f\ f' : U \vdash D\}\ \{g : W \vdash C\}\ (eq_1 : W \equiv \mathsf{sub}\ p\ \eta\ D)\ (eq_2 : f \doteq f') \\
&\quad \to \mathsf{cut}\ f\ g\ eq_1 \doteq \mathsf{cut}\ f'\ g\ eq_1 \\
&\mathsf{cut}{\doteq}_2 : \{f : U \vdash D\}\{g\ g' : W \vdash C\}\ (eq_1 : W \equiv \mathsf{sub}\ p\ \eta\ D)\ (eq_2 : g \doteq g') \\
&\quad \to \mathsf{cut}\ f\ g\ eq_1 \doteq \mathsf{cut}\ f\ g'\ eq_1
\end{aligned}
$$

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 $\mathsf{var}(D) \subseteq \mathsf{var}(U) \cap \mathsf{var}(\mathcal{C}, C)$.

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 $\mathsf{var}(D) \subseteq \mathsf{var}(U) \cap \mathsf{var}(\mathcal{C}, C)$.
- 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 $\mathrm{var}(D) \subseteq \mathrm{var}(U) \cap \mathrm{var}(\mathcal{C}, C)$.
- 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 $\mathrm{cut}(h, g) \doteq f$.

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. $\circeq$ 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. $\circeq$ to richer substructural logics.
- The Agda formalization is freely available online at: https://github.com/cswphilo/nonassociative-Lambek/tree/main/code.