# The Mostowski Collapse Theorem

Arthur Freitas Ramos
David Barros Hulak
Ruy J. G. B. de Queiroz

October 26, 2026

**Abstract**

This entry formalizes the Mostowski collapse theorem in Isabelle/ZF [3, 4]. For a set equipped with a well-founded extensional relation, the development defines the collapsing map by well-founded recursion, proves that its range is transitive, and shows that the map is an order isomorphism between the original relation and membership on the transitive range. The construction is also proved unique among maps satisfying the same recursive equation. The result is a standard bridge from abstract well-founded extensional structures to transitive membership structures, and complements the existing Isabelle/ZF and AFP developments on ordinals, cardinals, forcing, and transitive models [2, 1]. AI assistance was used for proof engineering. The final definitions, statements, and proofs are checked by Isabelle.

## Contents

**1 The Mostowski Collapse Theorem** **1**

**theory** *Mostowski-Collapse*
**imports** *ZF*
**begin**

## 1 The Mostowski Collapse Theorem

The Mostowski collapse theorem says that every set-sized well-founded extensional relation is isomorphic to membership on a transitive set. The collapsing map is obtained by well-founded recursion: each object is sent to the set of the collapses of its predecessors. We work with a set $A$ and collapse the restriction of a relation $r$ to $A$. The assumptions are therefore $wf[A](r)$ and extensionality on $A$; the final isomorphism is stated for $r \cap A*A$.

**definition** *extensional-on* :: $[i, i] \Rightarrow o$ **where**
*extensional-on*$(A,r) \equiv$
$\forall x{\in}A.\ \forall y{\in}A.\ (\forall z{\in}A.\ \langle z,x\rangle \in r \longleftrightarrow \langle z,y\rangle \in r) \longrightarrow x = y$

**definition** *collapse* :: $[i,\ i,\ i] \Rightarrow i$ **where**
  $collapse(A,r,x) \equiv$
    $wfrec[A](r,\ x,\ \lambda y\ f.\ f\ ``\ ((r\ -``\ \{y\})\ \cap\ A))$

**definition** *collapse-map* :: $[i,\ i] \Rightarrow i$ **where**
  $collapse\text{-}map(A,r) \equiv (\lambda x{\in}A.\ collapse(A,r,x))$

**definition** *collapse-range* :: $[i,\ i] \Rightarrow i$ **where**
  $collapse\text{-}range(A,r) \equiv range(collapse\text{-}map(A,r))$

**lemma** *collapse-unfold*:
  **assumes** $wf[A](r)\ x \in A$
  **shows** $collapse(A,r,x) = \{collapse(A,r,y).\ y \in (r\ -``\ \{x\})\ \cap\ A\}$
  **unfolding** *collapse-def*
  **using** *assms* **by** (*simp add*: *wfrec-on image-lam*)

**lemma** *collapse-memI*:
  **assumes** $wf[A](r)\ x \in A\ y \in A\ \langle y,x\rangle \in r$
  **shows** $collapse(A,r,y) \in collapse(A,r,x)$
**proof** −
  **have** $y \in (r\ -``\ \{x\})\ \cap\ A$
    **using** *assms* **by** (*simp add*: *vimage-singleton-iff*)
  **with** *collapse-unfold* [*OF assms*(*1*,*2*)] **show** *?thesis*
    **by** *force*
**qed**

**lemma** *collapse-memE*:
  **assumes** $wf[A](r)\ x \in A\ u \in collapse(A,r,x)$
  **obtains** $y$ **where** $y \in A\ \langle y,x\rangle \in r\ u = collapse(A,r,y)$
  **using** *assms* **by** (*auto simp*: *collapse-unfold vimage-singleton-iff*)

**lemma** *collapse-injective*:
  **assumes** *wf*: $wf[A](r)$ **and** *ext*: $extensional\text{-}on(A,r)$
    **and** *xA*: $x \in A$ **and** *yA*: $y \in A$
    **and** *eq*: $collapse(A,r,x) = collapse(A,r,y)$
  **shows** $x = y$
**proof** −
  **have** $\forall\, y{\in}A.\ collapse(A,r,x) = collapse(A,r,y) \longrightarrow x = y$
    **using** *wf xA*
  **proof** (*rule wf-on-induct-raw*)
    **fix** $x$
    **assume** *xA'*: $x \in A$
      **and** *IH*: $\forall\, z{\in}A.\ \langle z,x\rangle \in r \longrightarrow$
        $(\forall\, y{\in}A.\ collapse(A,r,z) = collapse(A,r,y) \longrightarrow z = y)$
    **show** $\forall\, y{\in}A.\ collapse(A,r,x) = collapse(A,r,y) \longrightarrow x = y$
    **proof**
      **fix** $y$
      **assume** *yA'*: $y \in A$

```
      show collapse(A,r,x) = collapse(A,r,y) ⟶ x = y
      proof
        assume eqxy: collapse(A,r,x) = collapse(A,r,y)
        have pred-eq: ∀ z∈A. ⟨z,x⟩ ∈ r ⟷ ⟨z,y⟩ ∈ r
        proof
          fix z
          assume zA: z ∈ A
          show ⟨z,x⟩ ∈ r ⟷ ⟨z,y⟩ ∈ r
          proof
            assume zx: ⟨z,x⟩ ∈ r
            have collapse(A,r,z) ∈ collapse(A,r,x)
              using wf xA' zA zx by (rule collapse-memI)
            then have collapse(A,r,z) ∈ collapse(A,r,y)
              using eqxy by simp
            then obtain w where wA: w ∈ A and wy: ⟨w,y⟩ ∈ r
              and zw: collapse(A,r,z) = collapse(A,r,w)
              by (rule collapse-memE [OF wf yA'])
            then show ⟨z,y⟩ ∈ r
              using IH zA zx wA zw by force
          next
            assume zy: ⟨z,y⟩ ∈ r
            have collapse(A,r,z) ∈ collapse(A,r,y)
              using wf yA' zA zy by (rule collapse-memI)
            then have collapse(A,r,z) ∈ collapse(A,r,x)
              using eqxy by simp
            then obtain w where wA: w ∈ A and wx: ⟨w,x⟩ ∈ r
              and zw: collapse(A,r,z) = collapse(A,r,w)
              by (rule collapse-memE [OF wf xA'])
            then show ⟨z,x⟩ ∈ r
              using IH zA wx wA zw by force
          qed
        qed
        then show x = y
          using ext xA' yA' by (auto simp: extensional-on-def)
      qed
    qed
  qed
  then show ?thesis
    using yA eq by simp
qed

lemma collapse-map-type:
  collapse-map(A,r) ∈ A → collapse-range(A,r)
  unfolding collapse-map-def collapse-range-def
  by (intro lam-type rangeI) (auto simp: lam-def)

lemma collapse-map-apply [simp]:
  x ∈ A ⟹ collapse-map(A,r) ‘ x = collapse(A,r,x)
  by (simp add: collapse-map-def)
```

**lemma** *collapse-map-inj*:
**assumes** *wf*[*A*](*r*) *extensional-on*(*A*,*r*)
**shows** *collapse-map*(*A*,*r*) ∈ *inj*(*A*, *collapse-range*(*A*,*r*))
**unfolding** *inj-def*
**proof**
**show** *collapse-map*(*A*,*r*) ∈ *A* → *collapse-range*(*A*,*r*)
**by** (*rule collapse-map-type*)
**next**
**show** ∀ *w*∈*A*. ∀ *x*∈*A*. *collapse-map*(*A*,*r*)‘*w* = *collapse-map*(*A*,*r*)‘*x* ⟶ *w* = *x*
**proof** (*intro strip*)
**fix** *w x*
**assume** *wA*: *w* ∈ *A* **and** *xA*: *x* ∈ *A* **and** *eq*: *collapse-map*(*A*,*r*)‘*w* = *collapse-map*(*A*,*r*)‘*x*
**then have** *collapse*(*A*,*r*,*w*) = *collapse*(*A*,*r*,*x*)
**by** *simp*
**then show** *w* = *x*
**by** (*rule collapse-injective* [*OF assms wA xA*])
**qed**
**qed**

**lemma** *collapse-map-bij*:
**assumes** *wf*[*A*](*r*) *extensional-on*(*A*,*r*)
**shows** *collapse-map*(*A*,*r*) ∈ *bij*(*A*, *collapse-range*(*A*,*r*))
**using** *collapse-map-inj* [*OF assms*]
**unfolding** *collapse-range-def*
**by** (*rule inj-bij-range*)

**lemma** *collapse-range-iff*:
*u* ∈ *collapse-range*(*A*,*r*) ⟷ (∃ *x* ∈ *A*. *u* = *collapse*(*A*,*r*,*x*))
**unfolding** *collapse-range-def collapse-map-def range-def lam-def*
**by** *blast*

**lemma** *collapse-range-transitive*:
**assumes** *wf*: *wf*[*A*](*r*)
**shows** *Transset*(*collapse-range*(*A*,*r*))
**unfolding** *Transset-def*
**by** (*force simp*: *collapse-range-iff elim*: *collapse-memE* [*OF wf*])

**lemma** *collapse-mem-iff*:
**assumes** *wf*: *wf*[*A*](*r*) **and** *ext*: *extensional-on*(*A*,*r*)
**and** *xA*: *x* ∈ *A* **and** *yA*: *y* ∈ *A*
**shows** *collapse*(*A*,*r*,*x*) ∈ *collapse*(*A*,*r*,*y*) ⟷ ⟨*x*,*y*⟩ ∈ *r*
**proof**
**assume** *collapse*(*A*,*r*,*x*) ∈ *collapse*(*A*,*r*,*y*)
**then obtain** *z* **where** *zA*: *z* ∈ *A* **and** *zy*: ⟨*z*,*y*⟩ ∈ *r*
**and** *xz*: *collapse*(*A*,*r*,*x*) = *collapse*(*A*,*r*,*z*)
**by** (*rule collapse-memE* [*OF wf yA*])
**have** *x* = *z*

```
    using collapse-injective [OF wf ext xA zA xz] .
  then show ⟨x,y⟩ ∈ r
    using zy by simp
next
  assume ⟨x,y⟩ ∈ r
  then show collapse(A,r,x) ∈ collapse(A,r,y)
    by (rule collapse-memI [OF wf yA xA])
qed

lemma collapse-ord-iso:
  assumes wf: wf[A](r) and ext: extensional-on(A,r)
  shows collapse-map(A,r) ∈
    ord-iso(A, r ∩ A*A, collapse-range(A,r), Memrel(collapse-range(A,r)))
proof (rule ord-isoI)
  show collapse-map(A,r) ∈ bij(A, collapse-range(A,r))
    using wf ext by (rule collapse-map-bij)
next
  fix x y
  assume xA: x ∈ A and yA: y ∈ A
  show ⟨x,y⟩ ∈ r ∩ A*A ⟷
      ⟨collapse-map(A,r)`x, collapse-map(A,r)`y⟩ ∈ Memrel(collapse-range(A,r))
    using xA yA collapse-mem-iff [OF wf ext xA yA]
    by (auto simp: collapse-range-iff)
qed

theorem mostowski-collapse:
  assumes wf[A](r) extensional-on(A,r)
  shows Transset(collapse-range(A,r)) ∧
    collapse-map(A,r) ∈
      ord-iso(A, r ∩ A*A, collapse-range(A,r), Memrel(collapse-range(A,r)))
  using assms
  by (simp add: collapse-range-transitive collapse-ord-iso)

theorem mostowski-collapse-exists:
  assumes wf[A](r) extensional-on(A,r)
  shows ∃ M f. Transset(M) ∧ f ∈ ord-iso(A, r ∩ A*A, M, Memrel(M))
  using mostowski-collapse [OF assms] by blast

lemma collapse-unique:
  assumes wf: wf[A](r)
    and ftype: f ∈ A → B
    and frec: ⋀x. x ∈ A ⟹ f`x = f `` ((r -`` {x}) ∩ A)
    and xA: x ∈ A
  shows f`x = collapse(A,r,x)
  using wf xA
proof (rule wf-on-induct-raw)
  fix x
  assume xA': x ∈ A
    and IH: ∀ y∈A. ⟨y,x⟩ ∈ r ⟶ f ` y = collapse(A,r,y)
```

```
  have f‘x = f `` ((r −`` {x}) ∩ A)
    using frec xA′ by simp
  also have ... = {collapse(A,r,y). y ∈ (r −`` {x}) ∩ A}
  proof (rule equalityI)
    show f `` ((r −`` {x}) ∩ A) ⊆
      {collapse(A,r,y). y ∈ (r −`` {x}) ∩ A}
    proof
      fix u
      assume u ∈ f `` ((r −`` {x}) ∩ A)
      then obtain y where pair: ⟨y,u⟩ ∈ f
        and ypred: y ∈ (r −`` {x}) ∩ A
        by (rule imageE)
      have uy: u = f‘y
        using pair ftype by (simp add: apply-equality)
      have f‘y = collapse(A,r,y)
        using IH ypred by (simp add: vimage-singleton-iff)
      then have u = collapse(A,r,y)
        using uy by simp
      moreover have y ∈ (r −`` {x}) ∩ A
        by (rule ypred)
      ultimately show u ∈ {collapse(A,r,y). y ∈ (r −`` {x}) ∩ A}
        by (rule RepFun-eqI)
    qed
  next
    show {collapse(A,r,y). y ∈ (r −`` {x}) ∩ A} ⊆
      f `` ((r −`` {x}) ∩ A)
    proof
      fix u
      assume u ∈ {collapse(A,r,y). y ∈ (r −`` {x}) ∩ A}
      then obtain y where ypred: y ∈ (r −`` {x}) ∩ A
        and uy: u = collapse(A,r,y)
        by (rule RepFunE)
      have y ∈ A
        using ypred by simp
      then have pair: ⟨y,f‘y⟩ ∈ f
        by (rule apply-Pair [OF ftype])
      have f‘y = collapse(A,r,y)
        using IH ypred by (simp add: vimage-singleton-iff)
      then have ⟨y,u⟩ ∈ f
        using uy pair by simp
      then show u ∈ f `` ((r −`` {x}) ∩ A)
        using ypred by (rule imageI)
    qed
  qed
  also have ... = collapse(A,r,x)
    using collapse-unfold [OF wf xA′] by simp
  finally show f‘x = collapse(A,r,x) .
qed
```

**lemma** *collapse-map-unique*:
**assumes** *wf*: $wf[A](r)$
**and** *ftype*: $f \in A \to B$
**and** *frec*: $\bigwedge x.\ x \in A \Longrightarrow f`x = f\ ``\ ((r\ -``\ \{x\}) \cap A)$
**shows** $f = \mathit{collapse\text{-}map}(A,r)$
**unfolding** *collapse-map-def*
**using** *collapse-unique* [*OF wf ftype frec*]
**by** (*force intro*: *fun-extension ftype lam-type*)

**lemma** *collapse-range-unique*:
**assumes** *wf*: $wf[A](r)$
**and** *ftype*: $f \in A \to B$
**and** *frec*: $\bigwedge x.\ x \in A \Longrightarrow f`x = f\ ``\ ((r\ -``\ \{x\}) \cap A)$
**and** $M$: $M = \mathit{range}(f)$
**shows** $M = \mathit{collapse\text{-}range}(A,r)$
**using** *collapse-map-unique* [*OF wf ftype frec*] $M$
**unfolding** *collapse-range-def* **by** *simp*

**end**

# References


[1] T. Jech. *Set Theory: The Third Millennium Edition, Revised and Expanded*. Springer Monographs in Mathematics. Springer, Berlin, Heidelberg, 3 edition, 2003. DOI: `https://doi.org/10.1007/3-540-44761-X`.

[2] K. Kunen. *Set Theory: An Introduction to Independence Proofs*, volume 102 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1980.

[3] A. Mostowski. An undecidable arithmetical statement. *Fundamenta Mathematicae*, 36(1):143–164, 1949. DOI: `https://doi.org/10.4064/fm-36-1-143-164`.

[4] A. Mostowski. *Constructible Sets with Applications*, volume 57 of *Studies in Logic and the Foundations of Mathematics*. North-Holland; PWN–Polish Scientific Publishers, Amsterdam; Warsaw, 1969.