A positive perspective on term representation: work in progress

Jui-Hsuan Wu (Ray) and Dale Miller

Inria Saclay & LIX, Institut Polytechnique de Paris

LFMTP 2022, Haifa, Israel

1 August 2022
Outline

Introduction

Focusing, polarization, and synthetic inference rules

Annotating synthetic rules and proofs
• Terms (or expressions) exist in various settings.
Terms

- Terms (or expressions) exist in various settings.
  Mathematics (equations, formulas, proofs, etc) / Programming languages (compilers, interpreters, etc) / Proof assistants.
Terms

- Terms (or expressions) exist in various settings. Mathematics (equations, formulas, proofs, etc) / Programming languages (compilers, interpreters, etc) / Proof assistants.

- There are different formats for terms:
  
  (1 + 2) + (1 + (1 + 2))
  
  let x = 1 + 2 in let y = (1 + (1 + 2)) in x + y
  
  let x = 1 + 2 in let y = 1 + x in x + y
• Terms (or expressions) exist in various settings. Mathematics (equations, formulas, proofs, etc) / Programming languages (compilers, interpreters, etc) / Proof assistants.

• There are different formats for terms:
  \[ (1 + 2) + (1 + (1 + 2)) \]
  let \( x = 1 + 2 \) in let \( y = (1 + (1 + 2)) \) in \( x + y \)
  let \( x = 1 + 2 \) in let \( y = 1 + x \) in \( x + y \)

• Even some graphical representations:
  (Labelled) Trees, Directed acyclic graphs (DAGs), etc.
Terms

- Terms (or expressions) exist in various settings.
  Mathematics (equations, formulas, proofs, etc) / Programming languages (compilers, interpreters, etc) / Proof assistants.

- There are different formats for terms:
  
  \[(1 + 2) + (1 + (1 + 2))\]
  
  ```
  let x = 1 + 2 in let y = (1 + (1 + 2)) in x + y
  let x = 1 + 2 in let y = 1 + x in x + y
  ```

- Even some graphical representations:
  (Labelled) Trees, Directed acyclic graphs (DAGs), etc.

- What to do with terms? Equality, substitution, evaluation, etc.
Proof theory for term representations

- A framework for describing/unifying/justifying different term structures.
  - Possible interaction between different formats for terms!
Proof theory for term representations

• A framework for describing/unifying/justifying different term structures.
  ▶ Possible interaction between different formats for terms!

• Why proof theory?
Proof theory for term representations

- A framework for describing/unifying/justifying different term structures.
  - Possible interaction between different formats for terms!

- Why proof theory?
  - Highly principled and mathematically sound means for describing syntactic structures.
Proof theory for term representations

• A framework for describing/unifying/justifying different term structures.
  ▶ Possible interaction between different formats for terms!

• Why proof theory?
  ▶ Highly principled and mathematically sound means for describing syntactic structures.
  ▶ Proofs-as-terms, but not proofs-as-programs!
Proof theory for term representations

• A framework for describing/unifying/justifying different term structures.
  ▶ Possible interaction between different formats for terms!

• Why proof theory?
  ▶ Highly principled and mathematically sound means for describing syntactic structures.
  ▶ Proofs-as-terms, but not proofs-as-programs!

• Which proof system to use?
Proof theory for term representations

• A framework for describing/unifying/justifying different term structures.
  ▶ Possible interaction between different formats for terms!

• Why proof theory?
  ▶ Highly principled and mathematically sound means for describing syntactic structures.
  ▶ Proofs-as-terms, but not proofs-as-programs!

• Which proof system to use?
  • Sequent calculus? Too little structure, too much non-essential information (rule permutation).
Proof theory for term representations

- A framework for describing/unifying/justifying different term structures.
  - Possible interaction between different formats for terms!

- Why proof theory?
  - Highly principled and mathematically sound means for describing syntactic structures.
  - Proofs-as-terms, but not proofs-as-programs!

- Which proof system to use?
  - Sequent calculus? Too little structure, too much non-essential information (rule permutation).
  - Focused proof system \(LJF\):
    - Focusing: large-scale rules.
    - Polarization: flexibility on forms of proofs (terms).
• Introduced by Andreoli (1992) to reduce non-determinism in proof search for $LL$. 

• Rule invertible $\iff$ non-invertible

• Information non-essential $\iff$ essential

• Phase negative $\uparrow \iff$ positive $\downarrow$

• Two-phase structure of focused proofs.

• Applied to LJ and LK: LJT, LJQ, LKT, LKQ, etc.

• Polarization: LJF and LKF by Liang and Miller (2009).

• Large-scale rules (not phases!): synthetic inference rules.

J.-H. Wu (Ray) and D. Miller

A positive perspective on term representation: work in progress

5 / 23
Focusing

- Introduced by Andreoli (1992) to reduce non-determinism in proof search for $LL$.

<table>
<thead>
<tr>
<th>Rule</th>
<th>invertible ↔ non-invertible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>non-essential ↔ essential</td>
</tr>
<tr>
<td>Phase</td>
<td>negative ⇑ ↔ positive ⇩</td>
</tr>
</tbody>
</table>

⇒ Two-phase structure of focused proofs.
Focusing

- Introduced by Andreoli (1992) to reduce non-determinism in proof search for $LL$.

<table>
<thead>
<tr>
<th>Rule</th>
<th>invertible $\iff$ non-invertible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>non-essential $\iff$ essential</td>
</tr>
<tr>
<td>Phase</td>
<td>negative $\uparrow$ $\iff$ positive $\downarrow$</td>
</tr>
</tbody>
</table>

$\Rightarrow$ Two-phase structure of focused proofs.

- Applied to LJ and LK: LJT, LJQ, LKT, LKQ, etc.
Focusing

- Introduced by Andreoli (1992) to reduce non-determinism in proof search for $LL$.

<table>
<thead>
<tr>
<th>Rule</th>
<th>invertible $\leftrightarrow$ non-invertible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>non-essential $\leftrightarrow$ essential</td>
</tr>
<tr>
<td>Phase</td>
<td>negative $\uparrow$ $\leftrightarrow$ positive $\downarrow$</td>
</tr>
</tbody>
</table>

$\Rightarrow$ Two-phase structure of focused proofs.

- Applied to LJ and LK: LJT, LJQ, LKT, LKQ, etc.

- Polarization: $LJF$ and $LKF$ by Liang and Miller (2009).
Focusing

- Introduced by Andreoli (1992) to reduce non-determinism in proof search for $LL$.

<table>
<thead>
<tr>
<th>Rule</th>
<th>invertible</th>
<th>⇔</th>
<th>non-invertible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>non-essential</td>
<td>⇔</td>
<td>essential</td>
</tr>
<tr>
<td>Phase</td>
<td>negative $\uparrow$</td>
<td>⇔</td>
<td>positive $\downarrow$</td>
</tr>
</tbody>
</table>

⇒ Two-phase structure of focused proofs.

- Applied to LJ and LK: LJT, LJQ, LKT, LKQ, etc.

- Polarization: $LJF$ and $LKF$ by Liang and Miller (2009).

- Large-scale rules (not phases!): synthetic inference rules.
Two-phase structure, borders, and large-scale rules

borders

\[ \vdash A^\perp, A \quad \vdash B, B^\perp \quad \vdash A^\perp, B^\perp, A \otimes B \quad \vdash A^\perp, B^\perp \oplus (C^\perp \otimes D^\perp), A \otimes B \]

\[ \vdash \vdash C, C^\perp \quad \vdash D^\perp, D \quad \vdash C, D, C^\perp \otimes D^\perp \quad \vdash C \otimes D, B^\perp \oplus (C^\perp \otimes D^\perp) \quad \vdash C \otimes D, (B^\perp \oplus (C^\perp \otimes D^\perp)), (A \otimes B) \& (A \otimes (C \otimes D)) \]

large-scale rule

(= synthetic inference rule)

decide: choose a formula to put under focus

\[ \vdash A^\perp, A \quad \vdash B, B^\perp \]

\[ \vdash A^\perp, B^\perp, A \otimes B \]

\[ \vdash A^\perp, B^\perp \oplus (C^\perp \otimes D^\perp), A \otimes B \]
The \textit{LJF} system with only implication

- Formulas are built using atomic formulas and implication.
The *LJF* system with only implication

- Formulas are built using atomic formulas and implication.
- We work with **polarized** formulas.
  - Implications are negative.
  - Atomic formulas are either **positive** or **negative**.
    (forward-chaining / backchaining)
The *LJF* system with only implication

- Formulas are built using atomic formulas and implication.
- We work with polarized formulas.
  - Implications are negative.
  - Atomic formulas are either positive or negative.
  (forward-chaining / backchaining)
- A polarized formula (resp. theory) is a formula (resp. theory) together with an atomic bias assignment $\delta : \mathcal{A} \rightarrow \{+, -\}$. 

J.-H. Wu (Ray) and D. Miller

A positive perspective on term representation: work in progress
The *LJF* system with only implication

- Formulas are built using atomic formulas and implication.
- We work with **polarized** formulas.
  - Implications are negative.
  - Atomic formulas are either **positive** or **negative**.
    (forward-chaining / backchaining)
- A polarized formula (resp. theory) is a formula (resp. theory) together with an **atomic bias assignment** $\delta : \mathcal{A} \to \{+, -\}$.
- Different polarizations do not affect provability, but give **different forms of proofs**.
  - If a sequent is provable in *LJF* for some polarization, then it is provable for all such polarizations.
The *LJF* system with only implication

Decide, Release, and Store Rules

\[
\frac{N, \Gamma \downarrow N \vdash A}{N, \Gamma \vdash A} \quad D_I \\
\frac{\Gamma \vdash P \downarrow}{\Gamma \vdash P} \quad D_r \\
\frac{\Gamma \vdash P \vdash A}{\Gamma \downarrow P \vdash A} \quad R_I \\
\frac{\Gamma \vdash N \uparrow}{\Gamma \vdash N \downarrow} \quad R_r
\]

\[
\frac{\Gamma, C \uparrow \Theta \vdash \Delta' \uparrow \Delta}{\Gamma \uparrow \Theta, C \vdash \Delta' \uparrow \Delta} \quad S_I \\
\frac{\Gamma \uparrow \Theta \vdash A}{\Gamma \uparrow \Theta \vdash A} \quad S_r
\]

Initial Rules

\[
\frac{A \text{ positive}}{A, \Gamma \vdash A \downarrow} \quad I_r \\
\frac{A \text{ negative}}{\Gamma \downarrow A \vdash A} \quad I_l
\]

Introduction Rules for Implication

\[
\frac{\Gamma \vdash B \downarrow \quad \Gamma \downarrow B' \vdash A}{\Gamma \downarrow B \supset B' \vdash A} \quad \supset L \\
\frac{\Gamma \uparrow \Theta, B \vdash B' \uparrow}{\Gamma \uparrow \Theta \vdash B \supset B' \uparrow} \quad \supset R
\]
**Synthetic inference rules**

**Synthetic inference rule** = large-scale rule = \(\downarrow\)-phase + \(\uparrow\)-phase

**Definition**

A *left synthetic inference rule* for \(B\) is an inference rule of the form

\[
\begin{align*}
\Gamma_1 \vdash A_1 & \quad \ldots \quad \Gamma_n \vdash A_n \\
\Gamma \vdash A
\end{align*}
\]

justified by a derivation (in \(LJF\)) of the form

\[
\begin{align*}
\Gamma_1 \vdash A_1 & \quad \ldots \quad \Gamma_n \vdash A_n \\
& \quad \quad \quad \ldots \quad \uparrow \text{phase} \\
& \quad \quad \quad \ldots \quad \downarrow \text{phase} \\
\Gamma \downarrow B \vdash A
\end{align*}
\]

\[
\frac{D_l}{\Gamma \vdash A}
\]
Axioms as rules

Definition

Let $\mathcal{T}$ be a finite polarized theory of order 2 or less, We define $LJ\langle \mathcal{T} \rangle$ to be the extension of $LJ$ with the left synthetic inference rules for $\mathcal{T}$. More precisely, for every left synthetic inference rule

\[
\frac{B, \Gamma_1 \vdash A_1 \quad \ldots \quad B, \Gamma_n \vdash A_n}{B, \Gamma \vdash A} \quad B
\]

with $B \in \mathcal{T}$, the inference rule

\[
\frac{\Gamma_1 \vdash A_1 \quad \ldots \quad \Gamma_n \vdash A_n}{\Gamma \vdash A} \quad B
\]

is added to $LJ\langle \mathcal{T} \rangle$. 
**Axioms as rules**

**Definition**

Let $\mathcal{T}$ be a finite polarized theory of order 2 or less. We define $LJ\langle\mathcal{T}\rangle$ to be the extension of $LJ$ with the left synthetic inference rules for $\mathcal{T}$. More precisely, for every left synthetic inference rule

$$
\frac{B, \Gamma_1 \vdash A_1 \quad \ldots \quad B, \Gamma_n \vdash A_n}{B \vdash A}
$$

with $B \in \mathcal{T}$, the inference rule

$$
\frac{\Gamma_1 \vdash A_1 \quad \ldots \quad \Gamma_n \vdash A_n}{\Gamma \vdash A}
$$

is added to $LJ\langle\mathcal{T}\rangle$.

**Theorem**

$\mathcal{T}, \Gamma \vdash B$ provable in $LJ \iff \Gamma \vdash B$ provable in $LJ\langle\mathcal{T}\rangle$. 
An example

Let $\mathcal{T}$ be the collection of formulas

$D_1 = a_0 \supset a_1, \cdots, D_n = a_0 \supset \cdots \supset a_n, \cdots$ where $a_i$ are atomic.
An example

Let $\mathcal{T}$ be the collection of formulas
$D_1 = a_0 \supset a_1, \ldots, D_n = a_0 \supset \cdots \supset a_n, \ldots$ where $a_i$ are atomic.

If $a_i$ are all given the negative bias,
An example

Let $\mathcal{T}$ be the collection of formulas

$D_1 = a_0 \supset a_1, \cdots, D_n = a_0 \supset \cdots \supset a_n, \cdots$ where $a_i$ are atomic.

If $a_i$ are all given the **negative** bias, the inference rules in $LJ\langle \mathcal{T} \rangle$ include

$$
\Gamma \vdash a_0 \quad \cdots \quad \Gamma \vdash a_{n-1}
$$

$$
\Gamma \vdash a_n
$$
An example

Let $\mathcal{T}$ be the collection of formulas
\[ D_1 = a_0 \supset a_1, \cdots, D_n = a_0 \supset \cdots \supset a_n, \cdots \] where $a_i$ are atomic.

If $a_i$ are all given the negative bias, the inference rules in $LJ\langle \mathcal{T} \rangle$ include

\[
\begin{array}{c}
\Gamma \vdash a_0 \\
\vdots \\
\Gamma \vdash a_{n-1}
\end{array}
\]

\[
\Gamma \vdash a_n
\]

"backchaining"
An example

Let $\mathcal{T}$ be the collection of formulas $D_1 = a_0 \supset a_1, \cdots, D_n = a_0 \supset \cdots \supset a_n, \cdots$ where $a_i$ are atomic.

If $a_i$ are all given the negative bias, the inference rules in $LJ(\mathcal{T})$ include

\[
\Gamma \vdash a_0 \quad \cdots \quad \Gamma \vdash a_{n-1} \\
\quad \Gamma \vdash a_n
\]

"backchaining"

If $a_i$ are all given the positive bias,
An example

Let $\mathcal{T}$ be the collection of formulas
$D_1 = a_0 \supset a_1, \cdots, D_n = a_0 \supset \cdots \supset a_n, \cdots$ where $a_i$ are atomic.

If $a_i$ are all given the negative bias, the inference rules in $LJ\langle \mathcal{T} \rangle$ include

$$
\frac{\Gamma \vdash a_0 \quad \cdots \quad \Gamma \vdash a_{n-1}}{
\Gamma \vdash a_n}
$$

"backchaining"

If $a_i$ are all given the positive bias, the inference rules in $LJ\langle \mathcal{T} \rangle$ include

$$
\frac{\Gamma, a_0, \cdots, a_{n-1}, a_n \vdash A}{\Gamma, a_0, \cdots, a_{n-1} \vdash A}
$$
An example

Let $\mathcal{T}$ be the collection of formulas
$D_1 = a_0 \supset a_1, \cdots, D_n = a_0 \supset \cdots \supset a_n, \cdots$ where $a_i$ are atomic.

If $a_i$ are all given the negative bias, the inference rules in $LJ\langle \mathcal{T} \rangle$ include

$$
\Gamma \vdash a_0 \quad \cdots \quad \Gamma \vdash a_{n-1} \\
\Gamma \vdash a_n
$$

"backchaining"

If $a_i$ are all given the positive bias, the inference rules in $LJ\langle \mathcal{T} \rangle$ include

$$
\Gamma, a_0, \cdots, a_{n-1}, a_n \vdash A \\
\Gamma, a_0, \cdots, a_{n-1} \vdash A
$$

"forward-chaining"
What are the proofs of $a_0 \vdash a_n$?
Backchaining and Forward-chaining

What are the proofs of $a_0 \vdash a_n$?

When $a_i$ are all given the **negative** bias, we have:

\[
\begin{array}{c}
\Gamma \vdash a_0 \\
\Gamma \vdash a_1 \\
\Gamma \vdash a_2 \\
\vdots \\
\Gamma \vdash a_{n-1} \\
\Gamma \vdash a_n
\end{array}
\]

▷ a **unique** proof of exponential size
What are the proofs of $a_0 \vdash a_n$?

When $a_i$ are all given the negative bias, we have:

$$
\begin{align*}
\Gamma \vdash a_0 & \quad \Gamma \vdash a_0 \quad \Gamma \vdash a_1 \\
\Gamma \vdash a_1 & \quad \Gamma \vdash a_2 & \quad \ldots & \quad \Gamma \vdash a_0 \quad \ldots \quad \Gamma \vdash a_{n-1} \\
& \quad \Gamma \vdash a_n
\end{align*}
$$

▷ a unique proof of exponential size

When $a_i$ are all given the positive bias, we have:

$$
\begin{align*}
\Gamma, a_0, a_1 \vdash A & \quad \Gamma, a_0, a_1, a_2 \vdash A \\
\Gamma, a_0 \vdash A & \quad \Gamma, a_0, a_1 \vdash A \quad \ldots \quad \Gamma, a_0, \ldots, a_{n-1}, a_n \vdash A \\
& \quad \Gamma, a_0, \ldots, a_{n-1} \vdash A
\end{align*}
$$

▷ a shortest proof of linear size
Annotating rules and proofs

Consider the inference rules in the previous example and annotate them.

\[
\begin{align*}
\Gamma & \vdash a_0 & \Gamma & \vdash a_0 & \Gamma & \vdash a_1 & \Gamma & \vdash a_2 \\
\Gamma & \vdash a_1 & & \Gamma & \vdash a_2 & \ldots \\
\Gamma & \vdash a_0 & \ldots & \Gamma & \vdash a_{n-1} & \\
\Gamma & \vdash a_n
\end{align*}
\]

Consider the proofs of $a_0 \vdash a_4$. 
Annotating rules and proofs

Consider the inference rules in the previous example and annotate them.

\[
\frac{\Gamma \vdash t_0 : a_0}{\Gamma \vdash E_1 t_0 : a_1} \quad \frac{\Gamma \vdash t_0 : a_0}{\Gamma \vdash E_2 t_0 t_1 : a_2} \quad \ldots
\]

\[
\frac{\Gamma \vdash t_0 : a_0 \quad \ldots \quad \Gamma \vdash t_{n-1} : a_{n-1}}{\Gamma \vdash E_n t_0 \cdots t_{n-1} : a_n}
\]

Consider the proofs of \( a_0 \vdash a_4 \).
Annotating rules and proofs

Consider the inference rules in the previous example and annotate them.

\[
\begin{align*}
\Gamma \vdash t_0 : a_0 & \quad \Gamma \vdash t_0 : a_0 \quad \Gamma \vdash t_1 : a_1 \\
\Gamma \vdash E_1 t_0 : a_1 & \quad \Gamma \vdash E_2 t_0 t_1 : a_2 \\
\Gamma \vdash t_0 : a_0 & \quad \ldots \quad \Gamma \vdash t_{n-1} : a_{n-1} \\
\Gamma \vdash E_n t_0 \cdots t_{n-1} : a_n
\end{align*}
\]

Consider the proofs of \( d_0 : a_0 \vdash t : a_4 \).
Annotating rules and proofs

Consider the inference rules in the previous example and annotate them.

\[
\begin{align*}
\Gamma & \vdash t_0 : a_0 \\
\Gamma & \vdash E_1 t_0 : a_1 \\
\Gamma & \vdash t_0 : a_0 \\
\Gamma & \vdash t_1 : a_1 \\
\Gamma & \vdash E_2 t_0 t_1 : a_2 \\
\Gamma & \vdash t_0 : a_0 \\
\Gamma & \vdash \cdots \\
\Gamma & \vdash t_{n-1} : a_{n-1} \\
\Gamma & \vdash E_n t_0 \cdots t_{n-1} : a_n
\end{align*}
\]

Consider the proofs of \( d_0 : a_0 \vdash t : a_4 \).

The term annotating the unique proof is

\[
(E_4 (E_3 (E_2 (E_1 d_0) (E_1 d_0)) (E_1 d_0)))
\]

\[
(E_2 (E_1 d_0) (E_1 d_0)))
\]

\[
(E_3 (E_2 (E_1 d_0) (E_1 d_0))
\]

\[
(E_2 (E_1 d_0) (E_1 d_0))))
\]
Consider the inference rules in the previous example and annotate them.

\[
\Gamma, a_0, a_1 \vdash A \\
\Gamma, a_0 \vdash A
\]

\[
\Gamma, a_0, a_1, a_2 \vdash A \\
\Gamma, a_0, a_1 \vdash A
\]

\[\ldots\]

\[
\Gamma, a_0, \ldots, a_{n-1}, a_n \vdash A \\
\Gamma, a_0, \ldots, a_{n-1} \vdash A
\]

Consider the proofs of \(a_0 \vdash a_4\).
Annotating rules and proofs

Consider the inference rules in the previous example and annotate them.

\[
\begin{align*}
\Gamma, x_0 : a_0, x_1 : a_1 & \vdash t : A \\
\Gamma, x_0 : a_0 & \vdash F_1 x_0 (\lambda x_1 . t) : A
\end{align*}
\]

\[
\begin{align*}
\Gamma, x_0 : a_0, x_1 : a_1 & \vdash t : A \\
\Gamma, x_0 : a_0, x_1 : a_1, x_2 : a_2 & \vdash t : A
\end{align*}
\]

\[
\begin{align*}
\Gamma, x_0 : a_0, x_1 : a_1 & \vdash F_2 x_0 x_1 (\lambda x_2 . t) : A \\
\Gamma, x_0 : a_0, x_1 : a_1, x_2 : a_2 & \vdash t : A
\end{align*}
\]

\[
\begin{align*}
\Gamma, x_0 : a_0 & \vdash F_n x_0 \cdots x_{n-1} (\lambda x_n . t) : A
\end{align*}
\]

Consider the proofs of \(d_0 : a_0 \vdash t : a_4\).
Annotating rules and proofs

Consider the inference rules in the previous example and annotate them.

\[ \Gamma, x_0 : a_0, x_1 : a_1 \vdash t : A \]
\[ \Gamma, x_0 : a_0 \vdash F_1 x_0(\lambda x_1.t) : A \]
\[ \Gamma, x_0 : a_0, x_1 : a_1, x_2 : a_2 \vdash t : A \]
\[ \Gamma, x_0 : a_0, x_1 : a_1 \vdash F_2 x_0 x_1(\lambda x_2.t) : A \]
\[ \Gamma, x_0 : a_0, \cdots, x_{n-1} : a_{n-1}, x_n : a_n \vdash t : A \]
\[ \Gamma, x_0 : a_0, \cdots, x_{n-1} : a_{n-1} \vdash F_n x_0 \cdots x_{n-1}(\lambda x_n.t) : A \]

Consider the proofs of \( d_0 : a_0 \vdash t : a_4 \).

The term annotating the shortest proof is

\[
(F_1 \ d_0 \quad (\lambda x_1. \\
(F_2 \ d_0 \ x_1 \quad (\lambda x_2. \\
(F_3 \ d_0 \ x_1 \ x_2 \quad (\lambda x_3. \\
(F_4 \ d_0 \ x_1 \ x_2 \ x_3 \ (\lambda x_4. x_4))))))))
\]
Encodings of untyped $\lambda$-terms

We use a primitive type (atomic formula) $D$ for untyped $\lambda$-terms.
Encodings of untyped $\lambda$-terms

We use a primitive type (atomic formula) $D$ for untyped $\lambda$-terms.

We fix a theory $T = \{ \Phi : D \supset D \supset D, \Psi : (D \supset D) \supset D \}$ and consider proofs of sequents of the form $T, x_1 : D, \cdots, x_k : D \vdash t : D$.
Encodings of untyped $\lambda$-terms

We use a primitive type (atomic formula) $D$ for untyped $\lambda$-terms.

We fix a theory $T = \{ \Phi : D \supset D \supset D, \Psi : (D \supset D) \supset D \}$ and consider proofs of sequents of the form $T, x_1 : D, \cdots, x_k : D \vdash t : D$

When $D$ is given the negative bias, we have the following synthetic inference rules:

$$
\begin{align*}
\Gamma \vdash D & \quad \Gamma \vdash D \\
\hline
\Gamma \vdash D & \quad \Phi \\
\end{align*}
$$

$$
\begin{align*}
\Gamma, D \vdash D & \\
\hline
\Gamma \vdash D & \quad \Psi \\
\end{align*}
$$

and the initial rule.
Encodings of untyped $\lambda$-terms

We use a primitive type (atomic formula) $D$ for untyped $\lambda$-terms.

We fix a theory $\mathcal{T} = \{ \Phi : D \supset D \supset D, \Psi : (D \supset D) \supset D \}$ and consider proofs of sequents of the form $\mathcal{T}, x_1 : D, \cdots, x_k : D \vdash t : D$

When $D$ is given the negative bias, we have the following synthetic inference rules:

\[
\frac{\Gamma \vdash t : D \quad \Gamma \vdash u : D}{\Gamma \vdash \Phi} \\
\frac{\Gamma \vdash D \quad \Gamma \vdash D}{\Gamma \vdash \Psi}
\]

and the initial rule.
Encodings of untyped $\lambda$-terms

We use a primitive type (atomic formula) $D$ for untyped $\lambda$-terms.

We fix a theory $\mathcal{T} = \{ \Phi : D \supset D \supset D, \Psi : (D \supset D) \supset D \}$ and consider proofs of sequents of the form $\mathcal{T}, x_1 : D, \cdots, x_k : D \vdash t : D$

When $D$ is given the negative bias, we have the following synthetic inference rules:

\[
\frac{\Gamma \vdash t : D \quad \Gamma \vdash u : D}{\Gamma \vdash \Phi \ t \ u : D} \quad \Phi
\]

\[
\frac{\Gamma, D \vdash D}{\Gamma \vdash D} \quad \Psi
\]

and the initial rule.
We use a primitive type (atomic formula) $D$ for untyped $\lambda$-terms. We fix a theory $T = \{\Phi : D \supset D \supset D, \Psi : (D \supset D) \supset D\}$ and consider proofs of sequents of the form $\Gamma, x_1 : D, \cdots, x_k : D \vdash t : D$.

When $D$ is given the negative bias, we have the following synthetic inference rules:

\[
\begin{align*}
\Gamma \vdash t : D & \quad \Gamma \vdash u : D & \quad \Phi \\
\hline
\Gamma \vdash \Phi \ t \ u : D
\end{align*}
\]

\[
\begin{align*}
\Gamma, x : D \vdash t : D & \quad \Psi \\
\hline
\Gamma \vdash D
\end{align*}
\]

and the initial rule.
Encodings of untyped $\lambda$-terms

We use a primitive type (atomic formula) $D$ for untyped $\lambda$-terms.

We fix a theory $\mathcal{T} = \{ \Phi : D \vdash D \vdash D, \Psi : (D \vdash D) \vdash D \}$ and consider proofs of sequents of the form $\mathcal{T}, x_1 : D, \cdots, x_k : D \vdash t : D$

When $D$ is given the negative bias, we have the following synthetic inference rules:

$$
\Gamma \vdash t : D \quad \Gamma \vdash u : D \quad \Phi
\quad \frac{}{\Gamma \vdash \Phi \ t \ u : D}
$$

$$
\Gamma, x : D \vdash t : D \quad \Psi
\quad \frac{}{\Gamma \vdash \Psi \ (\lambda x. t) : D}
$$

and the initial rule.
Encodings of untyped \( \lambda \)-terms

We use a primitive type (atomic formula) \( D \) for untyped \( \lambda \)-terms.

We fix a theory \( T = \{ \Phi : D \supset D \supset D, \psi : (D \supset D) \supset D \} \) and consider proofs of sequents of the form \( T, x_1 : D, \cdots, x_k : D \vdash t : D \)

When \( D \) is given the positive bias, we have the following synthetic inference rules:

\[
\begin{align*}
\Gamma, D, D, D & \vdash D & \Phi \\
\Gamma, D & \vdash D & \psi \\
\Gamma & \vdash D \\
\end{align*}
\]

and the initial rule.
Encodings of untyped $\lambda$-terms

We use a primitive type (atomic formula) $D$ for untyped $\lambda$-terms.

We fix a theory $\mathcal{T} = \{ \Phi : D \supset D \supset D, \Psi : (D \supset D) \supset D \}$ and consider proofs of sequents of the form $\mathcal{T}, x_1 : D, \ldots, x_k : D \vdash t : D$

When $D$ is given the positive bias, we have the following synthetic inference rules:

$$\frac{\Gamma, x : D, y : D, z : D \vdash t : D}{\Gamma, D, D \vdash D} \quad \Phi$$

$$\frac{\Gamma, D \vdash D \quad \Gamma, D \vdash D}{\Gamma \vdash D} \quad \Psi$$

and the initial rule.
Encodings of untyped $\lambda$-terms

We use a primitive type (atomic formula) $D$ for untyped $\lambda$-terms.

We fix a theory $\mathcal{T} = \{\Phi : D \supset D \supset D, \psi : (D \supset D) \supset D\}$ and consider proofs of sequents of the form $\mathcal{T}, x_1 : D, \cdots, x_k : D \vdash t : D$.

When $D$ is given the positive bias, we have the following synthetic inference rules:

\[
\frac{\Gamma, x : D, y : D, z : D \vdash t : D}{\Gamma, x : D, y : D \vdash \Phi \times y \, (\lambda z.t) : D} \quad \Phi
\]

\[
\frac{\Gamma, D \vdash D}{\Gamma \vdash D} \quad \Psi
\]

and the initial rule.
Encodings of untyped $\lambda$-terms

We use a primitive type (atomic formula) $D$ for untyped $\lambda$-terms.

We fix a theory $T = \{ \Phi : D \supset D \supset D, \Psi : (D \supset D) \supset D \}$ and consider proofs of sequents of the form $T, x_1 : D, \cdots, x_k : D \vdash t : D$

When $D$ is given the positive bias, we have the following synthetic inference rules:

\[
\begin{align*}
\Gamma, x : D, y : D, z : D & \vdash t : D \\
\Phi & \\
\Gamma, x : D, y : D & \vdash \Phi \times y (\lambda z. t) : D \\
\end{align*}
\]

\[
\begin{align*}
\Gamma, x : D & \vdash t : D \\
\Gamma, y : D & \vdash u : D \\
\Psi & \\
\Gamma & \vdash D
\end{align*}
\]

and the initial rule.
Encodings of untyped $\lambda$-terms

We use a primitive type (atomic formula) $D$ for untyped $\lambda$-terms.

We fix a theory $\mathcal{T} = \{ \Phi : D \supset D \supset D, \Psi : (D \supset D) \supset D \}$ and consider proofs of sequents of the form $\mathcal{T}, x_1 : D, \cdots, x_k : D \vdash t : D$

When $D$ is given the positive bias, we have the following synthetic inference rules:

$$
\frac{
\Gamma, x : D, y : D, z : D \vdash t : D
}{
\Gamma, x : D, y : D \vdash \Phi x y (\lambda z. t) : D
}\quad \Phi

\frac{
\Gamma, x : D \vdash t : D \quad \Gamma, y : D \vdash u : D
}{
\Gamma \vdash \Psi (\lambda x. t) (\lambda y. u) : D
}\quad \Psi
$$

and the initial rule.
Two formats for untyped $\lambda$-terms

Two different polarity assignments give two different term structures:

- $D$ is negative:
  \[
  \begin{align*}
  x & \quad \text{nvar } x \quad x \\
  \Phi \ t \ u & \quad \text{napp } t \ u \quad tu \\
  \Psi (\lambda x.t) & \quad \text{nabs } (x\ t) \quad \lambda x.t \\
  \end{align*}
  \]
  $\rightarrow$ Top-down / tree-like structure

- $D$ is positive:
  \[
  \begin{align*}
  x & \quad \text{pvar } x \quad x \\
  \Phi x y (\lambda z.t) & \quad \text{papp } x \ y \ (z\ t) \quad name \ z = xy \ in \ t \\
  \Psi (\lambda x.t)(\lambda y.s) & \quad \text{pabs } (x\ t) \ (y\ s) \quad name \ y = \lambda x.t \ in \ s \\
  \end{align*}
  \]
  $\rightarrow$ Bottom-up / DAG structure
Some examples for the positive-bias syntax

\[
\text{name } y = \text{app } x \ x \ \text{in name } z = \text{app } y \ y \ \text{in } z
\]

▷ Arguments of app are all names
name y = app x x in name z = app y y in z

- Arguments of app are all names

name y1 = app x x in name y2 = app x x in name z = app y1 y2 in z

- Redundant naming
Some examples for the positive-bias syntax

name y = app x x in name z = app y y in z

▷ Arguments of app are all names

name y1 = app x x in name y2 = app x x in
name z = app y1 y2 in z

▷ Redundant naming

name y1 = app x x in name y2 = app y y in
name z = app y1 y1 in z

▷ Vacuous naming
Some examples for the positive-bias syntax

name \( y = \text{app} \ x \ x \) in name \( z = \text{app} \ y \ y \) in \( z \)

▷ Arguments of \text{app} are all \textit{names}

name \( y_1 = \text{app} \ x \ x \) in name \( y_2 = \text{app} \ x \ x \) in
name \( z = \text{app} \ y_1 \ y_2 \) in \( z \)

▷ Redundant naming

name \( y_1 = \text{app} \ x \ x \) in \( \text{name} \ y_2 = \text{app} \ y \ y \) in
name \( z = \text{app} \ y_1 \ y_1 \) in \( z \)

▷ Vacuous naming

name \( y_1 = \text{app} \ x \ x \) in name \( y_2 = \text{app} \ y \ y \) in
name \( z = \text{app} \ y_1 \ y_2 \) in \( z \)

name \( z = \text{abs} \ (x\ \text{name} \ y_1 = \text{app} \ y \ y \) in \( y_1 \) in \( z \)

▷ Parallel naming
Some examples for the positive-bias syntax

name y = app x x in name z = app y y in z

▷ Arguments of app are all names

name y1 = app x x in name y2 = app x x in
name z = app y1 y2 in z

▷ Redundant naming

name y1 = app x x in name y2 = app y y in
name z = app y1 y1 in z

▷ Vacuous naming

name y1 = app x x and y2 = app y y in
name z = app y1 y2 in z

name z = abs (x\ name y1 = app y y in y1) in z

▷ Parallel naming
Some examples for the positive-bias syntax

name y = app x x in name z = app y y in z

- Arguments of app are all names

name y1 = app x x in name y2 = app x x in name z = app y1 y2 in z

- Redundant naming

name y1 = app x x in name y2 = app y y in name z = app y1 y1 in z

- Vacuous naming

name y1 = app x x and y2 = app y y in name z = app y1 y2 in z

name y1 = app y y in name z = abs (x\ y1) in z

- Parallel naming
Cut-elimination for $LJ\langle \mathcal{T} \rangle$

The following theorem\(^1\) states that cut is admissible for the extensions of $LJ$ with polarized theories based on synthetic inference rules.

**Theorem (Cut admissibility for $LJ\langle \mathcal{T} \rangle$)**

Let $\mathcal{T}$ be a finite polarized theory of order 2 or less. Then the cut rule is admissible for the proof system $LJ\langle \mathcal{T} \rangle$.

---

The following theorem\(^1\) states that cut is admissible for the extensions of \(LJ\) with polarized theories based on synthetic inference rules.

**Theorem (Cut admissibility for \(LJ\langle T\rangle\))**

Let \(T\) be a finite polarized theory of order 2 or less. Then the cut rule is admissible for the proof system \(LJ\langle T\rangle\).

The proof is based on a cut elimination procedure for \(LJF\)

- This defines the notion of substitution for terms.

The following theorem\(^1\) states that cut is admissible for the extensions of \(LJ\) with polarized theories based on synthetic inference rules.

**Theorem (Cut admissibility for \(LJ\langle T\rangle\))**

Let \(T\) be a finite polarized theory of order 2 or less. Then the cut rule is admissible for the proof system \(LJ\langle T\rangle\).

The proof is based on a cut elimination procedure for \(LJF\)

- This defines the notion of substitution for terms.

When we restrict to atomic cut formulas, the cut elimination procedure can be presented in a big-step style.

- Cuts are permuted with synthetic rules instead of \(LJF\) rules.

---

Untyped $\lambda$-terms (substitution)

The cut-elimination procedure of $LJF$ gives us the following definitions of substitutions.

\[
\text{type nsubst, psubs} \quad \text{tm} \rightarrow (\text{val} \rightarrow \text{tm}) \rightarrow \text{tm} \rightarrow \text{o}.
\]

\[
\begin{align*}
n\text{subst } T & \left( x \backslash \text{nvar } x \right) T. \\
n\text{subst } T & \left( x \backslash \text{nvar } Y \right) \left( \text{nvar } Y \right) . \\
n\text{subst } T & \left( x \backslash \text{napp } \left( R \times \right) \left( S \times \right) \right) \left( \text{napp } R' \left( S' \right) \right) \left( \text{napp } R' \left( S' \right) \right) : \left( \text{napp } R' \left( S' \right) \right) \\
& \quad \text{nsubst } T \left( R \times \right) \left( R' \times \right) \left( S \times \right) \left( S' \times \right) . \\
n\text{subst } T & \left( x \backslash \text{nabs } y \backslash \left( R \times y \right) \right) \left( \text{nabs } y \backslash \left( R' \times y \right) \right) : \left( \text{nabs } y \backslash \left( R' \times y \right) \right) \\
& \quad \text{pi } y \backslash \text{nsubst } T \left( x \backslash \left( R \times y \right) \right) \left( R' \times y \right) . \\
p\text{subst } \left( \text{papp } U V K \right) & \left( \text{papp } U V H \right) : \left( \text{papp } U V H \right) \\
& \quad \text{pi } x \backslash \text{psubs} \left( K \times \right) \left( R \times \right) \left( H \times \right) . \\
p\text{subst } \left( \text{pabs } S K \right) & \left( \text{pabs } S H \right) : \left( \text{pabs } S H \right) \\
& \quad \text{pi } x \backslash \text{psubs} \left( K \times \right) \left( R \times \right) \left( H \times \right) . \\
p\text{subst } \left( \text{pvar } U \right) & \left( R \times \right) \left( R \times \right) .
\end{align*}
\]
An example

```
name y = app x x in
name z = app y y in z
name y' = app a a in
name z' = app y' y' in z
```
An example

```
name y = app x x in
name z = app y y in z
```

```
name y' = app a a in
name z' = app y' y' in z'
```
An example

name y = app x x in
name z = app y y in z

name y' = app a a in
name z' = app y' y' in
name y = app z' z' in
name z = app y y in z

name y' = app a a in
name z' = app y' y' in z'
Equality on terms

We have now two different formats for untyped $\lambda$-terms.

When should two such expressions be considered the same?
Equality on terms

We have now two different formats for untyped λ-terms.

When should two such expressions be considered the same?

”White box” approach:

▷ Look at the actual syntax of proof expressions.
   ⇒ not working since we have two different sets of synthetic inference rules.
Equality on terms

We have now two different formats for untyped $\lambda$-terms.

When should two such expressions be considered the same?

"White box" approach:
- Look at the actual syntax of proof expressions.
  - not working since we have two different sets of synthetic inference rules.

"Black box" approach:
- Describe traces by probing a term: exponential cost.
  - Bisimulation on graphical representations.
Graphical representations

The positive-bias syntax is closely related to some graphical representations.

▶ *name* introduces *new nodes* and gives them a *label*. 
The positive-bias syntax is closely related to some graphical representations.

> `name` introduces **new nodes** and gives them a **label**.

Here is an example:

\[
\lambda \, ^{x_3} x_3 \downarrow \, ^{x_2} x_2 \\
\downarrow \, ^{x_1} x_1 \\
\text{name } x_3 = \text{abs (x\ name x_1 = app x x in \\
\text{name x_2 = app x_1 x_1 in x_2) in x_3}
\]

\[
. 
\]
The positive-bias syntax is closely related to some graphical representations.

▷ name introduces new nodes and gives them a label.

Here is an example:

\[
\lambda x^3 \rightarrow x^2 \rightarrow x_1 \rightarrow \\
\text{name } x_3 = \\
\text{abs } (x \backslash \text{name } x_1 = \text{app } x \ x \ \text{in} \\
\text{name } x_2 = \text{app } x_1 \ x_1 \ \text{in} \ x_2) \ \text{in} \ x_3
\]

Bisimulation on graphs allows to check sharing equality in linear time\(^2\).

\(^2\)Andrea Condoluci, Beniamino Accattoli, and Claudio Sacerdoti Coen. Sharing equality is linear. *PPDP 2019.*
Parallel naming can be captured by graphical representations:

\[ \text{name } y_1 = \text{app } x \ x \ \text{in} \ \text{name } y_2 = \text{app } y \ y \ \text{in} \]
\[ \text{name } z = \text{app } y_1 \ y_2 \ \text{in} \ z \]
\[ \text{name } y_2 = \text{app } y \ y \ \text{in} \ \text{name } y_1 = \text{app } x \ x \ \text{in} \]
\[ \text{name } z = \text{app } y_1 \ y_2 \ \text{in} \ z \]

\[ \text{name } z = \text{abs } (x \ \text{name } y_1 = \text{app } y \ y \ \text{in} y_1) \ \text{in} \]
\[ \text{name } y_1 = \text{app } y \ y \ \text{in} \ \text{name } z = \text{abs } (x \ \text{in} y_1) \ \text{in} \]
\[ \text{name } z = \text{abs } (x \ \text{in} y_1) \ \text{in} \ z \]

\[ \text{name } y_1 = \text{app } y \ y \ \text{in} \ \text{name } z = \text{abs } (x \ \text{in} y_1) \ \text{in} \ z \]
Related and future work

- Generalize to full $LJF$. 
Related and future work

- Generalize to full \textit{LJF}.
- Multi-focusing:
  - Parallel actions (parallel naming).
  - Maximal multi-focused proofs $\leftrightarrow$ graphical structures.
  - Conjecture: MMF proofs are isomorphic to some graphical structure in the case for untyped $\lambda$-terms.
Related and future work

- Generalize to full $LJF$.
- Multi-focusing:
  - Parallel actions (parallel naming).
  - Maximal multi-focused proofs $\leftrightarrow$ graphical structures.
  - Conjecture: MMF proofs are isomorphic to some graphical structure in the case for untyped $\lambda$-terms.
- Big-step cut-elimination for arbitrary cut formulas.
  - At the level of synthetic rules (not phases!).
Related and future work

- Generalize to full $LJF$.
- Multi-focusing:
  - Parallel actions (parallel naming).
  - Maximal multi-focused proofs $\leftrightarrow$ graphical structures.
  - Conjecture: MMF proofs are isomorphic to some graphical structure in the case for untyped $\lambda$-terms.
- Big-step cut-elimination for arbitrary cut formulas.
  - At the level of synthetic rules (not phases!).
- Connection with the literature in programming language theory (A-normal form, etc).
Related and future work

- Generalize to full $LJF$.
- Multi-focusing:
  - Parallel actions (parallel naming).
  - Maximal multi-focused proofs $\leftrightarrow$ graphical structures.
  - Conjecture: MMF proofs are isomorphic to some graphical structure in the case for untyped $\lambda$-terms.
- Big-step cut-elimination for arbitrary cut formulas.
  - At the level of synthetic rules (not phases!).
- Connection with the literature in programming language theory (A-normal form, etc).
- There exist some other frameworks for term structures, such as terms-as-graphs by Grabmayer. Are there some connections or overlaps?
Related and future work

- Generalize to full \textit{LJF}.
- Multi-focusing:
  - Parallel actions (parallel naming).
  - Maximal multi-focused proofs $\leftrightarrow$ graphical structures.
  - Conjecture: MMF proofs are isomorphic to some graphical structure in the case for untyped $\lambda$-terms.
- Big-step cut-elimination for arbitrary cut formulas.
  - At the level of \textit{synthetic rules} (not phases!).
- Connection with the literature in programming language theory (A-normal form, etc).
- There exist some other frameworks for term structures, such as terms-as-graphs by Grabmayer. Are there some connections or overlaps?
- \textbf{Proof-theoretic} methods for checking term equality.