Well Founded Relations and Recursion

Roger Bishop Jones

Abstract

Fixed points, well founded relations and a recursion theorem.

http://www.rbjones.com/rbjpub/pp/doc/t005.pdf

Created 2004/07/15

Last Modified Date: 2013/01/10 16:35:58

http://www.rbjones.com/rbjpub/pp/doc/t005.pdf

Id: t005.doc,v 1.18 2013/01/10 16:35:58 rbj Exp

Contents

References

[1] Roger Bishop Jones. Introduction to Work in Progress. *RBJones.com*, 2010. `http://www.rbjones.com/rbjpub/pp/doc/t000.pdf`.

1 Introduction

For context and motivation see [1].

There are in this document two treatments of well-foundedness and the recursion theorem which differ primarily only in the type of the relations which they deal with.

Actually, they don't look different at all. Must check out whether there are *any* differences!

They were at one time in separate documents but have now been brought together in the one document as a step towards rationalisation.

The material on transitive closure is common to both.

2 Transitive Closure

Elementary results about transitive relations and transitive closure.

The new theory *tc* is first created.

SML

```
open_theory "hol";
force_new_theory "tc";
set_pc "hol";
```

2.1 Definitions

There is in hol4 a theory of relations in which transitive closure is defined in the obvious way, similar to the way in which it is defined here. There is also a package providing support for defining relations using rules, and an example in which a Church-Rosser result for the pure combinatory logic is obtained very concisely by using a definition of reflexive transitive relation through that package instead of the one in the theory of relations. The main advantage of this alternative approach is that it gives automatically induction principles for reasoning about reflexive transitive closures which are not available in the theory of relations.

The following development begins in a similar vein to the hol4 theory of relations but then continues to obtain the results delivered by defining reflexive transitive closure using the hol4 relation definition package. Of these the most important are the induction principles, but I have replicated the other principles which are automatically obtained in hol4.

The results below which involve decomposition of transitive closure into a path of direct reductions represent a more cumbersome approach to proofs about transitive closures which demand some kind of induction. It may be best in considering such proofs, first to look at the induction principles which follow these decompositions.

HOL Constant

```
trans: ('a → 'a → BOOL) → BOOL
─────────────────────────────
∀r• trans r ⇔ ∀ s t u• r s t ∧ r t u ⇒ r s u
```

HOL Constant

$\boldsymbol{tc}$: $('a \to 'a \to BOOL) \to ('a \to 'a \to BOOL)$

$\forall r\bullet\ tc\ r = \lambda\ s\ t\bullet\ \forall tr\bullet\ trans\ tr \land (\forall v\ u\bullet\ r\ v\ u \Rightarrow tr\ v\ u) \Rightarrow tr\ s\ t$

2.2 Theorems

$\boldsymbol{tran\_tc\_thm}$ =
$\vdash \forall r\bullet\ trans\ (tc\ r)$

$\boldsymbol{tran\_tc\_thm2}$ =
$\vdash \forall\ r\ x\ y\ z\bullet\ tc\ r\ x\ y \land tc\ r\ y\ z$
$\Rightarrow tc\ r\ x\ z$

$\boldsymbol{tc\_incr\_thm}$ =
$\vdash \forall r\ x\ y\ \bullet\ r\ x\ y$
$\Rightarrow tc\ r\ x\ y$

$\boldsymbol{tc\_decomp\_thm}$ =
$\vdash \forall\ r\ x\ y\bullet\ tc\ r\ x\ y \land \neg\ r\ x\ y$
$\Rightarrow \exists z\bullet\ tc\ r\ x\ z \land r\ z\ y$

$\boldsymbol{tc\_decomp\_thm2}$ =
$\vdash \forall\ r\ x\ y\bullet\ tc\ r\ x\ y$
$\Rightarrow (\exists\ f\ n\bullet\ x = f\ 0 \land y = f\ n \land (\forall\ m\bullet\ m < n \Rightarrow r\ (f\ m)\ (f\ (m + 1))))$

$\boldsymbol{tc\_decomp\_thm3}$ =
$\vdash \forall\ r\ x\ y\bullet\ tc\ r\ x\ y$
$\Rightarrow (\exists\ f\ n\bullet\ x = f\ 0 \land y = f\ (n + 1)$
$\land (\forall\ m\bullet\ m \le n \Rightarrow r\ (f\ m)\ (f\ (m + 1))))$

$\boldsymbol{tc\_decomp\_thm4}$ =
$\vdash \forall\ r\ x\ y\bullet\ (\exists\ f\ n\bullet\ x = f\ 0 \land y = f\ (n + 1) \land (\forall\ m\bullet\ m \le n \Rightarrow r\ (f\ m)\ (f\ (m + 1))))$
$\Rightarrow tc\ r\ x\ y$

$\boldsymbol{tc\_decomp\_thm5}$ =
$\vdash \forall\ r\ x\ y\bullet\ tc\ r\ x\ y \land \neg\ r\ x\ y$
$\Rightarrow (\exists\ z\bullet\ r\ x\ z \land tc\ r\ z\ y)$

$\boldsymbol{tc\_\Leftrightarrow\_thm}$ =
$\vdash \forall\ r\ x\ y\bullet\ tc\ r\ x\ y$
$\Leftrightarrow (\exists\ f\ n\bullet\ x = f\ 0 \land y = f\ (n + 1) \land (\forall\ m\bullet\ m \le n \Rightarrow r\ (f\ m)\ (f\ (m + 1))))$

$\boldsymbol{tc\_mono\_thm}$ =
$\vdash \forall\ r1\ r2\bullet\ (\forall\ x\ y\bullet\ r1\ x\ y \Rightarrow r2\ x\ y)$
$\Rightarrow (\forall\ x\ y\bullet\ tc\ r1\ x\ y \Rightarrow tc\ r2\ x\ y)$

$\boldsymbol{tc\_p\_thm}$ =
$\vdash \forall\ r\ p\bullet\ (\forall\ x\ y\bullet\ r\ x\ y \Rightarrow p\ x)$
$\Rightarrow (\forall\ x\ y\bullet\ tc\ r\ x\ y \Rightarrow p\ x)$

$tc\_induced\_thm$ =
$\vdash \forall\ r\ u\ x \bullet\ tc\ (\lambda\ x\ y \bullet\ r\ (f\ x)\ (f\ y))\ u\ x$
$\Rightarrow tc\ r\ (f\ u)\ (f\ x)$
$tran\_tc\_id\_thm$ =
$\vdash \forall\ r \bullet\ trans\ r \Rightarrow tc\ r = r$

tc_decomp_thm6 =
$\vdash \forall\ r\ x\ y \bullet\ tc\ r\ x\ y \Rightarrow r\ x\ y \lor (\exists\ z \bullet\ tc\ r\ x\ z \land r\ z\ y)$
tc_decomp_thm7 =
$\vdash \forall\ r\ x\ y \bullet\ tc\ r\ x\ y \Rightarrow r\ x\ y \lor (\exists\ z \bullet\ r\ x\ z \land tc\ r\ z\ y)$

tc_ind0 =
$\vdash \forall\ R\ r \bullet\ (\forall\ x\ y \bullet\ r\ x\ y \Rightarrow R\ x\ y) \land (\forall\ x\ y\ z \bullet\ R\ x\ y \land R\ y\ z \Rightarrow R\ x\ z)$
$\Rightarrow (\forall\ x\ y \bullet\ tc\ r\ x\ y \Rightarrow R\ x\ y)$

tc_ind1 =
$\vdash \forall\ R\ r \bullet\ (\forall\ x\ y \bullet\ r\ x\ y \Rightarrow R\ x\ y) \land (\forall\ x\ y\ z \bullet\ R\ x\ y \land r\ y\ z \Rightarrow R\ x\ z)$
$\Rightarrow (\forall\ x\ y \bullet\ tc\ r\ x\ y \Rightarrow R\ x\ y)$

tc_ind2 =
$\vdash \forall\ R\ r \bullet\ (\forall\ x\ y \bullet\ r\ x\ y \Rightarrow R\ x\ y) \land (\forall\ x\ y\ z \bullet\ r\ x\ y \land R\ y\ z \Rightarrow R\ x\ z)$
$\Rightarrow (\forall\ x\ y \bullet\ tc\ r\ x\ y \Rightarrow R\ x\ y)$

HOL Constant

$rtc: ('a \rightarrow 'a \rightarrow BOOL) \rightarrow ('a \rightarrow 'a \rightarrow BOOL)$

$\forall r \bullet\ rtc\ r = \lambda\ s\ t \bullet\ s = t \lor tc\ r\ s\ t$

$tran\_rtc\_thm$ =
$\vdash \forall r \bullet\ trans\ (rtc\ r)$

$tran\_rtc\_thm2$ =
$\vdash \forall\ r\ s\ t\ u \bullet\ rtc\ r\ s\ t \land rtc\ r\ t\ u \Rightarrow rtc\ r\ s\ u$

rtc_incr_thm =
$\vdash \forall r\ x\ y \bullet\ r\ x\ y \lor x = y \Rightarrow rtc\ r\ x\ y$

rtc_decomp_thm =
$\vdash \forall\ R\ a0\ a1 \bullet\ rtc\ R\ a0\ a1 \Rightarrow a1 = a0 \lor (\exists\ y \bullet\ R\ a0\ y \land rtc\ R\ y\ a1)$
rtc_decomp_thm2 =
$\vdash \forall\ R\ a0\ a1 \bullet\ rtc\ R\ a0\ a1 \Rightarrow a1 = a0 \lor R\ a0\ a1 \lor (\exists\ y \bullet\ R\ a0\ y \land tc\ R\ y\ a1)$
rtc_decomp_thm3 =
$\vdash \forall\ R\ a0\ a1 \bullet\ rtc\ R\ a0\ a1 \Rightarrow a1 = a0 \lor R\ a0\ a1 \lor (\exists\ y \bullet\ tc\ R\ a0\ y \land R\ y\ a1)$

```
rtc_mono_thm =
    ⊢ ∀ r1 r2• (∀ x y• r1 x y ⇒ r2 x y)
     ⇒ (∀ x y• rtc r1 x y ⇒ rtc r2 x y)

rtc_ind0 =
    ⊢ ∀ R r• (∀ x y• r x y ⇒ R x y) ∧ (∀ x• R x x)
        ∧ (∀ x y z• R x y ∧ R y z ⇒ R x z)
         ⇒ (∀ x y• rtc r x y ⇒ R x y)

rtc_ind1 =
    ⊢ ∀ R r• (∀ x• R x x) ∧ (∀ x y z• R x y ∧ r y z ⇒ R x z)
     ⇒ (∀ x y• rtc r x y ⇒ R x y)

rtc_ind =
    ⊢ ∀ r R• (∀ x• R x x) ∧ (∀ x y z• r x y ∧ R y z ⇒ R x z)
     ⇒ (∀ x y• rtc r x y ⇒ R x y)
```

```
rtc_rules =
    ⊢ ∀ r• (∀ x• rtc r x x) ∧ (∀ x y z• r x y ∧ rtc r y z ⇒ rtc r x z)
```

We are now able to obtain a stronger induction principle.

```
rtc_strongind =
    ⊢ ∀ r rtc'• (∀ x• rtc' x x)
        ∧ (∀ x y z• r x y ∧ rtc r y z ∧ rtc' y z ⇒ rtc' x z)
         ⇒ (∀ a0 a1• rtc r a0 a1 ⇒ rtc' a0 a1)
```

2.3 Induction Tactics

SML

```
fun rel_induction_tac (thm : THM): TACTIC =
(    let fun bad_thm thm = thm_fail "REL_INDUCTION_T" 29021 [thm];
        val ([r,rtc], body) = (strip_∀ (concl thm))
            handle Fail _ => bad_thm thm;
        val (prem, conc) = dest_⇒ body
            handle Fail _ => bad_thm thm;
        fun match tm = simple_ho_match [] tm conc
            handle Fail _ => bad_thm thm;
    in fn (asms, tm) =>
        let val (tys, tms) = match tm;
            val nth = (conv_rule (MAP_C β_conv) (inst_term_rule tms
                    (inst_type_rule tys (all_∀_elim thm))));
        in bc_tac [nth] (asms, tm)
        end
    end
);
```

SML

```
val rtc_ind_tac = rel_induction_tac rtc_ind;
val rtc_strongind_tac = rel_induction_tac rtc_strongind;
```

```
rtc_cases =
    ⊢ ∀ R a0 a1• rtc R a0 a1 ⇔ a1 = a0 ∨ (∃ y• R a0 y ∧ rtc R y a1)
```

3 Well Founded Relations (I)

SML

```
open_theory "tc";
force_new_theory "wfrel";
set_pc "hol";
```

Definition of well-founded and transitive-well-founded and proof that the transitive closure of a well-founded relation is transitive-well-founded.

HOL Constant

```
well_founded: ('a → 'a → BOOL) → BOOL
────────────────────────────────────
∀r• well_founded r ⇔ ∀ s • (∀ x • (∀ y • r y x ⇒ s y) ⇒ s x) ⇒ ∀ x • s x
```

HOL Constant

```
twfp: ('a → 'a → BOOL) → BOOL
────────────────────────────────────
∀r• twfp r ⇔ well_founded r ∧ trans r
```

The first thing I need to prove here is that the transitive closure of a well-founded relation is also well-founded. This provides a form of induction with a stronger induction hypothesis.

Naturally we would expect this to be proven inductively and the question is therefore what property to use in the inductive proof, the observation that the transitive closure of a relation is well-founded is not explicitly the ascription of a property to the field of the relation. The obvious method is to relativise the required result to the transitive closure of a set, giving a property of sets, and then to prove that this property is hereditary if the relation is well-founded.

```
tcwf_lemma1 =
    ⊢ ∀ s r• well_founded r
            ⇒ (∀ x• (∀ y• tc r y x ⇒ (∀ z• tc r z y ⇒ s z) ⇒ s y)
                ⇒ (∀ y• tc r y x ⇒ s y))

wf_lemma =
    ⊢ ∀ r• well_founded r ⇒ (∀ s• (∀ t• (∀ u• r u t ⇒ s u) ⇒ s t) ⇒ (∀ e• s e))

tcwf_lemma2 =
    ⊢ ∀ r• well_founded r
            ⇒ (∀ s• (∀ t• (∀ u• tc r u t ⇒ s u) ⇒ s t) ⇒ (∀ e• s e))

wf_tc_wf_thm = ⊢ ∀ r• well_founded r ⇒ well_founded (tc r)
```

Now we prove that if the transitive closure of a relation is well-founded then so must be the relation.

```
tc_wf_wf_thm = ⊢ ∀ r• well_founded (tc r) ⇒ well_founded r
```

3.1 Induction Tactics etc.

We here define a general tactic for performing induction using some well-founded relation. The following function (I think these things are called "THM-TACTICAL"s) must be given a theorem which asserts that some relation is well-founded, and then a THM-TACTIC (which determines what is done with the induction assumption), and then a term which is the variable to induct over, and will then facilitate an inductive proof of the current goal using that theorem.

SML

```
fun WF_INDUCTION_T2 (thm : THM) (ttac : THM -> TACTIC) : TERM -> TACTIC =
 let fun bad_thm thm = thm_fail "WF_INDUCTION_T2" 29021 [thm];
    val (wf, r) = (dest_app (concl thm))
        handle Fail _ => bad_thm thm;
    val sthm = ∀_elim r wf_lemma
        handle Fail _ => bad_thm thm;
    val ithm = ⇒_elim sthm thm
        handle Fail _ => bad_thm thm;
 in GEN_INDUCTION_T ithm ttac
 end;

fun WFCV_INDUCTION_T (thm : THM) (ttac : THM -> TACTIC) : TERM -> TACTIC =
 let fun bad_thm thm = thm_fail "WFCV_INDUCTION_T" 29021 [thm];
    val (wf, r) = (dest_app (concl thm))
        handle Fail _ => bad_thm thm;
    val sthm = ∀_elim r tcwf_lemma2
        handle Fail _ => bad_thm thm;
    val ithm = ⇒_elim sthm thm
        handle Fail _ => bad_thm thm;
 in GEN_INDUCTION_T ithm ttac
 end;
```

And now we make a tactic out of that (basically by saying "strip the induction hypothesis into the assumptions").

SML

```
fun wf_induction_tac (thm : THM) : TERM -> TACTIC = (
    let val ttac = (WF_INDUCTION_T2 thm strip_asm_tac)
            handle ex => reraise ex "wf_induction_tac";
    in
    fn tm =>
    let val tac = (ttac tm) handle ex => reraise ex "wf_induction_tac";
    in fn gl => ((tac gl) handle ex => reraise ex "wf_induction_tac")
    end
```

```
|    end
|);
|
|fun wfcv_induction_tac (thm : THM) : TERM -> TACTIC = (
|    let val ttac = (WFCV_INDUCTION_T thm strip_asm_tac)
|            handle ex => reraise ex "wfcv_induction_tac";
|    in
|    fn tm =>
|    let val tac = (ttac tm) handle ex => reraise ex "wfcv_induction_tac";
|    in fn gl => ((tac gl) handle ex => reraise ex "wfcv_induction_tac")
|    end
|    end
|);
```

3.2 Well-foundedness and Induction

The following proof shows how the above induction tactic can be used. The theorem can be paraphrased loosely along the lines that there are no bottomless descending chains in a well-founded relation. We think of a bottomless descending chain as a non-empty set (represented by a property called "p") every element of which is preceded by an element under the transitive closure of r.

Now a shorter formulation of bottomless pits.

Next we prove the converse, that the lack of bottomless pits entails well-foundedness.

Now with second order foundation.

Try a weaker hypothesis.

3.3 Bottomless Pits and Minimal Elements

The following theorem states something like that if there are no unending downward chains then every "set" has a minimal element.

A second order version with the weaker bottomless pits can be formulated as follows:

It follows that all non-empty collections of predecessors under a well-founded relation have minimal elements.

But the converse does not hold.

3.4 Restrictions of Well-Founded Relations

In this section we show that a restriction of a well-founded relation is well-founded.

3.5 Well Founded Recursion

I have already proved a recursion theorem fairly closely following the formulation and proof devised by Tobias Nipkow for Isabelle-HOL. There are two reasons for my wanting a different version of this result. The Nipkow derived version works with relations rather than functions, and in my version the relations are ProofPower sets of pairs (I think in the original they were probably properties of pairs). This is probably all easily modded into one which works directly with functions but I though it should be possible also to do a neater proof (the "proof" of the recursion theorem in Kunen is just a couple of lines).

The end result certainly looks nicer, we'll have to see whether it works out well in practice. In particular the fixpoint operator simply takes a functional as an argument and delivers the fixed point as a result. The functional which you give it as an argument, in the simple cases, is just what you get by abstracting the right hand side of a recursive definition on the name of the function (more complicated of course if a pattern matching definition is used). The relation with respect to which the recursion is well-founded need only be mentioned when attempting to prove that this does yield a fixed point.

Another minor improvement is that I do not require the relation to be transitive.

This is the end result:

$\boldsymbol{fixp\_thm1} = \vdash \forall f\ r\bullet\ well\_founded\ r \land f\ respects\ r \Rightarrow \exists g\bullet\ f\ g = g$

The proof is shorter than (my version of) the original, but by less than 20 percent. I'm sure there's lots of scope for improvement. (The isabelle version is much shorter than either.)

3.5.1 Defining the Fixed Point Operator

The main part of this is the proof that functionals which are well-founded with respect to some well-founded relation have fixed points. This done, the operator "fix" is defined, which yields such a fixed point.

SML

```
declare_infix (240, "respects");
```

HOL Constant

$\$\boldsymbol{respects}: (('a \to 'b) \to ('a \to 'b)) \to ('a \to 'a \to BOOL) \to BOOL$

$\forall\ f\ r\ \bullet\ f\ respects\ r \Leftrightarrow \forall g\ h\ x\bullet\ (\forall y\bullet\ (tc\ r)\ y\ x \Rightarrow g\ y = h\ y) \Rightarrow f\ g\ x = f\ h\ x$

HOL Constant

$\boldsymbol{fixed\_below}: (('a \to 'b) \to ('a \to 'b)) \to ('a \to 'a \to BOOL) \to ('a \to 'b) \to 'a \to BOOL$

$\forall f\ r\ g\ x\bullet\ fixed\_below\ f\ r\ g\ x \Leftrightarrow \forall y\bullet\ tc\ r\ y\ x \Rightarrow f\ g\ y = g\ y$

HOL Constant

$\boldsymbol{fixed\_at}: (('a \to 'b) \to ('a \to 'b)) \to ('a \to 'a \to BOOL) \to ('a \to 'b) \to 'a \to BOOL$

$\forall f\ r\ g\ x\bullet\ fixed\_at\ f\ r\ g\ x \Leftrightarrow fixed\_below\ f\ r\ g\ x \land f\ g\ x = g\ x$

HOL Constant

$\boldsymbol{fix}: (('a \to 'b) \to ('a \to 'b)) \to 'a \to 'b$

$\forall f\ r\bullet\ well\_founded\ r \land f\ respects\ r \Rightarrow f\ (fix\ f) = fix\ f$

3.5.2 Partial Functions

Having reformulated the recursion theorem to work with total functions in HOL rather than relations, I later decided that I needed a version which supported the definition of functions over a subset of a type.

The application I am thinking of here is as follows.

A new type is to be defined. The carrier is defined using induction. One of the primitive operators over the new type must be defined inductively. If it weren't primitive it could be defined by well founded induction over the new type, but given that it is primitive it has to be defined over the representation *set*. I'm guessing a function is still required rather than a relation (it probably doesn't make much difference) but either way it will only be nicely behaved over the representation set.

I'm not sure that I have an example of that kind, but here is a better example. If you want to code something into some membership structure, e.g. "godelising" the syntax of a language to prove a Tarski-like definability result, you don't want to make a new type of this inductively defined set, but you will need to define functions by recursion over the set.

There are some other things I want to try out at the same time.

They are:

- recovering the "well-founded" relation from the functor which is required to respect it, i.e. recovering the relation which it respects.
- taking a fixed point which is a function not defined over the whole type, not even defined over some specified subset, but which is defined over the well-founded part of the dependency relation of the defining functor.

The possibility has arisen to take a fixpoint of this kind without consideration of well-foundedness, but taking a closure of the empty set under some functor derived from the defining functor. I haven't yet got a very clear idea on this one, and don't know how closely this material comes to it.

Anyway, for starters I will try to formulate the revised fixedpoint conjecture.

3.5.3 Extracting a Minimal Respected Relation

HOL Constant

$\boldsymbol{ResRelOfFunctor}: (('a \to 'a) \to ('a \to 'a)) \to ('a \to 'a \to BOOL)$

$\forall f\ x\ y \bullet\ ResRelOfFunctor\ f\ x\ y \Leftrightarrow$
$\quad \exists g\ v \bullet\ \neg\ f\ g\ y = f\ (\lambda z \bullet\ if\ z = x\ then\ v\ else\ g\ z)\ y$

3.5.4 The Well-founded Part of a Relation

HOL Constant

$\boldsymbol{WfDomOf}: ('a \to 'a \to BOOL) \to ('a \to BOOL)$

$\forall r \bullet\ WfDomOf\ r =$
$\quad (\lambda\ x \bullet\ \forall p \bullet\ (\forall v \bullet\ (\forall w \bullet\ r\ w\ v \Rightarrow p\ w) \Rightarrow p\ v) \Rightarrow p\ x)$

HOL Constant

$$\boldsymbol{WfPartOf}: ('a \to 'a \to BOOL) \to ('a \to 'a \to BOOL)$$

$$\forall r \bullet\ WfPartOf\ r = (\lambda x\ y \bullet\ r\ x\ y \land WfDomOf\ r\ y)$$

Now we want a conjecture to the effect that any functor has a partial fixed point, i.e. a function whose behaviour over the well-founded part of its respected relation is fixed under the functor.

However, there is no point in doing that without checking that these definitions work.

SML

```
val ResRelOfFunctor_def = get_spec ⌜ResRelOfFunctor⌝;
val WfPartOf_def = get_spec ⌜WfPartOf⌝;
val WfDomOf_def = get_spec ⌜WfDomOf⌝;
```

3.5.5 Respect Theorems

Some theorems which help to prove that functions respect relations.

My first applications of the recursion theorem are in set theory, typically involving recursion which respects membership or its transitive closure.

3.5.6 The Inverse of a Relation

The following function takes a relation and a function and returns a function which maps each element in the domain of the relation to the relation which holds between a predecessor of that element and its value under the function. i.e. it maps the function over the predecessors of the element and returns the result as a relation. It may therefore be used to rephrase primitive recursive definitions, and so the result which follows may be used to establish the existence of functions defined by primitive recursion.

HOL Constant

$$\boldsymbol{relmap} : ('a \to 'a \to BOOL) \to ('a \to 'b) \to ('a \to ('a \to 'b \to BOOL))$$

$$\forall r\ f \bullet\ relmap\ r\ f = \lambda x\ y\ z \bullet\ r\ y\ x \land z = f\ y$$

4 Well-Founded Relations (II)

This is a transcription of the treatment of well-foundedness on which "galactic" set theory was based (from rbjpub/pp/x002.xml).

One of the principle well-founded relations of interest in this application is $\in_g$, which has type

$$⌜:GS \to GS \to BOOL⌝$$

so I would like a version of "well-founded" which has type:

$$⌜:('a \to 'a \to BOOL) \to BOOL⌝$$

The new theory *wf_relp* is first created.

SML

```
open_theory "hol";
force_new_theory "wf_relp";
new_parent "tc";
```

4.1 Well-Founded Relations

Definition of well-founded and transitive-well-founded and proof that the transitive closure of a well-founded relation is transitive-well-founded.

HOL Constant

$$\textbf{well\_founded}: ('a \to 'a \to BOOL) \to BOOL$$

$$\forall r \bullet\ well\_founded\ r \Leftrightarrow \forall\ s \bullet (\forall\ x \bullet (\forall\ y \bullet r\ y\ x \Rightarrow s\ y) \Rightarrow s\ x) \Rightarrow \forall\ x \bullet s\ x$$

HOL Constant

$$\textbf{twfp}: ('a \to 'a \to BOOL) \to BOOL$$

$$\forall r \bullet\ twfp\ r \Leftrightarrow well\_founded\ r \land trans\ r$$

The first thing I need to prove here is that the transitive closure of a well-founded relation is also well-founded. This provides a form of induction with a stronger induction hypothesis. Naturally we would expect this to be proven inductively and the question is therefore what property to use in the inductive proof, the observation that the transitive closure of a relation is well-founded is not explicitly the ascription of a property to the field of the relation. The obvious method is to relativise the required result to the transitive closure of a set, giving a property of sets, and then to prove that this property is hereditary if the relation is well-founded.

$\textbf{tcwf\_lemma1} =$

$\quad \vdash \forall s\ r \bullet\ well\_founded\ r$

$\qquad \Rightarrow \forall x \bullet (\forall y \bullet\ tc\ r\ y\ x \Rightarrow (\forall z \bullet\ tc\ r\ z\ y \Rightarrow s\ z) \Rightarrow s\ y)$

$\qquad \Rightarrow (\forall y \bullet\ tc\ r\ y\ x \Rightarrow s\ y)$

$\textbf{wf\_lemma2} =$

$\quad \vdash \forall\ r \bullet\ well\_founded\ r \Rightarrow (\forall\ s \bullet (\forall\ t \bullet (\forall\ u \bullet\ r\ u\ t \Rightarrow s\ u) \Rightarrow s\ t) \Rightarrow (\forall\ e \bullet\ s\ e))$

$\textbf{tcwf\_lemma2} =$

$\quad \vdash \forall r \bullet\ well\_founded\ r$

$\qquad \Rightarrow \forall s \bullet (\forall t \bullet (\forall u \bullet\ tc\ r\ u\ t \Rightarrow s\ u) \Rightarrow s\ t)$

$\qquad \Rightarrow (\forall e \bullet\ s\ e)$

$\textbf{wf\_tc\_wf\_thm} =$

$\quad \vdash\ \forall r \bullet\ well\_founded\ (tc\ r) \Rightarrow well\_founded\ r$

Now we prove that if the transitive closure of a relation is well-founded then so must be the relation.

$\textbf{tc\_wf\_wf\_thm} =$

$\quad \vdash \forall r \bullet\ well\_founded\ (tc\ r) \Rightarrow well\_founded\ r$

$\textbf{tc\_wf\_twf\_thm} =$

$\quad \vdash \forall r \bullet\ well\_founded\ r \Rightarrow twfp\ (tc\ r)$

We here define a general tactic for performing induction using some well-founded relation.

The following function (I think these things are called "THM-TACTICAL"s) must be given a theorem which asserts that some relation is well-founded, and then a THM-TACTIC (which determines what is done with the induction assumption), and then a term which is the variable to induct over, and will then facilitate an inductive proof of the current goal using that theorem.

SML

```
fun WF_INDUCTION_T (thm : THM) (ttac : THM -> TACTIC) : TERM -> TACTIC =
(   let fun bad_thm thm = thm_fail "WF_INDUCTION_T" 29021 [thm];
        val (wf, r) = (dest_app (concl thm))
            handle Fail _ => bad_thm thm;
        val sthm = ∀_elim r wf_lemma2
            handle Fail _ => bad_thm thm;
        val ithm = ⇒_elim sthm thm
            handle Fail _ => bad_thm thm;
    in GEN_INDUCTION_T ithm ttac
    end
);

fun WFCV_INDUCTION_T (thm : THM) (ttac : THM -> TACTIC) : TERM -> TACTIC =
(   let fun bad_thm thm = thm_fail "WFCV_INDUCTION_T" 29021 [thm];
        val (wf, r) = (dest_app (concl thm))
            handle Fail _ => bad_thm thm;
        val sthm = ∀_elim r tcwf_lemma2
            handle Fail _ => bad_thm thm;
        val ithm = ⇒_elim sthm thm
            handle Fail _ => bad_thm thm;
    in GEN_INDUCTION_T ithm ttac
    end
);
```

And now we make a tactic out of that (basically by saying "strip the induction hypothesis into the assumptions").

SML

```
fun wf_induction_tac (thm : THM) : TERM -> TACTIC = (
    let val ttac = (WF_INDUCTION_T thm strip_asm_tac)
            handle ex => reraise ex "wf_induction_tac";
    in
    fn tm =>
    let val tac = (ttac tm) handle ex => reraise ex "wf_induction_tac";
    in fn gl => ((tac gl) handle ex => reraise ex "wf_induction_tac")
    end
    end
);

fun wfcv_induction_tac (thm : THM) : TERM -> TACTIC = (
```

```
let val ttac = (WFCV_INDUCTION_T thm strip_asm_tac)
        handle ex => reraise ex "wfcv_induction_tac";
in
fn tm =>
let val tac = (ttac tm) handle ex => reraise ex "wfcv_induction_tac";
in fn gl => ((tac gl) handle ex => reraise ex "wfcv_induction_tac")
end
end
);
```

4.1.1 Well-foundedness and Induction

The following proof shows how the above induction tactic can be used. The theorem can be paraphrased loosely along the lines that there are no bottomless descending chains in a well-founded relation. We think of a bottomless descending chain as a non-empty set (represented by a property called "p") every element of which is preceded by an element under the transitive closure of r.

wf_nochain_thm =

$\vdash \forall r\bullet\ well\_founded\ r$

$\quad \Rightarrow \forall x\bullet\ \neg\exists p\ v\bullet\ p\ v \land \forall y\bullet\ p\ y \Rightarrow tc\ r\ y\ x \land \exists z\bullet\ p\ z \land r\ z\ y$

Now a shorter formulation of bottomless pits.

wf_wf_thm =

$\vdash \forall r\bullet\ well\_founded\ r$

$\quad \Rightarrow \neg\exists p\ v\bullet\ p\ v \land \forall y\bullet\ p\ y \Rightarrow \exists z\bullet\ p\ z \land r\ z\ y$

nochain_wf_thm =

$\vdash \forall r\bullet\ (\forall x\bullet\ \neg\exists p\ v\bullet\ p\ v \land \forall y\bullet\ p\ y \Rightarrow tc\ r\ y\ x \land \exists z\bullet\ p\ z \land r\ z\ y)$

$\quad \Rightarrow well\_founded\ r$

wf_⇔_nochain_thm =

$\vdash \forall r\bullet\ well\_founded\ r$

$\quad \Leftrightarrow (\forall\ x\bullet\ \neg\ (\exists\ p\ v\bullet\ p\ v \land (\forall\ y\bullet\ p\ y \Rightarrow tc\ r\ y\ x \land (\exists\ z\bullet\ p\ z \land r\ z\ y))))$

Now with second order foundation.

wf_induct_thm =

$\vdash (\neg\exists p\ v\bullet\ p\ v \land \forall y\bullet\ p\ y \Rightarrow \exists z\bullet\ p\ z \land r\ z\ y)$

$\quad \Rightarrow well\_founded\ r$

Try a weaker hypothesis.

nochain_wf_thm2 =

$\vdash \forall r\bullet\ (\forall x\bullet\ \neg\exists p\ v\bullet\ p\ v \land \forall y\bullet\ p\ y \Rightarrow \exists z\bullet\ p\ z \land r\ z\ y)$

$\quad \Rightarrow well\_founded\ r$

4.1.2 Bottomless Pits and Minimal Elements

The following theorem states something like that if there are no unending downward chains then every "set" has a minimal element.

$\boldsymbol{nochain\_min\_thm} =$
$\vdash \forall r\bullet (\forall x\bullet\ \neg\exists p\ v\bullet\ p\ v \wedge \forall y\bullet\ p\ y \Rightarrow tc\ r\ y\ x \wedge \exists z\bullet\ p\ z \wedge r\ z\ y)$
$\quad \Rightarrow \forall x\bullet\ (\exists y\bullet\ r\ y\ x) \Rightarrow \exists z\bullet\ r\ z\ x \wedge \neg\exists v\bullet\ r\ v\ z \wedge r\ v\ x$

A second order version with the weaker bottomless pits can be formulated as follows:

$\boldsymbol{nochain\_min\_thm2} =$
$\vdash \forall r\bullet (\forall x\bullet\ \neg\exists p\ v\bullet\ p\ v \wedge \forall y\bullet\ p\ y \Rightarrow \exists z\bullet\ p\ z \wedge r\ z\ y)$
$\quad \Rightarrow \forall p\bullet\ (\exists y\bullet\ p\ y) \Rightarrow \exists z\bullet\ p\ z \wedge \neg\exists v\bullet\ r\ v\ z \wedge p\ v$

It follows that all non-empty collections of predecessors under a well-founded relation have minimal elements.

$\boldsymbol{wf\_min\_thm} =$
$\vdash \forall r\bullet\ well\_founded\ r$
$\quad \Rightarrow \forall x\bullet\ (\exists y\bullet\ r\ y\ x) \Rightarrow \exists z\bullet\ r\ z\ x \wedge \neg\exists v\bullet\ r\ v\ z \wedge r\ v\ x$

But the converse does not hold.

$\boldsymbol{minr\_not\_wf\_thm} =$
$\vdash \exists r{:}\ BOOL\rightarrow BOOL\rightarrow BOOL\bullet$
$\quad (\forall x\bullet\ (\exists y\bullet\ r\ y\ x) \Rightarrow \exists z\bullet\ r\ z\ x \wedge \neg\exists v\bullet\ r\ v\ z \wedge r\ v\ x)$
$\quad \wedge \neg\ well\_founded\ r$

4.2 Some Consequences of Well Foundedness

$\boldsymbol{wf\_not\_refl\_thm} =$
$\vdash \forall\ r\bullet\ well\_founded\ r \Rightarrow \neg\ (\exists\ x\bullet\ r\ x\ x)$

4.3 Ways of Constructing Well Founded Relations

In this section we show that a restriction of a well-founded relation is well-founded.

$\boldsymbol{wf\_restrict\_wf\_thm} =$
$\vdash \forall r\bullet\ well\_founded\ r \Rightarrow \forall r2\bullet\ well\_founded\ (\lambda x\ y\bullet\ r2\ x\ y \wedge r\ x\ y)$

$\boldsymbol{wf\_image\_wf\_thm} =$
$\vdash \forall r\bullet\ well\_founded\ r \Rightarrow \forall f\bullet\ well\_founded\ (\lambda x\ y\bullet\ r\ (f\ x)\ (f\ y))$

4.4 Proof Context

In this section I will create a decent proof context for recursive definitions, eventually.

4.4.1 Proof Context

SML

```
(∗ commit_pc "'wf_relp"; ∗)
```

4.5 Recursion Theorem

SML

```
open_theory "wf_relp";
force_new_theory "wf_recp";
```

4.5.1 Defining the Fixed Point Operator

The main part of this is the proof that functionals which are well-founded with respect to some well-founded relation have fixed points. This done, the operator "fix" is defined, which yields such a fixed point.

SML

```
declare_infix (240, "respects");
```

HOL Constant

$\$\boldsymbol{respects}: (('a \to 'b) \to ('a \to 'b)) \to ('a \to 'a \to BOOL) \to BOOL$

$\forall\ f\ r \bullet f\ respects\ r \Leftrightarrow \forall g\ h\ x\bullet\ (\forall y\bullet\ (tc\ r)\ y\ x \Rightarrow g\ y = h\ y) \Rightarrow f\ g\ x = f\ h\ x$

HOL Constant

$\boldsymbol{fixed\_below}: (('a \to 'b) \to ('a \to 'b)) \to ('a \to 'a \to BOOL) \to ('a \to 'b) \to 'a \to BOOL$

$\forall f\ r\ g\ x\bullet\ fixed\_below\ f\ r\ g\ x \Leftrightarrow \forall y\bullet\ tc\ r\ y\ x \Rightarrow f\ g\ y = g\ y$

HOL Constant

$\boldsymbol{fixed\_at}: (('a \to 'b) \to ('a \to 'b)) \to ('a \to 'a \to BOOL) \to ('a \to 'b) \to 'a \to BOOL$

$\forall f\ r\ g\ x\bullet\ fixed\_at\ f\ r\ g\ x \Leftrightarrow fixed\_below\ f\ r\ g\ x \land f\ g\ x = g\ x$

HOL Constant

$\boldsymbol{fix}: (('a \to 'b) \to ('a \to 'b)) \to 'a \to 'b$

$\forall f\ r\bullet\ well\_founded\ r \land f\ respects\ r \Rightarrow f\ (fix\ f) = fix\ f$

4.5.2 The Inverse of a Relation

The following function takes a relation and a function and returns a function which maps each element in the domain of the relation to the relation which holds between a predecessor of that element and its value under the function. i.e. it maps the function over the predecessors of the element and returns the result as a relation. It may therefore be used to rephrase primitive recursive definitions, and so the result which follows may be used to establish the existence of functions defined by primitive recursion.

HOL Constant

$\boldsymbol{relmap} : ('a \to 'a \to BOOL) \to ('a \to 'b) \to ('a \to ('a \to 'b \to BOOL))$

$\forall r\ f \bullet\ relmap\ r\ f = \lambda x\ y\ z \bullet\ r\ y\ x \land z = f\ y$

4.5.3 Proof Context

SML

```
(* commit_pc "'wf_relp"; *)
```

5 The Theory tc

5.1 Parents

hol

5.2 Children

wf_relp *wfrel*

5.3 Constants

trans	$('a \to 'a \to BOOL) \to BOOL$
tc	$('a \to 'a \to BOOL) \to 'a \to 'a \to BOOL$
rtc	$('a \to 'a \to BOOL) \to 'a \to 'a \to BOOL$

5.4 Definitions

trans $\vdash \forall\ r\bullet\ trans\ r \Leftrightarrow (\forall\ s\ t\ u\bullet\ r\ s\ t \wedge r\ t\ u \Rightarrow r\ s\ u)$

tc
$$\vdash \forall\ r \bullet\ tc\ r = (\lambda\ s\ t \bullet \forall\ tr \bullet\ trans\ tr \wedge (\forall\ v\ u\bullet\ r\ v\ u \Rightarrow tr\ v\ u) \Rightarrow tr\ s\ t)$$

rtc $\vdash \forall\ r\bullet\ rtc\ r = (\lambda\ s\ t\bullet\ s = t \vee tc\ r\ s\ t)$

5.5 Theorems

tran_tc_thm $\vdash \forall\ r\bullet\ trans\ (tc\ r)$

tran_tc_thm2 $\vdash \forall\ r\ x\ y\ z\bullet\ tc\ r\ x\ y \wedge tc\ r\ y\ z \Rightarrow tc\ r\ x\ z$

tc_incr_thm $\vdash \forall\ r\ x\ y\bullet\ r\ x\ y \Rightarrow tc\ r\ x\ y$

tc_decomp_thm
$$\vdash \forall\ r\ x\ y\bullet\ tc\ r\ x\ y \wedge \neg\ r\ x\ y \Rightarrow (\exists\ z\bullet\ tc\ r\ x\ z \wedge r\ z\ y)$$

tc_decomp_thm2
$$\vdash \forall\ r\ x\ y \bullet\ tc\ r\ x\ y \Rightarrow (\exists\ f\ n \bullet\ x = f\ 0 \wedge y = f\ n \wedge (\forall\ m\bullet\ m < n \Rightarrow r\ (f\ m)\ (f\ (m + 1))))$$

tc_decomp_thm3
$$\vdash \forall\ r\ x\ y \bullet\ tc\ r\ x\ y \Rightarrow (\exists\ f\ n \bullet\ x = f\ 0 \wedge y = f\ (n + 1) \wedge (\forall\ m\bullet\ m \le n \Rightarrow r\ (f\ m)\ (f\ (m + 1))))$$

tc_decomp_thm4
$$\vdash \forall\ r\ x\ y \bullet\ (\exists\ f\ n$$

$\bullet\ x = f\ 0$
$\quad \land\ y = f\ (n + 1)$
$\quad \land\ (\forall\ m\bullet\ m \le n \Rightarrow r\ (f\ m)\ (f\ (m + 1))))$
$\Rightarrow tc\ r\ x\ y$

tc_⇔_thm $\vdash \forall\ r\ x\ y$
$\bullet\ tc\ r\ x\ y$
$\quad \Leftrightarrow (\exists\ f\ n$
$\quad \bullet\ x = f\ 0$
$\quad\quad \land\ y = f\ (n + 1)$
$\quad\quad \land\ (\forall\ m\bullet\ m \le n \Rightarrow r\ (f\ m)\ (f\ (m + 1))))$

***tc_decomp_thm*5**
$\vdash \forall\ r\ x\ y\bullet\ tc\ r\ x\ y \land \neg\ r\ x\ y \Rightarrow (\exists\ z\bullet\ r\ x\ z \land tc\ r\ z\ y)$

tc_mono_thm $\vdash \forall\ r1\ r2$
$\bullet\ (\forall\ x\ y\bullet\ r1\ x\ y \Rightarrow r2\ x\ y)$
$\quad \Rightarrow (\forall\ x\ y\bullet\ tc\ r1\ x\ y \Rightarrow tc\ r2\ x\ y)$

tc_p_thm $\vdash \forall\ r\ p\bullet\ (\forall\ x\ y\bullet\ r\ x\ y \Rightarrow p\ x) \Rightarrow (\forall\ x\ y\bullet\ tc\ r\ x\ y \Rightarrow p\ x)$

tc_induced_thm
$\vdash \forall\ r\ u\ x$
$\bullet\ tc\ (\lambda\ x\ y\bullet\ r\ (f\ x)\ (f\ y))\ u\ x \Rightarrow tc\ r\ (f\ u)\ (f\ x)$

tran_tc_id_thm
$\vdash \forall\ r\bullet\ trans\ r \Rightarrow tc\ r = r$

***tc_decomp_thm*6**
$\vdash \forall\ r\ x\ y\bullet\ tc\ r\ x\ y \Rightarrow r\ x\ y \lor (\exists\ z\bullet\ tc\ r\ x\ z \land r\ z\ y)$

***tc_decomp_thm*7**
$\vdash \forall\ r\ x\ y\bullet\ tc\ r\ x\ y \Rightarrow r\ x\ y \lor (\exists\ z\bullet\ r\ x\ z \land tc\ r\ z\ y)$

***tc_ind*0** $\vdash \forall\ R\ r$
$\bullet\ (\forall\ x\ y\bullet\ r\ x\ y \Rightarrow R\ x\ y)$
$\quad \land\ (\forall\ x\ y\ z\bullet\ R\ x\ y \land R\ y\ z \Rightarrow R\ x\ z)$
$\quad \Rightarrow (\forall\ x\ y\bullet\ tc\ r\ x\ y \Rightarrow R\ x\ y)$

***tc_ind*1** $\vdash \forall\ R\ r$
$\bullet\ (\forall\ x\ y\bullet\ r\ x\ y \Rightarrow R\ x\ y)$
$\quad \land\ (\forall\ x\ y\ z\bullet\ R\ x\ y \land r\ y\ z \Rightarrow R\ x\ z)$
$\quad \Rightarrow (\forall\ x\ y\bullet\ tc\ r\ x\ y \Rightarrow R\ x\ y)$

***tc_ind*2** $\vdash \forall\ R\ r$
$\bullet\ (\forall\ x\ y\bullet\ r\ x\ y \Rightarrow R\ x\ y)$
$\quad \land\ (\forall\ x\ y\ z\bullet\ r\ x\ y \land R\ y\ z \Rightarrow R\ x\ z)$
$\quad \Rightarrow (\forall\ x\ y\bullet\ tc\ r\ x\ y \Rightarrow R\ x\ y)$

tran_rtc_thm $\vdash \forall\ r\bullet\ trans\ (rtc\ r)$

***tran_rtc_thm*2**
$\vdash \forall\ r\ s\ t\ u\bullet\ rtc\ r\ s\ t \land rtc\ r\ t\ u \Rightarrow rtc\ r\ s\ u$

rtc_incr_thm $\vdash \forall\ r\ x\ y\bullet\ r\ x\ y \lor x = y \Rightarrow rtc\ r\ x\ y$

rtc_decomp_thm
$\vdash \forall\ R\ a0\ a1$
$\bullet\ rtc\ R\ a0\ a1 \Rightarrow a1 = a0 \lor (\exists\ y\bullet\ R\ a0\ y \land rtc\ R\ y\ a1)$

***rtc_decomp_thm*2**
$\vdash \forall\ R\ a0\ a1$
$\bullet\ rtc\ R\ a0\ a1$
$\quad \Rightarrow a1 = a0 \lor R\ a0\ a1 \lor (\exists\ y\bullet\ R\ a0\ y \land tc\ R\ y\ a1)$

***rtc_decomp_thm*3**
$\vdash \forall\ R\ a0\ a1$
$\bullet\ rtc\ R\ a0\ a1$

$\Rightarrow a1 = a0 \lor R\ a0\ a1 \lor (\exists\ y \bullet\ tc\ R\ a0\ y \land R\ y\ a1)$

rtc_mono_thm $\vdash \forall\ r1\ r2$
$\bullet\ (\forall\ x\ y \bullet\ r1\ x\ y \Rightarrow r2\ x\ y)$
$\Rightarrow (\forall\ x\ y \bullet\ rtc\ r1\ x\ y \Rightarrow rtc\ r2\ x\ y)$

rtc_ind0 $\vdash \forall\ R\ r$
$\bullet\ (\forall\ x\ y \bullet\ r\ x\ y \Rightarrow R\ x\ y)$
$\land (\forall\ x \bullet\ R\ x\ x)$
$\land (\forall\ x\ y\ z \bullet\ R\ x\ y \land R\ y\ z \Rightarrow R\ x\ z)$
$\Rightarrow (\forall\ x\ y \bullet\ rtc\ r\ x\ y \Rightarrow R\ x\ y)$

rtc_ind1 $\vdash \forall\ R\ r$
$\bullet\ (\forall\ x \bullet\ R\ x\ x) \land (\forall\ x\ y\ z \bullet\ R\ x\ y \land r\ y\ z \Rightarrow R\ x\ z)$
$\Rightarrow (\forall\ x\ y \bullet\ rtc\ r\ x\ y \Rightarrow R\ x\ y)$

rtc_ind $\vdash \forall\ r\ R$
$\bullet\ (\forall\ x \bullet\ R\ x\ x) \land (\forall\ x\ y\ z \bullet\ r\ x\ y \land R\ y\ z \Rightarrow R\ x\ z)$
$\Rightarrow (\forall\ x\ y \bullet\ rtc\ r\ x\ y \Rightarrow R\ x\ y)$

rtc_rules $\vdash \forall\ r$
$\bullet\ (\forall\ x \bullet\ rtc\ r\ x\ x)$
$\land (\forall\ x\ y\ z \bullet\ r\ x\ y \land rtc\ r\ y\ z \Rightarrow rtc\ r\ x\ z)$

rtc_strongind
$\vdash \forall\ r\ rtc'$
$\bullet\ (\forall\ x \bullet\ rtc'\ x\ x)$
$\land (\forall\ x\ y\ z$
$\bullet\ r\ x\ y \land rtc\ r\ y\ z \land rtc'\ y\ z \Rightarrow rtc'\ x\ z)$
$\Rightarrow (\forall\ a0\ a1 \bullet\ rtc\ r\ a0\ a1 \Rightarrow rtc'\ a0\ a1)$

rtc_cases $\vdash \forall\ R\ a0\ a1$
$\bullet\ rtc\ R\ a0\ a1 \Leftrightarrow a1 = a0 \lor (\exists\ y \bullet\ R\ a0\ y \land rtc\ R\ y\ a1)$

6 The Theory wfrel

6.1 Parents

tc

6.2 Constants

well_founded	$('a \to 'a \to BOOL) \to BOOL$
twfp	$('a \to 'a \to BOOL) \to BOOL$
$***respects***	$(('a \to 'b) \to 'a \to 'b) \to ('a \to 'a \to BOOL) \to BOOL$
fixed_below	$(('a \to 'b) \to 'a \to 'b) \to ('a \to 'a \to BOOL) \to ('a \to 'b) \to 'a \to BOOL$
fixed_at	$(('a \to 'b) \to 'a \to 'b) \to ('a \to 'a \to BOOL) \to ('a \to 'b) \to 'a \to BOOL$
fix	$(('a \to 'b) \to 'a \to 'b) \to 'a \to 'b$
ResRelOfFunctor	$(('a \to 'a) \to 'a \to 'a) \to 'a \to 'a \to BOOL$
WfDomOf	$('a \to 'a \to BOOL) \to 'a \to BOOL$
WfPartOf	$('a \to 'a \to BOOL) \to 'a \to 'a \to BOOL$
relmap	$('a \to 'a \to BOOL) \to ('a \to 'b) \to 'a \to 'a \to 'b \to BOOL$

6.3 Fixity

Right Infix 240:

respects

6.4 Definitions

well_founded $\vdash \forall\ r \bullet\ well\_founded\ r \Leftrightarrow (\forall\ s \bullet (\forall\ x\bullet\ (\forall\ y\bullet\ r\ y\ x \Rightarrow s\ y) \Rightarrow s\ x) \Rightarrow (\forall\ x\bullet\ s\ x))$

twfp $\vdash \forall\ r\bullet\ twfp\ r \Leftrightarrow well\_founded\ r \land trans\ r$

respects $\vdash \forall\ f\ r \bullet\ f\ respects\ r \Leftrightarrow (\forall\ g\ h\ x \bullet (\forall\ y\bullet\ tc\ r\ y\ x \Rightarrow g\ y = h\ y) \Rightarrow f\ g\ x = f\ h\ x)$

fixed_below $\vdash \forall\ f\ r\ g\ x \bullet\ fixed\_below\ f\ r\ g\ x \Leftrightarrow (\forall\ y\bullet\ tc\ r\ y\ x \Rightarrow f\ g\ y = g\ y)$

fixed_at $\vdash \forall\ f\ r\ g\ x \bullet\ fixed\_at\ f\ r\ g\ x \Leftrightarrow fixed\_below\ f\ r\ g\ x \land f\ g\ x = g\ x$

fix $\vdash ConstSpec$

$$(\lambda\ fix' \bullet \forall\ f\ r \bullet well\_founded\ r \land f\ respects\ r \Rightarrow f\ (fix'\ f) = fix'\ f)\ fix$$

ResRelOfFunctor

$$\vdash \forall\ f\ x\ y \bullet ResRelOfFunctor\ f\ x\ y \Leftrightarrow (\exists\ g\ v \bullet \neg\ f\ g\ y = f\ (\lambda\ z \bullet\ if\ z = x\ then\ v\ else\ g\ z)\ y)$$

WfDomOf

$$\vdash \forall\ r \bullet WfDomOf\ r = (\lambda\ x \bullet \forall\ p \bullet (\forall\ v \bullet (\forall\ w \bullet r\ w\ v \Rightarrow p\ w) \Rightarrow p\ v) \Rightarrow p\ x)$$

WfPartOf

$$\vdash \forall\ r \bullet WfPartOf\ r = (\lambda\ x\ y \bullet r\ x\ y \land WfDomOf\ r\ y)$$

relmap

$$\vdash \forall\ r\ f \bullet relmap\ r\ f = (\lambda\ x\ y\ z \bullet r\ y\ x \land z = f\ y)$$

6.5 Theorems

tcwf_lemma1

$$\vdash \forall\ s\ r \bullet well\_founded\ r \Rightarrow (\forall\ x \bullet (\forall\ y \bullet tc\ r\ y\ x \Rightarrow (\forall\ z \bullet tc\ r\ z\ y \Rightarrow s\ z) \Rightarrow s\ y) \Rightarrow (\forall\ y \bullet tc\ r\ y\ x \Rightarrow s\ y))$$

wf_lemma

$$\vdash \forall\ r \bullet well\_founded\ r \Rightarrow (\forall\ s \bullet (\forall\ t \bullet (\forall\ u \bullet r\ u\ t \Rightarrow s\ u) \Rightarrow s\ t) \Rightarrow (\forall\ e \bullet s\ e))$$

tcwf_lemma2

$$\vdash \forall\ r \bullet well\_founded\ r \Rightarrow (\forall\ s \bullet (\forall\ t \bullet (\forall\ u \bullet tc\ r\ u\ t \Rightarrow s\ u) \Rightarrow s\ t) \Rightarrow (\forall\ e \bullet s\ e))$$

wf_tc_wf_thm

$$\vdash \forall\ r \bullet well\_founded\ r \Rightarrow well\_founded\ (tc\ r)$$

tc_wf_wf_thm

$$\vdash \forall\ r \bullet well\_founded\ (tc\ r) \Rightarrow well\_founded\ r$$

tc_wf_twf_thm

$$\vdash \forall\ r \bullet well\_founded\ r \Rightarrow twfp\ (tc\ r)$$

wf_nochain_thm

$$\vdash \forall\ r \bullet well\_founded\ r \Rightarrow (\forall\ x \bullet \neg\ (\exists\ p\ v \bullet p\ v \land (\forall\ y \bullet p\ y \Rightarrow tc\ r\ y\ x \land (\exists\ z \bullet p\ z \land r\ z\ y))))$$

wf_wf_thm

$$\vdash \forall\ r \bullet well\_founded\ r \Rightarrow \neg\ (\exists\ p\ v \bullet p\ v \land (\forall\ y \bullet p\ y \Rightarrow (\exists\ z \bullet p\ z \land r\ z\ y)))$$

nochain_wf_thm

$$\vdash \forall\ r$$

$\bullet\ (\forall\ x$
$\quad \bullet\ \neg\ (\exists\ p\ v$
$\qquad \bullet\ p\ v$
$\qquad\quad \land\ (\forall\ y$
$\qquad\quad \bullet\ p\ y \Rightarrow tc\ r\ y\ x \land (\exists\ z \bullet\ p\ z \land r\ z\ y))))$
$\quad \Rightarrow well\_founded\ r$

wf_induct_thm

$\vdash \neg\ (\exists\ p\ v \bullet\ p\ v \land (\forall\ y \bullet\ p\ y \Rightarrow (\exists\ z \bullet\ p\ z \land r\ z\ y)))$
$\quad \Rightarrow well\_founded\ r$

nochain_wf_thm2

$\vdash \forall\ r$
$\bullet\ (\forall\ x$
$\quad \bullet\ \neg\ (\exists\ p\ v$
$\qquad \bullet\ p\ v \land (\forall\ y \bullet\ p\ y \Rightarrow (\exists\ z \bullet\ p\ z \land r\ z\ y))))$
$\quad \Rightarrow well\_founded\ r$

nochain_min_thm

$\vdash \forall\ r$
$\bullet\ (\forall\ x$
$\quad \bullet\ \neg\ (\exists\ p\ v$
$\qquad \bullet\ p\ v$
$\qquad\quad \land\ (\forall\ y$
$\qquad\quad \bullet\ p\ y \Rightarrow tc\ r\ y\ x \land (\exists\ z \bullet\ p\ z \land r\ z\ y))))$
$\quad \Rightarrow (\forall\ x$
$\quad \bullet\ (\exists\ y \bullet\ r\ y\ x)$
$\qquad \Rightarrow (\exists\ z \bullet\ r\ z\ x \land \neg\ (\exists\ v \bullet\ r\ v\ z \land r\ v\ x)))$

nochain_min_thm2

$\vdash \forall\ r$
$\bullet\ (\forall\ x$
$\quad \bullet\ \neg\ (\exists\ p\ v$
$\qquad \bullet\ p\ v \land (\forall\ y \bullet\ p\ y \Rightarrow (\exists\ z \bullet\ p\ z \land r\ z\ y))))$
$\quad \Rightarrow (\forall\ p$
$\quad \bullet\ (\exists\ y \bullet\ p\ y) \Rightarrow (\exists\ z \bullet\ p\ z \land \neg\ (\exists\ v \bullet\ r\ v\ z \land p\ v)))$

wf_min_thm $\vdash \forall\ r$
$\bullet\ well\_founded\ r$
$\quad \Rightarrow (\forall\ x$
$\quad \bullet\ (\exists\ y \bullet\ r\ y\ x)$
$\qquad \Rightarrow (\exists\ z \bullet\ r\ z\ x \land \neg\ (\exists\ v \bullet\ r\ v\ z \land r\ v\ x)))$

minr_not_wf_thm

$\vdash \exists\ r$
$\bullet\ (\forall\ x$
$\quad \bullet\ (\exists\ y \bullet\ r\ y\ x)$
$\qquad \Rightarrow (\exists\ z \bullet\ r\ z\ x \land \neg\ (\exists\ v \bullet\ r\ v\ z \land r\ v\ x)))$
$\quad \land\ \neg\ well\_founded\ r$

wf_restrict_wf_thm

$\vdash \forall\ r$
$\bullet\ well\_founded\ r$
$\quad \Rightarrow (\forall\ r2 \bullet\ well\_founded\ (\lambda\ x\ y \bullet\ r2\ x\ y \land r\ x\ y))$

fixed_below_lemma1

$\vdash \forall\ f\ r$
$\bullet\ well\_founded\ r \land f\ respects\ r$
$\quad \Rightarrow (\forall\ x\ g\ y$

$\bullet$ *fixed_below* $f\ r\ g\ x \land$ *tc* $r\ y\ x$
$\Rightarrow$ *fixed_below* $f\ r\ g\ y)$

***fixed_at_lemma*1**
$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall x\ g$
$\bullet$ *fixed_below* $f\ r\ g\ x \Rightarrow$ *fixed_at* $f\ r\ (f\ g)\ x)$

***fixed_at_lemma*2**
$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall x\ g$
$\bullet$ *fixed_below* $f\ r\ g\ x$
$\Rightarrow (\forall y \bullet$ *tc* $r\ y\ x \Rightarrow$ *fixed_at* $f\ r\ g\ y))$

***fixed_at_lemma*3**
$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall x\ g$
$\bullet (\forall y \bullet$ *tc* $r\ y\ x \Rightarrow$ *fixed_at* $f\ r\ g\ y)$
$\Rightarrow$ *fixed_below* $f\ r\ g\ x)$

***fixed_below_lemma*2**
$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall x\ g\ h$
$\bullet$ *fixed_below* $f\ r\ g\ x \land$ *fixed_below* $f\ r\ h\ x$
$\Rightarrow (\forall z \bullet$ *tc* $r\ z\ x \Rightarrow h\ z = g\ z))$

***fixed_at_lemma*4**
$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall g\ x$
$\bullet$ *fixed_at* $f\ r\ g\ x$
$\Rightarrow (\forall y \bullet$ *tc* $r\ y\ x \Rightarrow$ *fixed_at* $f\ r\ g\ y))$

***fixed_at_lemma*5**
$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall g\ h\ x$
$\bullet$ *fixed_at* $f\ r\ g\ x \land$ *fixed_at* $f\ r\ h\ x$
$\Rightarrow g\ x = h\ x)$

***fixed_below_lemma*3**
$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall x$
$\bullet (\forall y \bullet$ *tc* $r\ y\ x \Rightarrow (\exists g \bullet$ *fixed_at* $f\ r\ g\ y))$
$\Rightarrow (\exists g \bullet$ *fixed_below* $f\ r\ g\ x))$

***fixed_below_lemma*4**
$\vdash \forall r\ f$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall x \bullet \exists g \bullet$ *fixed_below* $f\ r\ g\ x)$

***fixed_at_lemma*6**
$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall x \bullet \exists g \bullet$ *fixed_at* $f\ r\ g\ x)$

fixed_lemma1 $\vdash \forall\ f\ r$
$\bullet\ well\_founded\ r \land f\ respects\ r$
$\Rightarrow (\forall\ x$
$\bullet\ fixed\_at$
f
r
$(\lambda\ x\bullet\ (\epsilon\ h\bullet\ fixed\_at\ f\ r\ h\ x)\ x)$
$x)$

fixp_thm1 $\vdash \forall\ f\ r\bullet\ well\_founded\ r \land f\ respects\ r \Rightarrow (\exists\ g\bullet\ f\ g = g)$

relmap_respect_thm
$\vdash \forall\ r\ g\bullet\ (\lambda\ f\bullet\ g\ o\ relmap\ r\ f)\ respects\ r$

mono_respects_thm
$\vdash \forall\ f\ r1\ r2$
$\bullet\ f\ respects\ r1 \land (\forall\ x\ y\bullet\ r1\ x\ y \Rightarrow r2\ x\ y)$
$\Rightarrow f\ respects\ r2$

7 The Theory wf_relp

7.1 Parents

tc *hol*

7.2 Children

fixp *gst−ax* *wf_recp*

7.3 Constants

well_founded	$('a \to 'a \to BOOL) \to BOOL$
twfp	$('a \to 'a \to BOOL) \to BOOL$

7.4 Definitions

well_founded $\vdash \forall\ r$
$\bullet\ well\_founded\ r$
$\Leftrightarrow (\forall\ s$
$\bullet\ (\forall\ x \bullet\ (\forall\ y \bullet\ r\ y\ x \Rightarrow s\ y) \Rightarrow s\ x) \Rightarrow (\forall\ x \bullet\ s\ x))$

twfp $\vdash \forall\ r \bullet\ twfp\ r \Leftrightarrow well\_founded\ r \land trans\ r$

7.5 Theorems

tcwf_lemma1 $\vdash \forall\ s\ r$
$\bullet\ well\_founded\ r$
$\Rightarrow (\forall\ x$
$\bullet\ (\forall\ y \bullet\ tc\ r\ y\ x \Rightarrow (\forall\ z \bullet\ tc\ r\ z\ y \Rightarrow s\ z) \Rightarrow s\ y)$
$\Rightarrow (\forall\ y \bullet\ tc\ r\ y\ x \Rightarrow s\ y))$

wf_lemma2 $\vdash \forall\ r$
$\bullet\ well\_founded\ r$
$\Rightarrow (\forall\ s$
$\bullet\ (\forall\ t \bullet\ (\forall\ u \bullet\ r\ u\ t \Rightarrow s\ u) \Rightarrow s\ t) \Rightarrow (\forall\ e \bullet\ s\ e))$

tcwf_lemma2 $\vdash \forall\ r$
$\bullet\ well\_founded\ r$
$\Rightarrow (\forall\ s$
$\bullet\ (\forall\ t \bullet\ (\forall\ u \bullet\ tc\ r\ u\ t \Rightarrow s\ u) \Rightarrow s\ t)$
$\Rightarrow (\forall\ e \bullet\ s\ e))$

wf_tc_wf_thm $\vdash \forall\ r \bullet\ well\_founded\ r \Rightarrow well\_founded\ (tc\ r)$

tc_wf_wf_thm $\vdash \forall\ r \bullet\ well\_founded\ (tc\ r) \Rightarrow well\_founded\ r$

tc_wf_twf_thm
$\vdash \forall\ r \bullet\ well\_founded\ r \Rightarrow twfp\ (tc\ r)$

wf_nochain_thm
$\vdash \forall\ r$
$\bullet\ well\_founded\ r$
$\Rightarrow (\forall\ x$
$\bullet\ \neg\ (\exists\ p\ v$
$\bullet\ p\ v$
$\land (\forall\ y$

$\bullet\ p\ y \Rightarrow tc\ r\ y\ x \land (\exists\ z \bullet\ p\ z \land r\ z\ y))))$

wf_wf_thm $\vdash \forall\ r$
$\bullet\ well\_founded\ r$
$\Rightarrow \neg\ (\exists\ p\ v$
$\bullet\ p\ v \land (\forall\ y \bullet\ p\ y \Rightarrow (\exists\ z \bullet\ p\ z \land r\ z\ y)))$

nochain_wf_thm
$\vdash \forall\ r$
$\bullet\ (\forall\ x$
$\bullet\ \neg\ (\exists\ p\ v$
$\bullet\ p\ v$
$\land (\forall\ y$
$\bullet\ p\ y \Rightarrow tc\ r\ y\ x \land (\exists\ z \bullet\ p\ z \land r\ z\ y))))$
$\Rightarrow well\_founded\ r$

wf_⇔_nochain_thm
$\vdash \forall\ r$
$\bullet\ well\_founded\ r$
$\Leftrightarrow (\forall\ x$
$\bullet\ \neg\ (\exists\ p\ v$
$\bullet\ p\ v$
$\land (\forall\ y$
$\bullet\ p\ y \Rightarrow tc\ r\ y\ x \land (\exists\ z \bullet\ p\ z \land r\ z\ y))))$

wf_induct_thm
$\vdash \neg\ (\exists\ p\ v \bullet\ p\ v \land (\forall\ y \bullet\ p\ y \Rightarrow (\exists\ z \bullet\ p\ z \land r\ z\ y)))$
$\Rightarrow well\_founded\ r$

nochain_wf_thm2
$\vdash \forall\ r$
$\bullet\ (\forall\ x$
$\bullet\ \neg\ (\exists\ p\ v$
$\bullet\ p\ v \land (\forall\ y \bullet\ p\ y \Rightarrow (\exists\ z \bullet\ p\ z \land r\ z\ y))))$
$\Rightarrow well\_founded\ r$

nochain_min_thm
$\vdash \forall\ r$
$\bullet\ (\forall\ x$
$\bullet\ \neg\ (\exists\ p\ v$
$\bullet\ p\ v$
$\land (\forall\ y$
$\bullet\ p\ y \Rightarrow tc\ r\ y\ x \land (\exists\ z \bullet\ p\ z \land r\ z\ y))))$
$\Rightarrow (\forall\ x$
$\bullet\ (\exists\ y \bullet\ r\ y\ x)$
$\Rightarrow (\exists\ z \bullet\ r\ z\ x \land \neg\ (\exists\ v \bullet\ r\ v\ z \land r\ v\ x)))$

nochain_min_thm2
$\vdash \forall\ r$
$\bullet\ (\forall\ x$
$\bullet\ \neg\ (\exists\ p\ v$
$\bullet\ p\ v \land (\forall\ y \bullet\ p\ y \Rightarrow (\exists\ z \bullet\ p\ z \land r\ z\ y))))$
$\Rightarrow (\forall\ p$
$\bullet\ (\exists\ y \bullet\ p\ y) \Rightarrow (\exists\ z \bullet\ p\ z \land \neg\ (\exists\ v \bullet\ r\ v\ z \land p\ v)))$

wf_min_thm $\vdash \forall\ r$
$\bullet\ well\_founded\ r$
$\Rightarrow (\forall\ x$
$\bullet\ (\exists\ y \bullet\ r\ y\ x)$

$\Rightarrow (\exists\ z\bullet\ r\ z\ x \wedge \neg\ (\exists\ v\bullet\ r\ v\ z \wedge r\ v\ x)))$

minr_not_wf_thm

$\vdash \exists\ r$
$\bullet\ (\forall\ x$
$\bullet\ (\exists\ y\bullet\ r\ y\ x)$
$\Rightarrow (\exists\ z\bullet\ r\ z\ x \wedge \neg\ (\exists\ v\bullet\ r\ v\ z \wedge r\ v\ x)))$
$\wedge \neg\ well\_founded\ r$

wf_not_refl_thm

$\vdash \forall\ r\bullet\ well\_founded\ r \Rightarrow \neg\ (\exists\ x\bullet\ r\ x\ x)$

wf_restrict_wf_thm

$\vdash \forall\ r$
$\bullet\ well\_founded\ r$
$\Rightarrow (\forall\ r2\bullet\ well\_founded\ (\lambda\ x\ y\bullet\ r2\ x\ y \wedge r\ x\ y))$

wf_image_wf_thm

$\vdash \forall\ r$
$\bullet\ well\_founded\ r$
$\Rightarrow (\forall\ f\bullet\ well\_founded\ (\lambda\ x\ y\bullet\ r\ (f\ x)\ (f\ y)))$

8 The Theory wf_recp

8.1 Parents

wf_relp

8.2 Children

gst−ax

8.3 Constants

\$respects	$(('a \to 'b) \to 'a \to 'b) \to ('a \to 'a \to BOOL) \to BOOL$
fixed_below	$(('a \to 'b) \to 'a \to 'b)$ $\to ('a \to 'a \to BOOL)$ $\to ('a \to 'b)$ $\to 'a$ $\to BOOL$
fixed_at	$(('a \to 'b) \to 'a \to 'b)$ $\to ('a \to 'a \to BOOL)$ $\to ('a \to 'b)$ $\to 'a$ $\to BOOL$
fix	$(('a \to 'b) \to 'a \to 'b) \to 'a \to 'b$
relmap	$('a \to 'a \to BOOL) \to ('a \to 'b) \to 'a \to 'a \to 'b \to BOOL$

8.4 Fixity

Right Infix 240:

respects

8.5 Definitions

respects	$\vdash \forall\ f\ r$ $\bullet\ f\ respects\ r$ $\Leftrightarrow (\forall\ g\ h\ x$ $\bullet\ (\forall\ y\bullet\ tc\ r\ y\ x \Rightarrow g\ y = h\ y) \Rightarrow f\ g\ x = f\ h\ x)$
fixed_below	$\vdash \forall\ f\ r\ g\ x$ $\bullet\ fixed\_below\ f\ r\ g\ x \Leftrightarrow (\forall\ y\bullet\ tc\ r\ y\ x \Rightarrow f\ g\ y = g\ y)$
fixed_at	$\vdash \forall\ f\ r\ g\ x$ $\bullet\ fixed\_at\ f\ r\ g\ x$ $\Leftrightarrow fixed\_below\ f\ r\ g\ x \land f\ g\ x = g\ x$
fix	$\vdash \forall\ f\ r$ $\bullet\ well\_founded\ r \land f\ respects\ r \Rightarrow f\ (fix\ f) = fix\ f$
relmap	$\vdash \forall\ r\ f\bullet\ relmap\ r\ f = (\lambda\ x\ y\ z\bullet\ r\ y\ x \land z = f\ y)$

8.6 Theorems

***fixed_below_lemma*1**

$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall\ x\ g\ y$
$\bullet$ *fixed_below* $f\ r\ g\ x \land tc\ r\ y\ x$
$\Rightarrow$ *fixed_below* $f\ r\ g\ y)$

***fixed_at_lemma*1**

$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall\ x\ g$
$\bullet$ *fixed_below* $f\ r\ g\ x \Rightarrow$ *fixed_at* $f\ r\ (f\ g)\ x)$

***fixed_at_lemma*2**

$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall\ x\ g$
$\bullet$ *fixed_below* $f\ r\ g\ x$
$\Rightarrow (\forall\ y\bullet\ tc\ r\ y\ x \Rightarrow$ *fixed_at* $f\ r\ g\ y))$

***fixed_at_lemma*3**

$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall\ x\ g$
$\bullet (\forall\ y\bullet\ tc\ r\ y\ x \Rightarrow$ *fixed_at* $f\ r\ g\ y)$
$\Rightarrow$ *fixed_below* $f\ r\ g\ x)$

***fixed_below_lemma*2**

$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall\ x\ g\ h$
$\bullet$ *fixed_below* $f\ r\ g\ x \land$ *fixed_below* $f\ r\ h\ x$
$\Rightarrow (\forall\ z\bullet\ tc\ r\ z\ x \Rightarrow h\ z = g\ z))$

***fixed_at_lemma*4**

$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall\ g\ x$
$\bullet$ *fixed_at* $f\ r\ g\ x$
$\Rightarrow (\forall\ y\bullet\ tc\ r\ y\ x \Rightarrow$ *fixed_at* $f\ r\ g\ y))$

***fixed_at_lemma*5**

$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall\ g\ h\ x$
$\bullet$ *fixed_at* $f\ r\ g\ x \land$ *fixed_at* $f\ r\ h\ x$
$\Rightarrow g\ x = h\ x)$

***fixed_below_lemma*3**

$\vdash \forall f\ r$
$\bullet$ *well_founded* $r \land f$ *respects* r
$\Rightarrow (\forall\ x$
$\bullet (\forall\ y\bullet\ tc\ r\ y\ x \Rightarrow (\exists\ g\bullet$ *fixed_at* $f\ r\ g\ y))$
$\Rightarrow (\exists\ g\bullet$ *fixed_below* $f\ r\ g\ x))$

***fixed_below_lemma*4**

$\vdash \forall r\ f$
$\bullet$ *well_founded* $r \land f$ *respects* r

$\Rightarrow (\forall\ x\bullet\ \exists\ g\bullet\ \mathit{fixed\_below}\ f\ \ r\ \ g\ \ x)$

***fixed_at_lemma*6**

$\vdash \forall\ f\ \ r$

$\bullet\ \mathit{well\_founded}\ \ r\ \wedge\ f\ \ \mathit{respects}\ \ r$

$\Rightarrow (\forall\ x\bullet\ \exists\ g\bullet\ \mathit{fixed\_at}\ \ f\ \ r\ \ g\ \ x)$

***fixed_lemma*1** $\vdash \forall\ f\ \ r$

$\bullet\ \mathit{well\_founded}\ \ r\ \wedge\ f\ \ \mathit{respects}\ \ r$

$\Rightarrow (\forall\ x$

$\bullet\ \mathit{fixed\_at}$

f

r

$(\lambda\ x\bullet\ (\epsilon\ h\bullet\ \mathit{fixed\_at}\ \ f\ \ r\ \ h\ \ x)\ \ x)$

$x)$

***fixp_thm*1** $\vdash \forall\ f\ \ r\bullet\ \mathit{well\_founded}\ \ r\ \wedge\ f\ \ \mathit{respects}\ \ r \Rightarrow (\exists\ g\bullet\ f\ \ g\ =\ g)$

relmap_respect_thm

$\vdash \forall\ r\ \ g\bullet\ (\lambda\ f\bullet\ g\ \ o\ \ \mathit{relmap}\ \ r\ \ f)\ \ \mathit{respects}\ \ r$

9 INDEX