# Proof Terms for Higher-Order Rewriting and Their Equivalence

October 28th, 2022

**Pablo Barenbaum**
Universidad Nacional de Quilmes (CONICET)
Universidad de Buenos Aires
Argentina

Eduardo Bonelli
Stevens Institute of Technology
USA

# **First-order** proof terms

# Proof terms for **first-order** rewriting

A well-known first-order term rewriting system

$$\begin{array}{lcrcl} \varrho(x) & : & \mathbf{add}(\mathbf{zero}, x) & \rightharpoonup & x \\ \vartheta(x, y) & : & \mathbf{add}(\mathbf{suc}(x), y) & \rightharpoonup & \mathbf{suc}(\mathbf{add}(x, y)) \end{array}$$

Some first-order proof terms

$$\vartheta(\mathbf{zero}, \mathbf{suc}(\mathbf{zero})) : \mathbf{add}(\mathbf{suc}(\mathbf{zero}), \mathbf{suc}(\mathbf{zero})) \rightharpoonup \mathbf{suc}(\mathbf{add}(\mathbf{zero}, \mathbf{suc}(\mathbf{zero})))$$

$$\mathbf{suc}(\varrho(\mathbf{suc}(\mathbf{zero}))) : \mathbf{suc}(\mathbf{add}(\mathbf{zero}, \mathbf{suc}(\mathbf{zero}))) \rightharpoonup \mathbf{suc}(\mathbf{suc}(\mathbf{zero}))$$

$$\begin{array}{l} \vartheta(\mathbf{zero}, \mathbf{suc}(\mathbf{zero})) \ ; \ \mathbf{suc}(\varrho(\mathbf{suc}(\mathbf{zero}))) \\ \qquad : \mathbf{add}(\mathbf{suc}(\mathbf{zero}), \mathbf{suc}(\mathbf{zero})) \rightharpoonup \mathbf{suc}(\mathbf{suc}(\mathbf{zero})) \end{array}$$

# Proof terms for **first-order** rewriting

## First-order proof terms (formal syntax)

$$
\begin{array}{rclll}
\rho & ::= & \mathbf{c}(\rho_1, \ldots, \rho_n) & \text{congruence} & \mathbf{c} \text{ is any } n\text{-ary } \textbf{function symbol} \\
& \mid & \varrho(\rho_1, \ldots, \rho_n) & \text{rule application} & \varrho \text{ is any } n\text{-ary } \textbf{rule symbol} \\
& \mid & \rho_1 \,\mathbf{;}\, \rho_2 & \text{composition} &
\end{array}
$$

## Rewriting judgment

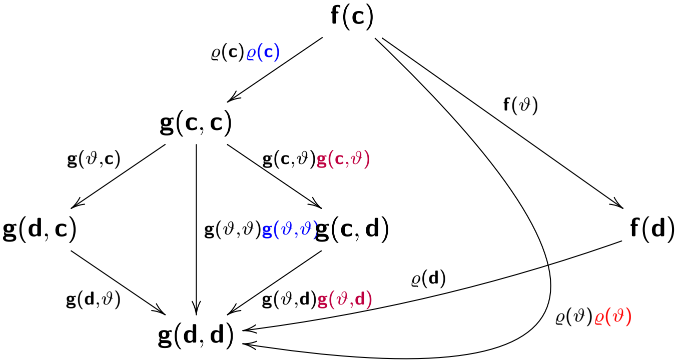
$$
\frac{\ldots \rho_i : s_i \rightharpoonup t_i \ldots}{\mathbf{c}(\rho_1, \ldots, \rho_n) : \mathbf{c}(s_1, \ldots, s_n) \rightharpoonup \mathbf{c}(t_1, \ldots, t_n)}
$$

$$
\frac{(\varrho(x_1, \ldots, x_n) : s \to t) \in \mathcal{R} \qquad \ldots \rho_i : s_i \rightharpoonup t_i \ldots}{\varrho(\rho_1, \ldots, \rho_n) : s\{x_i \backslash s_i\}_{i \in 1..n} \rightharpoonup t\{x_i \backslash t_i\}_{i \in 1..n}}
$$

$$
\frac{\rho : s_1 \rightharpoonup s_2 \qquad \sigma : s_2 \rightharpoonup s_3}{\rho \,\mathbf{;}\, \sigma : s_1 \rightharpoonup s_3}
$$

# Permutation equivalence of reductions (example)

$$\begin{array}{llcl} \varrho(x) & : & \mathbf{f}(x) & \rightharpoonup & \mathbf{g}(x,x) \\ \vartheta & : & \mathbf{c} & \rightharpoonup & \mathbf{d} \end{array}$$




$$\varrho(\vartheta) \quad \approx \quad \varrho(\mathbf{c})\ ;\ \mathbf{g}(\vartheta,\vartheta) \quad \approx \quad \varrho(\mathbf{c})\ ;\ (\mathbf{g}(\mathbf{c},\vartheta)\ ;\ \mathbf{g}(\vartheta,\mathbf{d}))$$

# Permutation equivalence of reductions (important remark)

- If $\rho \approx \sigma$ then $\rho$ and $\sigma$ have the same source and target:

$$\rho : s \rightarrowtriangle t \qquad \text{and} \qquad \sigma : s \rightarrowtriangle t$$

- But the converse does not hold, for instance, if:

$$\varrho(x) \quad : \quad \mathbf{f}(x) \quad \rightarrowtriangle \quad x$$

then:

$$\begin{tikzcd}[row sep=large] \mathbf{f}(\mathbf{f}(\mathbf{c})) \arrow[d, bend right=60, "\varrho(\mathbf{f}(\mathbf{c}))"'] \arrow[d, bend left=60, "\mathbf{f}(\varrho(\mathbf{c}))"] \\ \mathbf{f}(\mathbf{c}) \end{tikzcd}$$

and $\mathbf{f}(\varrho(\mathbf{c})) \not\approx \varrho(\mathbf{f}(\mathbf{c}))$.

# Projection of reductions (example)

$$
\begin{array}{lclcl}
\varrho(x) & : & \mathbf{f}(x) & \rightharpoonup & \mathbf{g}(x,x) \\
\vartheta & : & \mathbf{c} & \rightharpoonup & \mathbf{d}
\end{array}
$$

$$
\begin{tikzcd}
 & & \mathbf{f}(\mathbf{c}) \arrow[dl, "\varrho(\mathbf{c})"'] \arrow[ddrr, "\mathbf{f}(\vartheta)"] \arrow[dddl, bend left=60, "\varrho(\vartheta)"] & & \\
 & \mathbf{g}(\mathbf{c},\mathbf{c}) \arrow[dl, "\mathbf{g}(\vartheta{,}\mathbf{c})"'] \arrow[dd, "\mathbf{g}(\vartheta{,}\vartheta)"] \arrow[dr, "\mathbf{g}(\mathbf{c}{,}\vartheta)"] & & & \\
\mathbf{g}(\mathbf{d},\mathbf{c}) \arrow[dr, "\mathbf{g}(\mathbf{d}{,}\vartheta)"'] & & \mathbf{g}(\mathbf{c},\mathbf{d}) \arrow[dl, "\mathbf{g}(\vartheta{,}\mathbf{d})"] & & \mathbf{f}(\mathbf{d}) \arrow[dlll, "\varrho(\mathbf{d})"'] \\
 & \mathbf{g}(\mathbf{d},\mathbf{d}) & & &
\end{tikzcd}
$$

$$
\begin{array}{lcl}
\varrho(\vartheta)/\varrho(\mathbf{c}) & = & \mathbf{g}(\vartheta,\vartheta) \\
\varrho(\vartheta)/(\varrho(\mathbf{c}) \ ;\ \mathbf{g}(\mathbf{c},\vartheta)) & = & \mathbf{g}(\vartheta,\mathbf{d}) \\
\varrho(\vartheta)/(\varrho(\mathbf{c}) \ ;\ (\mathbf{g}(\mathbf{c},\vartheta) \ ;\ \mathbf{g}(\vartheta,\mathbf{d}))) & = & \mathbf{g}(\mathbf{d},\mathbf{d})
\end{array}
\qquad
\varrho(\mathbf{c})/\varrho(\vartheta) = \mathbf{g}(\mathbf{d},\mathbf{d})
$$

# Equivalent notions of equivalence

## Theorem (de Vrijer, van Oostrom)

The following are equivalent:

1. **Permutation equivalence:** $\rho \approx \sigma$.
2. **Projection equivalence:** $\rho/\sigma$ and $\sigma/\rho$ are empty.
   Here "empty" means that it contains no rule symbols.

## Basic historical notes

- Permutation equivalence and projection equivalence had been studied and shown equivalent by Jean-Jacques Lévy ($\sim$1978). (But without proof terms).
- Proof terms were introduced in the work of José Meseguer. ($\sim$1992; keyword: "rewriting logic").
- Proof terms were extensively studied by Roel de Vrijer and Vincent van Oostrom ($\sim$2002) to study notions of equivalence between reductions, including also **standardization equivalence** and **labeling equivalence**. (See *e.g.* the Terese book, Chapter 8).

# **Higher-order** proof terms

# Higher-order rewriting systems (à la Nipkow)

## A well-known higher-order rewriting system

$$\mathbf{app}\,(\mathbf{lam}\,f)\,x \quad \rightharpoonup \quad f\,x$$

The object language is encoded in **higher-order abstract syntax**:

- First-order terms become simply-typed $\lambda$-terms:

$$\mathbf{app} : \iota \to \iota \to \iota \qquad \mathbf{lam} : (\iota \to \iota) \to \iota \qquad f : \iota \to \iota \qquad x : \iota$$

- Terms are considered up to $\beta\eta$-equivalence.
- In HRSs, left-hand sides of rules must be *patterns*.
- HRSs **strictly generalize** first-order term rewriting systems.
- We work with **orthogonal** HRSs: left-linear, no critical pairs.
- Orthogonal HRSs are confluent.
- HRSs were introduced by Tobias Nipkow ($\sim$1991).
  There are other flavors of HORSs (*e.g.* Klop's CRSs).

# Proof terms for **higher-order** rewriting

## Example

$$\beta \quad : \quad \lambda f.\lambda x.\,\mathbf{app}\,(\mathbf{lam}\,f)\,x \quad \rightharpoonup \quad \lambda f.\lambda x.\,f\,x \quad : (\iota \to \iota) \to \iota \to \iota$$

The reduction step of the object language:

$$\lambda x.(\lambda z.z\,(z\,x))\,I \to \lambda x.I\,(I\,x)$$

can be encoded as the higher-order proof term:

$$\mathbf{lam}\,(\lambda x.\beta\,\underbrace{(\lambda z.\mathbf{app}\,z\,(\mathbf{app}\,z\,x))}_{\iota\to\iota}\,\underbrace{(\mathbf{lam}(\lambda x.x))}_{\iota}) : s \rightharpoonup t$$

with

$$\begin{array}{rcl} s & = & \mathbf{lam}\,(\lambda x.\mathbf{app}\,(\mathbf{lam}\,(\lambda z.\mathbf{app}\,z\,(\mathbf{app}\,z\,x)))\,(\mathbf{lam}(\lambda x.x))) \\ t & = & \mathbf{lam}\,(\lambda x.(\lambda z.\mathbf{app}\,z\,(\mathbf{app}\,z\,x))\,(\mathbf{lam}(\lambda x.x))) \\ & =_{\beta\eta} & \mathbf{lam}\,(\lambda x.\mathbf{app}\,(\mathbf{lam}(\lambda x.x))\,(\mathbf{app}\,(\mathbf{lam}(\lambda x.x))\,x)) \end{array}$$

# Proof terms for **higher-order** rewriting

## Higher-order proof terms (formal syntax)

$$
\begin{array}{rcll}
\rho & ::= & x & \text{variable} \\
& | & \mathbf{c} & \text{constant} \\
& | & \varrho & \text{rule symbol} \\
& | & \lambda x.\rho & \text{abstraction} \\
& | & \rho_1\, \rho_2 & \text{application} \\
& | & \rho_1 \,\mathbf{;}\, \rho_2 & \text{composition}
\end{array}
$$

## Rewriting judgment

$$
\frac{}{x : x \rightarrow x} \qquad \frac{}{\mathbf{c} : \mathbf{c} \rightarrow \mathbf{c}} \qquad \frac{(\varrho : s \rightarrow t) \in \mathcal{R}}{\varrho : s \rightarrow t} \qquad \frac{\rho : s \rightarrow t}{\lambda x.\rho : \lambda x.s \rightarrow \lambda x.t}
$$

$$
\frac{\rho_1 : s_1 \rightarrow t_1 \qquad \rho_2 : s_2 \rightarrow t_2}{\rho_1\, \rho_2 : s_1\, s_2 \rightarrow t_1\, t_2} \qquad \frac{\rho_1 : s_1 \rightarrow s_2 \qquad \rho_2 : s_2 \rightarrow s_3}{\rho_1 \,\mathbf{;}\, \rho_2 : s_1 \rightarrow s_3}
$$

$$
\frac{s =_{\beta\eta} s' \qquad \rho : s' \rightarrow t' \qquad t' =_{\beta\eta} t}{\rho : s \rightarrow t}
$$

# A stumbling block

Proof terms for higher-order rewriting were studied by Bruggink ($\sim$2008).

## What does "$(\lambda x.\rho)\,\sigma$" mean?

$$(\lambda x.\rho)\,\sigma \;\overset{?}{\approx}\; \rho\{x\backslash\sigma\}$$

## As noted by Bruggink, this is not sound

Suppose that $\rho : s \to t$ is such that $s \neq t$. Then:

$$\begin{array}{lclcl}
x & : & x & \rightharpoonup & x \\
x \,;\, x & : & x & \rightharpoonup & x \\
\lambda x.(x \,;\, x) & : & \lambda x.x & \rightharpoonup & \lambda x.x \\
(\lambda x.(x \,;\, x))\,\rho & : & (\lambda x.x)\,s & \rightharpoonup & (\lambda x.x)\,t \\
(\lambda x.(x \,;\, x))\,\rho & : & s & \rightharpoonup & t
\end{array}$$

But $\rho \,;\, \rho$ is not well-typed, as $\rho$ cannot be composed with itself.

Bruggink sidesteps the problem by allowing compositions ("**;**") only at the toplevel.

# Permutation equivalence for higher-order proof terms

## Definition

$$
\begin{array}{rcll}
\rho^{\mathsf{src}} \,;\, \rho & \approx & \rho & \\
\rho \,;\, \rho^{\mathsf{tgt}} & \approx & \rho & \\
(\rho \,;\, \sigma) \,;\, \tau & \approx & \rho \,;\, (\sigma \,;\, \tau) & \\
(\lambda x.\rho) \,;\, (\lambda x.\sigma) & \approx & \lambda x.(\rho \,;\, \sigma) & \\
(\rho_1\rho_2) \,;\, (\sigma_1\sigma_2) & \approx & (\rho_1 \,;\, \sigma_1)(\rho_2 \,;\, \sigma_2) & \\
(\lambda x.s)\,\rho & \approx & s\{x \mathbin{\backslash\!\!\backslash} \rho\} & \\
(\lambda x.\rho)\,s & \approx & \rho\{x \backslash s\} & \\
\lambda x.\rho\, x & \approx & \rho & \text{if } x \notin \mathsf{fv}(\rho)
\end{array}
$$

- $\rho^{\mathsf{src}}$ and $\rho^{\mathsf{tgt}}$ denote the source and the target term of $\rho$.
- $s\{x \mathbin{\backslash\!\!\backslash} \rho\}$ substitutes a variable in a $\lambda$-term for a proof term (yielding a proof term).
- $\rho\{x \backslash s\}$ substitutes a variable in a proof term for a $\lambda$-term (yielding a proof term).

# Permutation equivalence for higher-order proof terms

## Example

$$
\begin{array}{lclcl}
\varrho & : & \lambda z.\mathbf{mu}\, z & \rightharpoonup & \lambda z.z\,(\mathbf{mu}\, z) & : & (\iota \to \iota) \to \iota \\
\vartheta & : & \mathbf{f} & \rightharpoonup & \mathbf{g} & : & \iota \to \iota
\end{array}
$$

Then:

$$\varrho\,\vartheta : \mathbf{mu}\,\mathbf{f} \rightharpoonup \mathbf{g}\,(\mathbf{mu}\,\mathbf{g})$$

And:

$$
\begin{array}{lll}
& \varrho\,\vartheta & \\
\approx & (\varrho \,;\, (\lambda z.z\,(\mathbf{mu}\, z)))\,\vartheta & \text{as } \lambda z.z\,(\mathbf{mu}\, z) \text{ is the target of } \varrho \\
\approx & (\varrho \,;\, (\lambda z.z\,(\mathbf{mu}\, z)))\,(\mathbf{f} \,;\, \vartheta) & \text{as } \mathbf{f} \text{ is the source of } \vartheta \\
\approx & \varrho\,\mathbf{f} \,;\, (\lambda z.z\,(\mathbf{mu}\, z))\,\vartheta & \text{by the application rule} \\
\approx & \varrho\,\mathbf{f} \,;\, \vartheta\,(\mathbf{mu}\,\vartheta) & \text{by the term/rewrite } \beta\text{-like rule}
\end{array}
$$

## Proposition

$$(\lambda x.\rho)\,\sigma \;\approx\; \rho\{x\backslash\sigma^{\mathsf{src}}\} \,;\, \rho^{\mathsf{tgt}}\{x\|\sigma\} \;\approx\; \rho^{\mathsf{src}}\{x\|\sigma\} \,;\, \rho\{x\backslash\sigma^{\mathsf{tgt}}\}$$

# Flattening

## Definition

We have proposed a *flattening* relation between higher-order proof terms:

$$\begin{array}{rcll}
\lambda x.(\rho \,;\, \sigma) & \overset{\flat}{\mapsto} & (\lambda x.\rho) \,;\, (\lambda x.\sigma) & \\
(\rho \,;\, \sigma)\, \mu & \overset{\flat}{\mapsto} & (\rho\, \mu^{\mathsf{src}}) \,;\, (\sigma\, \mu) & \\
\mu\, (\rho \,;\, \sigma) & \overset{\flat}{\mapsto} & (\mu\, \rho) \,;\, (\mu^{\mathsf{tgt}}\, \sigma) & \\
(\rho_1 \,;\, \rho_2)\, (\sigma_1 \,;\, \sigma_2) & \overset{\flat}{\mapsto} & ((\rho_1 \,;\, \rho_2)\, \sigma_1^{\mathsf{src}}) \,;\, (\rho_2^{\mathsf{tgt}}\, (\sigma_1 \,;\, \sigma_2)) & \\
(\lambda x.\mu)\, \nu & \overset{\flat}{\mapsto} & \mu\{x \backslash \nu\} & \\
\lambda x.\mu\, x & \overset{\flat}{\mapsto} & \mu & \text{if } x \notin \mathsf{fv}(\mu)
\end{array}$$

where $\mu, \nu, \ldots$ stand for **multisteps**, that is, multisteps without occurrences of the composition operator "**;**".

## Theorem

Flattening is **confluent** and **strongly normalizing**.

The normal forms are called **flat proof terms**.
Compositions only appear at the toplevel, as in Bruggink's work.

# Flat permutation equivalence

A notion of permutation equivalence **between flat proof terms** can be defined as follows:

$$\begin{array}{lcll} (\rho \,;\, \sigma) \,;\, \tau & \sim & \rho \,;\, (\sigma \,;\, \tau) & \\ \mu & \sim & \mu_1^\flat \,;\, \mu_2^\flat & \quad \text{if } \mu \Leftrightarrow \mu_1 \,;\, \mu_2 \end{array}$$

where $\mu \Leftrightarrow \mu_1 \,;\, \mu_2$ is a ternary relation meaning that the multistep $\mu$ can be "split" as the composition of the multisteps $\mu_1$ and $\mu_2$.

## Example

If, as before:

$$\begin{array}{ccrcccl} \varrho & : & \lambda z.\mathbf{mu}\, z & \rightharpoonup & \lambda z.z\,(\mathbf{mu}\, z) & : & (\iota \to \iota) \to \iota \\ \vartheta & : & \mathbf{f} & \rightharpoonup & \mathbf{g} & : & \iota \to \iota \end{array}$$

Then, for example:

$$\varrho\,\vartheta \sim \varrho\,\mathbf{f} \,;\, \vartheta\,(\mathbf{mu}\,\vartheta) \qquad \text{since } \varrho\,\vartheta \Leftrightarrow \varrho\,\mathbf{f} \,;\, (\lambda z.z\,(\mathbf{mu}\, z))\,\vartheta$$

## Theorem (Flat permutation equivalence)

$\rho \approx \sigma$ if and only if $\rho^\flat \sim \sigma^\flat$.

# Projection

A notion of **projection** can be defined for **multisteps** (no composition):

$$
\frac{}{x \mathbin{/\!\!/\!\!/} x \Rightarrow x}
\qquad
\frac{}{\mathbf{c} \mathbin{/\!\!/\!\!/} \mathbf{c} \Rightarrow \mathbf{c}}
\qquad
\frac{}{\varrho \mathbin{/\!\!/\!\!/} \varrho \Rightarrow \varrho^{\mathsf{tgt}}}
\qquad
\frac{}{\varrho \mathbin{/\!\!/\!\!/} \varrho^{\mathsf{src}} \Rightarrow \varrho}
$$

$$
\frac{}{\varrho^{\mathsf{src}} \mathbin{/\!\!/\!\!/} \varrho \Rightarrow \varrho^{\mathsf{tgt}}}
\qquad
\frac{\mu \mathbin{/\!\!/\!\!/} \nu \Rightarrow \xi}{\lambda x.\mu \mathbin{/\!\!/\!\!/} \lambda x.\nu \Rightarrow \lambda x.\xi}
\qquad
\frac{\mu_1 \mathbin{/\!\!/\!\!/} \nu_1 \Rightarrow \xi_1 \qquad \mu_2 \mathbin{/\!\!/\!\!/} \nu_2 \Rightarrow \xi_2}{\mu_1\,\mu_2 \mathbin{/\!\!/\!\!/} \nu_1\,\nu_2 \Rightarrow \xi_1\,\xi_2}
$$

This can be extended to **flat** proof terms in a typical way:

$$
\begin{array}{rcll}
\mu^\flat \mathbin{/\!\!/} \nu^\flat \mu^\flat \mathbin{/\!\!/} \nu^\flat & \overset{\mathrm{def}}{=} \; \overset{\mathrm{def}}{=} & \xi^\flat \xi^\flat & \text{if } \mu \mathbin{/\!\!/\!\!/} \nu \Rightarrow \xi \\
\rho \mathbin{/\!\!/} (\sigma \,;\, \tau) & \overset{\mathrm{def}}{=} & (\rho \mathbin{/\!\!/} \sigma) \mathbin{/\!\!/} \tau & \\
(\rho \,;\, \sigma) \mathbin{/\!\!/} \tau & \overset{\mathrm{def}}{=} & (\rho \mathbin{/\!\!/} \tau) \,;\, (\sigma \mathbin{/\!\!/} (\tau \mathbin{/\!\!/} \rho)) &
\end{array}
$$

(The first equation uses pattern matching against LHSs of rewrite rules).(The first equation uses pattern matching against LHSs of rewrite rules).

Finally, it can be extended to **arbitrary** proof terms by flattening first:

$$
\rho / \sigma \overset{\mathrm{def}}{=} \rho^\flat \mathbin{/\!\!/} \sigma^\flat
$$

# Projection equivalence

**Theorem (Projection equivalence)**

$\rho \approx \sigma$ if and only if $\rho/\sigma$ and $\sigma/\rho$ are empty.

Again, "empty" means that it contains no rule symbols.

# Future work

- Formulate a standardization procedure.
- Study labeling equivalence.
- Relate with 2-categorical models (Hirschowitz, 2013).