Confluence of a Good Calculus with Explicit Substitutions Formalized

Flávio L. C. de Moura and Leandro Oliveira Rezende

September 19, 2018

Universidade de Brasília

Intuition

The λex-**calculus**

- Developed by Delia Kesner [1]

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

$t[x/u][y/v]$	$=_C$	$t[y/v][x/u],$	if $y \notin \mathtt{fv}(u)$ and $x \notin \mathtt{fv}(v)$
$(\lambda x.t)\ u$	$\rightarrow_{\mathtt{B}}$	$t[x/u]$	
$x[x/u]$	$\rightarrow_{\mathtt{Var}}$	u	
$t[x/u]$	$\rightarrow_{\mathtt{Gc}}$	$t,$	if $x \notin \mathtt{fv}(t)$
$(t\ v)[x/u]$	$\rightarrow_{\mathtt{App}}$	$t[x/u]\ v[x/u]$	
$(\lambda y.t)[x/u]$	$\rightarrow_{\mathtt{Lamb}}$	$\lambda y.t[x/u]$	
$t[x/u][y/v]$	$\rightarrow_{\mathtt{Comp}}$	$t[y/v][x/u[y/v]],$	if $y \in \mathtt{fv}(u)$

- **Goal**: Formal proof of confluence (with code extraction).
  - Proof assistant: Coq.
  - Framework: Locally nameless representation (Arthur Charguéraud).
    - $+$ No need for α-conversion.
    - $+$ No need referential contexts (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 : ∀ x, term (pterm_fvar x)
  | term_app : ∀ t1 t2, term t1 → term t2 →
                    term (pterm_app t1 t2)
  | term_abs : ∀ L t1, (∀ x, x \notin L → term (t1 ^ x)) →
                                    term (pterm_abs t1)
  | term_sub : ∀ L t1 t2, (∀ x, x \notin L → term (t1 ^ x)) →
                        term t2 → term (pterm_sub t1 t2).
```

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

```
Inductive eqc : Rel pterm :=
| eqc_def: ∀ t u v, term u → term v → eqc (t[u][v]) ((& t)[v][u]).

Definition eqc_ctx (t u: pterm) := ES_contextual_closure eqc t u.

Definition eqC (t : pterm) (u : pterm) := refltrans eqc_ctx t u.
Notation "t =_C u" := (eqC t u) (at level 66).
```

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

```
Inductive rule_b : Rel pterm :=
    reg_rule_b : ∀ (t u:pterm),
        rule_b (pterm_app(pterm_abs t) u) (t[u]).
```

```
Definition b_ctx t u := ES_contextual_closure rule_b t u.
Notation "t →ʙ u" := (b_ctx t u) (at level 66).
```

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

```
Inductive sys_x : Rel pterm :=
| reg_rule_var : ∀ t, sys_x (pterm_bvar 0 [t]) t
| reg_rule_gc : ∀ t u, sys_x (t[u]) t
| reg_rule_app : ∀ t1 t2 u,
    sys_x ((pterm_app t1 t2)[u]) (pterm_app (t1[u]) (t2[u]))
| reg_rule_lamb : ∀ t u,
    sys_x ((pterm_abs t)[u]) (pterm_abs ((& t)[u]))
| reg_rule_comp : ∀ t u v, has_free_index 0 u →
    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 $(A, \to)$ be an abstract rewriting system (ARS). The system $(A, \to)$ has the Z property, if there exists a map $f : A \to A$ such that:
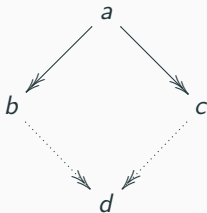
$$a \longrightarrow b$$
$$f(a) \dashrightarrow f(b)$$
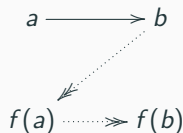
Definition *Zprop* {*A*:Type} (*R*: *Rel A*) := $\exists$ *f*:*A* $\to$ *A*, $\forall$ *a b*, *R a b* $\to$ ((*refltrans R*) *b* (*f a*) $\wedge$ (*refltrans R*) (*f a*) (*f b*)).
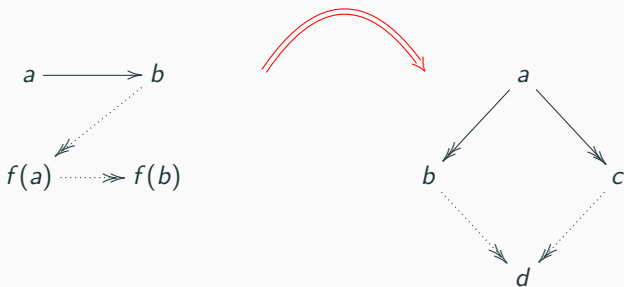
Definition (Confluence)

An ARS $(A, \rightarrow)$ is confluent if

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

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.