



Confluence of a Good Calculus with Explicit Substitutions Formalized

Flávio L. C. de Moura and Leandro Oliveira Rezende

September 20, 2018

Universidade de Brasília

Explicit Substitutions

- Starting point: λ -calculus

$$\mathcal{T} ::= x \mid \lambda x. \mathcal{T} \mid \mathcal{T} \mathcal{T}$$

$$(\lambda x. t) \ u \rightarrow_{\beta} t\{x/u\}$$

Explicit Substitutions

- Starting point: λ -calculus

$$\mathcal{T} ::= x \mid \lambda x. \mathcal{T} \mid \mathcal{T} \mathcal{T}$$

$$(\lambda x. t) \ u \rightarrow_{\beta} t\{x/u\}$$

- Extending the λ -calculus with an explicit substitution operator:
calculi with explicit substitutions

$$\mathcal{T} ::= x \mid \lambda x. \mathcal{T} \mid \mathcal{T} \mathcal{T} \mid \mathcal{T}[x/\mathcal{T}]$$

Explicit Substitutions

- Starting point: λ -calculus

$$\mathcal{T} ::= x \mid \lambda x. \mathcal{T} \mid \mathcal{T} \mathcal{T}$$

$$(\lambda x. t) u \rightarrow_{\beta} t\{x/u\}$$

- Extending the λ -calculus with an explicit substitution operator:
calculi with explicit substitutions

$$\mathcal{T} ::= x \mid \lambda x. \mathcal{T} \mid \mathcal{T} \mathcal{T} \mid \mathcal{T}[x/\mathcal{T}]$$

$$(\lambda x. t) u \rightarrow t[x/u]$$

Explicit Substitutions

- Starting point: λ -calculus

$$\mathcal{T} ::= x \mid \lambda x. \mathcal{T} \mid \mathcal{T} \mathcal{T}$$

$$(\lambda x. t) u \rightarrow_{\beta} t\{x/u\}$$

- Extending the λ -calculus with an explicit substitution operator:
calculi with explicit substitutions

$$\mathcal{T} ::= x \mid \lambda x. \mathcal{T} \mid \mathcal{T} \mathcal{T} \mid \mathcal{T}[x/\mathcal{T}]$$

$$(\lambda x. t) u \rightarrow t[x/u]$$

$$x[x/u] \rightarrow u$$

$$y[x/u] \rightarrow y$$

$$(\lambda y. t')[x/u] \rightarrow \lambda y. t'[x/u]$$

$$(t' t'')[x/u] \rightarrow t'[x/u] t''[x/u]$$

Terms $a, b \quad ::= \quad \underline{1} \mid \lambda a \mid a \ b \mid a[s]$

Substituições $s, t \quad ::= \quad id \mid \uparrow \mid s.t \mid s \circ t$

- Developed by Delia Kesner [1]

$$\mathcal{T} ::= x \mid \lambda x. \mathcal{T} \mid \mathcal{T} \mathcal{T} \mid \mathcal{T}[x/\mathcal{T}]$$

$$t[x/u][y/v] =_C t[y/v][x/u], \quad \text{if } y \notin \text{fv}(u) \text{ and } x \notin \text{fv}(v)$$

$$(\lambda x. t) u \rightarrow_B t[x/u]$$

$$x[x/u] \rightarrow_{\text{Var}} u$$

$$t[x/u] \rightarrow_{\text{Gc}} t, \quad \text{if } x \notin \text{fv}(t)$$

$$(t \ v)[x/u] \rightarrow_{\text{App}} t[x/u] \ v[x/u]$$

$$(\lambda y. t)[x/u] \rightarrow_{\text{Lamb}} \lambda y. t[x/u]$$

$$t[x/u][y/v] \rightarrow_{\text{Comp}} t[y/v][x/u[y/v]], \quad \text{if } y \in \text{fv}(u)$$

- **Goal:** Formal proof of confluence (with code extraction).
 - Proof assistant: Coq.
 - Framework: Locally nameless representation (Arthur Charguéraud).
 - + No need for α -conversion (bound variables are De Bruijn indexes).
 - + No need referential contexts (named free variables).
 - The whole library is non-constructive.

$$\mathcal{T} ::= x \mid \mathcal{T}\mathcal{T} \mid \lambda x.\mathcal{T} \mid \mathcal{T}[x/\mathcal{T}]$$

Inductive *pterm* : Set :=

- | *pterm_bvar* : nat → *pterm*
- | *pterm_fvar* : var → *pterm*
- | *pterm_app* : *pterm* → *pterm* → *pterm*
- | *pterm_abs* : *pterm* → *pterm*
- | *pterm_sub* : *pterm* → *pterm* → *pterm*.

Inductive *term* : *pterm* → Prop :=

- | *term_var* : $\forall x, \text{term } (\text{pterm_fvar } x)$
- | *term_app* : $\forall t1\ t2, \text{term } t1 \rightarrow \text{term } t2 \rightarrow$
 $\text{term } (\text{pterm_app } t1\ t2)$
- | *term_abs* : $\forall L\ t1, (\forall x, x \notin L \rightarrow \text{term } (t1 \wedge x)) \rightarrow$
 $\text{term } (\text{pterm_abs } t1)$
- | *term_sub* : $\forall L\ t1\ t2, (\forall x, x \notin L \rightarrow \text{term } (t1 \wedge x)) \rightarrow$
 $\text{term } t2 \rightarrow \text{term } (\text{pterm_sub } t1\ t2).$

$$t[x/u][y/v] =_C t[y/v][x/u], \text{ if } y \notin \text{fv}(u) \text{ and } x \notin \text{fv}(v)$$

Inductive $\text{eqc} : \text{Rel } \text{pterm} :=$

| $\text{eqc_def} : \forall t u v, \text{term } u \rightarrow \text{term } v \rightarrow \text{eqc } (t[u][v]) ((\& t)[v][u]).$

Definition $\text{eqc_ctx } (t u : \text{pterm}) := \text{ES_contextual_closure } \text{eqc } t u.$

Definition $\text{eqC } (t : \text{pterm}) (u : \text{pterm}) := \text{refltrans } \text{eqc_ctx } t u.$

Notation " $t =_C u$ " $:= (\text{eqC } t u)$ (at **level** 66).

$$(\lambda x. t) \ u \rightarrow_B t[x/u]$$

Inductive $\text{rule_b} : \text{Rel } \text{pterm} :=$
 $\text{reg_rule_b} : \forall (t \ u : \text{pterm}),$
 $\text{rule_b } (\text{pterm_app}(\text{pterm_abs } t) \ u) (t[u]).$

Definition $b_ctx \ t \ u := \text{ES_contextual_closure } \text{rule_b } t \ u.$

Notation " $t \rightarrow_B u$ " $:= (b_ctx \ t \ u) \text{ (at level 66)}.$

$$\begin{array}{lll}
 x[x/u] & \rightarrow_{\text{Var}} & u \\
 t[x/u] & \rightarrow_{\text{Gc}} & t, \quad \text{if } x \notin \text{fv}(t) \\
 (t \ v)[x/u] & \rightarrow_{\text{App}} & t[x/u] \ v[x/u] \\
 (\lambda y. t)[x/u] & \rightarrow_{\text{Lamb}} & \lambda y. t[x/u] \\
 t[x/u][y/v] & \rightarrow_{\text{Comp}} & t[y/v][x/u[y/v]], \quad \text{if } y \in \text{fv}(u)
 \end{array}$$

Inductive $\text{sys_x} : \text{Rel } \text{pterm} :=$

| $\text{reg_rule_var} : \forall t, \text{sys_x} (\text{pterm_bvar } 0 \ [t]) \ t$
| $\text{reg_rule_gc} : \forall t \ u, \text{sys_x} (t[u]) \ t$
| $\text{reg_rule_app} : \forall t1 \ t2 \ u,$
 $\text{sys_x} ((\text{pterm_app } t1 \ t2)[u]) (\text{pterm_app } (t1[u]) (t2[u]))$
| $\text{reg_rule_lamb} : \forall t \ u,$
 $\text{sys_x} ((\text{pterm_abs } t)[u]) (\text{pterm_abs } ((\& \ t)[u]))$
| $\text{reg_rule_comp} : \forall t \ u \ v, \text{has_free_index } 0 \ u \rightarrow$
 $\text{sys_x} (t[u][v]) (((\& \ t)[v])[\ u \ [\ v \]]).$

Corollary *lex_is_confluent*: *Confl lex*.

Proof.

 apply *Zprop_implies_Confl*.

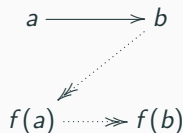
 apply *Zlex*.

Qed.

Confluence and the Z Property

Definition (Z Property)

Let (A, \rightarrow) be an abstract rewriting system (ARS). The system (A, \rightarrow) has the Z property, if there exists a map $f : A \rightarrow A$ such that:

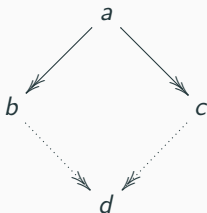


Definition $Zprop \{A:Type\} (R: Rel A) := \exists f:A \rightarrow A, \forall a b, R a b \rightarrow ((refltrans R) b (f a) \wedge (refltrans R) (f a) (f b))$.

Confluence and the Z Property

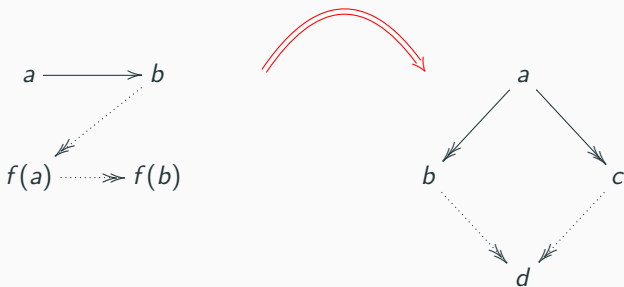
Definition (Confluence)

An ARS (A, \rightarrow) is confluent if



Definition $\text{Confl } \{A:\text{Type}\} (R: \text{Rel } A) := \forall a b c, (\text{refltrans } R) a b \rightarrow (\text{refltrans } R) a c \rightarrow (\exists d, (\text{refltrans } R) b d \wedge (\text{refltrans } R) c d).$

Confluence and the Z Property



Theorem $Zprop_implies_Confl \{A:Type\}: \forall R: Rel\ A, Zprop\ R \rightarrow Confl\ R.$



D. Kesner.

A Theory of Explicit Substitutions with Safe and Full Composition.

5(3:1):1–29, 2009.