A Formalization of the (Compositional) Z Property

Flávio L. C. de Moura and Leandro O. Rezende
Dept. de Ciência da Computação
Universidade de Brasília
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. In addition, we formalized our proof and an extension of the Z property, known as the Compositional Z, in the Coq proof assistant.

1 Introduction

Confluence is an important and undecidable property concerning the determinism of the computational process. This means that independently of the choice of the evaluation path, the result is always the same. 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 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.
- We formalize an extension of the Z property, known as compositional Z property, as presented in [NF16]. The proofs are presented by interleaving Coq code followed by an explanation in English of the corresponding 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.

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

2 The Z property implies Confluence

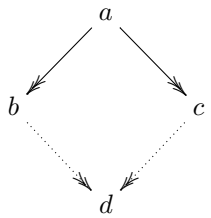
An ARS, say (A, R), is a pair composed of a set A and binary relation over this set $R : A \times A$. Let $a, b \in A$. We write $a \to_R b$ (or $R\ a\ b$ in the Coq syntax below) to denote that $(a, b) \in R$, and in this case, we say that a R-reduces to b in one step. The reflexive transitive closure of a relation R, written as $\twoheadrightarrow_R$, is defined by the following inference rules:

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

where a, b and c are universally quantified variables as explicitly stated in the corresponding Coq definition:

```
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:



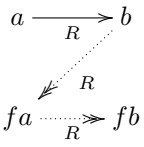
Formally, this means that if an expression a can be reduced in two different ways to b and c, then there exists an expression d such that both b and c reduce to d. The existential quantification is expressed by the dotted lines in the diagram. This notion is defined in the Coq system as follows:

```
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 [DvO08], 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 2.1. *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:



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 [Kes09] (that follows [DvO08]) and the one in [FNvOS16] 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. This proof is formalized in the Coq proof assistant, and the whole formalization is available in a GitHub repository[1]. The following proof is the first contribution of this work:

Theorem 2.1. *[DvO08] If there exists a mapping satisfying the Z property for an abstract rewriting system, then it is confluent.*

Proof. Let $(A, \to)$ be an ARS, and $f : A \to A$ a function that is Z for $\to$. Let $b \twoheadleftarrow a \twoheadrightarrow c$ be an arbitrary divergence. The proof proceeds by induction on the reduction $a \twoheadrightarrow b$, and according to the definition of the reflexive transitive closure of a relation, we have two cases: in the first case, $a = b$ and we are done. Otherwise, $a \to b' \twoheadrightarrow b$ for some term b'. We now want to proceed by induction on the reduction $a \twoheadrightarrow c$, for which the first case is also trivial, but the other case cannot be proved using the last generated induction hypothesis. In fact, suppose that $a \to c' \twoheadrightarrow c$, for some term c'. The induction hypothesis now refers to c' and the hypothesis $a \to b'$ is transformed in the condition $c' \to b'$, which is not provable, therefore we get stuck. The way to circumvent this problem is by reorganizing the proof context by removing the hypothesis $a \to b$ before starting the inner induction on $a \twoheadrightarrow c$. The Z property will allow this reorganization because the hypothesis $a \to b$ will be replaced by $b \twoheadrightarrow (f\ a)$ that is straightforward from the Z property, and by $a \twoheadrightarrow (f\ a)$ that is obtained from the transitivity of $\to$. The new context allows us to proceed by induction on the reduction $a \twoheadrightarrow c$, as expected because the hypothesis $a \twoheadrightarrow (f\ a)$ is transformed in the condition $c' \twoheadrightarrow (f\ c')$ which can be proved using the Z property. The details of the whole proof can be found in the technical report[1]. □

The interesting point of the above proof is that it reveals the difficulties that are usually made invisible by graphical proofs. In the next section, we present the formalization of an extension of the Z property known as Compositional Z.

3 An extension of the Z property: The Compositional Z property

In this section we present a formalization of the *Compositional Z*, as presented in [NF16], which is an interesting property because it allows a kind of modular approach to the Z property when the reduction relation can be split into smaller relations. More precisely, given an ARS $(A, \to_R)$, one must be able to decompose the relation $\to_R$ into two parts, say $\to_1$ and $\to_2$ such that $\to_R = \to_1 \cup \to_2$. This kind of decomposition can be done in several interesting situations such as the λ-calculus with $\beta\eta$-reduction[Bar84], extensions of the λ-calculus with explicit substitutions[ACCL91], the $\lambda\mu$-calculus[Par92], etc. But before presenting the full definition of the Compositional Z, we need to define the *weak Z property*:

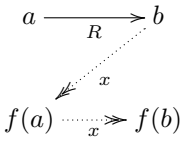


Figure 1: The weak Z property

Definition 3.1. *Let $(A, \to_R)$ be an ARS and $\to'_R$ be another relation on A. A mapping f satisfies the weak Z property for $\to_R$ by $\to'_R$ if $a \to_R b$ implies $b \twoheadrightarrow'_R f(a)$ and $f(a) \twoheadrightarrow'_R f(b)$ (cf. Figure 1). Therefore, a mapping f satisfies the Z property for $\to_R$ if it satisfies the weak Z property by itself.*

When f satisfies the weak Z property, we also say that f is weakly Z, and the corresponding definition in Coq is given as follows:

```
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)).
```

The compositional Z is then defined as follows:

Definition 3.2. *[NF16] Let $(A, \to)$ be an ARS such that $\to = \to_1 \cup \to_2$. The $(A, \to)$ satisfies the compositional Z if there exist mappings $f_1 : A \to A$ and $f_2 : A \to A$ such that:*

[1] https://github.com/flaviodemoura/Zproperty

1. f_1 *is* Z *for* $\rightarrow_1$

2. $a \rightarrow_1 b$ *implies* $f_2\ a \twoheadrightarrow f_2\ b$

3. $a \twoheadrightarrow f_2\ a$ *holds for any* $a \in Im(f_1)$

4. $f_2 \circ f_1$ *is weakly* Z *for* $\rightarrow_2$ *by* $\rightarrow$.

The corresponding definition in the Coq proof assistant is below, where the composition of $f_2 \circ f_1$ is written as *f2 # f1*, i.e. $(f2\ \#\ f1)x = f2(f1\ x)$, and the union $\rightarrow_1 \cup \rightarrow_2$ written as *R1 !_! R2*.

`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 a*) (*f2 b*)) ∧ (∀ *a b*, *b* = *f1 a* → (*refltrans R*) *b* (*f2 b*)) ∧ (*f_is_weak_Z R2 R* (*f2 # f1*)).

The next theorem is proved by presenting the Coq proof interleaved with natural language explaining the corresponding Coq steps. An immediate consequence of this theorem is the confluence of the reduction relation R.

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

`intros` *R H*. | Let R be a relation over A, and H the hypothesis that R satisfies the compositional Z.

`unfold` *Z_prop*. `unfold` *Z_comp* `in` *H*. `destruct` *H* `as` [*R1* [*R2* [*f1* [*f2* [*Hunion* [*H1* [*H2* [*H3 H4*]]]]]]]]. | After unfolding the definitions *Z_prop* and *Z_comp*, we need to prove the existence of a map, say f, that is Z as shown by the current proof context:

```
1 subgoal (ID 207)

  A : Type
  R, R1, R2 : Rel A
  f1, f2 : A -> A
  Hunion : forall x y : A, R x y <-> (R1 !_! R2) x y
  H1 : f_is_Z R1 f1
  H2 : forall a b : A, R1 a b -> refltrans R ((f2 # f1) a) ((f2 # f1) b)
  H3 : forall a b : A, b = f1 a -> refltrans R b (f2 b)
  H4 : f_is_weak_Z R2 R (f2 # f1)
  ============================
  exists f : A -> A, forall a b : A, R a b -> refltrans R b (f a) /\ refltrans R
(f a) (f b)
```

∃ (*f2 # f1*). | We will prove that the composition $(f_2 \circ f_1)$ is Z.

`intros` *a b HR*. | Let a and b be elements of A, and suppose that a R-reduces to b in one step, i.e. that $a \rightarrow_R b$ and call HR this hypothesis.

`apply` *Hunion* `in` *HR*. `inversion` *HR*; `subst`. `clear` *H*. | Since R is the union of $R1$ and $R2$, one has that a reduces to b in one step via either $R1$ or $R2$. Therefore, there are two cases to consider:

- `split`. | Firstly, suppose that a $R1$-reduces in one step to b, i.e. $a \rightarrow_{R1} b$.

  + `apply` *refltrans_composition* `with` (*f1 a*). | In order to prove that $b \twoheadrightarrow_R (f_2(f_1\ a))$, we first need to show that $b \twoheadrightarrow_{R1} (f_1\ a)$, and then that $(f_1\ a) \twoheadrightarrow_R (f_2(f_1\ a))$.

    × `apply` *H1* `in` *H*. `destruct` *H* `as` [*Hb Hf*]. `apply` (*refltrans_union R1 R2*) `in` *Hb*.
    `apply` *refltrans_union_equiv* `with` *R1 R2*.
    ** `apply` *Hunion*.
    ** `apply` *Hb*. | The proof of $b \twoheadrightarrow_{R1} (f_1\ a)$ is done from the fact that f_1 is Z for $R1$. An ancillary lemma is used to allow for the reflexive closure of R to be used on Hunion.

    × `apply` *H3* `with` *a*; `reflexivity`. | The proof that $(f_1\ a) \twoheadrightarrow_R (f_2(f_1\ a))$ is a direct consequence of the hypothesis $H3$.

  + `apply` *H2*; `assumption`. | The proof that $(f_2(f_1\ a))$ R-reduces to $(f_2(f_1\ b))$ is a direct consequence of the hypothesis $H2$.

- `apply` *H4*; `assumption`. | Finally, when a $R2$-reduces in one step to b one concludes the proof using the assumption that $(f_2 \circ f_1)$ is weak Z.

`Qed`.

Rewriting Systems with equations is another interesting and non-trivial topic [Win89, Ter03]. The confluence of rewriting systems with an equivalence relation can also be proved by a variant of the compositional Z, known as Z property modulo [AK12]. We define the predicate *Z_comp_eq* corresponding to the Z property modulo, and prove that it implies the compositional Z.

```
Definition Z_comp_eq {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)).

Lemma Z_comp_eq_implies_Z_comp {A:Type}: ∀ (R : Rel A), Z_comp_eq R → Z_comp R.
Proof.
  intros R Heq. unfold Z_comp_eq in Heq.
```
Let R be a relation that satisfies the predicate *Z_comp_eq*.

```
  destruct Heq as [R1 [R2 [f1 [f2 [Hunion [H1 [H2 [H3 H4]]]]]]]].
```
Call Hi the ith hypothesis ($1 \leq i \leq 4$) of the definition of the predicate *Z_comp_eq*, as in the previous definition.

```
  unfold Z_comp. ∃ (R1, R2, f1, f2).
```
Following the definition of the predicate *Z_comp*, we will show that the relations *R1* and *R2*, and the functions *f1* and *f2* satisfy the required conditions.

```
  split.
  - assumption.
```
The first condition, which states that R is the union of $R1$ with $R2$ is an hypothesis, and there is nothing to do.

```
  - split.
    + unfold f_is_Z. intros a b H; split.
```
As the second condition, we need to prove that *f1* is Z for *R1*. So, assuming that $a \rightarrow_{R1} b$, we will prove that $b \twoheadrightarrow_{R1} (f_1\ a)$ and $(f_1\ a) \twoheadrightarrow_{R1} (f_1\ b)$.

```
      × apply H1 in H. rewrite H. apply H2.
```
From hypothesis $H1$, we know that $f_1\ a = f_1\ b$, since $a \rightarrow_{R1} b$. Therefore, we need to prove that $b \twoheadrightarrow_{R1} (f_1\ b)$ that is straightforward from hypothesis $H2$.

```
      × apply H1 in H. rewrite H. apply refl.
```
Similarly to the previous case, using the fact that $f_1\ a = f_1\ b$ we complete this case by the reflexivity of $\twoheadrightarrow_{R1}$.

```
    + split.
      × intros a b H. unfold comp. apply H1 in H. rewrite H. apply refl.
```
The proof that $(f_2(f_1\ a)) \twoheadrightarrow_R (f_2(f_1\ b))$ is straightforward because $(f_2(f_1\ a)) = (f_2(f_1\ b))$ by hypothesis $H1$.

```
      × split; assumption.
```
Finally, the last two conditions of *Z_comp_eq* and *Z_comp* coincide, and we are done.

`Qed`.

4 Conclusion, Related Work and Future Work

In this work we presented a constructive proof that the Z property implies confluence, and this proof was formalized in the Coq proof assistant[2]. Moreover, our formalization includes an extension of the Z property, known as compositional Z property, as presented in [NF16].

The Z property was presented by V. van Oostrom as a sufficient condition for an ARS to be confluent [FNvOS16], 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. In [FNvOS16], B. Felgenhauer et.al. formalized the Z property in Isabelle/HOL via semiconfluence. Recently, [van21] shows the flexibility and applicability of the Z property to prove confluence and normalisation based on a syntax-free version of the classical notion of development.

As future work, this formalization will be used to prove the confluence property of calculi with explicit substitutions using different approaches such as nominal and locally nameless representations.

References

[ACCL91] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. *JFP*, 1(4):375–416, 1991.

[AK12] B. Accattoli and D. Kesner. The permutative lambda calculus. In Nikolaj Bjørner and Andrei Voronkov, editors, *LPAR*, volume 7180 of *Lecture Notes in Computer Science*, pages 23–36. Springer, 2012.

[Bar84] H. P. Barendregt. *The Lambda Calculus : Its Syntax and Semantics (revised edition)*. North Holland, 1984.

[DvO08] 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.

[FNvOS16] B. Felgenhauer *et. al.* The Z property. *Archive of Formal Proofs*, 2016.

[Kes09] D. Kesner. A Theory of Explicit Substitutions with Safe and Full Composition. *Logical Methods in Computer Science*, 5(3:1):1–29, 2009.

[NF16] K. Nakazawa and K. Fujita. Compositional Z: confluence proofs for permutative conversion. *Studia Logica*, 104(6):1205–1224, 2016.

[Par92] M. Parigot. Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. In Andrei Voronkov, editor, *Logic Programming and Automated Reasoning,International Conference LPAR'92, St. Petersburg, Russia, July 15-20, 1992, Proceedings*, volume 624 of *LNCS*, pages 190–201. Springer, 1992.

[Ter03] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[van21] V. van Oostrom. Z; syntax-free developments. In Naoki Kobayashi, editor, *6th International Conference on Formal Structures for Computation and Deduction (FSCD 2021)*, volume 195 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:22, Dagstuhl, Germany, 2021.

[Win89] F. Winkler. *Equational Theorem Proving and Rewrite Rule Systems*, pages 26–39. Informatik-Fachberichte. Springer Berlin Heidelberg, 1989.

[2]The corresponding files are available at https://github.com/flaviodemoura/Zproperty