Focusing for code inference, a tutorial

Gabriel Scherer, Didier Rémy

Northeastern University

May 31, 2016
Context

A formal look at code inference (program synthesis).

\[ \Gamma \vdash ? : A \]

We are searching among well-typed programs for a fixed \((\Gamma, A)\).
Context

A formal look at **code inference** (program synthesis).

\[ \Gamma \vdash ? : A \]

We are searching among well-typed programs for a fixed \((\Gamma, A)\).

Many of these programs are equivalent; considering duplicates is useless.
Context

A formal look at code inference (program synthesis).

\[ \Gamma \vdash \ ? : A \]

We are searching among well-typed programs for a fixed \((\Gamma, A)\).

Many of these programs are equivalent; considering duplicates is useless.

Question: is there a restricted judgment \(\Gamma \vdash_{\text{foc}} t : A\) that is...
Context

A formal look at code inference (program synthesis).

\[ \Gamma \vdash ? : A \]

We are searching among well-typed programs for a fixed \((\Gamma, A)\).

Many of these programs are equivalent; considering duplicates is useless.

Question: is there a restricted judgment \(\Gamma \vdash_{\text{foc}} t : A\) that is

- **complete**: For any \(\Gamma \vdash t : A\), there is an equivalent \(\Gamma \vdash_{\text{foc}} t' : A\).
Context

A formal look at **code inference** (program synthesis).

\[ \Gamma \vdash ? : A \]

We are searching among well-typed programs for a fixed \((\Gamma, A)\).

Many of these programs are equivalent; considering duplicates is useless.

**Question:** is there a restricted judgment \(\Gamma \vdash_{\text{foc}} t : A\) that is

- **complete**: For any \(\Gamma \vdash t : A\), there is an equivalent \(\Gamma \vdash_{\text{foc}} t' : A\).
- **canonical**: If \(\Gamma \vdash_{\text{foc}} t : A\) and \(\Gamma \vdash_{\text{foc}} u : A\) and \(t \neq u\), then \(t\) and \(u\) are not equivalent.
Context

A formal look at **code inference** (program synthesis).

\[ \Gamma \vdash ? : A \]

We are searching among well-typed programs for a fixed \((\Gamma, A)\).

Many of these programs are equivalent; considering duplicates is useless.

Question: is there a restricted judgment \(\Gamma \vdash_{foc} t : A\) that is

- **complete**: For any \(\Gamma \vdash t : A\), there is an equivalent \(\Gamma \vdash_{foc} t' : A\).
- **canonical**: If \(\Gamma \vdash_{foc} t : A\) and \(\Gamma \vdash_{foc} u : A\) and \(t \neq u\), then \(t\) and \(u\) are not equivalent.

?
Context

A formal look at code inference (program synthesis).

\[ \Gamma \vdash ? : A \]

We are searching among well-typed programs for a fixed \((\Gamma, A)\).

Many of these programs are equivalent; considering duplicates is useless.

Question: is there a restricted judgment \(\Gamma \vdash_{\text{foc}} t : A\) that is

- complete: For any \(\Gamma \vdash t : A\), there is an equivalent \(\Gamma \vdash_{\text{foc}} t' : A\).
- canonical: If \(\Gamma \vdash_{\text{foc}} t : A\) and \(\Gamma \vdash_{\text{foc}} u : A\) and \(t \neq u\), then \(t\) and \(u\) are not equivalent.

\?

Focusing: not the complete answer (not canonical), but a good step forward.
Simply-typed lambda-calculus

\[
\begin{align*}
\Gamma, x : A & \vdash x : A \\
\Gamma, x : A & \vdash t : B \\
\Gamma & \vdash \lambda x. t : A \rightarrow B \\
\Gamma & \vdash t : A \rightarrow B 
& \Gamma \vdash u : A \\
\Gamma & \vdash t u : B \\
\Gamma & \vdash t_1 : A_1 \\
\Gamma & \vdash t_2 : A_2 \\
\Gamma & \vdash (t_1, t_2) : A_1 \times A_2 \\
\Gamma & \vdash t : A_1 \times A_2 \\
\Gamma & \vdash \pi_i t : A_i \\
\Gamma & \vdash t : A_i \\
\Gamma & \vdash \sigma_i t : A_1 + A_2 \\
\Gamma \vdash \text{match } t \text{ with} \\
\Gamma, x_1 : A_1 & \vdash u_1 : C \\
\Gamma, x_2 : A_2 & \vdash u_2 : C \\
\Gamma & \vdash \sigma_1 x_1 \rightarrow u_1 \\
\Gamma & \vdash \sigma_2 x_2 \rightarrow u_2 : C \\
\end{align*}
\]

(plus units 0 and 1)
\( \lambda \mapsto \text{sequents} \)

\[
\Gamma \vdash t : A_1 \times A_2 \quad \Rightarrow \quad \Gamma \vdash \pi_1 \ t : A_1 \\
\]  

\[
\Gamma \vdash t : A_1 \times A_2 \quad \Rightarrow \quad \Gamma \vdash A_1 \times A_2 \\
\]

\[
\Gamma, A_1 \vdash C \\
\]

\[
(, ) \text{ is non-disjoint union} \\
\]
Invertible vs. non-invertible rules. Positives vs. negatives.
Invertible phase

\[
\begin{align*}
? \\
\frac{X + Y \vdash X}{X + Y \vdash X + Y}
\end{align*}
\]

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

**Focusing restriction 1: invertible phases**

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

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

Focusing restriction 1: invertible phases

Invertible 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 \((\lambda f. \lambda x. f x)\).

After all invertible rules, negative context, positive goal.
Non-invertible phases

After all invertible rules, negative context, positive goal.
Only step forward: select a formula, apply some non-invertible rules on it.
Non-invertible phases

After all invertible rules, negative context, positive goal.
Only step forward: select a formula, apply some non-invertible rules on it.

Focusing restriction 2: non-invertible phase
When a principal formula is selected for non-invertible rule, they should be applied as long as possible – until its polarity changes.
Non-invertible phases

After all invertible rules, negative context, positive goal.

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

Focusing restriction 2: non-invertible phase

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

Completeness: this restriction preserves provability. Non-trivial!

Example of removed redundancy:

\[
\begin{align*}
X_2, & \quad Y_1 \vdash A \\
X_2 \times X_3, & \quad Y_1 \vdash A \\
X_2 \times X_3, & \quad Y_1 \times Y_2 \vdash A \\
X_1 \times X_2 \times X_3, & \quad Y_1 \times Y_2 \vdash A
\end{align*}
\]
Demo Time

\[ \vdash (1 \rightarrow X \rightarrow (Y + Z)) \rightarrow X \rightarrow (Y \rightarrow W) \rightarrow (Z + W) \]

invertible rules
Demo Time


\[
( 1 \rightarrow X \rightarrow ( Y + Z ))) \vdash X \rightarrow (Y \rightarrow W) \rightarrow (Z + W)
\]

invertible rules
( 1 \rightarrow X \rightarrow ( Y + Z ) ), \quad X \vdash ( Y \rightarrow W ) \rightarrow ( Z + W )

invertible rules
Demo Time

(1 \rightarrow X \rightarrow (Y + Z)), \quad X, \quad Y \rightarrow W \vdash Z + W

invertible rules
Demo Time

\[(1 \to X \to (Y + Z)), \quad X, \quad Y \to W \vdash Z + W\]

choice of focus
choice of focus
Demo Time

(1 → X → (Y + Z)), X, Y → W ⊢ Z + W

non-invertible rules
(1 \rightarrow X \rightarrow (Y + Z)), \quad X, \quad Y \rightarrow W \vdash Z + W

non-invertible rules
Demo Time

(1 \to X \to (Y + Z)), \quad X, \quad Y \to W \vdash Z + W

invertible rules
Demo Time

\[
\frac{Y, Y \rightarrow W \vdash Z + W \quad \quad Z \vdash Z + W}{(1 \rightarrow X \rightarrow (Y + Z)), \quad X, \quad Y \rightarrow W \vdash Z + W}
\]

invertible rules
Demo Time

\[
Y, Y \rightarrow W \vdash Z + W \\
Z \vdash Z + W
\]

(1 \rightarrow X \rightarrow (Y + Z)), \ X, \ Y \rightarrow W \vdash Z + W

choice of focus
Demo Time

\[
\begin{array}{c}
Y, Y \rightarrow W \vdash Z + W \\
Z \vdash Z + W \\
(1 \rightarrow X \rightarrow (Y + Z)), X, Y \rightarrow W \vdash Z + W
\end{array}
\]

conclusion
Focused proofs are structured in alternating phases, invertible (boring) and non-invertible (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.
Focused normal forms for $\lambda$-calculus

(Grammar with type annotations)

$$v ::= \text{values}$$
\[
| \lambda x. v \\
| (v_1, v_2) \\
| \text{match } x \text{ with } \sigma_1 x \rightarrow v_1, \sigma_2 x \rightarrow v_2 \\
| (f : P | X)
\]

$$n ::= \text{negative neutrals}$$
\[
| (x : N) \\
| \pi_i n \\
| n p
\]

$$f ::= \text{focused forms}$$
\[
| \text{let } (x : P) = n \text{ in } v \\
| (n : X^-) \\
| (p : P)
\]

$$p ::= \text{positive neutrals}$$
\[
| (x : X^+) \\
| \sigma_i p \\
| (v : N)
\]