Multi-focusing on extensional rewriting with sums (introduction)

Gabriel Scherer

Gallium – INRIA

March 11, 2014
... but why?

Current research topic: does a given type have a unique inhabitant (modulo program equivalence)?
... but why?

Current research topic: does a given type have a unique inhabitant (modulo program equivalence)?

\[(\lambda(x) \ t) \ u \to_{\beta} t[u/x] \quad (t : A \to B) =_{\eta} \lambda(x) \ t \ x\]

\[\pi_i (t_1, t_2) \to_{\beta} t_i \quad (t : A \ast B) =_{\eta} (\pi_1 \ t, \pi_2 \ t)\]
... but why?

Current research topic: does a given type have a \textbf{unique} inhabitant (modulo program equivalence)?

\[
(\lambda(x) \ t) \ u \rightarrow_{\beta} t[u/x] \quad (t : A \rightarrow B) =_{\eta} \lambda(x) \ t \ x
\]

\[
\pi_i (t_1, t_2) \rightarrow_{\beta} t_i \quad (t : A \ast B) =_{\eta} (\pi_1 t, \pi_2 t)
\]

\[
\delta(\sigma; t, x_1.u_1, x_2.u_2) \rightarrow_{\beta} u_i[t/x_i]
\]

\[
(t : A + B) =_{\eta} \delta(t, x_1.\sigma_1 x_1, x_2.\sigma_2 x_2)
\]
... but why?

Current research topic: does a given type have a \textbf{unique} inhabitant (modulo program equivalence)?

\[
(\lambda(x) \, t) \, u \rightarrow_{\beta} t[u/x] \quad (t : A \rightarrow B) =_{\eta} \lambda(x) \, t \, x
\]

\[
\pi_i \, (t_1, t_2) \rightarrow_{\beta} t_i \quad (t : A \ast B) =_{\eta} (\pi_1 \, t, \pi_2 \, t)
\]

\[
\delta(\sigma; t, \, x_1.u_1, \, x_2.u_2) \rightarrow_{\beta} u_i[t/x_i]
\]

\[
(t : A + B) =_{\eta} \delta(t, \, x_1.\sigma_1 \, x_1, \, x_2.\sigma_2 \, x_2)
\]

\[
(t, \, u) \overset{?}{=} \delta(t, \, x_1.(\sigma_1 \, x_1, \, u), \, x_2.(\sigma_2 \, x_2, \, u))
\]
... but why?

Current research topic: does a given type have a **unique** inhabitant (modulo program equivalence)?

\[(\lambda(x) t) \ u \rightarrow_{\beta} t[u/x] \quad (t : A \rightarrow B) =_{\eta} \lambda(x) t \ x\]

\[\pi_i (t_1, t_2) \rightarrow_{\beta} t_i \quad (t : A \ast B) =_{\eta} (\pi_1 t, \pi_2 t)\]

\[\delta(\sigma_i t, x_1.u_1, x_2.u_2) \rightarrow_{\beta} u_i[t/x_i]\]

\[(t : A + B) =_{\eta} \delta(t, x_1.\sigma_1 x_1, x_2.\sigma_2 x_2)\]

\[(t, u) \overset{?}{=} \delta(t, x_1.(\sigma_1 x_1, u), x_2.(\sigma_2 x_2, u)) \quad K = (\Box, u)\]
... but why?

Current research topic: does a given type have a unique inhabitant (modulo program equivalence)?

\[
(\lambda(x) \ t) \ u \rightarrow_{\beta} t[u/x] \quad (t : A \rightarrow B) =_{\eta} \lambda(x) \ t \ x
\]

\[
\pi_i (t_1, t_2) \rightarrow_{\beta} t_i \quad (t : A \ast B) =_{\eta} (\pi_1 \ t, \pi_2 \ t)
\]

\[
\delta(\sigma_i \ t, x_1.u_1, x_2.u_2) \rightarrow_{\beta} u_i[t/x_i]
\]

\[
\forall(K[A_1 + A_2] : B), \quad K[t] =_{\eta} \delta(t, x_1.K[\sigma_1 \ x_1], x_2.K[\sigma_2 \ x_2])
\]
... but why?

Current research topic: does a given type have a unique inhabitant (modulo program equivalence)?

\[(\lambda (x) \; t) \; u \rightarrow_\beta t[u/x] \quad (t : A \rightarrow B) =_\eta \lambda (x) \; t \; x\]

\[\pi_i \; (t_1, \; t_2) \rightarrow_\beta t_i \quad (t : A \times B) =_\eta (\pi_1 \; t, \; \pi_2 \; t)\]

\[\delta(\sigma_i \; t, \; x_1.u_1, \; x_2.u_2) \rightarrow_\beta u_i[t/x_i]\]

\[\forall (K[A_1 + A_2] : B), \quad K[t] =_\eta \delta(t, \; x_1.K[\sigma_1 \; x_1], \; x_2.K[\sigma_2 \; x_2])\]

- Sum equivalence looks hard. Can we implement it?
- Are there representations of programs (proofs) that quotient over those equivalences?
My paper in one slide

The equivalence algorithm of

Sam Lindley.
Extensional rewriting with sums.

and the normalization of proof representations in

Kaustuv Chaudhuri, Dale Miller, and Alexis Saurin.
Canonical sequent proofs via multi-focusing.

are doing (almost) the same thing
– and we had not noticed.
In this talk

Sam Lindley’s rewriting-based algorithm is the first simple solution (first solution: Neil Ghani, 1995) to deciding sum equivalences.

It’s easy to understand and follow. But to me it felt a bit arbitrary.

On the other hand, (multi-)focusing is beautiful, but requires some background knowledge.

Providing it is the purpose of this talk.
Sequent calculus
(Can be done in natural deduction, but less regular)

\[ \Gamma \vdash A \quad \Gamma, B \vdash C \]
\[ \Gamma, A \rightarrow B \vdash C \]

\[ \Gamma, A \vdash B \]
\[ \Gamma \vdash A \rightarrow B \]

\[ \Gamma, A_i \vdash C \]
\[ \Gamma, A_1 \ast A_2 \vdash C \]

\[ \Gamma, A_1 \vdash C \quad \Gamma, A_2 \vdash C \]
\[ \Gamma \vdash A_1 \ast A_2 \]

\[ \Gamma \vdash A_i \]
\[ \Gamma \vdash A_1 + A_2 \]

Inversible vs. non-inversible rules.
Negatives (interesting on the left): products, arrow, atoms.
Positives (interesting on the right): sum, atoms.
Inversible phase

\[
\frac{?}{X + Y \vdash X}
\]
\[
\frac{X + Y \vdash X}{X + Y \vdash X + Y}
\]

If applied too early, non-inversible rules can ruin your proof.
Inversible phase

\[
\frac{?}{X + Y \vdash X} \\
\frac{X + Y \vdash X}{X + Y \vdash X + Y}
\]

If applied too early, non-inversible rules can ruin your proof.

Focusing restriction 1: inverse phases

Inversible rules must be applied as soon and as long as possible – and their order does not matter.
Inversible phase

If applied too early, non-inversible rules can ruin your proof.

Focusing restriction 1: inversible phases

Inversible rules must be applied as soon and as long as possible – and their order does not matter.

Imposing this restriction gives a single proof of \( (X \rightarrow Y) \rightarrow (X \rightarrow Y) \) instead of two \( \lambda(f) f \) and \( \lambda(f) \lambda(x) f x \).
Non-inversible phases

After all inversible rules, $\Gamma_n \vdash A_p$

Only step forward: select a formula, apply some non-inversible rules on it.
Non-inversible phases

After all inversible rules, $\Gamma_n \vdash A_p$

Only step forward: select a formula, apply some non-inversible rules on it.

Focusing restriction 2: non-inversible phase

When a principal formula is selected for non-inversible rule, they should be applied as long as possible – until its polarity changes.
Non-inversible phases

After all inversible rules, $\Gamma_n \vdash A_p$

Only step forward: select a formula, apply some non-inversible rules on it.

Focusing restriction 2: non-inversible phase

When a principal formula is selected for non-inversible rule, they should be applied as long as possible – until its polarity changes.

Completeness: this restriction preserves provability. Non-trivial!

Example of removed redundancy:

$$
\frac{X_2, \quad Y_1 \vdash A}{X_2 \ast X_3, \quad Y_1 \vdash A}
\frac{X_2 \ast X_3, \quad Y_1 \ast Y_2 \vdash A}{X_1 \ast X_2 \ast X_3, \quad Y_1 \ast Y_2 \vdash A}
$$
This was focusing

Focused proofs are structured in alternating phases, inversible (boring) and non-inversible (focus).

Phases are forced to be as long as possible – to eliminate duplicate proofs.

The idea is independent from the proof system. Applies to sequent calculus or natural deduction; intuitionistic, classical, linear, you-name-it logic.

On proof terms, these restrictions correspond to $\beta\eta$-normal forms (at least for products and arrows). But the fun is in the search.
(1 \rightarrow X \rightarrow (Y + Z)) \rightarrow X \rightarrow (Y \rightarrow W) \rightarrow (Z + W)
Restrictive syntax

So far we’ve defined **focused proofs** as a subset of proofs in our system. Some people prefer to give them a syntax that enforces their structure.

\[
\frac{Γ; Δ, A ⊨ B}{Γ; Δ ⊨ A \to B}
\]

\[
\frac{Γ; Δ ⊨ A \quad Γ; Δ ⊨ B}{Γ; Δ ⊨ A \ast B}
\]

\[
\frac{Γ; A, Δ ⊨ C \quad Γ; B, Δ ⊨ C}{Γ; A + B, Δ ⊨ C}
\]

\[
\frac{Γ, A_n; Δ ⊨ C}{Γ; A_n, Δ ⊨ C}
\]
Restrictive syntax
So far we’ve defined **focused proofs** as a subset of proofs in our system. Some people prefer to give them a syntax that enforces their structure.

\[
\begin{align*}
\Gamma; \Delta, A \vdash B & \quad \Gamma; \Delta \vdash A \quad \Gamma; \Delta \vdash B \\
\hline
\Gamma; \Delta \vdash A \rightarrow B & \quad \Gamma; \Delta \vdash A \ast B
\end{align*}
\]

\[
\begin{align*}
\Gamma; A, \Delta \vdash C & \quad \Gamma; B, \Delta \vdash C \\
\hline
\Gamma; A + B, \Delta \vdash C & \quad \Gamma; A_n, \Delta \vdash C
\end{align*}
\]

\[
\begin{align*}
\Gamma_n \vdash \boxed{C_p} & \\
\hline
\Gamma_n; \emptyset \vdash C_p
\end{align*}
\]

\[
\begin{align*}
\Gamma_n, B_n \vdash \boxed{B_n} \vdash C_p & \\
\hline
\Gamma_n, B_n; \emptyset \vdash C_p
\end{align*}
\]
Restrictive syntax

So far we’ve defined **focused proofs** as a subset of proofs in our system. Some people prefer to give them a syntax that enforces their structure.

\[
\begin{align*}
\Gamma; \Delta, A & \vdash B \\
\hline
\Gamma; \Delta & \vdash A \rightarrow B
\end{align*}
\]

\[
\begin{align*}
\Gamma; \Delta & \vdash A \\
\hline
\Gamma; \Delta & \vdash A \ast B
\end{align*}
\]

\[
\begin{align*}
\Gamma; A, \Delta & \vdash C \\
\hline
\Gamma; A + B, \Delta & \vdash C
\end{align*}
\]

\[
\begin{align*}
\Gamma; A_n, \Delta & \vdash C \\
\hline
\Gamma; A_n & \vdash C
\end{align*}
\]

\[
\begin{align*}
\Gamma_n, B_n & \vdash \boxed{B_n} \\
\hline
\Gamma_n & \vdash C_p
\end{align*}
\]

\[
\begin{align*}
\Gamma_n & \vdash \boxed{C_p} \\
\hline
\Gamma_n; \emptyset & \vdash C_p
\end{align*}
\]

\[
\begin{align*}
\Gamma & \vdash A \\
\hline
\Gamma & \vdash \boxed{B} \\
\Gamma & \vdash A \rightarrow B \\
\hline
\Gamma & \vdash C
\end{align*}
\]
Restrictive syntax

So far we’ve defined **focused proofs** as a subset of proofs in our system. Some people prefer to give them a syntax that enforces their structure.
Success stories

Focusing was introduced by Andreoli in 1992. Revolution in logic programming.

Forward-chaining and backward-chaining expressed in a single system by assigning polarities to atoms.
Funnier stuff (magic sets?) with dynamic polarity changes.

Syntethic connectives: state-of-the-art automated theorem proving for non-classical logics (+ Jumbo connectives, Paul Blain Levy, 2006)

Lazy vs. strict: focusing, polarization (Zeilberger (2008), Munch-Maccagnoni (2013)).

A sequent calculus with cut-free search bisimilar to DPLL (Lengrand, 2013).
This is **not** the end

\[(A + A) \rightarrow A\]

\[(1 \rightarrow (A + A)) \rightarrow A\]

\[\lambda(x) \delta(x, 1, y.y, y.y)\]

\[\lambda(x) \delta(x, 1, y.\delta(x, 1, y'.y', y'.y'), y.y)\]

\[\lambda(x) \delta(x, 1, y.y, y.\delta(x, 1, y'.y, y'.y))\]

\[\ldots\]
Multi-focusing

Sometimes several independent foci are possible to make progress in a proof.

Multi-focusing (Miller and Saurin, 2007): do them all at once, in parallel.
Maximal multi-focusing

Given a focused proof, it is possible to put focused sequences in parallel to exhibit some parallelism – without changing the operational meaning of the proof, seen as a pure program.

Does there exists a maximally parallel multi-focused proof?
Maximal multi-focusing

Given a focused proof, it is possible to put focused sequences in parallel to exhibit some parallelism – without changing the operational meaning of the proof, seen as a pure program.

Does there exists a maximally parallel multi-focused proof?

Yes. (In the good logics)
Maximal multi-focusing

Given a focused proof, it is possible to put focused sequences in parallel to exhibit some parallelism – without changing the operational meaning of the proof, seen as a pure program.

Does there exists a maximally parallel multi-focused proof?

Yes. (In the good logics)

Maximally multi-focusing is a powerful notion of canonical structure for proof.

- linear logic: proof nets (Chaudhuri, Miller, Saurin, 2008)
- first-order classical logic: expansion proofs (Chaudhuri, Hetzl, Miller, 2013)

“Evolution rather than revolution” (Dale Miller)
Computing a maximal proof

**Preemptive** rewriting temporarily breaks the focused structure to move foci as far down as possible.

\[
\begin{align*}
\{ & I_3 \\
& NI_3 \\
& I_2 \\
& NI_2 \\
& I_1 \\
& NI_1 \\
\}
\end{align*}
\]

\[
\begin{align*}
& \{ \text{I2} \\
& \text{NI2} \\
& \text{I1} \\
& \text{NI1} \\
\} \\
\end{align*}
\]

\[
\begin{align*}
& \{ \text{I3} \\
& \text{NI3} \\
\} \\
\end{align*}
\]

\[
? \begin{align*}
& \{ \text{I2} \\
& \text{NI2} \\
& \text{I1} \quad \text{I3} \\
& \text{NI1} \quad \text{NI3} \\
\} \\
\end{align*}
\]

(This is the heart of the correspondence with Sam Lindley's work)
Computing a maximal proof

**Preemptive** rewriting temporarily breaks the focused structure to move foci as far down as possible.

\[
\left\{ \begin{array}{c} I_2 \\ NI_2 \\ I_1 \\ NI_1 \end{array} \right\} \rightarrow^* \left\{ \begin{array}{c} I_2 \\ NI_2 \\ I_1 \\ NI_1 \end{array} \right\} = \left\{ \begin{array}{c} I_3  \\ NI_3 \end{array} \right\} \rightarrow^* \left\{ \begin{array}{c} I_3  \\ NI_3; (I_2) \end{array} \right\} = \left\{ \begin{array}{c} I_2 \\ NI_2 \\ I_3 \\ NI_3 \\ I_1 \\ NI_1 \end{array} \right\}
\]

(This is the heart of the correspondence with Sam Lindley's work)
Computing a maximal proof

**Preemptive** rewriting temporarily breaks the focused structure to move foci as far down as possible.

\[
\begin{align*}
\{I_3, NI_3\} & \rightarrow^* \{I_2, NI_3; (I_2)\} & \rightarrow^* \{I_2, NI_2, I_3, NI_3\} \\
\{I_2, NI_2, I_1, NI_1\} & \rightarrow^* \{I_2, I_1, NI_1, NI_3\} & \{I_2, NI_2, I_1, I_3, NI_1, NI_3\} \\
\end{align*}
\]
Computing a maximal proof

Preemptive rewriting temporarily breaks the focused structure to move foci as far down as possible.

\[
\begin{align*}
\{ & I_3 \\
& NI_3 \\
i & I_2 \\
& NI_2 \\
i & I_1 \\
& NI_1 \\
\} & \rightarrow^* & \{ & I_2 \\
& NI_3; (I_2) \\
i & I_1 \\
& NI_1 \\
\} & \rightarrow^* & \{ & I_2 \\
& NI_2 \\
& I_1 \\
& NI_1 \\
\} \\
\rightarrow^* & \{ & I_2 \\
& NI_2; (I_3) \\
i & I_1 \\
& NI_3; (I_1) \\
\} & \rightarrow^* & \{ & I_2 \\
& NI_1 \\
& I_3 \\
& NI_3 \\
\} \\
? & \{ & I_2 \\
& NI_2 \\
& I_1 \\
& I_3 \\
& NI_1 \\
& NI_3 \\
\}
\end{align*}
\]
Computing a maximal proof

**Preemptive** rewriting temporarily breaks the focused structure to move foci as far down as possible.

\[
\begin{align*}
\left\{ \begin{array}{c}
I_3 \\
NI_3 \\
I_2 \\
NI_2 \\
I_1 \\
NI_1 \\
\end{array} \right\} & \rightarrow^* \left\{ \begin{array}{c}
I_2 \\
NI_2 \\
I_1 \\
NI_1 \\
\end{array} \right\} \\
& \rightarrow^* \left\{ \begin{array}{c}
I_2 \\
NI_2 \\
I_1 \\
NI_1 \\
\end{array} \right\} \\
& \rightarrow^* \left\{ \begin{array}{c}
I_2 \\
NI_2 \\
I_1 \\
NI_1 \\
\end{array} \right\}
\end{align*}
\]
Computing a maximal proof

Preemptive rewriting temporarily breaks the focused structure to move foci as far down as possible.

\[
\begin{align*}
&\{ I_3 \\ NI_3 \} \\
&\{ I_2 \\ NI_2 \\ I_1 \\ NI_1 \} \quad \rightarrow^* \quad \{ I_2 \\ NI_3; (I_2) \} \\
&\{ I_3 \} \\
&\{ I_2 \\ NI_2; (I_3) \\ I_1 \\ NI_3; (I_1) \} \quad \rightarrow^* \quad \{ I_2 \\ NI_2; (I_3) \\ I_1 \\ NI_1 \} \quad \rightarrow^* \quad \{ I_2 \\ NI_2 \\ I_1 \\ NI_3 \} \\
&\{ I_3 \} \\
&\{ I_2 \\ NI_2; (I_3) \\ I_1 \\ NI_3; (I_1) \} \quad \rightarrow^* \quad \{ I_2 \\ NI_2; (I_3) \\ I_1 \\ NI_1 \} \quad \Delta \quad \{ I_2 \\ NI_2 \\ I_1 \\ I_3 \\ NI_3 \}
\end{align*}
\]
Computing a maximal proof

**Preemptive** rewriting temporarily breaks the focused structure to move foci as far down as possible.

\[
\begin{align*}
&\left\{ \begin{array}{l}
I_3 \\
NI_3
\end{array} \right\} \\
&\quad \quad \left\{ \begin{array}{l}
I_2 \\
NI_2 \\
I_1 \\
NI_1
\end{array} \right\} \\
&\quad \quad \rightarrow^* \\
&\quad \quad \left\{ \begin{array}{l}
I_2 \\
NI_2 \\
I_1 \\
NI_1
\end{array} \right\} \\
&\quad \quad \left\{ \begin{array}{l}
I_3
\end{array} \right\}
\end{align*}
\]

\[
\begin{align*}
&\left\{ \begin{array}{l}
I_2 \\
NI_2; (I_3)
\end{array} \right\} \\
&\quad \quad \left\{ \begin{array}{l}
I_3
\end{array} \right\} \\
&\quad \quad \rightarrow^* \\
&\quad \quad \left\{ \begin{array}{l}
I_2 \\
NI_2; (I_3)
\end{array} \right\} \\
&\quad \quad \rightarrow^* \\
&\quad \quad \left\{ \begin{array}{l}
I_2 \\
NI_2 \\
I_1 \\
NI_1
\end{array} \right\} \\
&\quad \quad \left\{ \begin{array}{l}
I_3
\end{array} \right\}
\end{align*}
\]

(This is the heart of the correspondence with Sam Lindley’s work)
Conclusion

Focusing imposes extra structure on proofs, based on rules permutability (inversible, non invertible).

Multi-focusing is a natural generalization of focusing, which gives very strong canonicity.

Existing equivalence-checking algorithms can be logically justified as maximalization techniques.

Multi-focusing may help me decide unicity of types.