

Confluence via the Z Property in Coq

Flávio L. C. de Moura and Leandro O. Rezende

Departamento de Ciência da Computação, Universidade de Brasília, Brazil
flaviomoura@unb.br, l-ordo.ab.chao@hotmail.com

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 constructive proof that the Z property implies confluence. The known proofs of this fact usually rely on the law of the excluded middle, but its use is not necessary, as shown in our proof, which circumvents this necessity by using nested induction on the inductive step of the original proof. 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. 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 constructive and based on nested induction.
- The proof that the Z property implies confluence is formalized in the Coq proof assistant, and the presentation is made interleaving English followed by the corresponding Coq 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 [7].

An ARS, say (A, R) , is defined as a pair composed of a set A and binary relation over this set $R : A \times A$. Let $a, b \in A$. We write $a R b$ or $a \rightarrow_R b$ to denote that $(a, b) \in R$, and we say that a R -reduces to b in one step. The arrow notation will be preferred because it is more convenient for expressing reductions, so the reflexive transitive closure of a relation R , written as \rightarrow_R , is defined by the following inference rules:

$$\frac{}{a \rightarrow_R a} \text{ (refl)} \qquad \frac{a \rightarrow_R b \quad b \rightarrow_R c}{a \rightarrow_R c} \text{ (rtrans)}$$

where a, b and c are universally quantified variables as one makes explicit 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 rules named (*refl*) and (*rtrans*) are called *constructors* in the Coq definition. The first constructor states the reflexivity axiom for \rightarrow_R , while *rtrans* extends the reflexive transitive closure of *R*, if one has at least a one-step reduction. As a first example, let's have a look at the proof of transitivity of \rightarrow_R :

Lemma 1. *Let \rightarrow_R be a binary relation over a set A . If $t \rightarrow_R u$ and $u \rightarrow_R v$ then $t \rightarrow_R v$, for all $t, u, v \in A$.*

Although its proof is simple, it will help us explain the way in which we will relate English annotations with the proof steps. The corresponding lemma in Coq, named *refltrans_composition*, is stated as follows:

Lemma *refltrans_composition* {*A*} (*R*: Rel *A*): $\forall t\ u\ v, \text{refltrans } R\ t\ u \rightarrow \text{refltrans } R\ u\ v \rightarrow \text{refltrans } R\ t\ v$.

```

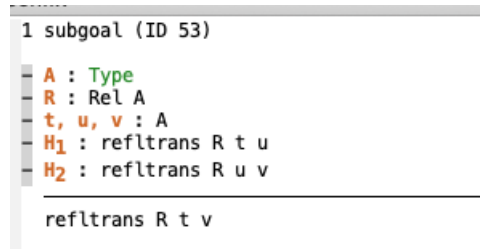
1. Proof.
2.  intros t u v.
3.  intros H1 H2.
4.  induction H1.
5.  - assumption.
6.  - apply rtrans with b.
7.  + assumption.
8.  + apply IHrefltrans; assumption.
9. Qed.

```

This work is not a Coq tutorial, but our idea is that it should also be readable for those unfamiliar with the Coq proof Assistant. In addition, this paper is built directly from a Coq proof script, which means that we are forced to present the ideas and the results in a more organized and systematic way that is not necessarily the more pedagogical one. Coq proofs are written between the reserved words **Proof** and **Qed** (lines 1 and 9), and each proof command finishes with a dot. Proofs can be structured with bullets (- in the first level, + in the second level, * in the third level, ** in the fourth level, and so on). The corresponding informal proof proceed as follows:

Proof.

Let $t, u, v \in A$, i.e. they are elements of type *A*, or elements of the set *A* (line 2). Call *H1* (resp. *H2*) the hypothesis that $t \rightarrow_R u$ (resp. $u \rightarrow_R v$) (line 3). The proof proceeds by induction on the hypothesis *H1* (line 4), i.e. by induction on $t \rightarrow_R u$. The structure of the proof context determines the shape of the induction hypothesis, and this fact will be essential to understand the inductive proof of the next theorem. As shown in Figure 1, *H1* and *H2* are the only hypothesis (the other lines are just declaration of variables), therefore the induction hypothesis subsumes *H2*.



```

1 subgoal (ID 53)
- A : Type
- R : Rel A
- t, u, v : A
- H1 : refltrans R t u
- H2 : refltrans R u v
-----
refltrans R t v

```

Fig. 1. Transitivity of \rightarrow_R

The first case is when $t \rightarrow_R u$ is generated by the constructor *refl*, which is an axiom and hence we are done (line 5). The second case, i.e. the recursive case is more interesting because $t \rightarrow_R u$ is now generated by *rtrans* (line 6). This means that there exists an element, say *b*, such that $t \rightarrow_R b$ and $b \rightarrow_R u$. Therefore,

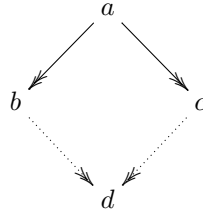
in order to prove that $t \rightarrow_R u$, we can apply the rule *rtrans* taking b as the intermediary term. The proof of the recursive case can be better visualized by the corresponding deduction tree:

$$\frac{\frac{\frac{\frac{\forall x y z, x \rightarrow_R y \rightarrow y \rightarrow_R z \rightarrow x \rightarrow_R z}{t \rightarrow_R b \rightarrow b \rightarrow_R u \rightarrow t \rightarrow_R u} rtrans}{b \rightarrow_R u \rightarrow t \rightarrow_R u} (\forall_e) \quad \frac{t \rightarrow_R b}{t \rightarrow_R b} H}{b \rightarrow_R u \rightarrow t \rightarrow_R u} MP \quad \frac{\frac{u \rightarrow_R v \rightarrow b \rightarrow_R u}{u \rightarrow_R v} IH \quad \frac{u \rightarrow_R v}{u \rightarrow_R v} H2}{b \rightarrow_R u} MP}{t \rightarrow_R u} MP$$

Each branch of the above tree corresponds to a new goal in the Coq proof. Therefore, we have two subcases (or subgoals) to prove: In this subgoal we need to prove that $t \rightarrow_R b$, which we have as hypothesis (line 7). In the second subgoal (line 8), we need to prove that $b \rightarrow_R u$. To do so, we apply the induction hypothesis *IHrefltrans*: $u \rightarrow_R v \rightarrow b \rightarrow_R u$, where $u \rightarrow_R v$ is the hypothesis *H2*. \square

This example is interesting because it shows how Coq works, how each command line (also known as tactics or tacticals depending on its structure) corresponds, in general, to several steps of natural deduction rules.

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:

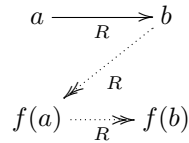


Formally, this means that if an expression a can be reduced in two different ways to the expressions b and c , then there exists an expression d such that both b and c reduce to d . The existential quantification is expressed by the dotted lines in the diagram. This notion is defined in the Coq system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

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

In [4], V. van Oostrom gives a sufficient condition for an ARS to be confluent, known as the *Z Property*:

Definition 1. Let (A, \rightarrow_R) be an ARS. Then (A, \rightarrow_R) has the *Z property*, if there exists a map $f : A \rightarrow A$ such that the following diagram holds:



The corresponding Coq definition is given as:

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

Alternatively, for a given function f , one can say that f satisfies the Z property, or that f is Z, if the above conditions hold for f :

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

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 [6] (that follows [4]) and the one in [5] in

the sense that it does not rely on analyzing whether a term is in normal form or not, avoiding necessity of the law of the excluded middle. As a result, we have an elegant inductive proof of the fact that if a binary relation has the Z property then it is confluent. In addition, we formalized this proof in the Coq proof assistant. In [5], B. Felgenhauer et.al. formalized in Isabelle/HOL the Z property and its relation to confluence. In what follows, we present the theorem and its proof interleaving Coq code and the corresponding comments.

Theorem *Z_prop_implies_Confl* {A:Type}: $\forall R: \text{Rel } A, \text{Z_prop } R \rightarrow \text{Confl } R$.

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 :  $\exists f : A \rightarrow A,$ 
               $\forall a b : A, R a b \rightarrow \text{refltrans } R b (f a) \wedge \text{refltrans } R (f a) (f b)$ 
-  $\forall a b c : A,$ 
  refltrans R a b  $\rightarrow$ 
  refltrans R a c  $\rightarrow \exists d : A, \text{refltrans } R b d \wedge \text{refltrans } R c d$ 
```

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

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:

```
1 subgoal (ID 103)
- A : Type
- R : Rel A
- g : A  $\rightarrow$  A
- HZ_prop :  $\forall a b : A, R a b \rightarrow \text{refltrans } R b (g a) \wedge \text{refltrans } R (g a) (g b)$ 
- a, b, c : A
- Hrefl1 : refltrans R a b
- Hrefl2 : refltrans R a c
-  $\exists d : A, \text{refltrans } R b d \wedge \text{refltrans } R c d$ 
```

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 $a \rightarrow_R b$.

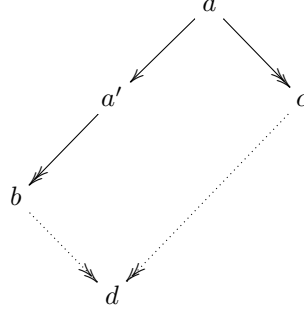
- **intros** *c Hrefl2*. In the first case, we have that $b = a$ since we are in the reflexive case. This means that we have to prove that there exists *d*, such that $a \rightarrow_R d$ and $c \rightarrow_R d$.

$\exists c$; **split**. Taking *d* as *c*, the proof is simplified to $a \rightarrow_R c$ and $c \rightarrow_R c$.

 + **assumption**. The first component is exactly the hypothesis *Hrefl2* and,

 + **apply** *refl*. $c \rightarrow_R c$ corresponds to an application of the *refl* axiom.

The interesting part of the proof is then given by the inductive case, i.e. when $a \rightarrow_R b$ 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.



The induction hypothesis states that every divergence from a' that reduces to b from one side converges: $IHHrefl1 : \forall c_0 : A, a' \rightarrow_R c_0 \rightarrow (\exists d : A, b \rightarrow_R d \wedge c_0 \rightarrow_R d)$. Now, we'd like apply induction on the hypothesis $Hrefl2$ (a “tto_R c”), but the current proof context has the hypothesis $H : a \rightarrow_R a'$ (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 $IHHrefl2$ would assume that there is a term a'' such that $a \rightarrow_R a'' \rightarrow_R c$ and would require that $a'' \rightarrow_R a'$, 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.

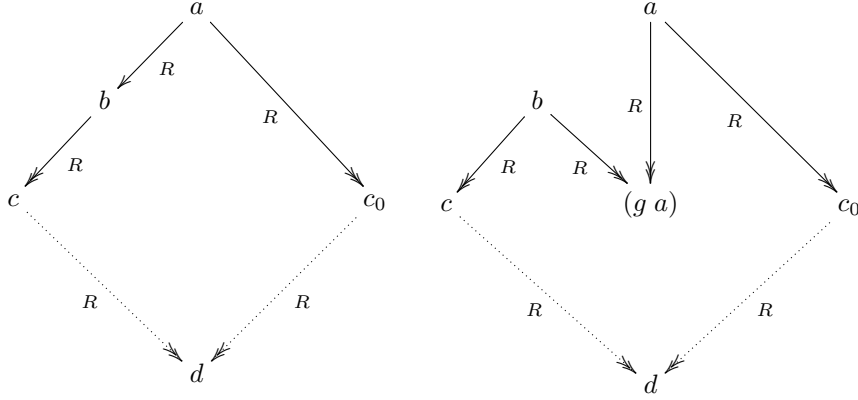
- `intros c0 Hrefl2`. Let c_0 be a reduct of a , and $Hrefl2$ be the hypothesis $a \rightarrow_R c_0$. So the reduction $a \rightarrow_R c$ in the above diagram is now $a \rightarrow_R c_0$ due to a renaming of variables automatically done by the Coq system. In addition, the reduction $a \rightarrow_R a' \rightarrow_R b$ is now $a \rightarrow_R b \rightarrow_R c$, 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 $Hrefl2 : a \rightarrow_R c_0$, we will derive $b \rightarrow_R (g a)$ and $a \rightarrow_R (g a)$ from the proof context so we can discard the hypothesis $H : a \rightarrow_R b$.

```
assert (Hbga: refltrans R b (g a)).
{ apply HZ_prop; assumption. } We call Hbga the reduction  $b \rightarrow_R (g a)$  that is directly obtained from
the Z property.
assert (Haga: refltrans R a (g a)).
{ apply rtrans with b; assumption. } Call Haga the reduction  $a \rightarrow_R (g a)$ , and prove it using the
transitivity of  $\rightarrow_R$ , since  $a \rightarrow_R b$  and  $b \rightarrow_R (g a)$ .
```

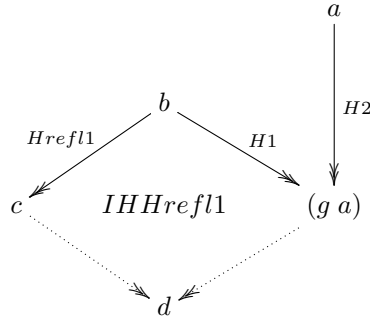
Diagrammatically, we change from the situation on the left to the one on the right:



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 $a \rightarrow_R c_0$, 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 $IHHrefl1$:



assert ($IHHrefl1_ga := IHHrefl1 (g a)$); apply $IHHrefl1_ga$ 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 $c \rightarrow_R x$ and $(g a) \rightarrow_R x$.

$\exists x$; **split.** We then take x to show that $c \rightarrow_R x$ and $a \rightarrow_R x$.

\times **apply H .** Note that $c \rightarrow_R x$ is already an hypothesis, and we are done.

\times **apply $refltrans_composition$ with $(g a)$; [assumption | apply H].** The proof of $a \rightarrow_R x$ is done by the transitivity of \rightarrow_R taking $(g a)$ as the intermediate step.

+ **intros $b0$ $Hrefl1$ $IHHrefl1$ $Hb0ga$.** The second subgoal corresponds to the case in which $a \rightarrow_R c_0$ is generated by the rule ($rtrans$). Therefore, there exists a term b such that $a \rightarrow_R b$ and $b \rightarrow_R c_0$. 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 $b0 \rightarrow_R (g a)$ 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 *IHHrefl2* with *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 *refltrans_composition* with $(g\ a)$; apply *HZ_prop*; assumption. 1. $b \rightarrow_R (g\ b)$: This is proved by the transitivity of the reflexive transitive closure of *R* using the hypothesis (H: $a \rightarrow_R b$) and *HZ_prop*: $\forall a\ b : a \rightarrow_R b \rightarrow (b \rightarrow_R (g\ a) \wedge (g\ a) \rightarrow_R (g\ b))$.

× assumption. 2. $b0 \rightarrow_R c$: This is exactly the hypothesis *Hrefl1*.

× assumption. 3. $\forall c0 : b0 \rightarrow_R c0 \rightarrow (\exists d : c \rightarrow_R d \wedge c0 \rightarrow_R d)$: This is exactly the induction hypothesis *IHHrefl1*.

× apply *refltrans_composition* with $(g\ a)$; [assumption | apply *HZ_prop*; assumption]. 4. $b0 \rightarrow_R (g\ b)$: This is proved by the transitivity of the reflexive transitive closure of *R* using the hypothesis (*H'*: $b0 \rightarrow_R (g\ a)$) and the fact that $(g\ a) \rightarrow_R (g\ b)$ that is obtained from the fact that *R* satisfies the Z property (hypothesis *HZ_prop*).

Qed.

Another proof

Lemma *refltrans_f_is_Z_refltrans* {A:Type}: $\forall (R: Rel\ A)\ a\ b\ f, f_is_Z\ R\ f \rightarrow (refltrans\ R)\ a\ b \rightarrow (refltrans\ R)\ (f\ a)\ (f\ b)$.

Proof.

```

intros R a b f H Hab.
unfold f_is_Z in H.
induction Hab.
- apply refl.
- apply H in H0.
  destruct H0 as [H1 H2].
  apply refltrans_composition with (f b); assumption.

```

Qed.

Lemma *refltrans_f_is_Z* {A:Type}: $\forall (R: Rel\ A)\ a\ f, f_is_Z\ R\ f \rightarrow (refltrans\ R)\ a\ (f\ a)$.

Proof.

```

intros R a f H.
unfold f_is_Z in H.

```

Admitted.

Theorem *Z_prop_implies_Confl2* {A:Type}: $\forall R: Rel\ A, Z_prop\ R \rightarrow Confl\ R$.

Proof.

```

intros R H.
unfold Z_prop in H.
destruct H as [g H].
unfold Confl.
intros a b c H1 H2.
generalize dependent c.
induction H1.
- intros c H1.
  ∃ c; split.
  + assumption.
  + apply refl.
- intros c0 H2.
  apply H in H0.
  destruct H0 as [Hga Hgb].
  assert (refltrans R (g a) (g c0)).
  {
    apply refltrans_f_is_Z_refltrans.
    - assumption.
    - assumption.
  }
  assert (refltrans R c0 (g c0)).
  {
    apply refltrans_f_is_Z; assumption.
  }
  assert (refltrans R b (g c0)).
  {
    apply refltrans_composition with (g a); assumption.
  }
  apply IHrefltrans in H4.
  destruct H4 as [d [H4 H5]].
  ∃ d; split.
  + assumption.
  + apply refltrans_composition with (g c0); assumption.

```

Qed.

An alternative proof that Z implies confluence is possible via the notion of semiconfluence, which is equivalent to confluence, as done in [5]. Unlike the proof in [5] and similarly to our previous proof, our proof of the Theorem that Z implies semiconfluence is constructive, but we will not explain it here due to lack of space; any interested reader can find it in the Coq file in our GitHub repository.

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

Theorem *Z_prop_implies_SemiConfl* {A:Type}: $\forall R: Rel A, Z_prop R \rightarrow SemiConfl R$.

Theorem *Semi-equiv_Confl* {A: Type}: $\forall R: Rel A, Confl R \leftrightarrow SemiConfl R$.

Corollary *Zprop_implies_Confl_via_SemiConfl* {A:Type}: $\forall R: Rel A, Z_prop R \rightarrow Confl R$.

Proof.

```

intros R HZ_prop.
apply Semi-equiv_Confl.
generalize dependent HZ_prop.
apply Z_prop_implies_SemiConfl.

```

Qed.

2 An extension of the Z property: Compositional Z

In this section we present a formalization of an extension of the Z property with compositional functions, known as *Compositional Z*, as presented in [7]. The compositional Z is an interesting property because it allows a kind of modular approach to the Z property in such a way that the reduction relation can be split into two parts. More precisely, given an ARS (A, \rightarrow_R) , one must be able to decompose the relation \rightarrow_R into two parts, say \rightarrow_1 and \rightarrow_2 such that $\rightarrow_R = \rightarrow_1 \cup \rightarrow_2$. This kind of decomposition can be done in several interesting situations such as the λ -calculus with $\beta\eta$ -reduction[3], extensions of the λ -calculus with explicit substitutions[1], the $\lambda\mu$ -calculus[8], etc. But before presenting the full definition of the Compositional Z, we need to define the *weak Z property*:

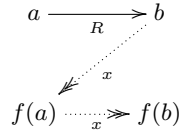


Fig. 2. The weak Z property

Definition 2. Let (A, \rightarrow_R) be an ARS and \rightarrow'_R a relation on A . A mapping f satisfies the weak Z property for \rightarrow_R by \rightarrow'_R if $a \rightarrow_R b$ implies $b \rightarrow'_R f(a)$ and $f(a) \rightarrow'_R f(b)$ (cf. Figure 2). Therefore, a mapping f satisfies the Z property for \rightarrow_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` $\{A\}$ $(R \ R': \text{Rel } A) (f: A \rightarrow A) := \forall a \ b, R \ a \ b \rightarrow ((\text{refltrans } R') \ b \ (f \ a) \wedge (\text{refltrans } R') \ (f \ a) \ (f \ b))$.

The compositional Z is an extension of the Z property for compositional functions, where composition is defined as usual:

Definition `comp` $\{A\}$ $(f1 \ f2: A \rightarrow A) := \text{fun } x:A \Rightarrow f1 \ (f2 \ x)$.

Notation `"f1 # f2"` $:= (\text{comp } f1 \ f2) \text{ (at level 40)}$.

and the disjoint union is inductively defined as:

Inductive `union` $\{A\}$ $(red1 \ red2: \text{Rel } A) : \text{Rel } A :=$
 $| \text{union_left}: \forall a \ b, red1 \ a \ b \rightarrow \text{union } red1 \ red2 \ a \ b$
 $| \text{union_right}: \forall a \ b, red2 \ a \ b \rightarrow \text{union } red1 \ red2 \ a \ b$.

Notation `"R1 !| R2"` $:= (\text{union } R1 \ R2) \text{ (at level 40)}$.

We are now ready to present the definition of the compositional Z:

Theorem 1. [7] Let (A, \rightarrow_R) be an ARS such that $\rightarrow_R = \rightarrow_1 \cup \rightarrow_2$. If there exists mappings $f_1, f_2 : A \rightarrow A$ such that

1. f_1 is Z for \rightarrow_1
2. $a \rightarrow_1 b$ implies $f_2(a) \rightarrow f_2(b)$
3. $a \rightarrow f_2(a)$ holds for any $a \in \text{Im}(f_1)$
4. $f_2 \circ f_1$ is weakly Z for \rightarrow_2 by \rightarrow_R

then $f_2 \circ f_1$ is Z for (A, \rightarrow_R) , and hence (A, \rightarrow_R) is confluent.

We define the predicate Z_comp that corresponds to the premises of Theorem 1, i.e. to the conjunction of items (i), (ii), (iii) and (iv) in addition to the fact that $\rightarrow_R = \rightarrow_1 \cup \rightarrow_2$, where \rightarrow_1 (resp. \rightarrow_2) is written as $R1$ (resp. $R2$):

Definition $Z_comp \{A:Type\} (R : Rel A) := \exists (R1 R2 : Rel A) (f1 f2 : A \rightarrow A), R = (R1 \text{ !_! } R2) \wedge f_is_Z R1 f1 \wedge (\forall a b, R1 a b \rightarrow (refltrans R) (f2 a) (f2 b)) \wedge (\forall a b, b = f1 a \rightarrow (refltrans R) b (f2 b)) \wedge (f_is_weak_Z R2 R (f2 \# f1))$.

As stated by Theorem 1, the compositional Z gives a sufficient condition for compositional functions to be Z. In other words, compositional Z implies Z, which is justified by the diagrams of Figure 3.



Fig. 3. Compositional Z implies Z

In what follows, we present our commented Coq proof of this fact:

Theorem $Z_comp_implies_Z_prop \{A:Type\} : \forall (R : Rel A), Z_comp R \rightarrow Z_prop R$.

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** $[R1 [R2 [f1 [f2 [Hunion [H1 [H2 [H3 [H4]]]]]]]]]$. Now unfold the definitions of Z_prop and Z_comp as presented before, and name the hypothesis of the compositional Z as in Theorem 1. We need to prove that there exists a map, say f , that is Z as shown by the current proof context:

```
1 subgoal (ID 167)
- A : Type
- R, R1, R2 : Rel A
- f1, f2 : A → A
- Hunion : R = R1 !\_! R2
- H1 : f_is_Z R1 f1
- H2 : ∀ a b : A, R1 a b → refltrans R (f2 a) (f2 b)
- H3 : ∀ a b : A, b = f1 a → refltrans R b (f2 b)
- H4 : f_is_weak_Z R2 R (f2 # f1)
-
  ∃ f : A → A,
    ∀ a b : A, R a b → refltrans R b (f a) ∧ refltrans R (f a) (f b)
```

$\exists (f2 \# f1)$. We will prove that the composition $f2 \circ f1$ 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.

inversion $Hunion$; **subst**. **clear** H . **inversion** HR ; **subst**. 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:

Rewriting Systems with equations is another interesting and non-trivial topic [10,9]. 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 [2].

Theorem 2. *Let (A, \rightarrow_R) be an ARS such that $\rightarrow_R = \rightarrow_1 \cup \rightarrow_2$. If there exist mappings $f_1, f_2 : A \rightarrow A$ such that*

1. $a \rightarrow_1 b$ implies $f_1(a) = f_1(b)$
2. $a \rightarrow_1 f_1(a)$, for all a
3. $a \rightarrow_R f_2(a)$ holds for any $a \in \text{Im}(f_1)$
4. $f_2 \circ f_1$ is weakly Z for \rightarrow_2 by \rightarrow_R

then $f_2 \circ f_1$ is Z for (A, \rightarrow_R) , and hence (A, \rightarrow_R) is confluent.

We define the predicate Z_comp_eq corresponding to the hypothesis of Theorem 2, and then we prove directly that if Z_comp_eq holds for a relation R then $Zprop\ R$ also holds. This approach differs from [7] that proves Theorem 2, which is a Corollary in [7], directly from Theorem 1

Definition $Z_comp_eq\ \{A:\text{Type}\}\ (R : \text{Rel}\ A) := \exists\ (R1\ R2 : \text{Rel}\ A)\ (f1\ f2 : A \rightarrow A),\ R = (R1\ !_!\ R2) \wedge (\forall\ a\ b,\ R1\ a\ b \rightarrow (f1\ a) = (f1\ b)) \wedge (\forall\ a,\ (\text{refltrans}\ R1)\ a\ (f1\ a)) \wedge (\forall\ b\ a,\ a = f1\ b \rightarrow (\text{refltrans}\ R)\ a\ (f2\ a)) \wedge (f_is_weak_Z\ R2\ R\ (f2\ \# f1)).$

Lemma $Z_comp_eq_implies_Z_prop\ \{A:\text{Type}\} : \forall\ (R : \text{Rel}\ A),\ Z_comp_eq\ R \rightarrow Z_prop\ R.$

Proof.

`intros R Heq. unfold Z_comp_eq in Heq.` Let R be a relation and suppose that R satisfies the predicate Z_comp_eq .

`destruct Heq as [R1 [R2 [f1 [f2 [Hunion [H1 [H2 [H3 H4]]]]]]].` Call H_i the i th hypothesis as in 2.

`unfold Z_prop. $\exists\ (f2\ \# f1)$.` From the definition of the predicate Z_prop , we need to find a map, say f that is Z. Let $(f_2 \circ f_1)$ be such map.

`intros a b Hab.` In order to prove that $(f_2 \circ f_1)$ is Z, let a and b be arbitrary elements of type A , and Hab be the hypothesis that $a \rightarrow_R b$.

`inversion Hunion; subst; clear H. inversion Hab; subst; clear Hab.` Since a R -reduces in one step to b and R is the union of the relations $R1$ and $R2$ then we consider two cases:

- `unfold comp; split.` The first case is when $a \rightarrow_{R1} b$. This is equivalent to say that $f_2 \circ f_1$ is weak Z for $R1$ by $R1 \cup R2$.

+ `apply refltrans_composition with (f1 b).` Therefore, we first prove that $b \rightarrow_{(R1 \cup R2)} (f_2(f_1 a))$, which can be reduced to $b \rightarrow_{(R1 \cup R2)} (f_1 b)$ and $(f_1 b) \rightarrow_{(R1 \cup R2)} (f_2(f_1 a))$ by the transitivity of refltrans .

`\times apply refltrans_union. apply H2.` From hypothesis $H2$, we know that $a \rightarrow_{R1} (f_1 a)$ for all a , and hence $a \rightarrow_{(R1 \cup R2)} (f_1 a)$ and we conclude.

`\times apply H1 in H. rewrite H. apply H3 with b; reflexivity.` The proof that $(f_1 b) \rightarrow_{(R1 \cup R2)} (f_2(f_1 a))$ is exactly the hypothesis $H3$.

+ `apply H1 in H. rewrite H. apply refl.` The proof that $(f_2(f_1 a)) \rightarrow_{(R1 \cup R2)} (f_2(f_1 b))$ is done using the reflexivity of refltrans because $(f_2(f_1 a)) = (f_2(f_1 b))$ by hypothesis $H1$.

- `apply H4; assumption.` When $a \rightarrow_{R2} b$ then we are done by hypothesis $H4$.

Qed.

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 [5], 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 [7]

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. [6]). In addition, we hope that our formalization can be used as a framework for proving confluence of others rewriting systems.

References

1. M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.
2. 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.
3. H. P. Barendregt. *The Lambda Calculus : Its Syntax and Semantics (revised edition)*. North Holland, 1984.
4. 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.
5. B. Felgenhauer, J. Nagele, V. van Oostrom, and C. Sternagel. The Z property. *Archive of Formal Proofs*, 2016.
6. D. Kesner. A Theory of Explicit Substitutions with Safe and Full Composition. *Logical Methods in Computer Science*, 5(3:1):1–29, 2009.
7. Koji Nakazawa and Ken-etsu Fujita. Compositional Z: confluence proofs for permutative conversion. *Studia Logica*, 104(6):1205–1224, 2016.
8. Michel 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 *Lecture Notes in Computer Science*, pages 190–201. Springer, 1992.
9. Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
10. Franz Winkler. *Equational Theorem Proving and Rewrite Rule Systems*, pages 26–39. Informatik-Fachberichte. Springer Berlin Heidelberg, 1989.