Confluence via the Z Property in Coq

Flávio L. C. de Moura[1] Leandro O. Rezende[2]

Departamento de Ciência da Computação, Universidade de Brasília, Brasília, Brazil

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. In addition, we formalized our proof and an extension of the Z property, known as the Compositional Z, in the Coq proof assistant.

Keywords: Rewriting systems, Confluence, Interactive theorem proving, Coq

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 English explanation followed by the corresponding Coq code 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 [6].

2 The Z property implies Confluence

An ARS is defined as a pair composed of a set and binary operation over this set. Given an ARS (A, R), where A is a set, $R : A \times A$ and $a, b : A$, we write $a\ R\ b$ or $a \to_R b$ to denote that $(a, b) \in R$, and we say that a R-reduces to b in one step . The arrow notation will be prefered because it is more convenient for expressing

[1] Email:flaviomoura@unb.br

[2] Email:l-ordo.ab.chao@hotmail.com

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

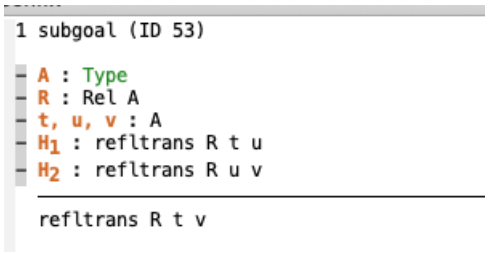


Figure 1. Transitivity of refltrans

reductions, so the reflexive transitive closure of a relation R, written as $\twoheadrightarrow_R$, is defined by the following inference rules:

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

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 rule names (*refl*) and (*rtrans*) are called *constructors* in the Coq definition. The first constructor states the reflexivity of $\twoheadrightarrow_R$, 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 $\twoheadrightarrow_R$:

Lemma 2.1 *Let $\rightarrow_R$ be a binary relation over a set A. For all $t, u, v \in A$, if $t \twoheadrightarrow_R u$ and $u \twoheadrightarrow_R v$ then $t \twoheadrightarrow_R v$.*

Although its simplicity, it will help us to explain the way 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):
  ∀ t u v, refltrans R t u → refltrans R u v → refltrans R t v.
```

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. In this way, we decided to comment the proof steps giving the general idea of what they do. It is done in such a way that both parts (informal and formal) are mostly independent, and those not interested in the Coq part can focus just on the informal part. Firstly, notice that proofs are written between the reserved words `Proof` and `Qed`. Each proof command finishes with a dot, and proofs can be structured with bullets.

`Proof`.
`intros` *t u v H1 H2*. Let *t*,*u* and *v* be elements of type A (or be elements of the set A), and assume that $t \twoheadrightarrow_R u$ (name this assumption *H1*) and $u \twoheadrightarrow_R v$ (assumption *H2*). The corresponding proof context shows the hypothesis and the goal separated by a horizontal line:

`induction` *H1*. The proof proceeds by induction on the hypothesis *H1*, i.e. induction on $t \twoheadrightarrow_R u$. The structure of the proof context models the induction hypothesis, and this fact will be essential to build the inductive proof of the next theorem. As shown in Figure 1, *H2* is the sole hypothesis different from *H1*, and hence the induction hypothesis will subsume *H2*.

- `assumption`. The first case is when $t \twoheadrightarrow_R u$ is generated by the constructor $refl$, which is an axiom and hence we are done.

- `apply rtrans with` *b*. The second case, i.e. the recursive case is more interesting because $t \twoheadrightarrow_R u$ is now generated by (*rtrans*). This means that there exists an element, say b, such that $t \rightarrow_R b$ and $b \twoheadrightarrow_R u$. Therefore, in order to prove that $t \twoheadrightarrow_R v$, we can apply the rule *rtrans* taking b as the intermediary term, and we have two subcases to prove:

  + `assumption`. In the first case we need to prove that $t \rightarrow_R b$, which we have as hypothesis.

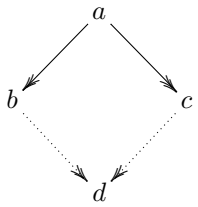
+ `apply` *IHrefltrans*; `assumption`. In the second case, we prove that $b \twoheadrightarrow_R v$ by the induction hypothesis $u \twoheadrightarrow_R v \to b \twoheadrightarrow_R v$, where $u \twoheadrightarrow_R v$ is the hypothesis *H2*. The proof of the recursive case may be better visualized by the corresponding deduction tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\forall x\ y\ z, x \to_R y \to y \twoheadrightarrow_R z \to x \twoheadrightarrow_R z}\ rtrans}{a \to_R b \to b \twoheadrightarrow_R v \to a \twoheadrightarrow_R v}\ (\forall_e) \quad \dfrac{}{a \to_R b}\ \text{H}}{b \twoheadrightarrow_R v \to a \twoheadrightarrow_R v}\ \text{MP} \quad \dfrac{\dfrac{}{c \twoheadrightarrow_R v \to b \twoheadrightarrow_R v}\ IH \quad \dfrac{}{c \twoheadrightarrow_R v}\ \text{H2}}{b \twoheadrightarrow_R v}\ \text{MP}}{a \twoheadrightarrow_R v}\ \text{MP}$$

`Qed`.

This example is interesting because it shows how Coq works, how tactics correspond in general to several steps of natural deduction rules, and how proofs can be structured with bullets.

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:

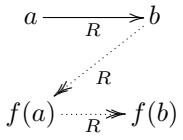


Therefore, if an expression a can be reduced in two different ways to the expressions b and c, then there exists an expression d such that both b and c reduces to d. The existential quantification is expressed by the dotted lines in the diagram. This notion is defined in the Coq system as follows:

```
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 [9], V. van Oostrom gives a suficient condition for an ARS to be confluent, known as the *Z Property*:

Definition 2.2 Let $(A, \to_R)$ be an abstract rewriting system (ARS). The system $(A, \to_R)$ has the Z property, if there exists a map $f : A \to A$ such that the following diagram holds:



The corresponding Coq definition is given as:

```
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)).
```

Alternatively, when f satisfies the Z property one says that f is Z:

```
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)).
```

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 [5] (that follows [9]) in the sense that it does not rely on 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 [4], B. Felgenhauer et.al. formalized the Z property in Isabelle/HOL. In what follows we present the theorem and its proof interleaving Coq code and the corresponding comments.

```
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, we get the following proof context:

```
1 subgoal (ID 95)

- A : Type
- R : Rel A
- HZ_prop : ∃ wb : A → A,
              ∀ a b : A,
              R a b → refltrans R b (wb a) ∧ refltrans R (wb a) (wb 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 $a \twoheadrightarrow_R c$.

`destruct` *HZ_prop* `as` [*g HZ_prop*]. In addition, by the hypothesis *HZ_prop*, we know 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 → A
- HZ_prop : ∀ a b : A, R a b → refltrans R b (g a) ∧ refltrans R (g a) (g b)
- a, b, c : A
- Hrefl1 : refltrans R a b
- Hrefl2 : refltrans R a c
------------------------------------------------------------------------
∃ d : A, refltrans R b d ∧ refltrans R c d
```

Now we need to show that there exists an element *d* such that both *b* and *c* *R*-reduces to *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 split in two subgoals, one for each possible way of constructing $a \twoheadrightarrow_R b$.

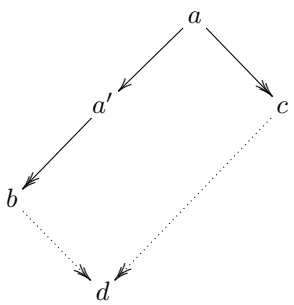
- `intros` *c Hrefl2*. In the first case, $a = b$ since we are in the reflexive case.

  ∃ *c*; `split`. Therefore, there is no divergence and this case is proved by taking *c* as *d*.

  + `assumption`. The goal is then the conjunction $a \twoheadrightarrow_R c \wedge c \twoheadrightarrow_R c$ whose first component is exactly the hypothesis *Hrefl2* and,

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

The interesting case is given by the inductive case, i.e. by the rule (*rtrans*), where the reduction from *a* to *b* is done in at least one step. Therefore, there exists an element a' such that the following diagram holds.

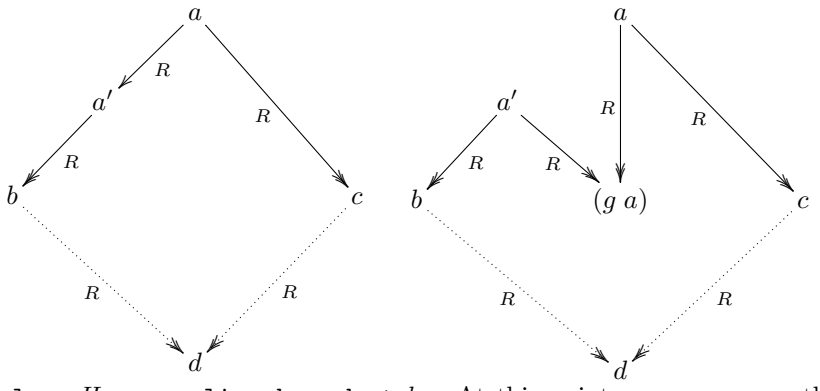


The induction hypothesis states that every divergence from a', that reduces to *b* from one side, converges: *IHHrefl1* : $\forall c_0 : A, a' \twoheadrightarrow_R c_0 \rightarrow (\exists d : A, b \twoheadrightarrow_R d \wedge c_0 \twoheadrightarrow_R d)$. The idea is to apply induction on the hypothesis *Hrefl2*, but the current proof context has the hypothesis H: $a \rightarrow_R a'$, which will generate, in the induction hypothesis, the condition that $a' \rightarrow_R c$, and this is not true in general. In order to circumvent this problem, we

need to remove this hypothesis, but the information that $a \rightarrow_R b$ is essential and cannot be simply removed. At this point, we use the Z property as shown below.

- `intros` *c0 Hrefl2*. Let c_0 be a reduct of b, and *Hrefl2* the fact that $a \twoheadrightarrow_R c_0$. So the reduction $a \twoheadrightarrow_R c$ in the above diagrams is now $a \twoheadrightarrow_R c_0$ due to a renaming of variables automatically done by the Coq system. Before applying induction to *Hrefl2*: $a \twoheadrightarrow_R c_0$, we will be replace the hypothesis H by two other properties that are proved from the Z property: $b \twoheadrightarrow_R (g\ a)$ and $a \twoheadrightarrow_R (g\ a)$.

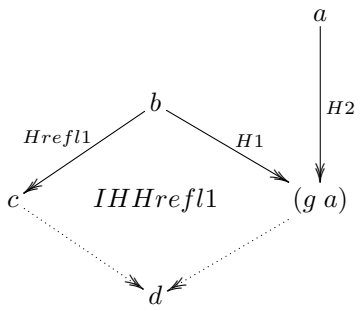
`assert` (*H1*: **refltrans** $R\ b\ (g\ a)$).
{ `apply` *HZ_prop*; `assumption`. } We call *H1* the reduction $b \twoheadrightarrow_R (g\ a)$ that is directly obtained from the Z property.
`assert` (*H2*: **refltrans** $R\ a\ (g\ a)$).
{ `apply` rtrans `with` b; `assumption`. } Call *H2* the reduction $a \twoheadrightarrow_R (g\ a)$, and prove it using the transitivity of $\twoheadrightarrow_R$, since $a \rightarrow_R b$ and $b \twoheadrightarrow_R (g\ a)$.
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.

`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 H1*. 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` *H1*. In order to apply *IHHrefl1*, we instantiate c_0 with $(g\ a)$.

`destruct` *H1*. 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 $a \twoheadrightarrow_R x$.

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

$\times$ `apply 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 intermediary step.

+ `intros` *b0 Hrefl1 IHHrefl1 H'*. 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 $a \rightarrow_R b$ 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` *IHHrefl2* `with` *b0*. The second goal, i.e. the inductive case can be proved by the second induction hypothesis *IHHrefl2*, and each of the 4 conditions generated by this hypothesis is solved as follows:

× `apply refltrans_composition with` $(g\ a)$; `apply` *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 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`.

An alternative proof that Z implies confluence is possible via the notion of semiconfluence, which is equivalent to confluence, as done in [4]. Our proof is also constructive, but we will not explain it here due to lack of space, but as the interested reader can visit the Coq file in our GitHub repository.

```
Lemma red_to_f {A: Type}: ∀ (R: Rel A) (f: A → A), f_is_Z R f → (∀ a b: A, refltrans R a b → refltrans R (f a) (f b)). Proof. intros R f H a b Hred. unfold f_is_Z in H. induction Hred. - apply refl. - apply refltrans_composition with (f b). + apply H; assumption. + assumption. 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. Proof. intros R HZ_prop. unfold Z_prop in HZ_prop. unfold SemiConfl. destruct HZ_prop. intros a b c Hrefl Hrefl'. assert (Haxa: refltrans R a (x a)). { apply rtrans with b. - assumption. - apply H. assumption. } apply H in Hrefl. destruct Hrefl. clear H1. generalize dependent b. induction Hrefl'. - intros. ∃ (x a). split; assumption. - intros. destruct IHHrefl' with b0. + apply refltrans_composition with (x a); apply H; assumption. + apply refltrans_composition with (x b). × apply refltrans_composition with (x a). ** assumption. ** apply H. assumption. × apply refl. + ∃ x0. assumption. Qed.

Theorem Semi_equiv_Confl {A: Type}: ∀ R: Rel A, Confl R ↔ SemiConfl R. Proof. unfold Confl. unfold SemiConfl. intro R. split. - intros. apply H with a. + apply rtrans with b. × assumption. × apply refl. + assumption. - intros. generalize dependent c. induction H0. + intros. ∃ c. split. × assumption. × apply refl. + intros. specialize (H a). specialize (H b). specialize (H c0). apply H in H0. × destruct H0. destruct H0. apply IHrefltrans in H0. destruct H0. destruct H0. ∃ x0. split. ** assumption. ** apply refltrans_composition with x; assumption. × assumption. Qed.
```

3 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 [6]. 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 $(A, \rightarrow)$, one must be able to decompose the relation $\rightarrow$ into two parts, say $\rightarrow_1$ and $\rightarrow_2$ such that $\rightarrow = \rightarrow_1 \cup \rightarrow_2$. The disjoint union can be inductively defined in Coq as follows:

```
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).
```

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[7], etc.

```
Lemma union_or {A}: ∀ (r1 r2: Rel A) (a b: A), (r1 !_! r2) a b ↔ (r1 a b) ∨ (r2 a b). Proof. intros
r1 r2 a b; split. - intro Hunion. inversion Hunion; subst. + left; assumption. + right; assumption. -
intro Hunion. inversion Hunion. + apply union_left; assumption. + apply union_right; assumption. Qed.
```

The compositional Z is defined in terms of a weaker property:

Definition 3.1 Let $(A, \rightarrow)$ be an ARS and $\rightarrow_x$ another relation on A. A mapping f satisfies the *weak Z property* for $\rightarrow$ by $\rightarrow_x$ if $a \rightarrow b$ implies $b \twoheadrightarrow_x f(a)$ and $f(a) \twoheadrightarrow_x f(b)$. Therefore, a mapping f satisfies the Z property for $\rightarrow$, 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 an extension of the Z property for compositional functions, where composition is defined as usual:

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

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

Theorem 3.2 *[6] Let $(A, \rightarrow)$ be an ARS such that $\rightarrow = \rightarrow_1 \cup \rightarrow_2$. If there exists mappings $f_1, f_2 : A \rightarrow A$ such that*

(i) *f_1 is Z for $\rightarrow_1$*

(ii) *$a \rightarrow_1 b$ implies $f_2(a) \twoheadrightarrow f_2(b)$*

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

(iv) *$f_2 \circ f_1$ is weakly Z for $\rightarrow_2$ by $\rightarrow$*

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

We define the predicate *Z_comp* that corresponds to the hypothesis of Theorem 3.2, where $\rightarrow_1$ (resp. $\rightarrow_2$) is written as *R1* (resp. *R2*):

```
Definition Z_comp {A:Type} (R :Rel A) := ∃ (R1 R2: Rel A) (f1 f2: A → A), R = (R1 !_! R2) ∧ 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)).
```

```
Lemma refltrans_union {A:Type}: ∀ (R R' :Rel A) (a b: A), refltrans R a b → refltrans (R !_! R') a b.
Proof. intros R R' a b Hrefl. induction Hrefl. - apply refl. - apply rtrans with b. + apply union_left;
assumption. + assumption. Qed.
```

As stated by Theorem 3.2, the compositional Z gives a sufficient condition for compositional functions to be Z. In other words, compositional Z implies Z, which can be seen by the diagrams of Figure 2

In what follows, we present our Coq proof of this fact in the same style of the first section by interleaving English followed by the corresponding Coq code.

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

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

Figure 2. Compositional Z implies Z

`intros` *R H*.

Now unfold the definitions of *Z_prop* and *Z_comp* as presented before, and name the hypothesis of compositional Z as in Theorem 3.2.

`unfold Z_prop. unfold Z_comp in` *H*. `destruct` *H* `as` [*R1* [*R2* [*f1* [*f2* [*Hunion* [*H1* [*H2* [*H3 H4*]]]]]]]].

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)
```

Take the composition *f2 # f1* as *f* as suggested by the above diagrams, and show that *f2 # f1* is Z.

∃ (*f2* `#` *f1*).

So, let *a* and *b* be elements of *A*, and suppose that *a* *R*-reduces to *b* in one step. Call *HR* this hypothesis.

`intros` *a b HR*.

Since *R* is the union of *R1* and *R2*, one has that *a* reduces to *b* in one step via either *R1* or *R2*.

`inversion` *Hunion*; `subst`. `clear` *H*. `inversion` *HR*; `subst`.

Firstly, suppose that *a* *R1*-reduces in one step to *b*.

`- split`.

In order to prove that *b* R-reduces to ((*f2 # f1*) *a*), we first need to show that *b* *R1*-reduces to (*f1 a*) as shown in Figure 2.

`+ apply refltrans_composition with` (*f1 a*). `× apply` *H1* `in` *H*. `destruct` *H*. `apply refltrans_union; assumption`.

The next step is then to prove that (*f1 a*) *R*-reduces to ((*f2 # f1*) *a*), which is a direct consequence of *H3*.

`× apply` *H3* `with` *a*; `reflexivity`.

The proof that ((*f2 # f1*) *a*) *R*-reduces to ((*f2 # f1*) *b*) is more tricky. Initially, note that, since *R1 a b* then we get that *refltrans R1* (*f1 a*) (*f1 b*) by the Z property.

`+ apply` *H1* `in` *H*. `destruct` *H*. `clear` *H HR*. `unfold comp`.

Now, the goal can be obtained from *H2* as long as *R1* (*f1 a*) (*f1 b*), but we only have that *refltrans R1* (*f1 a*) (*f1 b*). Therefore, we use induction on this hypothesis.

`induction` *H0*.

The reflexive case is trivial because *a* and *b* are equal.

× `apply refl.`

In the transitive case, we have that $(f1\ a)$ $R1$-reduces to $(f1\ b)$ in at least one step. The current proof context is as follows:

```
1 subgoal (ID 314)

- A : Type
- R1, R2 : Rel A
- f1, f2 : A → A
- H1 : f_is_Z R1 f1
- H4 : f_is_weak_Z R2 (R1 !_! R2) (f2 # f1)
- H3 : ∀ a b : A, b = f1 a → refltrans (R1 !_! R2) b (f2 b)
- H2 : ∀ a b : A, R1 a b → refltrans (R1 !_! R2) (f2 a) (f2 b)
- a, b, a0, b0, c : A
- H : R1 a0 b0
- H0 : refltrans R1 b0 c
- IHrefltrans : refltrans (R1 !_! R2) (f2 b0) (f2 c)
  ------------------------------------------------------------
  refltrans (R1 !_! R2) (f2 a0) (f2 c)
```

Therefore, there exists some $b0$ such that $R1\ a0\ b0$ and $refltrans\ R1\ b0\ c$ and we need to prove that $refltrans\ (R1\ !\_!\ R2)\ (f2\ a0)\ (f2\ c)$. This can be done in two steps by transitivity of $refltrans$ taking $(f2\ b0)$ as the intermediary term.

× `apply refltrans_composition with` $(f2\ b0)$.

The first subgoal is then $refltrans\ (R1\ !\_!\ R2)\ (f2\ a0)\ (f2\ b0)$ that is proved by hypothesis $H2$.

** `apply` $H2$; `assumption.`

And the second subgoal $refltrans\ (R1\ !\_!\ R2)\ (f2\ b0)\ (f2\ c)$ is proved by the induction hypothesis.

** `assumption.`

Finally, when a $R2$-reduces in one step to b one concludes the proof using the assumption that $(f2\ \#\ f1)$ is weak Z.

- `apply` $H4$; `assumption. Qed.`

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 suficient condition for confluence.

```
Corollary Z_comp_is_Confl {A}: ∀ (R: Rel A), Z_comp R → Confl R. Proof. intros R H. apply Z_comp_implies_Z_prop
in H. apply Z_prop_implies_Confl; assumption. Qed.
```

Rewriting Systems with equations is another interesting and non-trivial topic [10,8]. 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].

Corollary 3.3 *[6] Let $(A, \rightarrow)$ be an ARS such that $\rightarrow = \rightarrow_1 \cup \rightarrow_2$. If there exists mappings $f_1, f_2 : A \rightarrow A$ such that*

(i) $a \rightarrow_1 b$ *implies* $f_1(a) = f_1(b)$

(ii) $a \twoheadrightarrow_1 f_1(a), \forall a$

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

(iv) $f_2 \circ f_1$ *is weakly Z for* $\rightarrow_2$ *by* $\rightarrow$

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

We define the predicate Z_comp_eq corresponding to the hypothesis of Corollary 3.3, and then we prove directly that if Z_comp_eq holds for a relation R then $Zprop\ R$ also holds. This approach differs from [6] that proves Corollary 3.3 directly from Theorem 3.2

```
Definition Z_comp_eq {A:Type} (R :Rel A) := ∃ (R1 R2: Rel A) (f1 f2: A → A), R = (R1 !_! R2) ∧ (∀
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_prop {A:Type}: ∀ (R : Rel A), Z_comp_eq R → Z_prop R. Proof.`

Let R be a relation and suppose that R satisfies the predicate *Z_comp_eq*.

`intros R Heq. unfold Z_comp_eq in Heq.`

Call *Hi* the ith hypothesis as in 3.3.

`destruct Heq as [R1 [R2 [f1 [f2 [Hunion [H1 [H2 [H3 H4]]]]]]]].`

From the definition of the predicate *Z_prop*, we need to find a map, say f that is Z. Let (*f2 # f1*) be such map.

`unfold Z_prop. ∃ (f2 # f1).`

In order to prove that (*f2 # f1*) is Z, let a and b be arbitrary elements of type A, and *Hab* be the hypothesis that a R-reduces in one step to b.

`intros a b 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:

`inversion Hunion; subst; clear H. inversion Hab; subst; clear Hab.`

The first case is when a *R1*-reduces in one step to b.

`- unfold comp; split.`

This is equivalent to say that *f2 # f1* is weak Z for *R1* by *R1* !_! *R2*. Therefore, we first prove that *refltrans* (*R1* !_! *R2*) *b* (*f2* (*f1* *a*)), which can be reduced to *refltrans* (*R1* !_! *R2*) *b* (*f1* *b*) and *refltrans* (*R1* !_! *R2*) (*f1* *b*) (*f2* (*f1* *a*)) by the transitivity of *refltrans*.

`+ apply refltrans_composition with (f1 b).`

From hypothesis *H2*, we know that *refltrans R1 a* (*f1* *a*) for all a, and hence *refltrans* (*R1* !_! *R2*) *a* (*f1* *a*) and we conclude.

`× apply refltrans_union. apply H2.`

The proof that *refltrans* (*R1* !_! *R2*) (*f1* *b*) (*f2* (*f1* *a*)) is exactly the content of hypothesis *H3*.

`× apply H1 in H. rewrite H. apply H3 with b; reflexivity.`

The proof that *refltrans* (*R1* !_! *R2*) (*f2* (*f1* *a*)) (*f2* (*f1* *b*)) is done using the reflexivity of *refltrans* because (*f2* (*f1* *a*)) = (*f2* (*f1* *b*)) by hypothesis *H1*.

`+ apply H1 in H. rewrite H. apply refl.`

When a *R2*-reduces in one step to b then we are done by hypothesis *H4*.

`- apply H4; assumption. Qed.`

4 Conclusion

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] B. Felgenhauer, J. Nagele, V. van Oostrom, and C. Sternagel. The Z property. *Archive of Formal Proofs*, 2016, 2016.

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

[6] Koji Nakazawa and Ken-etsu Fujita. Compositional Z: confluence proofs for permutative conversion. 104(6):1205–1224, 2016.

[7] 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.

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

[9] Vincent van Oostrom. Z - draft: For your mind only. 2007.

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