A Formalization of the (Compositional) Z Property

Flávio L. C. de Moura

Departamento de Ciência da Computação, Universidade de Brasília, Brazil
flaviomoura@unb.br

Abstract. Rewriting theory is a well established model of computation equivalent to the Turing machines, and the most well known rewriting system is the λ-calculus. Confluence is an important and undecidable property related to the determinism of the computational process. Direct proofs of confluence are, in general, difficult to be done. Therefore, alternative characterizations of confluence can circumvent this difficulty for different contexts. This is the case of the so called Z property, which has been successfully used to prove confluence in several situations such as the λ-calculus with $\beta\eta$-reduction, extensions of the λ-calculus with explicit substitutions, the $\lambda\mu$-calculus, etc. In this work we present a direct and constructive proof that the Z property implies confluence, which is formalized in the Coq proof assistant. In addition, we formalized an extension of the Z property, known as the Compositional Z, that has a modular approach for compositional functions.

1 Introduction

Confluence is an important and undecidable property concerning the determinism of the computational process. In this sense, one says that a program is confluent if every two ways of evaluating it, result in the very same answer. In the particular case of Abstract Rewriting Systems (ARS), which are the focus of this work, confluence can be beautifully expressed by diagrams as we will see in the next section.

The contributions of this work are as follows:

- We present a new proof that the Z property implies confluence, which is direct and constructive.
- The proof that the Z property implies confluence is formalized in the Coq proof assistant, and the presentation is made interleaving Coq code followed by an explanation in English of the code. In this way, the annotations are done directly in the Coq files using the coqdoc annotation style. We believe that this approach is interesting for those that are not familiar with the Coq proof assistant because the Coq code followed by English explanations gives a good idea on how they relate to each other. This discipline also forces a better organization of the formalization and of the proofs so that the explanation in English is comprehensible.
- We formalize an extension of the Z property, known as compositional Z property, as presented in [4].

In this section, we present a formalization of the Z property in the context of ARS, which are sets with a binary relation. A binary relation is a predicate over a type A:

`Definition` Rel (A:`Type`) := $A \to A \to$ `Prop`.

If (A, R), is an ARS and $a, b \in A$ then we write $a \to_R b$ (or $R\ a\ b$ in the Coq syntax below) to denote that $(a, b) \in R$, and in this case, we say that a R-reduces to b in one step. The transitive closure of $\to_R$, written $\to_R^+$, is defined as usual by the following inference rules:

$$\frac{a \to_R b}{a \to_R^+ b}\ (singl) \qquad\qquad \frac{a \to_R b \qquad b \to_R^+ c}{a \to_R^+ c}\ (transit)$$

This definition corresponds to the following Coq code, where $\to_R$ (resp. $\to_R^+$) corresponds to R (resp. *trans R*):

```
Inductive trans {A} (R: Rel A) : Rel A :=
| singl: ∀ a b, R a b → trans R a b
| transit: ∀ b a c, R a b → trans R b c → trans R a c.
```

The reflexive transitive closure of $\to_R$, written $\twoheadrightarrow_R$, is defined by:

$$\frac{a \to_R b}{a \twoheadrightarrow_R b}\ (refl) \qquad\qquad \frac{a \to_R b \qquad b \twoheadrightarrow_R c}{a \twoheadrightarrow_R c}\ (rtrans)$$

This definition corresponds to the following Coq code, where $\twoheadrightarrow_R$ is written as *refltrans R*:

```
Inductive refltrans {A:Type} (R: Rel A) : A → A → Prop :=
| refl: ∀ a, refltrans R a a
| rtrans: ∀ a b c, R a b → refltrans R b c → refltrans R a c.
```

The reflexive transitive closure of a relation is used to define the notion of confluence: no matter how the reduction is done, the result will always be the same. In other words, every divergence is joinable as stated by the following diagram:

$$\begin{tikzcd}
 & a \arrow[dl, ->>] \arrow[dr, ->>] & \\
b \arrow[dr, dotted, ->>] & & c \arrow[dl, dotted, ->>] \\
 & d &
\end{tikzcd}$$

Formally, this means that if an expression a can be reduced in two different ways to b and c, then there exists an expression d such that both b and c reduce to d. The existential quantification is expressed by the dotted lines in the diagram. This notion is defined in the Coq system as follows:

```
Definition Confl {A:Type} (R: Rel A) := ∀ a b c, (refltrans R) a b → (refltrans R) a c → (∃ d, (refltrans
R) b d ∧ (refltrans R) c d).
```

In [1], V. van Oostrom gives a sufficient condition for an ARS to be confluent. This condition is based on the *Z Property* that is defined as follows:

Definition 1. *Let $(A, \to_R)$ be an ARS. A mapping $f : A \to A$ satisfies the Z property for $\to_R$, if $a \to_R b$ implies $b \twoheadrightarrow_R fa \twoheadrightarrow_R fb$, for any $a, b \in A$.*

The name of the property comes from the following diagrammatic representation of this definition:

$$\begin{tikzcd}
a \arrow[r, "R"'] & b \arrow[dl, dotted, ->>, "R"] \\
fa \arrow[r, dotted, ->>, "R"'] & fb
\end{tikzcd}$$

If a function f satisfies the Z property for $\to_R$ then we say that f is Z for $\to_R$, and the corresponding Coq definition is given by the following predicate:

```
Definition f_is_Z {A:Type} (R: Rel A) (f: A → A) := ∀ a b, R a b → ((refltrans R) b (f a) ∧ (refltrans
R) (f a) (f b)).
```

Alternatively, an ARS $(A, \to_R)$ satisfies the Z property if there exists a mapping $f : A \to A$ such that f is Z for $\to_R$:

```
Definition Z_prop {A:Type} (R: Rel A) := ∃ f:A → A, ∀ a b, R a b → ((refltrans R) b (f a) ∧ (refltrans
R) (f a) (f b)).
```

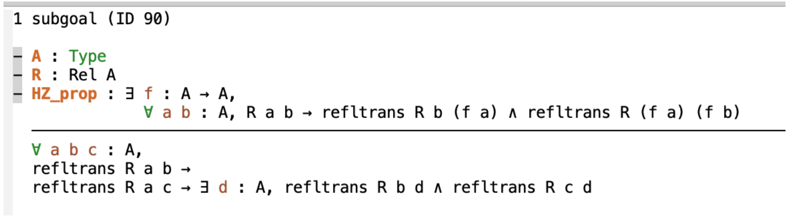
The first contribution of this work is a constructive proof of the fact that the Z property implies confluence. Our proof uses nested induction, and hence it differs from the one in [**?**] (that follows [1]) and the one in [**?**]

in the sense that it does not rely on the analyses of whether a term is in normal form or not, avoiding the necessity of the law of the excluded middle. As a result, we have an elegant inductive proof of the fact that if an ARS satisfies the Z property then it is confluent.

`Theorem` *Z_prop_implies_Confl* {*A*:`Type`}: ∀ *R*: *Rel A*, *Z_prop R* → *Confl R*.
`Proof`.

`intros` *R HZ_prop*. — Let R be a relation over A that satisfies the Z property, which will be denoted by *HZ_prop* for future reference.

`unfold` *Z_prop*, *Confl* `in` *. — Unfolding both definitions of *Z_prop* and *Confl*, we get the following proof context:



```
1 subgoal (ID 90)

- A : Type
- R : Rel A
- HZ_prop : ∃ f : A → A,
                ∀ a b : A, R a b → refltrans R b (f a) ∧ refltrans R (f a) (f b)
  ______________________________________________________________________________
  ∀ a b c : A,
  refltrans R a b →
  refltrans R a c → ∃ d : A, refltrans R b d ∧ refltrans R c d
```

`intros` *a b c Hrefl1 Hrefl2*. — Let a, b and c be elements of the set A, $Hrefl1$ the hypothesis that $a \twoheadrightarrow_R b$, and $Hrefl2$ the hypothesis that $a \twoheadrightarrow_R c$. We need to prove that there exists d such that $b \twoheadrightarrow_R d$ and $c \twoheadrightarrow_R d$.

`destruct` *HZ_prop* `as` [*g HZ_prop*]. — We know from the hypothesis *HZ_prop* that there exists a mapping f that is Z. Let's call g this mapping, and we get following proof context:
“includegraphics*scale*=0.6{figs/fig4.png}
The proof proceeds by nested induction, firstly on the length of the reduction from a to b, and then on the length of the reduction from a to c.

`generalize dependent` *c*. — Before the first induction, i.e. induction on $Hrefl1$, the element c needs to be generalized so that it can be afterwards instantiated with any reduct of a.

`induction` *Hrefl1*. — The induction on $Hrefl1$ corresponds to induction on the reflexive transitive closure of the relation R, and since $refltrans$ has two rules, the goal splits in two subgoals, one for each possible way of constructing $a \twoheadrightarrow_R b$.

- `intros` *c Hrefl2*. — In the first case, we have that $b = a$ since we are in the reflexive case. This means that we have to prove that there exists d, such that $a \twoheadrightarrow_R d$ and $c \twoheadrightarrow_R d$.

∃ *c*; `split`. — Taking d as c, the proof is simplified to $a \twoheadrightarrow_R c$ and $c \twoheadrightarrow_R c$.

+ `assumption`. — The first component is exactly the hypothesis $Hrefl2$ and

+ `apply` *refl*. — $c \twoheadrightarrow_R c$ corresponds to an application of the $refl$ axiom.

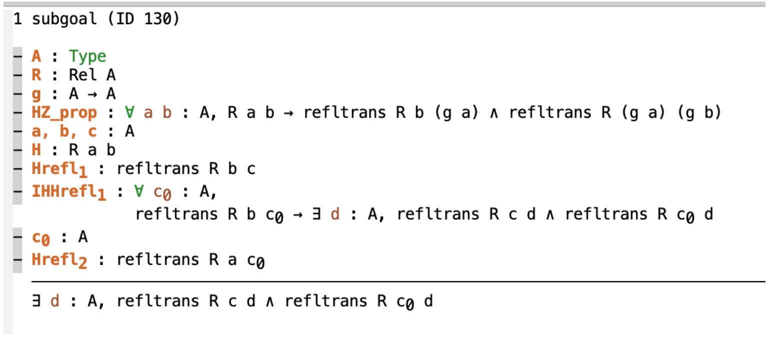
The interesting part of the proof is then given by the inductive case, i.e. when $a \twoheadrightarrow_R b$ is generated by the rule (*rtrans*). In this case, the reduction from a to b is done in at least one step, therefore there must exists an element a' such that the following diagram holds.

$$\begin{tikzcd}
 & & a \arrow[dl] \arrow[dr, two heads] & \\
 & a' \arrow[dl, two heads] & & c \arrow[ddll, dotted, two heads] \\
b \arrow[dr, dotted, two heads] & & & \\
 & d & &
\end{tikzcd}$$

The induction hypothesis states that every divergence from a' that reduces to b from one side converges: *IHHrefl1* : $\forall c_0 : A, a' \twoheadrightarrow_R c_0 \to (\exists d : A, b \twoheadrightarrow_R d \wedge c_0 \twoheadrightarrow_R d)$. Now, we'd like apply induction on the hypothesis *Hrefl2* (a“tto_R c), but the current proof context has the hypothesis H: $a \to_R a'$ (a reduces to a' in one step), and hence it is the sole hypothesis depending on a in the current proof context. If we were to apply induction on *Hrefl2* now, the generated induction hypothesis *IHrefl2* would assume that there is a term a'' such that $a \to_R a'' \twoheadrightarrow_R c$ and would require that $a'' \to_R a'$, which is generally false. In order to circumvent this problem, we need to discard the hypothesis H from our proof context, and replace it by another relevant information derived from the Z property as shown in what follows.

- `intros` *c0 Hrefl2*.

Let c_0 be a reduct of a, and $Hrefl2$ be the hypothesis $a \twoheadrightarrow_R c_0$. So the reduction $a \twoheadrightarrow_R c$ in the above diagram is now $a \twoheadrightarrow_R c_0$ due to a renaming of variables automatically done by the Coq system. In addition, the reduction $a \to_R a' \twoheadrightarrow_R b$ is now $a \to_R b \twoheadrightarrow_R c$, as shown below:



```
1 subgoal (ID 130)

A : Type
R : Rel A
g : A → A
HZ_prop : ∀ a b : A, R a b → refltrans R b (g a) ∧ refltrans R (g a) (g b)
a, b, c : A
H : R a b
Hrefl₁ : refltrans R b c
IHHrefl₁ : ∀ c₀ : A,
           refltrans R b c₀ → ∃ d : A, refltrans R c d ∧ refltrans R c₀ d
c₀ : A
Hrefl₂ : refltrans R a c₀
______________________________________
∃ d : A, refltrans R c d ∧ refltrans R c₀ d
```

Before applying induction to $Hrefl2$: $a \twoheadrightarrow_R c_0$, we will derive $b \twoheadrightarrow_R (g\ a)$ and $a \twoheadrightarrow_R (g\ a)$ from the proof context so we can discard the hypothesis H: $a \to_R$.

`assert` (*Hbga*: *refltrans R b* (*g a*)).
{ `apply` *HZ_prop*; `assumption`. }

We call $Hbga$ the reduction $b \twoheadrightarrow_R (g\ a)$ that is directly obtained from the Z property.

`assert` (*Haga*: *refltrans R a* (*g a*)).

`{ apply rtrans with b; assumption. }`

Call $Haga$ the reduction $a \twoheadrightarrow_R (g\ a)$, and prove it using the transitivity of $\twoheadrightarrow_R$, since $a \to_R b$ and $b \twoheadrightarrow_R (g\ a)$. Diagrammatically, we change from the situation on the top to the bottomone on the right:

$$\begin{tikzcd}
 & & a \arrow[dl, "R"] \arrow[ddrr, two heads, "R"] & & \\
 & b \arrow[dl, two heads, "R"] & & & \\
c \arrow[ddrr, dotted, two heads, "R"] & & & & c_0 \arrow[ddll, dotted, two heads, "R"'] \\
 & & & & \\
 & & d & &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & a \arrow[dd, two heads, "R"] \arrow[ddrr, two heads, "R"] & & \\
 & b \arrow[dl, two heads, "R"] \arrow[dr, two heads, "R"] & & & \\
c \arrow[ddrr, dotted, two heads, "R"] & & (g\ a) & & c_0 \arrow[ddll, dotted, two heads, "R"'] \\
 & & & & \\
 & & d & &
\end{tikzcd}$$

`clear H. generalize dependent b.`

At this point we can remove the hypothesis H from the context, and generalize b. Doing so, we generalize $IHHrefl1$, which, in conjunction with the hypotheses that depend on a (namely, $Hrefl2$, $Hbga$, and $Haga$), will form the four necessary conditions for use of the second inductive hypothesis, $IHHrefl2$.

`induction Hrefl2.`

Now we are ready to start the induction on the reduction $a \twoheadrightarrow_R c_0$, and we have two subgoals.

`+ intros b Hrefl1 IHHrefl1 Hbga.`

The first subgoal corresponds to the reflexive case that is closed by the induction hypothesis $IHHrefl1$:

$$\begin{tikzcd}
 & & a \arrow[dd, two heads, "H2"] \\
 & b \arrow[dl, two heads, "Hrefl1"'] \arrow[dr, two heads, "H1"] & \\
c \arrow[dr, dotted, two heads] & IHHrefl1 & (g\ a) \arrow[dl, dotted, two heads] \\
 & d &
\end{tikzcd}$$

`assert` $(\mathit{IHHrefl1\_ga} := \mathit{IHHrefl1}\ (g\ a))$;

`apply` $\mathit{IHHrefl1\_ga}$ `in` Hbga.

In order to apply $IHHrefl1$, we instantiate c_0 with $(g\ a)$.

`destruct` Hbga.

Therefore, there exists an element, say x, such that both $c \twoheadrightarrow_R x$ and $(g\ a) \twoheadrightarrow_R x$.

$\exists\ x$; `split`.

We then take x to show that $c \twoheadrightarrow_R x$ and $a \twoheadrightarrow_R x$.

× `apply` H.

Note that $c \twoheadrightarrow_R x$ is already an hypothesis, and we are done.

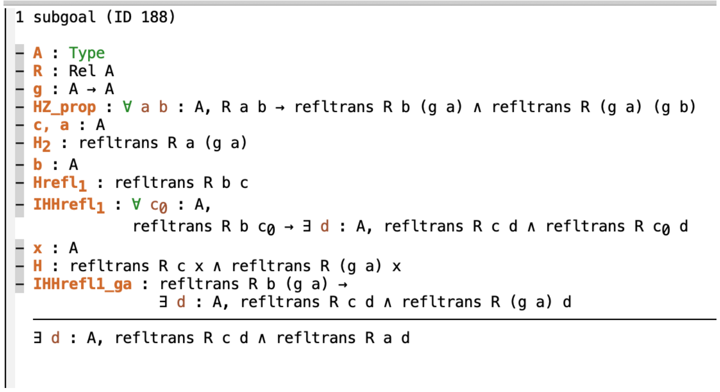
× `apply` $\mathit{refltrans\_composition}$ `with` $(g\ a)$;

[`assumption` | `apply` H].

The proof of $a \twoheadrightarrow_R x$ is done by the transitivity of $\twoheadrightarrow_R$ taking $(g\ a)$ as the intermediate step.

+ `intros` $\mathit{b0}\ \mathit{Hrefl1}\ \mathit{IHHrefl1}\ \mathit{Hb0ga}$.

The second subgoal corresponds to the case in which $a \twoheadrightarrow_R c_0$ is generated by the rule $(rtrans)$. Therefore, there exists a term b such that $a \to_R b$ and $b \twoheadrightarrow_R c_0$. The corresponding proof context after introducing the universally quantified variable $b0$, the hypothesis $Hrefl1$ and the induction hypothesis $IHHrefl1$ generated by the first outer induction and the fact that $b0 \twoheadrightarrow_R (g\ a)$ is given by:



```
1 subgoal (ID 188)

- A : Type
- R : Rel A
- g : A → A
- HZ_prop : ∀ a b : A, R a b → refltrans R b (g a) ∧ refltrans R (g a) (g b)
- c, a : A
- H2 : refltrans R a (g a)
- b : A
- Hrefl1 : refltrans R b c
- IHHrefl1 : ∀ c0 : A,
             refltrans R b c0 → ∃ d : A, refltrans R c d ∧ refltrans R c0 d
- x : A
- H : refltrans R c x ∧ refltrans R (g a) x
- IHHrefl1_ga : refltrans R b (g a) →
                ∃ d : A, refltrans R c d ∧ refltrans R (g a) d
____________________________________________________________________
  ∃ d : A, refltrans R c d ∧ refltrans R a d
```

`apply` $\mathit{IHHrefl2}$ `with` $\mathit{b0}$.

The second goal, i.e. the inductive case is the consequent on $IHHrefl2$, so we can apply $IHHrefl2$ to prove it. Doing so, we must prove the antecedent of $IHHrefl2$, which consists of four separate hypotheses that we must prove. Those hypotheses are as follows:

× `apply` $\mathit{refltrans\_composition}$ `with` $(g\ a)$;

`apply` $\mathit{HZ\_prop}$; `assumption`.

1. $b \twoheadrightarrow_R (g\ b)$: This is proved by the transitivity of the reflexive transitive closure of R using the hypothesis (H: $a \to_R b$) and HZ_prop: $\forall a\ b : a \to_R b \to (b \twoheadrightarrow_R (g\ a) \wedge (g\ a) \twoheadrightarrow_R (g\ b))$.

× `assumption`.

2. $b0 \twoheadrightarrow_R c$: This is exactly the hypothesis $Hrefl1$.

× `assumption`.

3. $\forall c0 : b0 \twoheadrightarrow_R c0 \to (\exists d : c \twoheadrightarrow_R d \wedge c0 \twoheadrightarrow_R d)$: This is exactly the induction hypothesis $IHHrefl1$.

× `apply` $\mathit{refltrans\_composition}$ `with` $(g\ a)$;

[assumption | apply HZ_prop; assumption].

4. $b0 \twoheadrightarrow_R (g\ b)$: This is proved by the transitivity of the reflexive transitive closure of R using the hypothesis H': $b0 \twoheadrightarrow_R (g\ a)$ and the fact that $(g\ a) \twoheadrightarrow_R (g\ b)$ that is obtained from the fact that R satisfies the Z property (hypothesis HZ_prop).

Qed.

Definition SemiConfl {A:Type} (R: Rel A) := ∀ a b c, R a b → (refltrans R) a c → (∃ d, (refltrans R) b d ∧ (refltrans R) c d).

Theorem Z_prop_implies_SemiConfl {A:Type}: ∀ R: Rel A, Z_prop R → SemiConfl R.

Theorem Semi_equiv_Confl {A: Type}: ∀ R: Rel A, Confl R ↔ SemiConfl R.

Corollary Zprop_implies_Confl_via_SemiConfl {A:Type}: ∀ R: Rel A, Z_prop R → Confl R.

2 An extension of the Z property: Compositional Z

Definition f_is_weak_Z {A} (R R': Rel A) (f: A → A) := ∀ a b, R a b → ((refltrans R') b (f a) ∧ (refltrans R') (f a) (f b)).

Definition comp {A} (f1 f2: A → A) := fun x:A ⇒ f1 (f2 x).
Notation "f1 # f2" := (comp f1 f2) (at level 40).

Inductive union {A} (red1 red2: Rel A) : Rel A :=
| union_left: ∀ a b, red1 a b → union red1 red2 a b
| union_right: ∀ a b, red2 a b → union red1 red2 a b.
Notation "R1 !_! R2" := (union R1 R2) (at level 40).

Lemma union_or {A}: ∀ (r1 r2: Rel A) (a b: A), (r1 !_! r2) a b ↔ (r1 a b) ∨ (r2 a b).
Require Import Setoid.
Require Import ZArith.

Lemma equiv_refltrans {A}: ∀ (R R1 R2: Rel A), (∀ x y, R x y ↔ (R1 !_! R2) x y) → ∀ x y, refltrans (R1 !_! R2) x y → refltrans R x y.

Definition Z_comp {A:Type} (R :Rel A) := ∃ (R1 R2: Rel A) (f1 f2: A → A), (∀ x y, R x y ↔ (R1 !_! R2) x y) ∧ f_is_Z R1 f1 ∧ (∀ a b, R1 a b → (refltrans R) ((f2 # f1) a) ((f2 # f1) b)) ∧ (∀ a b, b = f1 a → (refltrans R) b (f2 b)) ∧ (f_is_weak_Z R2 R (f2 # f1)).

Lemma refltrans_union {A:Type}: ∀ (R R' :Rel A) (a b: A), refltrans R a b → refltrans (R !_! R') a b.

Require Import Setoid.
Lemma refltrans_union_equiv {A}: ∀ (R R1 R2 : Rel A), (∀ (x y : A), (R x y ↔ (R1 !_! R2) x y)) → ∀ (x y: A), refltrans (R1 !_! R2) x y → refltrans R x y.

Theorem Z_comp_implies_Z_prop {A:Type}: ∀ (R :Rel A), Z_comp R → Z_prop R.

Now we can use the proofs of the theorems Z_comp_implies_Z_prop and Z_prop_implies_Confl to conclude that compositional Z is a sufficient condition for confluence.

Corollary Z_comp_is_Confl {A}: ∀ (R: Rel A), Z_comp R → Confl R.

Theorem Z_comp_thm {A:Type}: ∀ (R :Rel A) (R1 R2: Rel A) (f1 f2: A → A), (∀ x y, R x y ↔ (R1 !_! R2) x y) ∧ f_is_Z R1 f1 ∧ (∀ a b, R1 a b → (refltrans R) ((f2 # f1) a) ((f2 # f1) b)) ∧ (∀ a b, b = f1 a → (refltrans R) b (f2 b)) ∧ (f_is_weak_Z R2 R (f2 # f1)) → f_is_Z R (f2 # f1).

Corollary Z_comp_eq_corol {A:Type}: ∀ (R :Rel A) (R1 R2: Rel A) (f1 f2: A → A), (∀ x y, R x y ↔ (R1 !_! R2) x y) ∧ (∀ a b, R1 a b → (f1 a) = (f1 b)) ∧ (∀ a, (refltrans R1) a (f1 a)) ∧ (∀ b a, a = f1 b → (refltrans R) a (f2 a)) ∧ (f_is_weak_Z R2 R (f2 # f1)) → f_is_Z R (f2 # f1).

Definition Z_comp_eq $\{A{:}\mathtt{Type}\}$ $(R : Rel\ A) := \exists\ (R1\ R2{:}\ Rel\ A)\ (f1\ f2{:}\ A \to A), (\forall\ x\ y,\ R\ x\ y \leftrightarrow (R1\ !\_!\ R2)\ x\ y) \land (\forall\ a\ b,\ R1\ a\ b \to (f1\ a) = (f1\ b)) \land (\forall\ a, (refltrans\ R1)\ a\ (f1\ a)) \land (\forall\ b\ a,\ a = f1\ b \to (refltrans\ R)\ a\ (f2\ a)) \land (f\_is\_weak\_Z\ R2\ R\ (f2\ \#\ f1))$.

Lemma $Z\_comp\_eq\_implies\_Z\_comp$ $\{A{:}\mathtt{Type}\}$: $\forall\ (R : Rel\ A),\ Z\_comp\_eq\ R \to Z\_comp\ R$.

Lemma $Z\_comp\_eq\_implies\_Z\_prop$ $\{A{:}\mathtt{Type}\}$: $\forall\ (R : Rel\ A),\ Z\_comp\_eq\ R \to Z\_prop\ R$.

3 Conclusion

In this work we presented a constructive proof that the Z property implies confluence, an important property for rewriting systems. In addition, we formally proved this result in the Coq proof assistant. The corresponding files are available in our GitHub repository: https://github.com/flaviodemoura/Zproperty.

The Z property was presented by V. van Oostrom as a sufficient condition for an ARS to be confluent [2], and since then has been used to prove confluence in different contexts such as the λ-calculus with $\beta\eta$-reduction, extensions of the λ-calculus with explicit substitutions and the $\lambda\mu$-calculus. The Coq proofs of the main results are commented line by line which serve both as an informal presentation of the proofs (i.e. proofs explained in natural language) and as its formal counterpart. Moreover, we formalize an extension of the Z property, known as compositional Z property, as presented in [4]

As future work, this formalization will be used to prove the confluence property of a calculus with explicit substitution based on the λ_{ex}-calculus (cf. [3]). In addition, we hope that our formalization can be used as a framework for proving confluence of others rewriting systems.

References

1. P Dehornoy and V van Oostrom. Z, proving confluence by monotonic single-step upperbound functions. In *Logical Models of Reasoning and Computation (LMRC-08)*, 2008.
2. B. Felgenhauer, J. Nagele, V. van Oostrom, and C. Sternagel. The Z property. *Archive of Formal Proofs*, 2016.
3. D. Kesner. A Theory of Explicit Substitutions with Safe and Full Composition. *Logical Methods in Computer Science*, 5(3:1):1–29, 2009.
4. Koji Nakazawa and Ken-etsu Fujita. Compositional Z: confluence proofs for permutative conversion. *Studia Logica*, 104(6):1205–1224, 2016.