

The Independence of the Continuum Hypothesis in Isabelle/ZF

Emmanuel Gunther* Miguel Pagano* Pedro Sánchez Terraf*[†]
Matías Steinberg*

February 6, 2026

Abstract

We redeveloped our formalization of forcing in the set theory framework of Isabelle/ZF. Under the assumption of the existence of a countable transitive model of ZFC , we construct proper generic extensions that satisfy the Continuum Hypothesis and its negation.

Contents

1	Introduction	4
2	Forcing notions	4
2.1	Basic concepts	5
2.2	Towards Rasiowa-Sikorski Lemma (RSL)	9
3	Cohen forcing notions	10
3.1	Combinatorial results on Cohen posets	11
3.2	The well-founded relation <i>ed</i>	14
4	Well-founded relation on names	16
5	Concepts involved in instances of Replacement	24
5.1	Formulas used to prove some generic instances.	28
5.2	The relation <i>frecrel</i>	29
5.3	Definition of Forces	30
5.3.1	Definition of <i>forces</i> for equality and membership	30
5.3.2	The well-founded relation <i>forcerec</i>	31

*Universidad Nacional de Córdoba. Facultad de Matemática, Astronomía, Física y Computación.

[†]Centro de Investigación y Estudios de Matemática (CIEM-FaMAF), Conicet. Córdoba. Argentina. Supported by Secyt-UNC project 33620180100465CB.

5.4	<i>forc_at</i> , forcing for atomic formulas	31
5.5	Forcing for general formulas	33
5.5.1	The primitive recursion	34
5.6	The arity of <i>forces</i>	34
6	The ZFC axioms, internalized	38
6.1	The Axiom of Separation, internalized	39
6.2	The Axiom of Replacement, internalized	40
7	Interface between set models and Constructibility	45
7.1	Interface with <i>M_trivial</i>	46
7.2	Interface with <i>M_basic</i>	47
7.3	Interface with <i>M_trancl</i>	51
7.4	Interface with <i>M_eclose</i>	51
7.5	Interface for proving Collects and Replace in M.	53
7.6	More Instances of Separation	60
8	More Instances of Replacement	63
9	Further instances of axiom-schemes	79
10	Transitive set models of ZF	87
10.1	A forcing locale and generic filters	87
11	The definition of <i>forces</i>	89
11.1	The relation <i>frecrel</i>	89
11.2	Recursive expression of <i>forc_at</i>	94
11.3	Absoluteness of <i>forc_at</i>	94
11.4	Forcing for atomic formulas in context	96
12	Names and generic extensions	97
12.1	Values and check-names	98
13	The Forcing Theorems	103
13.1	The forcing relation in context	103
13.2	Kunen 2013, Lemma IV.2.37(a)	103
13.3	Kunen 2013, Lemma IV.2.37(a)	103
13.4	Kunen 2013, Lemma IV.2.37(b)	104
13.5	Kunen 2013, Lemma IV.2.38	104
13.6	The relation of forcing and atomic formulas	104
13.7	The relation of forcing and connectives	105
13.8	Kunen 2013, Lemma IV.2.29	106
13.9	Auxiliary results for Lemma IV.2.40(a)	106
13.10	Induction on names	107
13.11	Lemma IV.2.40(a), in full	108

13.12	Lemma IV.2.40(b)	108
13.13	The Strengthening Lemma	110
13.14	The Density Lemma	110
13.15	The Truth Lemma	110
13.16	The “Definition of forcing”	112
14	Ordinals in generic extensions	113
15	Auxiliary renamings for Separation	113
16	The Axiom of Separation in $M[G]$	116
17	The Axiom of Pairing in $M[G]$	117
18	The Axiom of Unions in $M[G]$	117
19	The Powerset Axiom in $M[G]$	118
20	The Axiom of Extensionality in $M[G]$	119
21	The Axiom of Foundation in $M[G]$	119
22	The Axiom of Replacement in $M[G]$	120
23	The Axiom of Infinity in $M[G]$	121
24	The Axiom of Choice in $M[G]$	121
24.1	$M[G]$ is a transitive model of ZF	123
25	Separative notions and proper extensions	123
26	A poset of successions	124
26.1	Cohen extension is proper	126
27	The existence of generic extensions	126
27.1	The generic extension is countable	126
27.2	Extensions of ctms of fragments of ZFC	127
28	Preservation of cardinals in generic extensions	128
28.1	Preservation by ccc forcing notions	131
29	Model of the negation of the Continuum Hypothesis	132
29.1	Non-absolute concepts between extensions	132
29.2	Cohen forcing is ccc	133
29.3	Models of fragments of $ZFC + \neg CH$	136

30 Preservation results for κ-closed forcing notions	137
30.1 $(\omega + 1)$ -Closed notions preserve countable sequences	141
31 Forcing extension satisfying the Continuum Hypothesis	141
31.1 Collapse forcing is sufficiently closed	142
31.2 Models of fragments of $ZFC + CH$	143
32 From M to \mathcal{V}	144
32.1 Locales of a class M hold in \mathcal{V}	144
33 Main definitions of the development	146
33.1 ZF	146
33.2 Relative concepts	149
33.3 Relativization of infinitary arithmetic	154
33.4 Forcing	155
34 Some demonstrations	158

1 Introduction

We formalize the theory of forcing. We work on top of the Isabelle/ZF framework developed by Paulson and Grabczewski [4]. Our mechanization is described in more detail in our papers [1] (LSFA 2018), [2], and [3] (IJCAR 2020).

The main entry point of the present session is `Definitions_Main.thy` (Section 33), in which a path from fundamental set theoretic concepts formalized in Isabelle reaching to our main theorems is expounded. Cross-references to major milestones are provided there.

In order to provide evidence for the correctness of several of our relativized definitions, we needed to assume the Axiom of Choice (AC) during the aforementioned theory. Nevertheless, the whole of our development is independent of AC , and the theory `CH.thy` already provides all of our results and does not import that axiom.

Release notes

Previous versions of this development can be found at <https://cs.famaf.unc.edu.ar/~pedro/forcing/>.

2 Forcing notions

This theory defines a locale for forcing notions, that is, preorders with a distinguished maximum element.

```

theory Forcing_Notions
  imports
    ZF-Constructible.Relative
    Delta_System_Lemma.ZF_Library
begin

```

```

hide_const (open) Order.pred

```

2.1 Basic concepts

We say that two elements p, q are *compatible* if they have a lower bound in P

definition $compat_in :: i \Rightarrow i \Rightarrow i \Rightarrow i \Rightarrow o$ **where**
 $compat_in(A, r, p, q) \equiv \exists d \in A . \langle d, p \rangle \in r \wedge \langle d, q \rangle \in r$

lemma $compat_inI$:
 $[[d \in A ; \langle d, p \rangle \in r ; \langle d, q \rangle \in r]] \Longrightarrow compat_in(A, r, p, q)$
 $\langle proof \rangle$

lemma $refl_compat$:
 $[[refl(A, r) ; \langle p, q \rangle \in r \mid p = q \mid \langle q, p \rangle \in r ; p \in A ; q \in A]] \Longrightarrow compat_in(A, r, p, q)$
 $\langle proof \rangle$

lemma $chain_compat$:
 $refl(A, r) \Longrightarrow linear(A, r) \Longrightarrow (\forall p \in A. \forall q \in A. compat_in(A, r, p, q))$
 $\langle proof \rangle$

lemma $subset_fun_image$: $f: N \rightarrow P \Longrightarrow f''N \subseteq P$
 $\langle proof \rangle$

lemma $refl_monot_domain$: $refl(B, r) \Longrightarrow A \subseteq B \Longrightarrow refl(A, r)$
 $\langle proof \rangle$

locale $forcing_notion =$
fixes P ($\langle \mathbb{P} \rangle$) **and** leq **and** one ($\langle \mathbf{1} \rangle$)
assumes one_in_P : $\mathbf{1} \in \mathbb{P}$
and leq_preord : $preorder_on(\mathbb{P}, leq)$
and one_max : $\forall p \in \mathbb{P}. \langle p, \mathbf{1} \rangle \in leq$
begin

abbreviation $Leq :: [i, i] \Rightarrow o$ (**infixl** $\langle \preceq \rangle$ 50)
where $x \preceq y \equiv \langle x, y \rangle \in leq$

lemma $refl_leq$:
 $r \in \mathbb{P} \Longrightarrow r \preceq r$
 $\langle proof \rangle$

A set D is *dense* if every element $p \in \mathbb{P}$ has a lower bound in D .

definition

dense :: $i \Rightarrow o$ **where**
dense(D) $\equiv \forall p \in \mathbb{P}. \exists d \in D. d \preceq p$

There is also a weaker definition which asks for a lower bound in D only for the elements below some fixed element q .

definition

dense_below :: $i \Rightarrow i \Rightarrow o$ **where**
dense_below(D, q) $\equiv \forall p \in \mathbb{P}. p \preceq q \longrightarrow (\exists d \in D. d \in \mathbb{P} \wedge d \preceq p)$

lemma *P_dense*: *dense*(\mathbb{P})
 ⟨*proof*⟩

definition

increasing :: $i \Rightarrow o$ **where**
increasing(F) $\equiv \forall x \in F. \forall p \in \mathbb{P}. x \preceq p \longrightarrow p \in F$

definition

compat :: $i \Rightarrow i \Rightarrow o$ **where**
compat(p, q) $\equiv \text{compat_in}(\mathbb{P}, \text{leq}, p, q)$

lemma *leq_transD*: $a \preceq b \Longrightarrow b \preceq c \Longrightarrow a \in \mathbb{P} \Longrightarrow b \in \mathbb{P} \Longrightarrow c \in \mathbb{P} \Longrightarrow a \preceq c$
 ⟨*proof*⟩

lemma *leq_transD'*: $A \subseteq \mathbb{P} \Longrightarrow a \preceq b \Longrightarrow b \preceq c \Longrightarrow a \in A \Longrightarrow b \in \mathbb{P} \Longrightarrow c \in \mathbb{P} \Longrightarrow a \preceq c$
 ⟨*proof*⟩

lemma *compatD[dest!]*: $\text{compat}(p, q) \Longrightarrow \exists d \in \mathbb{P}. d \preceq p \wedge d \preceq q$
 ⟨*proof*⟩

abbreviation *Incompatible* :: $[i, i] \Rightarrow o$ (**infixl** $\langle \perp \rangle$ 50)
where $p \perp q \equiv \neg \text{compat}(p, q)$

lemma *compatI[intro!]*: $d \in \mathbb{P} \Longrightarrow d \preceq p \Longrightarrow d \preceq q \Longrightarrow \text{compat}(p, q)$
 ⟨*proof*⟩

lemma *Incompatible_imp_not_eq*: $\llbracket p \perp q; p \in \mathbb{P}; q \in \mathbb{P} \rrbracket \Longrightarrow p \neq q$
 ⟨*proof*⟩

lemma *denseD [dest]*: $\text{dense}(D) \Longrightarrow p \in \mathbb{P} \Longrightarrow \exists d \in D. d \preceq p$
 ⟨*proof*⟩

lemma *denseI [intro!]*: $\llbracket \bigwedge p. p \in \mathbb{P} \Longrightarrow \exists d \in D. d \preceq p \rrbracket \Longrightarrow \text{dense}(D)$
 ⟨*proof*⟩

lemma *dense_belowD [dest]*:
assumes $\text{dense_below}(D, p)$ $q \in \mathbb{P}$ $q \preceq p$
shows $\exists d \in D. d \in \mathbb{P} \wedge d \preceq q$
 ⟨*proof*⟩

lemma *dense_belowI* [intro!]:

assumes $\bigwedge q. q \in \mathbb{P} \implies q \preceq p \implies \exists d \in D. d \in \mathbb{P} \wedge d \preceq q$
shows $\text{dense_below}(D, p)$
<proof>

lemma *dense_below_cong*: $p \in \mathbb{P} \implies D = D' \implies \text{dense_below}(D, p) \longleftrightarrow \text{dense_below}(D', p)$
<proof>

lemma *dense_below_cong'*: $p \in \mathbb{P} \implies [\bigwedge x. x \in \mathbb{P} \implies Q(x) \longleftrightarrow Q'(x)] \implies$
 $\text{dense_below}(\{q \in \mathbb{P}. Q(q)\}, p) \longleftrightarrow \text{dense_below}(\{q \in \mathbb{P}. Q'(q)\}, p)$
<proof>

lemma *dense_below_mono*: $p \in \mathbb{P} \implies D \subseteq D' \implies \text{dense_below}(D, p) \implies \text{dense_below}(D', p)$
<proof>

lemma *dense_below_under*:

assumes $\text{dense_below}(D, p) \ p \in \mathbb{P} \ q \in \mathbb{P} \ q \preceq p$
shows $\text{dense_below}(D, q)$
<proof>

lemma *ideal_dense_below*:

assumes $\bigwedge q. q \in \mathbb{P} \implies q \preceq p \implies q \in D$
shows $\text{dense_below}(D, p)$
<proof>

lemma *dense_below_dense_below*:

assumes $\text{dense_below}(\{q \in \mathbb{P}. \text{dense_below}(D, q)\}, p) \ p \in \mathbb{P}$
shows $\text{dense_below}(D, p)$
<proof>

A filter is an increasing set G with all its elements being compatible in G .

definition

filter :: $i \Rightarrow o$ **where**
 $\text{filter}(G) \equiv G \subseteq \mathbb{P} \wedge \text{increasing}(G) \wedge (\forall p \in G. \forall q \in G. \text{compat_in}(G, \text{leq}, p, q))$

lemma *filterD* : $\text{filter}(G) \implies x \in G \implies x \in \mathbb{P}$
<proof>

lemma *filter_subset_notion*[dest]: $\text{filter}(G) \implies G \subseteq \mathbb{P}$
<proof>

lemma *filter_leqD* : $\text{filter}(G) \implies x \in G \implies y \in \mathbb{P} \implies x \preceq y \implies y \in G$
<proof>

lemma *filter_imp_compat*: $\text{filter}(G) \implies p \in G \implies q \in G \implies \text{compat}(p, q)$
<proof>

lemma *low_bound_filter*: — says the compatibility is attained inside G

assumes $filter(G)$ **and** $p \in G$ **and** $q \in G$
shows $\exists r \in G. r \preceq p \wedge r \preceq q$
 $\langle proof \rangle$

We finally introduce the upward closure of a set and prove that the closure of A is a filter if its elements are compatible in A .

definition

$upclosure :: i \Rightarrow i$ **where**
 $upclosure(A) \equiv \{p \in \mathbb{P}. \exists a \in A. a \preceq p\}$

lemma $upclosureI$ [*intro*] : $p \in \mathbb{P} \Rightarrow a \in A \Rightarrow a \preceq p \Rightarrow p \in upclosure(A)$
 $\langle proof \rangle$

lemma $upclosureE$ [*elim*] :
 $p \in upclosure(A) \Rightarrow (\bigwedge x a. x \in \mathbb{P} \Rightarrow a \in A \Rightarrow a \preceq x \Rightarrow R) \Rightarrow R$
 $\langle proof \rangle$

lemma $upclosureD$ [*dest*] :
 $p \in upclosure(A) \Rightarrow \exists a \in A. (a \preceq p) \wedge p \in \mathbb{P}$
 $\langle proof \rangle$

lemma $upclosure_increasing$:
assumes $A \subseteq \mathbb{P}$
shows $increasing(upclosure(A))$
 $\langle proof \rangle$

lemma $upclosure_in_P$: $A \subseteq \mathbb{P} \Rightarrow upclosure(A) \subseteq \mathbb{P}$
 $\langle proof \rangle$

lemma $A_sub_upclosure$: $A \subseteq \mathbb{P} \Rightarrow A \subseteq upclosure(A)$
 $\langle proof \rangle$

lemma $elem_upclosure$: $A \subseteq \mathbb{P} \Rightarrow x \in A \Rightarrow x \in upclosure(A)$
 $\langle proof \rangle$

lemma $closure_compat_filter$:
assumes $A \subseteq \mathbb{P} (\forall p \in A. \forall q \in A. compat_in(A, leq, p, q))$
shows $filter(upclosure(A))$
 $\langle proof \rangle$

lemma aux_RS1 : $f \in N \rightarrow \mathbb{P} \Rightarrow n \in N \Rightarrow f^n \in upclosure(f^{“N})$
 $\langle proof \rangle$

lemma $decr_succ_decr$:
assumes $f \in nat \rightarrow \mathbb{P} preorder_on(\mathbb{P}, leq)$
 $\forall n \in nat. \langle f^{\cdot} succ(n), f^{\cdot} n \rangle \in leq$
 $m \in nat$
shows $n \in nat \Rightarrow n \leq m \Rightarrow \langle f^{\cdot} m, f^{\cdot} n \rangle \in leq$
 $\langle proof \rangle$

lemma *decr_seq_linear*:
assumes $\text{refl}(\mathbb{P}, \text{leq})$ $f \in \text{nat} \rightarrow \mathbb{P}$
 $\forall n \in \text{nat}. \langle f \text{ ' succ}(n), f \text{ ' } n \rangle \in \text{leq}$
 $\text{trans}[\mathbb{P}](\text{leq})$
shows $\text{linear}(f \text{ ' ' nat, leq})$
 $\langle \text{proof} \rangle$

end — *forcing_notion*

2.2 Towards Rasiowa-Sikorski Lemma (RSL)

locale *countable_generic* = *forcing_notion* +
fixes \mathcal{D}
assumes *countable_subs_of_P*: $\mathcal{D} \in \text{nat} \rightarrow \text{Pow}(\mathbb{P})$
and *seq_of_denses*: $\forall n \in \text{nat}. \text{dense}(\mathcal{D}'n)$

begin

definition

$D_generic :: i \Rightarrow o$ **where**
 $D_generic(G) \equiv \text{filter}(G) \wedge (\forall n \in \text{nat}. (\mathcal{D}'n) \cap G \neq \emptyset)$

The next lemma identifies a sufficient condition for obtaining RSL.

lemma *RS_sequence_imp_rasiowa_sikorski*:
assumes
 $p \in \mathbb{P}$ $f : \text{nat} \rightarrow \mathbb{P}$ $f \text{ ' } 0 = p$
 $\bigwedge n. n \in \text{nat} \implies f \text{ ' succ}(n) \preceq f \text{ ' } n \wedge f \text{ ' succ}(n) \in \mathcal{D}'n$
shows
 $\exists G. p \in G \wedge D_generic(G)$
 $\langle \text{proof} \rangle$

end — *countable_generic*

Now, the following recursive definition will fulfill the requirements of lemma *RS_sequence_imp_rasiowa_sikorski*

consts *RS_seq* :: $[i, i, i, i, i, i] \Rightarrow i$

primrec

$RS_seq(0, P, \text{leq}, p, \text{enum}, \mathcal{D}) = p$
 $RS_seq(\text{succ}(n), P, \text{leq}, p, \text{enum}, \mathcal{D}) =$
 $\text{enum}'(\mu m. \langle \text{enum}'m, RS_seq(n, P, \text{leq}, p, \text{enum}, \mathcal{D}) \rangle \in \text{leq} \wedge \text{enum}'m \in \mathcal{D}'n)$

context *countable_generic*

begin

lemma *countable_RS_sequence_aux*:

fixes p *enum*

defines $f(n) \equiv RS_seq(n, \mathbb{P}, \text{leq}, p, \text{enum}, \mathcal{D})$

and $Q(q, k, m) \equiv \text{enum}'m \preceq q \wedge \text{enum}'m \in \mathcal{D}'k$

assumes $n \in \text{nat}$ $p \in \mathbb{P}$ $\mathbb{P} \subseteq \text{range}(\text{enum})$ $\text{enum} : \text{nat} \rightarrow M$
 $\bigwedge x k. x \in \mathbb{P} \implies k \in \text{nat} \implies \exists q \in \mathbb{P}. q \preceq x \wedge q \in \mathcal{D} \text{ ' } k$
shows
 $f(\text{succ}(n)) \in \mathbb{P} \wedge f(\text{succ}(n)) \preceq f(n) \wedge f(\text{succ}(n)) \in \mathcal{D} \text{ ' } n$
 $\langle \text{proof} \rangle$

lemma *countable_RS_sequence*:

fixes $p \text{ enum}$
defines $f \equiv \lambda n \in \text{nat}. \text{RS_seq}(n, \mathbb{P}, \text{leq}, p, \text{enum}, \mathcal{D})$
and $Q(q, k, m) \equiv \text{enum} \text{ ' } m \preceq q \wedge \text{enum} \text{ ' } m \in \mathcal{D} \text{ ' } k$
assumes $n \in \text{nat}$ $p \in \mathbb{P}$ $\mathbb{P} \subseteq \text{range}(\text{enum})$ $\text{enum} : \text{nat} \rightarrow M$
shows
 $f \text{ ' } 0 = p$ $f \text{ ' } \text{succ}(n) \preceq f \text{ ' } n$ $\wedge f \text{ ' } \text{succ}(n) \in \mathcal{D} \text{ ' } n$ $f \text{ ' } \text{succ}(n) \in \mathbb{P}$
 $\langle \text{proof} \rangle$

lemma *RS_seq_type*:

assumes $n \in \text{nat}$ $p \in \mathbb{P}$ $\mathbb{P} \subseteq \text{range}(\text{enum})$ $\text{enum} : \text{nat} \rightarrow M$
shows $\text{RS_seq}(n, \mathbb{P}, \text{leq}, p, \text{enum}, \mathcal{D}) \in \mathbb{P}$
 $\langle \text{proof} \rangle$

lemma *RS_seq_funtype*:

assumes $p \in \mathbb{P}$ $\mathbb{P} \subseteq \text{range}(\text{enum})$ $\text{enum} : \text{nat} \rightarrow M$
shows $(\lambda n \in \text{nat}. \text{RS_seq}(n, \mathbb{P}, \text{leq}, p, \text{enum}, \mathcal{D})) : \text{nat} \rightarrow \mathbb{P}$
 $\langle \text{proof} \rangle$

lemmas *countable_rasiowa_sikorski* =

$\text{RS_sequence_imp_rasiowa_sikorski}[\text{OF_RS_seq_funtype countable_RS_sequence}(1,2)]$

end — *countable_generic*

end

3 Cohen forcing notions

theory *Cohen_Posets_Relative*

imports

Forcing_Notions

Transitive_Models.Delta_System_Relative

Transitive_Models.Partial_Functions_Relative

begin

locale *cohen_data* =

fixes $\kappa I J :: i$

assumes *zero_lt_kappa*: $0 < \kappa$

begin

lemmas *zero_lespoll_kappa* = $\text{zero_lespoll}[\text{OF } \text{zero_lt_kappa}]$

end — *cohen_data*

abbreviation

inj_dense :: $[i, i, i, i] \Rightarrow i$ **where**
inj_dense(I, J, w, x) \equiv
 $\{ p \in \text{Fn}(\omega, I \times \omega, J) . (\exists n \in \omega. \langle \langle w, n \rangle, 1 \rangle \in p \wedge \langle \langle x, n \rangle, 0 \rangle \in p) \}$

lemma *dense_inj_dense*:

assumes $w \in I \ x \in I \ w \neq x \ p \in \text{Fn}(\omega, I \times \omega, J) \ 0 \in J \ 1 \in J$
shows $\exists d \in \text{inj_dense}(I, J, w, x). \langle d, p \rangle \in \text{Fnle}(\omega, I \times \omega, J)$
 $\langle \text{proof} \rangle$

locale *add_reals* = *cohen_data* *nat* _ 2

3.1 Combinatorial results on Cohen posets

sublocale *cohen_data* \subseteq *forcing_notion* $\text{Fn}(\kappa, I, J) \ \text{Fnle}(\kappa, I, J) \ 0$
 $\langle \text{proof} \rangle$

context *cohen_data*

begin

lemma *compat_imp_Un_is_function*:

assumes $G \subseteq \text{Fn}(\kappa, I, J) \ \bigwedge p \ q. \ p \in G \Longrightarrow q \in G \Longrightarrow \text{compat}(p, q)$
shows $\text{function}(\bigcup G)$
 $\langle \text{proof} \rangle$

lemma *Un_filter_is_function*: $\text{filter}(G) \Longrightarrow \text{function}(\bigcup G)$

$\langle \text{proof} \rangle$

end — *cohen_data*

locale *M_cohen* = *M_delta* +

assumes

countable_lepoll_assms2:

$M(A') \Longrightarrow M(A) \Longrightarrow M(b) \Longrightarrow M(f) \Longrightarrow \text{separation}(M, \lambda y. \exists x \in A'. y = \langle x, \mu \ i. x \in \text{if_range_F_else_F}(\lambda a. \{p \in A . \text{domain}(p) = a\}, b, f, i))$

and

countable_lepoll_assms3:

$M(A) \Longrightarrow M(f) \Longrightarrow M(b) \Longrightarrow M(D) \Longrightarrow M(r') \Longrightarrow M(A') \Longrightarrow$
 $\text{separation}(M, \lambda y. \exists x \in A'. y = \langle x, \mu \ i. x \in \text{if_range_F_else_F}(\text{drSR_Y}(r', D, A), b, f, i))$

lemma (**in** *M_library*) *Fnle_rel_Aleph_rel1_closed*[*intro, simp*]:

$M(\text{Fnle}^M(\aleph_I^M, \aleph_I^M, \omega \rightarrow^M 2))$

$\langle \text{proof} \rangle$

locale *M_add_reals* = *M_cohen* + *add_reals*

begin

lemmas *zero_lesspoll_rel_kappa* = *zero_lesspoll_rel*[*OF zero_lt_kappa*]

end — *M_add_reals*

⟨*ML*⟩

context

notes *Un_assoc*[*simp*] *Un_transposition_aux2*[*simp*]

begin

⟨*ML*⟩

end

lemma (in *M_trivial*) *compat_in_abs*[*absolut*]:

assumes

$M(A) M(r) M(p) M(q)$

shows

$is_compat_in(M, A, r, p, q) \longleftrightarrow compat_in(A, r, p, q)$

⟨*proof*⟩

definition

antichain :: $i \Rightarrow i \Rightarrow i \Rightarrow o$ **where**

$antichain(P, leq, A) \equiv A \subseteq P \wedge (\forall p \in A. \forall q \in A. p \neq q \longrightarrow \neg compat_in(P, leq, p, q))$

⟨*ML*⟩

definition

ccc :: $i \Rightarrow i \Rightarrow o$ **where**

$ccc(P, leq) \equiv \forall A. antichain(P, leq, A) \longrightarrow |A| \leq nat$

abbreviation

antichain_rel_abbr :: $[i \Rightarrow o, i, i, i] \Rightarrow o$ (⟨*antichain*[']($_, _, _$)⟩) **where**

$antichain^M(P, leq, A) \equiv antichain_rel(M, P, leq, A)$

abbreviation

antichain_r_set :: $[i, i, i, i] \Rightarrow o$ (⟨*antichain*[']($_, _, _$)⟩) **where**

$antichain^M(P, leq, A) \equiv antichain_rel(\#\#M, P, leq, A)$

context *M_trivial*

begin

lemma *antichain_abs* [*absolut*]:

$\llbracket M(A); M(P); M(leq) \rrbracket \Longrightarrow antichain^M(P, leq, A) \longleftrightarrow antichain(P, leq, A)$

⟨*proof*⟩

end — *M_trivial*

⟨*ML*⟩

abbreviation

$ccc_rel_abbr :: [i \Rightarrow o, i, i] \Rightarrow o \langle ccc-'(_,_)' \rangle$ **where**
 $ccc_rel_abbr(M) \equiv ccc_rel(M)$

abbreviation

$ccc_r_set :: [i, i, i] \Rightarrow o \langle ccc-'(_,_)' \rangle$ **where**
 $ccc_r_set(M) \equiv ccc_rel(\#\#M)$

context $M_cardinals$ **begin****lemma** def_ccc_rel :**shows**

$ccc^M(P, leq) \longleftrightarrow (\forall A[M]. antichain^M(P, leq, A) \longrightarrow |A|^M \leq \omega)$
 $\langle proof \rangle$

end — $M_cardinals$ **context** $M_FiniteFun$ **begin****lemma** $Fnle_nat_closed$ [$intro, simp$]:

assumes $M(I) M(J)$

shows $M(Fnle(\omega, I, J))$

$\langle proof \rangle$

lemma Fn_nat_closed :

assumes $M(A) M(B)$ **shows** $M(Fn(\omega, A, B))$

$\langle proof \rangle$

end — $M_FiniteFun$ **context** M_add_reals **begin**

lemma $lam_replacement_drSR_Y$: $M(A) \Longrightarrow M(D) \Longrightarrow M(r') \Longrightarrow lam_replacement(M, drSR_Y(r', D, A))$

$\langle proof \rangle$

lemma (**in** M_trans) mem_F_bound3 :

fixes $F A$

defines $F \equiv dC_F$

shows $x \in F(A, c) \Longrightarrow c \in (range(f) \cup \{domain(x). x \in A\})$

$\langle proof \rangle$

lemma $ccc_rel_Fn_nat$:

assumes $M(I)$

shows $ccc^M(Fn(nat, I, 2), Fnle(nat, I, 2))$

<proof>

end — *M_add_reals*

end

theory *Edrel*

imports

Transitive_Models.ZF_Miscellanea

Transitive_Models.Recursion_Thms

begin

3.2 The well-founded relation *ed*

lemma *eclose_sing* : $x \in \text{eclose}(a) \implies x \in \text{eclose}(\{a\})$

<proof>

lemma *ecloseE* :

assumes $x \in \text{eclose}(A)$

shows $x \in A \vee (\exists B \in A . x \in \text{eclose}(B))$

<proof>

lemma *eclose_singE* : $x \in \text{eclose}(\{a\}) \implies x = a \vee x \in \text{eclose}(a)$

<proof>

lemma *in_eclose_sing* :

assumes $x \in \text{eclose}(\{a\})$ $a \in \text{eclose}(z)$

shows $x \in \text{eclose}(\{z\})$

<proof>

lemma *in_dom_in_eclose* :

assumes $x \in \text{domain}(z)$

shows $x \in \text{eclose}(z)$

<proof>

term *ed* is the well-founded relation on which *val* is defined.

definition *ed* :: $[i,i] \Rightarrow o$ **where**

$\text{ed}(x,y) \equiv x \in \text{domain}(y)$

definition *edrel* :: $i \Rightarrow i$ **where**

$\text{edrel}(A) \equiv \text{Rrel}(\text{ed},A)$

lemma *edI[intro!]*: $t \in \text{domain}(x) \implies \text{ed}(t,x)$

<proof>

lemma *edD[dest!]*: $\text{ed}(t,x) \implies t \in \text{domain}(x)$

<proof>

lemma *rank_ed*:

assumes $ed(y,x)$
shows $succ(rank(y)) \leq rank(x)$
 $\langle proof \rangle$

lemma $edrel_dest$ [$dest$]: $x \in edrel(A) \implies \exists a \in A. \exists b \in A. x = \langle a,b \rangle$
 $\langle proof \rangle$

lemma $edrelD$: $x \in edrel(A) \implies \exists a \in A. \exists b \in A. x = \langle a,b \rangle \wedge a \in domain(b)$
 $\langle proof \rangle$

lemma $edrelI$ [$intro!$]: $x \in A \implies y \in A \implies x \in domain(y) \implies \langle x,y \rangle \in edrel(A)$
 $\langle proof \rangle$

lemma $edrel_trans$: $Transset(A) \implies y \in A \implies x \in domain(y) \implies \langle x,y \rangle \in edrel(A)$
 $\langle proof \rangle$

lemma $domain_trans$: $Transset(A) \implies y \in A \implies x \in domain(y) \implies x \in A$
 $\langle proof \rangle$

lemma $relation_edrel$: $relation(edrel(A))$
 $\langle proof \rangle$

lemma $field_edrel$: $field(edrel(A)) \subseteq A$
 $\langle proof \rangle$

lemma $edrel_sub_memrel$: $edrel(A) \subseteq trancl(Memrel(eclose(A)))$
 $\langle proof \rangle$

lemma wf_edrel : $wf(edrel(A))$
 $\langle proof \rangle$

lemma $ed_induction$:
assumes $\bigwedge x. \llbracket \bigwedge y. ed(y,x) \implies Q(y) \rrbracket \implies Q(x)$
shows $Q(a)$
 $\langle proof \rangle$

lemma $dom_under_edrel_eclose$: $edrel(eclose(\{x\})) -'' \{x\} = domain(x)$
 $\langle proof \rangle$

lemma ed_eclose : $\langle y,z \rangle \in edrel(A) \implies y \in eclose(z)$
 $\langle proof \rangle$

lemma tr_edrel_eclose : $\langle y,z \rangle \in edrel(eclose(\{x\}))^+ \implies y \in eclose(z)$
 $\langle proof \rangle$

lemma $restrict_edrel_eq$:
assumes $z \in domain(x)$
shows $edrel(eclose(\{x\})) \cap eclose(\{z\}) \times eclose(\{z\}) = edrel(eclose(\{z\}))$
 $\langle proof \rangle$

```

lemma tr_edrel_subset :
  assumes  $z \in \text{domain}(x)$ 
  shows  $\text{tr\_down}(\text{edrel}(\text{eclose}(\{x\})), z) \subseteq \text{eclose}(\{z\})$ 
  <proof>

end

```

4 Well-founded relation on names

```

theory FrecR
  imports
    Transitive_Models.Discipline_Function
    Edrel
  begin

```

frecR is the well-founded relation on names that allows us to define forcing for atomic formulas.

```

definition
  ftype ::  $i \Rightarrow i$  where
    ftype  $\equiv$  fst

```

```

definition
  name1 ::  $i \Rightarrow i$  where
    name1( $x$ )  $\equiv$  fst(snd( $x$ ))

```

```

definition
  name2 ::  $i \Rightarrow i$  where
    name2( $x$ )  $\equiv$  fst(snd(snd( $x$ )))

```

```

definition
  cond_of ::  $i \Rightarrow i$  where
    cond_of( $x$ )  $\equiv$  snd(snd(snd(( $x$ ))))

```

```

lemma components_simp:
  ftype( $\langle f, n1, n2, c \rangle$ ) = f
  name1( $\langle f, n1, n2, c \rangle$ ) = n1
  name2( $\langle f, n1, n2, c \rangle$ ) = n2
  cond_of( $\langle f, n1, n2, c \rangle$ ) = c
  <proof>

```

```

definition eclose_n ::  $[i \Rightarrow i, i] \Rightarrow i$  where
  eclose_n(name,  $x$ ) = eclose( $\{ \text{name}(x) \}$ )

```

```

definition
  ecloseN ::  $i \Rightarrow i$  where
    ecloseN( $x$ ) = eclose_n(name1,  $x$ )  $\cup$  eclose_n(name2,  $x$ )

```

```

lemma components_in_eclose :

```

$n1 \in \text{ecloseN}(\langle f, n1, n2, c \rangle)$
 $n2 \in \text{ecloseN}(\langle f, n1, n2, c \rangle)$
 $\langle \text{proof} \rangle$

lemmas $\text{names_simp} = \text{components_simp}(2) \text{ components_simp}(3)$

lemma ecloseNI1 :
assumes $x \in \text{eclose}(n1) \vee x \in \text{eclose}(n2)$
shows $x \in \text{ecloseN}(\langle f, n1, n2, c \rangle)$
 $\langle \text{proof} \rangle$

lemmas $\text{ecloseNI} = \text{ecloseNI1}$

lemma ecloseN_mono :
assumes $u \in \text{ecloseN}(x) \text{ name1}(x) \in \text{ecloseN}(y) \text{ name2}(x) \in \text{ecloseN}(y)$
shows $u \in \text{ecloseN}(y)$
 $\langle \text{proof} \rangle$

definition
 $\text{is_ftype} :: (i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$ **where**
 $\text{is_ftype} \equiv \text{is_fst}$

definition
 $\text{ftype_fm} :: [i, i] \Rightarrow i$ **where**
 $\text{ftype_fm} \equiv \text{fst_fm}$

lemma is_ftype_iff_sats [iff_sats]:
assumes
 $\text{nth}(a, \text{env}) = x \text{ nth}(b, \text{env}) = y \text{ a} \in \text{nat} \text{ b} \in \text{nat} \text{ env} \in \text{list}(A)$
shows
 $\text{is_ftype}(\#\#A, x, y) \longleftrightarrow \text{sats}(A, \text{ftype_fm}(a, b), \text{env})$
 $\langle \text{proof} \rangle$

definition
 $\text{is_name1} :: (i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$ **where**
 $\text{is_name1}(M, x, t2) \equiv \text{is_hcomp}(M, \text{is_fst}(M), \text{is_snd}(M), x, t2)$

definition
 $\text{name1_fm} :: [i, i] \Rightarrow i$ **where**
 $\text{name1_fm}(x, t) \equiv \text{hcomp_fm}(\text{fst_fm}, \text{snd_fm}, x, t)$

lemma sats_name1_fm [simp]:
 $\llbracket x \in \text{nat}; y \in \text{nat}; \text{env} \in \text{list}(A) \rrbracket \Longrightarrow$
 $(A, \text{env} \models \text{name1_fm}(x, y)) \longleftrightarrow \text{is_name1}(\#\#A, \text{nth}(x, \text{env}), \text{nth}(y, \text{env}))$
 $\langle \text{proof} \rangle$

lemma is_name1_iff_sats [iff_sats]:
assumes
 $\text{nth}(a, \text{env}) = x \text{ nth}(b, \text{env}) = y \text{ a} \in \text{nat} \text{ b} \in \text{nat} \text{ env} \in \text{list}(A)$

shows

$is_name1(\#\#A,x,y) \longleftrightarrow A, env \models name1_fm(a,b)$
<proof>

definition

$is_snd_snd :: (i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$ **where**
 $is_snd_snd(M,x,t) \equiv is_hcomp(M, is_snd(M), is_snd(M), x, t)$

definition

$snd_snd_fm :: [i,i] \Rightarrow i$ **where**
 $snd_snd_fm(x,t) \equiv hcomp_fm(snd_fm, snd_snd_fm, x, t)$

lemma sats_snd2_fm [simp]:

$\llbracket x \in nat; y \in nat; env \in list(A) \rrbracket \Longrightarrow$
 $(A, env \models snd_snd_fm(x,y)) \longleftrightarrow is_snd_snd(\#\#A, nth(x,env), nth(y,env))$
<proof>

definition

$is_name2 :: (i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$ **where**
 $is_name2(M,x,t3) \equiv is_hcomp(M, is_fst(M), is_snd_snd(M), x, t3)$

definition

$name2_fm :: [i,i] \Rightarrow i$ **where**
 $name2_fm(x,t3) \equiv hcomp_fm(fst_fm, snd_snd_fm, x, t3)$

lemma sats_name2_fm :

$\llbracket x \in nat; y \in nat; env \in list(A) \rrbracket$
 $\Longrightarrow (A, env \models name2_fm(x,y)) \longleftrightarrow is_name2(\#\#A, nth(x,env), nth(y,env))$
<proof>

lemma is_name2_iff_sats [iff_sats]:

assumes

$nth(a,env) = x \quad nth(b,env) = y \quad a \in nat \quad b \in nat \quad env \in list(A)$

shows

$is_name2(\#\#A,x,y) \longleftrightarrow A, env \models name2_fm(a,b)$
<proof>

definition

$is_cond_of :: (i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$ **where**
 $is_cond_of(M,x,t4) \equiv is_hcomp(M, is_snd(M), is_snd_snd(M), x, t4)$

definition

$cond_of_fm :: [i,i] \Rightarrow i$ **where**
 $cond_of_fm(x,t4) \equiv hcomp_fm(snd_fm, snd_snd_fm, x, t4)$

lemma sats_cond_of_fm :

$\llbracket x \in nat; y \in nat; env \in list(A) \rrbracket \Longrightarrow$
 $(A, env \models cond_of_fm(x,y)) \longleftrightarrow is_cond_of(\#\#A, nth(x,env), nth(y,env))$
<proof>

lemma *is_cond_of_iff_sats* [*iff_sats*]:
assumes
 $nth(a,env) = x \quad nth(b,env) = y \quad a \in nat \quad b \in nat \quad env \in list(A)$
shows
 $is_cond_of(\#\#A,x,y) \longleftrightarrow A, env \models cond_of_fm(a,b)$
<proof>

lemma *components_type*[*TC*] :
assumes $a \in nat \quad b \in nat$
shows
 $f_type_fm(a,b) \in formula$
 $name1_fm(a,b) \in formula$
 $name2_fm(a,b) \in formula$
 $cond_of_fm(a,b) \in formula$
<proof>

lemmas *components_iff_sats* = *is_f_type_iff_sats is_name1_iff_sats is_name2_iff_sats is_cond_of_iff_sats*

lemmas *components_defs* = *f_type_fm_def snd_snd_fm_def hcomp_fm_def name1_fm_def name2_fm_def cond_of_fm_def*

definition
 $is_eclose_n :: [i \Rightarrow o, [i \Rightarrow o, i, i] \Rightarrow o, i, i] \Rightarrow o$ **where**
 $is_eclose_n(N, is_name, en, t) \equiv$
 $\exists n1[N]. \exists s1[N]. is_name(N, t, n1) \wedge is_singleton(N, n1, s1) \wedge is_eclose(N, s1, en)$

definition
 $eclose_n1_fm :: [i, i] \Rightarrow i$ **where**
 $eclose_n1_fm(m, t) \equiv Exists(Exists(And(And(name1_fm(t+\omega 2, 0), singleton_fm(0, 1)), is_eclose_fm(1, m+\omega 2))))$

definition
 $eclose_n2_fm :: [i, i] \Rightarrow i$ **where**
 $eclose_n2_fm(m, t) \equiv Exists(Exists(And(And(name2_fm(t+\omega 2, 0), singleton_fm(0, 1)), is_eclose_fm(1, m+\omega 2))))$

definition
 $is_ecloseN :: [i \Rightarrow o, i, i] \Rightarrow o$ **where**
 $is_ecloseN(N, t, en) \equiv \exists en1[N]. \exists en2[N].$
 $is_eclose_n(N, is_name1, en1, t) \wedge is_eclose_n(N, is_name2, en2, t) \wedge$
 $union(N, en1, en2, en)$

definition
 $ecloseN_fm :: [i, i] \Rightarrow i$ **where**
 $ecloseN_fm(en, t) \equiv Exists(Exists(And(eclose_n1_fm(1, t+\omega 2), And(eclose_n2_fm(0, t+\omega 2), union_fm(1, 0, en+\omega 2))))))$

lemma *ecloseN_fm_type* [TC] :
 $\llbracket en \in nat ; t \in nat \rrbracket \implies ecloseN_fm(en,t) \in formula$
 ⟨proof⟩

lemma *sats_ecloseN_fm* [simp]:
 $\llbracket en \in nat; t \in nat ; env \in list(A) \rrbracket$
 $\implies (A, env \models ecloseN_fm(en,t)) \longleftrightarrow is_ecloseN(\#\#A,nth(t,env),nth(en,env))$
 ⟨proof⟩

lemma *is_ecloseN_iff_sats* [iff_sats]:
 $\llbracket nth(en, env) = ena; nth(t, env) = ta; en \in nat; t \in nat ; env \in list(A) \rrbracket$
 $\implies is_ecloseN(\#\#A,ta,ena) \longleftrightarrow A, env \models ecloseN_fm(en,t)$
 ⟨proof⟩

definition

frecR :: $i \Rightarrow i \Rightarrow o$ where
 $frecR(x,y) \equiv$
 $(ftype(x) = 1 \wedge ftype(y) = 0$
 $\quad \wedge (name1(x) \in domain(name1(y)) \cup domain(name2(y)) \wedge (name2(x) =$
 $name1(y) \vee name2(x) = name2(y))))$
 $\vee (ftype(x) = 0 \wedge ftype(y) = 1 \wedge name1(x) = name1(y) \wedge name2(x) \in$
 $domain(name2(y)))$

lemma *frecR_ftypeD* :
assumes *frecR(x,y)*
shows $(ftype(x) = 0 \wedge ftype(y) = 1) \vee (ftype(x) = 1 \wedge ftype(y) = 0)$
 ⟨proof⟩

lemma *frecRI1*: $s \in domain(n1) \vee s \in domain(n2) \implies frecR(\langle 1, s, n1, q \rangle, \langle 0, n1, n2, q \rangle)$
 ⟨proof⟩

lemma *frecRI1'*: $s \in domain(n1) \cup domain(n2) \implies frecR(\langle 1, s, n1, q \rangle, \langle 0, n1, n2, q \rangle)$
 ⟨proof⟩

lemma *frecRI2*: $s \in domain(n1) \vee s \in domain(n2) \implies frecR(\langle 1, s, n2, q \rangle, \langle 0, n1, n2, q \rangle)$
 ⟨proof⟩

lemma *frecRI2'*: $s \in domain(n1) \cup domain(n2) \implies frecR(\langle 1, s, n2, q \rangle, \langle 0, n1, n2, q \rangle)$
 ⟨proof⟩

lemma *frecRI3*: $\langle s, r \rangle \in n2 \implies frecR(\langle 0, n1, s, q \rangle, \langle 1, n1, n2, q \rangle)$
 ⟨proof⟩

lemma *frecRI3'*: $s \in domain(n2) \implies frecR(\langle 0, n1, s, q \rangle, \langle 1, n1, n2, q \rangle)$

<proof>

lemma *frecR_D1* :

$frecR(x,y) \implies ftype(y) = 0 \implies ftype(x) = 1 \wedge$
 $(name1(x) \in domain(name1(y)) \cup domain(name2(y)) \wedge (name2(x) = name1(y)$
 $\vee name2(x) = name2(y)))$
<proof>

lemma *frecR_D2* :

$frecR(x,y) \implies ftype(y) = 1 \implies ftype(x) = 0 \wedge$
 $ftype(x) = 0 \wedge ftype(y) = 1 \wedge name1(x) = name1(y) \wedge name2(x) \in$
 $domain(name2(y))$
<proof>

lemma *frecR_DI* :

assumes $frecR(\langle a,b,c,d \rangle, \langle ftype(y), name1(y), name2(y), cond_of(y) \rangle)$
shows $frecR(\langle a,b,c,d \rangle, y)$
<proof>

<ML>

schematic_goal *sats_frecR_fm_auto*:

assumes
 $i \in nat \ j \in nat \ env \in list(A)$
shows
 $is_frecR(\#\#A, nth(i, env), nth(j, env)) \longleftrightarrow A, env \models ?fr_fm(i, j)$
<proof>

<ML>

Third item of Kunen's observations (p. 257) about the *trcl* relation.

lemma *eq_ftypep_not_frecR*:

assumes $ftype(x) = ftype(y)$
shows $\neg frecR(x, y)$
<proof>

definition

$rank_names :: i \Rightarrow i$ **where**
 $rank_names(x) \equiv max(rank(name1(x)), rank(name2(x)))$

lemma *rank_names_types* [TC]:

shows $Ord(rank_names(x))$
<proof>

definition

$mtype_form :: i \Rightarrow i$ **where**
 $mtype_form(x) \equiv if\ rank(name1(x)) < rank(name2(x))\ then\ 0\ else\ 2$

definition

type_form :: $i \Rightarrow i$ **where**
type_form(x) \equiv if *f*type(x) = 0 then 1 else *m*type_form(x)

lemma *type_form_tc* [TC]:
shows *type_form*(x) \in \mathcal{I}
 ⟨*proof*⟩

lemma *frecR_le_rnk_names* :
assumes *frecR*(x,y)
shows *rank_names*(x) \leq *rank_names*(y)
 ⟨*proof*⟩

definition
 Γ :: $i \Rightarrow i$ **where**
 $\Gamma(x) = \mathcal{I} ** \text{rank_names}(x) ++ \text{type_form}(x)$

lemma Γ_type [TC]:
shows *Ord*($\Gamma(x)$)
 ⟨*proof*⟩

lemma Γ_mono :
assumes *frecR*(x,y)
shows $\Gamma(x) < \Gamma(y)$
 ⟨*proof*⟩

definition
frecrel :: $i \Rightarrow i$ **where**
frecrel(A) \equiv *Rrel*(*frecR*, A)

lemma *frecrelI* :
assumes $x \in A$ $y \in A$ *frecR*(x,y)
shows $\langle x,y \rangle \in \text{frecrel}(A)$
 ⟨*proof*⟩

lemma *frecrelD* :
assumes $\langle x,y \rangle \in \text{frecrel}(A1 \times A2 \times A3 \times A4)$
shows
 *f*type(x) \in $A1$ *f*type(x) \in $A1$
 name1(x) \in $A2$ *name1*(y) \in $A2$
 name2(x) \in $A3$ *name2*(x) \in $A3$
 cond_of(x) \in $A4$ *cond_of*(y) \in $A4$
 frecR(x,y)
 ⟨*proof*⟩

lemma *wf_frecrel* :
shows *wf*(*frecrel*(A))
 ⟨*proof*⟩

lemma *core_induction_aux*:

fixes $A1\ A2 :: i$
assumes
 $Transset(A1)$
 $\bigwedge \tau\ \vartheta\ p. p \in A2 \implies \llbracket \bigwedge q\ \sigma. \llbracket q \in A2 ; \sigma \in domain(\vartheta) \rrbracket \implies Q(\theta, \tau, \sigma, q) \rrbracket \implies$
 $Q(1, \tau, \vartheta, p)$
 $\bigwedge \tau\ \vartheta\ p. p \in A2 \implies \llbracket \bigwedge q\ \sigma. \llbracket q \in A2 ; \sigma \in domain(\tau) \cup domain(\vartheta) \rrbracket \implies Q(1, \sigma, \tau, q)$
 $\wedge Q(1, \sigma, \vartheta, q) \rrbracket \implies Q(\theta, \tau, \vartheta, p)$
shows $a \in 2 \times A1 \times A1 \times A2 \implies Q(ftype(a), name1(a), name2(a), cond_of(a))$
 $\langle proof \rangle$

lemma $def_frecrel : frecrel(A) = \{z \in A \times A. \exists x\ y. z = \langle x, y \rangle \wedge frecR(x, y)\}$
 $\langle proof \rangle$

lemma $frecrel_fst_snd:$
 $frecrel(A) = \{z \in A \times A .$
 $ftype(fst(z)) = 1 \wedge$
 $ftype(snd(z)) = 0 \wedge name1(fst(z)) \in domain(name1(snd(z))) \cup do-$
 $main(name2(snd(z))) \wedge$
 $(name2(fst(z)) = name1(snd(z)) \vee name2(fst(z)) = name2(snd(z)))$
 $\vee (ftype(fst(z)) = 0 \wedge$
 $ftype(snd(z)) = 1 \wedge name1(fst(z)) = name1(snd(z)) \wedge name2(fst(z)) \in$
 $domain(name2(snd(z))))\}$
 $\langle proof \rangle$

end
theory $FrecR_Arities$
imports
 $FrecR$
begin

context
notes $FOL_arities[simp]$
begin

$\langle ML \rangle$
lemma $arity_fst_fm\ [arity] :$
 $\llbracket x \in nat ; t \in nat \rrbracket \implies arity(fst_fm(x, t)) = succ(x) \cup succ(t)$
 $\langle proof \rangle$

$\langle ML \rangle$
lemma $arity_snd_fm\ [arity] :$
 $\llbracket x \in nat ; t \in nat \rrbracket \implies arity(snd_fm(x, t)) = succ(x) \cup succ(t)$
 $\langle proof \rangle$

lemma $arity_snd_snd_fm\ [arity] :$
 $\llbracket x \in nat ; t \in nat \rrbracket \implies arity(snd_snd_fm(x, t)) = succ(x) \cup succ(t)$
 $\langle proof \rangle$

lemma $arity_ftype_fm\ [arity] :$

```

[[x∈nat ; t∈nat]] ⇒ arity(ftype_fm(x,t)) = succ(x) ∪ succ(t)
⟨proof⟩

lemma arity_name1_fm [arity] :
[[x∈nat ; t∈nat]] ⇒ arity(name1_fm(x,t)) = succ(x) ∪ succ(t)
⟨proof⟩

lemma arity_name2_fm [arity] :
[[x∈nat ; t∈nat]] ⇒ arity(name2_fm(x,t)) = succ(x) ∪ succ(t)
⟨proof⟩

lemma arity_cond_of_fm [arity] :
[[x∈nat ; t∈nat]] ⇒ arity(cond_of_fm(x,t)) = succ(x) ∪ succ(t)
⟨proof⟩

lemma arity_eclose_n1_fm [arity] :
[[x∈nat ; t∈nat]] ⇒ arity(eclose_n1_fm(x,t)) = succ(x) ∪ succ(t)
⟨proof⟩

lemma arity_eclose_n2_fm [arity] :
[[x∈nat ; t∈nat]] ⇒ arity(eclose_n2_fm(x,t)) = succ(x) ∪ succ(t)
⟨proof⟩

lemma arity_ecloseN_fm [arity] :
[[x∈nat ; t∈nat]] ⇒ arity(ecloseN_fm(x,t)) = succ(x) ∪ succ(t)
⟨proof⟩

lemma arity_frecl_fm [arity]:
[[a∈nat;b∈nat]] ⇒ arity(frecl_fm(a,b)) = succ(a) ∪ succ(b)
⟨proof⟩

end — FOL_arities

end

```

5 Concepts involved in instances of Replacement

```

theory Fm_Definitions
imports
  Transitive_Models.Renaming_Auto
  Transitive_Models.Aleph_Relative
  Frecl_Arities
begin

no_notation Aleph (⟨ℵ_⟩ [90] 90)

```

In this theory we put every concept that should be synthesized in a formula to have an instance of replacement.

The automatic synthesis of a concept /foo/ requires that every concept used to define /foo/ is already synthesized. We try to use our meta-programs to synthesize concepts: given the absolute concept /foo/ we relativize in relational form obtaining /is_foo/ and then we synthesize the formula /is_foo_fm/. The meta-program that synthesizes formulas also produces satisfaction lemmas.

Having one file to collect every formula needed for replacements breaks the reading flow: we need to introduce the concept in this theory in order to use the meta-programs; moreover there are some concepts for which we prove here the satisfaction lemmas manually, while for others we prove them on its theory.

```
declare arity_subset_fm [simp del] arity_ordinal_fm[simp del, arity] arity_transset_fm[simp del]
  FOL_arities[simp del]
```

⟨ML⟩

definition is_minimum' where

$$\begin{aligned} \text{is_minimum}'(M,R,X,u) \equiv & (M(u) \wedge u \in X \wedge (\forall v[M]. \exists a[M]. (v \in X \longrightarrow v \neq \\ & u \longrightarrow a \in R) \wedge \text{pair}(M, u, v, a))) \wedge \\ & (\exists x[M]. \\ & (M(x) \wedge x \in X \wedge (\forall v[M]. \exists a[M]. (v \in X \longrightarrow v \neq x \longrightarrow a \in R) \wedge \text{pair}(M, \\ & x, v, a))) \wedge \\ & (\forall y[M]. M(y) \wedge y \in X \wedge (\forall v[M]. \exists a[M]. (v \in X \longrightarrow v \neq y \longrightarrow a \in R) \wedge \\ & \text{pair}(M, y, v, a) \longrightarrow y = x)) \vee \\ & \neg (\exists x[M]. (M(x) \wedge x \in X \wedge (\forall v[M]. \exists a[M]. (v \in X \longrightarrow v \neq x \longrightarrow a \in R) \wedge \\ & \text{pair}(M, x, v, a))) \wedge \\ & (\forall y[M]. M(y) \wedge y \in X \wedge (\forall v[M]. \exists a[M]. (v \in X \longrightarrow v \neq y \longrightarrow a \in \\ & R) \wedge \text{pair}(M, y, v, a) \longrightarrow y = x)) \wedge \\ & \text{empty}(M, u) \end{aligned}$$

⟨ML⟩

lemma is_lambda_iff_sats[iff_sats]:

assumes is_F_iff_sats:

!!a0 a1 a2.

[[a0∈Aa; a1∈Aa; a2∈Aa]

==> is_F(a1, a0) <=> sats(Aa, is_F_fm, Cons(a0, Cons(a1, Cons(a2, env))))

shows

nth(A, env) = Ab ==>

nth(r, env) = ra ==>

A ∈ nat ==>

r ∈ nat ==>

env ∈ list(Aa) ==>

is_lambda(##Aa, Ab, is_F, ra) <=> Aa, env ⊨ lambda_fm(is_F_fm, A, r)

⟨proof⟩

lemma sats_is_wfrec_fm':

assumes MH_iff_sats:

$!!a0\ a1\ a2\ a3\ a4.$
 $[[a0 \in A; a1 \in A; a2 \in A; a3 \in A; a4 \in A]]$
 $\implies MH(a2, a1, a0) \longleftrightarrow \text{sats}(A, p, \text{Cons}(a0, \text{Cons}(a1, \text{Cons}(a2, \text{Cons}(a3, \text{Cons}(a4, \text{env}))))))$
shows
 $[[x \in \text{nat}; y \in \text{nat}; z \in \text{nat}; \text{env} \in \text{list}(A); 0 \in A]]$
 $\implies \text{sats}(A, \text{is_wfrec_fm}(p, x, y, z), \text{env}) \longleftrightarrow$
 $\text{is_wfrec}(\#\#A, MH, \text{nth}(x, \text{env}), \text{nth}(y, \text{env}), \text{nth}(z, \text{env}))$
 $\langle \text{proof} \rangle$

lemma $\text{is_wfrec_iff_sats}'[\text{iff_sats}]$:

assumes $MH\ \text{iff_sats}$:

$!!a0\ a1\ a2\ a3\ a4.$

$[[a0 \in Aa; a1 \in Aa; a2 \in Aa; a3 \in Aa; a4 \in Aa]]$

$\implies MH(a2, a1, a0) \longleftrightarrow \text{sats}(Aa, p, \text{Cons}(a0, \text{Cons}(a1, \text{Cons}(a2, \text{Cons}(a3, \text{Cons}(a4, \text{env}))))))$

$\text{nth}(x, \text{env}) = xx\ \text{nth}(y, \text{env}) = yy\ \text{nth}(z, \text{env}) = zz$

$x \in \text{nat}\ y \in \text{nat}\ z \in \text{nat}\ \text{env} \in \text{list}(Aa)\ 0 \in Aa$

shows

$\text{is_wfrec}(\#\#Aa, MH, xx, yy, zz) \longleftrightarrow Aa, \text{env} \models \text{is_wfrec_fm}(p, x, y, z)$

$\langle \text{proof} \rangle$

lemma $\text{is_wfrec_on_iff_sats}[\text{iff_sats}]$:

assumes $MH\ \text{iff_sats}$:

$!!a0\ a1\ a2\ a3\ a4.$

$[[a0 \in Aa; a1 \in Aa; a2 \in Aa; a3 \in Aa; a4 \in Aa]]$

$\implies MH(a2, a1, a0) \longleftrightarrow \text{sats}(Aa, p, \text{Cons}(a0, \text{Cons}(a1, \text{Cons}(a2, \text{Cons}(a3, \text{Cons}(a4, \text{env}))))))$

shows

$\text{nth}(x, \text{env}) = xx \implies$

$\text{nth}(y, \text{env}) = yy \implies$

$\text{nth}(z, \text{env}) = zz \implies$

$x \in \text{nat} \implies$

$y \in \text{nat} \implies$

$z \in \text{nat} \implies$

$\text{env} \in \text{list}(Aa) \implies$

$0 \in Aa \implies \text{is_wfrec_on}(\#\#Aa, MH, aa, xx, yy, zz) \longleftrightarrow Aa, \text{env} \models \text{is_wfrec_fm}(p, x, y, z)$

$\langle \text{proof} \rangle$

Formulas for particular replacement instances

Now we introduce some definitions used in the definition of check; which is defined by well-founded recursion using replacement in the recursive call.

definition

$rcheck :: i \Rightarrow i$ **where**

$rcheck(x) \equiv \text{Memrel}(\text{eclose}(\{x\}))^{\wedge+}$

$\langle ML \rangle$

definition

$PHcheck :: [i \Rightarrow o, i, i, i, i] \Rightarrow o$ **where**

$PHcheck(M, o, f, y, p) \equiv M(p) \wedge (\exists fy[M]. \text{fun_apply}(M, f, y, fy) \wedge \text{pair}(M, fy, o, p))$

$is_leq :: [i \Rightarrow o, i, i, i] \Rightarrow o$ **where**
 $is_leq(A, l, q, p) \equiv \exists qp[A]. (pair(A, q, p, qp) \wedge qp \in l)$

$\langle ML \rangle$

abbreviation

$fm_leq :: [i, i, i] \Rightarrow i$ ($\langle \cdot \leq _ \cdot \rangle$) **where**
 $fm_leq(A, l, B) \equiv is_leq_fm(l, A, B)$

5.1 Formulas used to prove some generic instances.

definition $\varrho_repl :: i \Rightarrow i$ **where**

$\varrho_repl(l) \equiv rsum(\{\langle 0, 1 \rangle, \langle 1, 0 \rangle\}, id(l), 2, 3, l)$

lemma $f_type : \{\langle 0, 1 \rangle, \langle 1, 0 \rangle\} \in 2 \rightarrow 3$
 $\langle proof \rangle$

hide_fact *Internalize.sum_type*

lemma $ren_type :$

assumes $l \in nat$

shows $\varrho_repl(l) : 2 +_{\omega} l \rightarrow 3 +_{\omega} l$

$\langle proof \rangle$

definition $Lambda_in_M_fm$ **where** $[simp]: Lambda_in_M_fm(\varphi, len) \equiv$

$\cdot(\exists \cdot pair_fm(1, 0, 2) \wedge$

$ren(\varphi) \text{ ‘ } (2 +_{\omega} len) \text{ ‘ } (3 +_{\omega} len) \text{ ‘ } \varrho_repl(len) \cdot) \wedge \cdot 0 \in len +_{\omega} 2 \cdot$

lemma $Lambda_in_M_fm_type[TC]: \varphi \in formula \Longrightarrow len \in nat \Longrightarrow Lambda_in_M_fm(\varphi, len) \in formula$

$\langle proof \rangle$

definition $\varrho_pair_repl :: i \Rightarrow i$ **where**

$\varrho_pair_repl(l) \equiv rsum(\{\langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 2, 3 \rangle\}, id(l), 3, 4, l)$

definition $LambdaPair_in_M_fm$ **where** $LambdaPair_in_M_fm(\varphi, len) \equiv$

$\cdot(\exists \cdot pair_fm(1, 0, 2) \wedge$

$ren((\exists (\exists \cdot fst(2) \text{ is } 0 \cdot \wedge \cdot snd(2) \text{ is } 1 \cdot \wedge ren(\varphi) \text{ ‘ } (3 +_{\omega} len) \text{ ‘ } (4 +_{\omega}$

$len) \text{ ‘ } \varrho_pair_repl(len) \cdot) \cdot) \text{ ‘ } (2 +_{\omega} len) \text{ ‘ }$

$(3 +_{\omega} len) \text{ ‘ }$

$\varrho_repl(len) \cdot) \wedge$

$\cdot 0 \in len +_{\omega} 2 \cdot$

lemma $f_type' : \{\langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 2, 3 \rangle\} \in 3 \rightarrow 4$

$\langle proof \rangle$

lemma $ren_type' :$

assumes $l \in nat$

shows $\varrho_pair_repl(l) : 3 +_{\omega} l \rightarrow 4 +_{\omega} l$

$\langle proof \rangle$

lemma *LambdaPair_in_M_fm_type*[TC]: $\varphi \in \text{formula} \implies \text{len} \in \text{nat} \implies \text{LambdaPair_in_M_fm}(\varphi, \text{len}) \in \text{formula}$
 ⟨proof⟩

5.2 The relation *frecrel*

definition

frecrelP :: $[i \Rightarrow o, i] \Rightarrow o$ **where**
frecrelP(M, xy) $\equiv (\exists x[M]. \exists y[M]. \text{pair}(M, x, y, xy) \wedge \text{is_frecrelR}(M, x, y))$

⟨ML⟩

definition

is_frecrel :: $[i \Rightarrow o, i, i] \Rightarrow o$ **where**
is_frecrel(M, A, r) $\equiv \exists A2[M]. \text{cartprod}(M, A, A, A2) \wedge \text{is_Collect}(M, A2, \text{frecrelP}(M), r)$

⟨ML⟩

definition

names_below :: $i \Rightarrow i \Rightarrow i$ **where**
names_below(P, x) $\equiv 2 \times \text{ecloseN}(x) \times \text{ecloseN}(x) \times P$

lemma *names_belowsD*:

assumes $x \in \text{names_below}(P, z)$

obtains $f\ n1\ n2\ p$ **where**

$x = \langle f, n1, n2, p \rangle$ $f \in 2$ $n1 \in \text{ecloseN}(z)$ $n2 \in \text{ecloseN}(z)$ $p \in P$

⟨proof⟩

⟨ML⟩

lemma *number2_iff* :

$(A)(c) \implies \text{number2}(A, c) \longleftrightarrow (\exists b[A]. \exists a[A]. \text{successor}(A, b, c) \wedge \text{successor}(A, a, b) \wedge \text{empty}(A, a))$

⟨proof⟩

⟨ML⟩

definition

is_tuple :: $[i \Rightarrow o, i, i, i, i, i] \Rightarrow o$ **where**
is_tuple($M, z, t1, t2, p, t$) $\equiv \exists t1t2p[M]. \exists t2p[M]. \text{pair}(M, t2, p, t2p) \wedge \text{pair}(M, t1, t2p, t1t2p)$
 \wedge

$\text{pair}(M, z, t1t2p, t)$

⟨ML⟩

5.3 Definition of Forces

5.3.1 Definition of forces for equality and membership

$p \Vdash \tau = \theta$ if for every $q \leq p$ both $q \Vdash \sigma \in \tau$ and $q \Vdash \sigma \in \theta$ hold for all $\sigma \in \text{dom}(\tau) \cup \text{dom}(\theta)$.

definition

$eq_case :: [i, i, i, i, i, i] \Rightarrow o$ **where**
 $eq_case(\tau, \vartheta, p, P, leq, f) \equiv \forall \sigma. \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \longrightarrow$
 $(\forall q. q \in P \wedge \langle q, p \rangle \in leq \longrightarrow (f' \langle 1, \sigma, \tau, q \rangle = 1 \longleftrightarrow f' \langle 1, \sigma, \vartheta, q \rangle = 1))$

$\langle ML \rangle$

$p \Vdash \tau \in \theta$ if for every $v \leq p$ there exist q, r , and σ such that $v \leq q, q \leq r$, $\langle \sigma, r \rangle \in \tau$, and $q \Vdash \pi = \sigma$.

definition

$mem_case :: [i, i, i, i, i, i] \Rightarrow o$ **where**
 $mem_case(\tau, \vartheta, p, P, leq, f) \equiv \forall v \in P. \langle v, p \rangle \in leq \longrightarrow$
 $(\exists q. \exists \sigma. \exists r. r \in P \wedge q \in P \wedge \langle q, v \rangle \in leq \wedge \langle \sigma, r \rangle \in \vartheta \wedge \langle q, r \rangle \in leq \wedge f' \langle 0, \tau, \sigma, q \rangle = 1)$

$\langle ML \rangle$

lemma $arity_eq_case_fm[arity]$:

assumes

$n1 \in \text{nat } n2 \in \text{nat } p \in \text{nat } P \in \text{nat } leq \in \text{nat } f \in \text{nat}$

shows

$arity(eq_case_fm(n1, n2, p, P, leq, f)) =$
 $\text{succ}(n1) \cup \text{succ}(n2) \cup \text{succ}(p) \cup \text{succ}(P) \cup \text{succ}(leq) \cup \text{succ}(f)$
 $\langle proof \rangle$

$\langle ML \rangle$

lemma $arity_mem_case_fm[arity]$:

assumes

$n1 \in \text{nat } n2 \in \text{nat } p \in \text{nat } P \in \text{nat } leq \in \text{nat } f \in \text{nat}$

shows

$arity(mem_case_fm(n1, n2, p, P, leq, f)) =$
 $\text{succ}(n1) \cup \text{succ}(n2) \cup \text{succ}(p) \cup \text{succ}(P) \cup \text{succ}(leq) \cup \text{succ}(f)$
 $\langle proof \rangle$

definition

$Hfrc :: [i, i, i, i] \Rightarrow o$ **where**
 $Hfrc(P, leq, fnnc, f) \equiv \exists ft. \exists \tau. \exists \vartheta. \exists p. p \in P \wedge fnnc = \langle ft, \tau, \vartheta, p \rangle \wedge$
 $(ft = 0 \wedge eq_case(\tau, \vartheta, p, P, leq, f)$
 $\vee ft = 1 \wedge mem_case(\tau, \vartheta, p, P, leq, f))$

$\langle ML \rangle$

definition

$is_Hfrc_at :: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$ **where**

$$\begin{aligned} is_Hfrc_at(M,P,leq,fnnc,f,b) &\equiv \\ &(\text{empty}(M,b) \wedge \neg is_Hfrc(M,P,leq,fnnc,f)) \\ &\vee (\text{number1}(M,b) \wedge is_Hfrc(M,P,leq,fnnc,f)) \end{aligned}$$

$\langle ML \rangle$

lemma *arity_Hfrc_fm*[arity] :

assumes

$$P \in nat \ leq \in nat \ fnnc \in nat \ f \in nat$$

shows

$$\text{arity}(Hfrc_fm(P,leq,fnnc,f)) = \text{succ}(P) \cup \text{succ}(leq) \cup \text{succ}(fnnc) \cup \text{succ}(f)$$

$\langle proof \rangle$

$\langle ML \rangle$

5.3.2 The well-founded relation *forcere*_l

definition

*forcere*_l :: $i \Rightarrow i \Rightarrow i$ **where**

$$\text{forcere}_l(P,x) \equiv \text{frecrel}(\text{names_below}(P,x)) \hat{+}$$

definition

*is_forcere*_l :: $[i \Rightarrow o, i, i, i] \Rightarrow o$ **where**

$$\begin{aligned} is_forcere_l(M,P,x,z) &\equiv \exists r[M]. \exists nb[M]. \text{tran_closure}(M,r,z) \wedge \\ &(\text{is_names_below}(M,P,x,nb) \wedge is_frecrel(M,nb,r)) \end{aligned}$$

$\langle ML \rangle$

5.4 *frc_at*, forcing for atomic formulas

definition

frc_at :: $[i, i, i] \Rightarrow i$ **where**

$$\begin{aligned} frc_at(P,leq,fnnc) &\equiv wfrec(\text{frecrel}(\text{names_below}(P,fnnc)),fnnc, \\ &\lambda x f. \text{bool_of_o}(Hfrc(P,leq,x,f))) \end{aligned}$$

— The relational form is defined manually because it uses *wfrec*.

definition

is_frc_at :: $[i \Rightarrow o, i, i, i, i] \Rightarrow o$ **where**

$$\begin{aligned} is_frc_at(M,P,leq,x,z) &\equiv \exists r[M]. is_forcere_l(M,P,x,r) \wedge \\ &is_wfrec(M,is_Hfrc_at(M,P,leq),r,x,z) \end{aligned}$$

definition

frc_at_fm :: $[i, i, i, i] \Rightarrow i$ **where**

$$\begin{aligned} frc_at_fm(p,l,x,z) &\equiv \text{Exists}(\text{And}(is_forcere_fm(\text{succ}(p),\text{succ}(x),0), \\ &is_wfrec_fm(Hfrc_at_fm(6+\omega p,6+\omega l,2,1,0),0,\text{succ}(x),\text{succ}(z)))))) \end{aligned}$$

lemma *frc_at_fm_type* [TC] :

$$\llbracket p \in nat; l \in nat; x \in nat; z \in nat \rrbracket \implies frc_at_fm(p,l,x,z) \in formula$$

$\langle proof \rangle$

lemma *arity_frc_at_fm*[arity] :

assumes $p \in \text{nat } l \in \text{nat } x \in \text{nat } z \in \text{nat}$
shows $\text{arity}(\text{frc_at_fm}(p,l,x,z)) = \text{succ}(p) \cup \text{succ}(l) \cup \text{succ}(x) \cup \text{succ}(z)$
 $\langle \text{proof} \rangle$

lemma sats_frc_at_fm :

assumes
 $p \in \text{nat } l \in \text{nat } i \in \text{nat } j \in \text{nat } \text{env} \in \text{list}(A) \ i < \text{length}(\text{env}) \ j < \text{length}(\text{env})$
shows
 $(A, \text{env} \models \text{frc_at_fm}(p,l,i,j)) \longleftrightarrow$
 $\text{is_frc_at}(\#\#A, \text{nth}(p, \text{env}), \text{nth}(l, \text{env}), \text{nth}(i, \text{env}), \text{nth}(j, \text{env}))$
 $\langle \text{proof} \rangle$

lemma $\text{frc_at_fm_iff_sats}$:

assumes $\text{nth}(i, \text{env}) = w \ \text{nth}(j, \text{env}) = x \ \text{nth}(k, \text{env}) = y \ \text{nth}(l, \text{env}) = z$
 $i \in \text{nat } j \in \text{nat } k \in \text{nat } l \in \text{nat } \text{env} \in \text{list}(A) \ k < \text{length}(\text{env}) \ l < \text{length}(\text{env})$
shows $\text{is_frc_at}(\#\#A, w, x, y, z) \longleftrightarrow (A, \text{env} \models \text{frc_at_fm}(i,j,k,l))$
 $\langle \text{proof} \rangle$

declare $\text{frc_at_fm_iff_sats}$ [iff_sats]

definition

$\text{forces_eq}' :: [i,i,i,i,i] \Rightarrow o$ **where**
 $\text{forces_eq}'(P,l,p,t1,t2) \equiv \text{frc_at}(P,l, \langle 0, t1, t2, p \rangle) = 1$

definition

$\text{forces_mem}' :: [i,i,i,i,i] \Rightarrow o$ **where**
 $\text{forces_mem}'(P,l,p,t1,t2) \equiv \text{frc_at}(P,l, \langle 1, t1, t2, p \rangle) = 1$

definition

$\text{forces_neq}' :: [i,i,i,i,i] \Rightarrow o$ **where**
 $\text{forces_neq}'(P,l,p,t1,t2) \equiv \neg (\exists q \in P. \langle q, p \rangle \in l \wedge \text{forces_eq}'(P,l,q,t1,t2))$

definition

$\text{forces_nmem}' :: [i,i,i,i,i] \Rightarrow o$ **where**
 $\text{forces_nmem}'(P,l,p,t1,t2) \equiv \neg (\exists q \in P. \langle q, p \rangle \in l \wedge \text{forces_mem}'(P,l,q,t1,t2))$

— The following definitions are explicitly defined to avoid the expansion of concepts.

definition

$\text{is_forces_eq}' :: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$ **where**
 $\text{is_forces_eq}'(M, P, l, p, t1, t2) \equiv \exists o[M]. \exists z[M]. \exists t[M]. \text{number1}(M, o) \wedge \text{empty}(M, z)$
 \wedge

$\text{is_tuple}(M, z, t1, t2, p, t) \wedge \text{is_frc_at}(M, P, l, t, o)$

definition

$\text{is_forces_mem}' :: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$ **where**
 $\text{is_forces_mem}'(M, P, l, p, t1, t2) \equiv \exists o[M]. \exists t[M]. \text{number1}(M, o) \wedge$
 $\text{is_tuple}(M, o, t1, t2, p, t) \wedge \text{is_frc_at}(M, P, l, t, o)$

definition

$is_forces_neq' :: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$ **where**
 $is_forces_neq'(M, P, l, p, t1, t2) \equiv$
 $\neg (\exists q[M]. q \in P \wedge (\exists qp[M]. pair(M, q, p, qp) \wedge qp \in l \wedge is_forces_eq'(M, P, l, q, t1, t2)))$

definition

$is_forces_nmem' :: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$ **where**
 $is_forces_nmem'(M, P, l, p, t1, t2) \equiv$
 $\neg (\exists q[M]. \exists qp[M]. q \in P \wedge pair(M, q, p, qp) \wedge qp \in l \wedge is_forces_mem'(M, P, l, q, t1, t2))$

$\langle ML \rangle$

context

notes $Un_assoc[simp]$ $Un_transposition_aux2[simp]$

begin

$\langle ML \rangle$

end

5.5 Forcing for general formulas

definition

$ren_forces_nand :: i \Rightarrow i$ **where**
 $ren_forces_nand(\varphi) \equiv Exists(And(Equal(0,1), iterates(\lambda p. incr_bv(p)'1, 2, \varphi)))$

lemma $ren_forces_nand_type[TC]$:

$\varphi \in formula \Longrightarrow ren_forces_nand(\varphi) \in formula$
 $\langle proof \rangle$

lemma $arity_ren_forces_nand$:

assumes $\varphi \in formula$
shows $arity(ren_forces_nand(\varphi)) \leq succ(arity(\varphi))$
 $\langle proof \rangle$

lemma $sats_ren_forces_nand$:

$[q, P, leq, o, p] @ env \in list(M) \Longrightarrow \varphi \in formula \Longrightarrow$
 $(M, [q, p, P, leq, o] @ env \models ren_forces_nand(\varphi)) \longleftrightarrow (M, [q, P, leq, o] @ env \models$
 $\varphi)$
 $\langle proof \rangle$

definition

$ren_forces_forall :: i \Rightarrow i$ **where**
 $ren_forces_forall(\varphi) \equiv$
 $Exists(Exists(Exists(Exists(Exists($
 $And(Equal(0,6), And(Equal(1,7), And(Equal(2,8), And(Equal(3,9),$
 $And(Equal(4,5), iterates(\lambda p. incr_bv(p)'5, 5, \varphi))))))))))$

lemma $arity_ren_forces_all$:

assumes $\varphi \in formula$
shows $arity(ren_forces_forall(\varphi)) = 5 \cup arity(\varphi)$

$\langle \text{proof} \rangle$

lemma *ren_forces_forall_type*[TC] :
 $\varphi \in \text{formula} \implies \text{ren_forces_forall}(\varphi) \in \text{formula}$
 $\langle \text{proof} \rangle$

lemma *sats_ren_forces_forall* :
 $[x, P, \text{leq}, o, p] @ \text{env} \in \text{list}(M) \implies \varphi \in \text{formula} \implies$
 $(M, [x, p, P, \text{leq}, o] @ \text{env} \models \text{ren_forces_forall}(\varphi)) \longleftrightarrow (M, [p, P, \text{leq}, o, x] @ \text{env}$
 $\models \varphi)$
 $\langle \text{proof} \rangle$

5.5.1 The primitive recursion

consts *forces'* :: $i \Rightarrow i$

primrec

$\text{forces}'(\text{Member}(x, y)) = \text{forces_mem_fm}(1, 2, 0, x +_{\omega} 4, y +_{\omega} 4)$
 $\text{forces}'(\text{Equal}(x, y)) = \text{forces_eq_fm}(1, 2, 0, x +_{\omega} 4, y +_{\omega} 4)$
 $\text{forces}'(\text{Nand}(p, q)) =$
 $\text{Neg}(\text{Exists}(\text{And}(\text{Member}(0, 2), \text{And}(\text{is_leq_fm}(3, 0, 1), \text{And}(\text{ren_forces_nand}(\text{forces}'(p)),$
 $\text{ren_forces_nand}(\text{forces}'(q)))))))$
 $\text{forces}'(\text{Forall}(p)) = \text{Forall}(\text{ren_forces_forall}(\text{forces}'(p)))$

definition

forces :: $i \Rightarrow i$ **where**
 $\text{forces}(\varphi) \equiv \text{And}(\text{Member}(0, 1), \text{forces}'(\varphi))$

lemma *forces'_type* [TC]: $\varphi \in \text{formula} \implies \text{forces}'(\varphi) \in \text{formula}$
 $\langle \text{proof} \rangle$

lemma *forces_type*[TC] : $\varphi \in \text{formula} \implies \text{forces}(\varphi) \in \text{formula}$
 $\langle \text{proof} \rangle$

5.6 The arity of forces

lemma *arity_forces_at*:

assumes $x \in \text{nat } y \in \text{nat}$
shows $\text{arity}(\text{forces}(\text{Member}(x, y))) = (\text{succ}(x) \cup \text{succ}(y)) +_{\omega} 4$
 $\text{arity}(\text{forces}(\text{Equal}(x, y))) = (\text{succ}(x) \cup \text{succ}(y)) +_{\omega} 4$
 $\langle \text{proof} \rangle$

lemma *arity_forces'*:

assumes $\varphi \in \text{formula}$
shows $\text{arity}(\text{forces}'(\varphi)) \leq \text{arity}(\varphi) +_{\omega} 4$
 $\langle \text{proof} \rangle$

lemma *arity_forces* :

assumes $\varphi \in \text{formula}$
shows $\text{arity}(\text{forces}(\varphi)) \leq 4 +_{\omega} \text{arity}(\varphi)$

$\langle ML \rangle$

definition *omap_wfrec_body* **where**

$omap_wfrec_body(A,r) \equiv (\cdot \exists \cdot image_fm(2, 0, 1) \wedge pred_set_fm(9+\omega A, 3, 9+\omega r, 0) \cdot \cdot)$

lemma *type_omap_wfrec_body_fm* : $A \in nat \implies r \in nat \implies omap_wfrec_body(A,r) \in formula$
 $\langle proof \rangle$

lemma *arity_omap_wfrec_aux* : $A \in nat \implies r \in nat \implies arity(omap_wfrec_body(A,r))$
 $= (9+\omega A) \cup (9+\omega r)$
 $\langle proof \rangle$

lemma *arity_omap_wfrec* : $A \in nat \implies r \in nat \implies$
 $arity(is_wfrec_fm(omap_wfrec_body(A,r), r+\omega 3, 1, 0)) = (4+\omega A) \cup (4+\omega r)$
 $\langle proof \rangle$

lemma *arity_isordermap* : $A \in nat \implies r \in nat \implies d \in nat \implies$
 $arity(is_ordermap_fm(A,r,d)) = succ(d) \cup (succ(A) \cup succ(r))$
 $\langle proof \rangle$

lemma *arity_is_ordertype* : $A \in nat \implies r \in nat \implies d \in nat \implies$
 $arity(is_ordertype_fm(A,r,d)) = succ(d) \cup (succ(A) \cup succ(r))$
 $\langle proof \rangle$

lemma *arity_is_order_body* : $arity(is_order_body_fm(0,1)) = 2$
 $\langle proof \rangle$

definition *H_order_pred* **where**

$H_order_pred(A,r) \equiv \lambda x f . f \text{ `` } Order.pred(A, x, r)$

$\langle ML \rangle$

definition *order_pred_wfrec_body* **where**

$order_pred_wfrec_body(M,A,r,z,x) \equiv \exists y[M].$

$pair(M, x, y, z) \wedge$

$(\exists f[M].$

$(\forall z[M].$

$z \in f \longleftrightarrow$

$(\exists xa[M].$

$\exists y[M].$

$\exists xaa[M].$

$\exists sx[M].$

$\exists r_sx[M].$

$\exists f_r_sx[M].$

$pair(M, xa, y, z) \wedge$

$pair(M, xa, x, xaa) \wedge$

$upair(M, xa, xa, sx) \wedge$

$$\begin{aligned}
& \text{pre_image}(M, r, sx, r_sx) \wedge \\
& \text{restriction}(M, f, r_sx, f_r_sx) \wedge \\
& xaa \in r \wedge (\exists a[M]. \text{image}(M, f_r_sx, a, y) \wedge \\
& \text{pred_set}(M, A, xa, r, a))) \wedge \\
& (\exists a[M]. \text{image}(M, f, a, y) \wedge \text{pred_set}(M, A, x, r, a))
\end{aligned}$$

$\langle ML \rangle$

definition *ordtype_replacement_fm* **where** *ordtype_replacement_fm* $\equiv (\cdot \exists \cdot \text{is_order_body_fm}(1, 0) \wedge \cdot \langle 1, 0 \rangle \text{ is } 2 \dots)$

definition *wfrec_ordertype_fm* **where** *wfrec_ordertype_fm* $\equiv \text{order_pred_wfrec_body_fm}(3, 2, 1, 0)$

definition *replacement_is_aleph_fm* **where** *replacement_is_aleph_fm* $\equiv \cdot 0 \text{ is ordinal} \cdot \wedge \cdot \aleph(0) \text{ is } 1 \cdot$

definition

funspace_succ_rep_intf **where**
funspace_succ_rep_intf $\equiv \lambda p z n. \exists f b. p = \langle f, b \rangle \ \& \ z = \{ \text{cons}(\langle n, b \rangle, f) \}$

$\langle ML \rangle$

definition *wfrec_Hfrc_at_fm* **where** *wfrec_Hfrc_at_fm* $\equiv (\cdot \exists \cdot \text{pair_fm}(1, 0, 2) \wedge \text{is_wfrec_fm}(\text{Hfrc_at_fm}(8, 9, 2, 1, 0), 5, 1, 0) \dots)$

definition *list_repl1_intf_fm* **where** *list_repl1_intf_fm* $\equiv (\cdot \exists \cdot \text{pair_fm}(1, 0, 2) \wedge \text{is_wfrec_fm}(\text{iterates_MH_fm}(\text{list_functor_fm}(13, 1, 0), 10, 2, 1, 0), 3, 1, 0) \dots)$

definition *list_repl2_intf_fm* **where** *list_repl2_intf_fm* $\equiv \cdot 0 \in 4 \cdot \wedge \text{is_iterates_fm}(\text{list_functor_fm}(13, 1, 0), 3, 0, 1) \cdot$

definition *formula_repl2_intf_fm* **where** *formula_repl2_intf_fm* $\equiv \cdot 0 \in 3 \cdot \wedge \text{is_iterates_fm}(\text{formula_functor_fm}(1, 0), 2, 0, 1) \cdot$

definition *eclose_abs_fm* **where** *eclose_abs_fm* $\equiv \cdot 0 \in 3 \cdot \wedge \text{is_iterates_fm}(\cdot \cup 1 \text{ is } 0 \cdot, 2, 0, 1) \cdot$

definition *powapply_repl_fm* **where** *powapply_repl_fm* $\equiv \text{is_Powapply_fm}(2, 0, 1)$

definition *wfrec_rank_fm* **where** *wfrec_rank_fm* $\equiv (\cdot \exists \cdot \text{pair_fm}(1, 0, 2) \wedge \text{is_wfrec_fm}(\text{is_Hrank_fm}(2, 1, 0), 3, 1, 0) \dots)$

definition *transrec_VFrom_fm* **where** *transrec_VFrom_fm* $\equiv (\cdot \exists \cdot \text{pair_fm}(1, 0, 2) \wedge \text{is_wfrec_fm}(\text{is_HVfrom_fm}(8, 2, 1, 0), 4, 1, 0) \dots)$

definition *wfrec_Hcheck_fm* **where** *wfrec_Hcheck_fm* $\equiv (\cdot \exists \cdot \text{pair_fm}(1, 0, 2) \wedge \text{is_wfrec_fm}(\text{is_Hcheck_fm}(8, 2, 1, 0), 4, 1, 0) \dots)$

definition *repl_PHcheck_fm* **where** *repl_PHcheck_fm* $\equiv \text{PHcheck_fm}(2, 3, 0, 1)$

definition *tl_repl_intf_fm* **where** *tl_repl_intf_fm* $\equiv (\cdot \exists \cdot \text{pair_fm}(1, 0, 2) \wedge \text{is_wfrec_fm}(\text{iterates_MH_fm}(\text{tl_fm}(1, 0), 9, 2, 1, 0), 3, 1, 0) \dots)$

definition *formula_repl1_intf_fm* **where** *formula_repl1_intf_fm* $\equiv (\cdot \exists \cdot \text{pair_fm}(1, 0, 2) \wedge \text{is_wfrec_fm}(\text{iterates_MH_fm}(\text{formula_functor_fm}(1, 0), 9, 2, 1, 0), 3, 1, 0) \dots)$

definition *eclose_closed_fm* **where** *eclose_closed_fm* $\equiv (\cdot \exists \cdot \text{pair_fm}(1, 0, 2) \wedge \text{is_wfrec_fm}(\text{iterates_MH_fm}(\cdot \cup 1 \text{ is } 0 \cdot, 9, 2, 1, 0), 3, 1, 0) \dots)$

definition *replacement_assm* **where**

$replacement_assm(M, env, \varphi) \equiv \varphi \in formula \longrightarrow env \in list(M) \longrightarrow$
 $arity(\varphi) \leq 2 +_{\omega} length(env) \longrightarrow$
 $strong_replacement(\#\#M, \lambda x y. (M, [x, y]@env \models \varphi))$

definition *ground_replacement_assm* **where**

$ground_replacement_assm(M, env, \varphi) \equiv replacement_assm(M, env, ground_repl_fm(\varphi))$

end

6 The ZFC axioms, internalized

theory *Internal_ZFC_Axioms*

imports

Fm_Definitions

begin

schematic_goal *ZF_union_auto*:

$Union_ax(\#\#A) \longleftrightarrow (A, [] \models ?zfunion)$
 $\langle proof \rangle$

$\langle ML \rangle$

notation *ZF_union_fm* ($\langle \cdot Union\ Ax \cdot \rangle$)

schematic_goal *ZF_power_auto*:

$power_ax(\#\#A) \longleftrightarrow (A, [] \models ?zfpow)$
 $\langle proof \rangle$

$\langle ML \rangle$

notation *ZF_power_fm* ($\langle \cdot Powerset\ Ax \cdot \rangle$)

schematic_goal *ZF_pairing_auto*:

$upair_ax(\#\#A) \longleftrightarrow (A, [] \models ?zfpair)$
 $\langle proof \rangle$

$\langle ML \rangle$

notation *ZF_pairing_fm* ($\langle \cdot Pairing \cdot \rangle$)

schematic_goal *ZF_foundation_auto*:

$foundation_ax(\#\#A) \longleftrightarrow (A, [] \models ?zffound)$
 $\langle proof \rangle$

$\langle ML \rangle$

notation *ZF_foundation_fm* ($\langle \cdot Foundation \cdot \rangle$)

schematic_goal *ZF_extensionality_auto*:

$extensionality(\#\#A) \longleftrightarrow (A, [] \models ?zfect)$
 $\langle proof \rangle$

$\langle ML \rangle$
notation $ZF_extensionality_fm$ ($\langle \cdot Extensionality \cdot \rangle$)

schematic_goal $ZF_infinity_auto$:
 $infinity_ax(\#\#A) \longleftrightarrow (A, [] \models (? \varphi(i,j,h)))$
 $\langle proof \rangle$

$\langle ML \rangle$
notation $ZF_infinity_fm$ ($\langle \cdot Infinity \cdot \rangle$)

schematic_goal ZF_choice_auto :
 $choice_ax(\#\#A) \longleftrightarrow (A, [] \models (? \varphi(i,j,h)))$
 $\langle proof \rangle$

$\langle ML \rangle$
notation ZF_choice_fm ($\langle \cdot AC \cdot \rangle$)

lemmas $ZFC_fm_defs = ZF_extensionality_fm_def ZF_foundation_fm_def ZF_pairing_fm_def$
 $ZF_union_fm_def ZF_infinity_fm_def ZF_power_fm_def ZF_choice_fm_def$

lemmas $ZFC_fm_sats = ZF_extensionality_auto ZF_foundation_auto ZF_pairing_auto$
 $ZF_union_auto ZF_infinity_auto ZF_power_auto ZF_choice_auto$

definition
 $ZF_fin :: i$ **where**
 $ZF_fin \equiv \{ \cdot Extensionality \cdot, \cdot Foundation \cdot, \cdot Pairing \cdot,$
 $\cdot Union\ Ax \cdot, \cdot Infinity \cdot, \cdot Powerset\ Ax \cdot \}$

6.1 The Axiom of Separation, internalized

lemma $iterates_Forall_type$ $[TC]$:
 $\llbracket n \in nat; p \in formula \rrbracket \implies Forall^n(p) \in formula$
 $\langle proof \rangle$

lemma $last_init_eq$:
assumes $l \in list(A)$ $length(l) = succ(n)$
shows $\exists a \in A. \exists l' \in list(A). l = l' @ [a]$
 $\langle proof \rangle$

lemma $take_drop_eq$:
assumes $l \in list(M)$
shows $\bigwedge n. n < succ(length(l)) \implies l = take(n,l) @ drop(n,l)$
 $\langle proof \rangle$

lemma $list_split$:
assumes $n \leq succ(length(rest))$ $rest \in list(M)$
shows $\exists re \in list(M). \exists st \in list(M). rest = re @ st \wedge length(re) = pred(n)$
 $\langle proof \rangle$

lemma *sats_nForall*:

assumes

$\varphi \in \text{formula}$

shows

$n \in \text{nat} \implies ms \in \text{list}(M) \implies$

$(M, ms \models (\text{Forall}^n(\varphi))) \longleftrightarrow$

$(\forall rest \in \text{list}(M). \text{length}(rest) = n \longrightarrow M, rest @ ms \models \varphi)$

$\langle \text{proof} \rangle$

definition

sep_body_fm :: $i \Rightarrow i$ **where**

$\text{sep_body_fm}(p) \equiv (\cdot \forall (\exists (\cdot \forall \cdot 0 \in 1 \leftrightarrow \cdot 0 \in 2 \wedge \text{incr_bv1}^2(p) \dots) \cdot))$

lemma *sep_body_fm_type* [TC]: $p \in \text{formula} \implies \text{sep_body_fm}(p) \in \text{formula}$

$\langle \text{proof} \rangle$

lemma *sats_sep_body_fm*:

assumes

$\varphi \in \text{formula} \ ms \in \text{list}(M) \ rest \in \text{list}(M)$

shows

$(M, rest @ ms \models \text{sep_body_fm}(\varphi)) \longleftrightarrow$

$\text{separation}(\#\#M, \lambda x. M, [x] @ rest @ ms \models \varphi)$

$\langle \text{proof} \rangle$

definition

ZF_separation_fm :: $i \Rightarrow i$ ($\langle \cdot \text{Separation}'(_) \cdot \rangle$) **where**

$\text{ZF_separation_fm}(p) \equiv \text{Forall}^{\wedge}(\text{pred}(\text{arity}(p)))(\text{sep_body_fm}(p))$

lemma *ZF_separation_fm_type* [TC]: $p \in \text{formula} \implies \text{ZF_separation_fm}(p) \in \text{formula}$

$\langle \text{proof} \rangle$

lemma *sats_ZF_separation_fm_iff*:

assumes

$\varphi \in \text{formula}$

shows

$(M, [] \models \cdot \text{Separation}(\varphi) \cdot)$

\longleftrightarrow

$(\forall env \in \text{list}(M). \text{arity}(\varphi) \leq 1 +_{\omega} \text{length}(env) \longrightarrow$

$\text{separation}(\#\#M, \lambda x. M, [x] @ env \models \varphi))$

$\langle \text{proof} \rangle$

6.2 The Axiom of Replacement, internalized

schematic_goal *sats_univalent_fm_auto*:

assumes

$Q_iff_sats: \bigwedge x y z. x \in A \implies y \in A \implies z \in A \implies$

$Q(x, z) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, env)))) \models Q1_fm)$

$\bigwedge x y z. x \in A \implies y \in A \implies z \in A \implies$
 $Q(x,y) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, \text{env})))) \models Q2_fm$

and

asms: $\text{nth}(i, \text{env}) = B \ i \in \text{nat} \ \text{env} \in \text{list}(A)$

shows

$\text{univalent}(\#\#A, B, Q) \longleftrightarrow A, \text{env} \models ?\text{ufm}(i)$

<proof>

<ML>

lemma *univalent_fm_type* [TC]: $q1 \in \text{formula} \implies q2 \in \text{formula} \implies i \in \text{nat} \implies$
 $\text{univalent_fm}(q2, q1, i) \in \text{formula}$

<proof>

lemma *sats_univalent_fm* :

assumes

$Q_iff_sats: \bigwedge x y z. x \in A \implies y \in A \implies z \in A \implies$
 $Q(x,z) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, \text{env})))) \models Q1_fm$

$\bigwedge x y z. x \in A \implies y \in A \implies z \in A \implies$
 $Q(x,y) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, \text{env})))) \models Q2_fm$

and

asms: $\text{nth}(i, \text{env}) = B \ i \in \text{nat} \ \text{env} \in \text{list}(A)$

shows

$(A, \text{env} \models \text{univalent_fm}(Q1_fm, Q2_fm, i)) \longleftrightarrow \text{univalent}(\#\#A, B, Q)$

<proof>

definition

swap_vars :: $i \Rightarrow i$ **where**

swap_vars(φ) \equiv

$\text{Exists}(\text{Exists}(\text{And}(\text{Equal}(0,3), \text{And}(\text{Equal}(1,2), \text{iterates}(\lambda p. \text{incr_bv}(p)'2, 2, \varphi))))))$

lemma *swap_vars_type*[TC] :

$\varphi \in \text{formula} \implies \text{swap_vars}(\varphi) \in \text{formula}$

<proof>

lemma *sats_swap_vars* :

$[x,y] @ \text{env} \in \text{list}(M) \implies \varphi \in \text{formula} \implies$

$(M, [x,y] @ \text{env} \models \text{swap_vars}(\varphi)) \longleftrightarrow M, [y,x] @ \text{env} \models \varphi$

<proof>

definition

univalent_Q1 :: $i \Rightarrow i$ **where**

univalent_Q1(φ) $\equiv \text{incr_bv1}(\text{swap_vars}(\varphi))$

definition

univalent_Q2 :: $i \Rightarrow i$ **where**

univalent_Q2(φ) $\equiv \text{incr_bv}(\text{swap_vars}(\varphi))'0$

lemma *univalent_Qs_type* [TC]:
assumes $\varphi \in \text{formula}$
shows $\text{univalent_Q1}(\varphi) \in \text{formula}$ $\text{univalent_Q2}(\varphi) \in \text{formula}$
 $\langle \text{proof} \rangle$

lemma *sats_univalent_fm_assm*:
assumes
 $x \in A$ $y \in A$ $z \in A$ $\text{env} \in \text{list}(A)$ $\varphi \in \text{formula}$
shows
 $(A, ([x,z] @ \text{env}) \models \varphi) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, \text{env})))) \models (\text{univalent_Q1}(\varphi))$
 $(A, ([x,y] @ \text{env}) \models \varphi) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, \text{env})))) \models (\text{univalent_Q2}(\varphi))$
 $\langle \text{proof} \rangle$

definition
 $\text{rep_body_fm} :: i \Rightarrow i$ **where**
 $\text{rep_body_fm}(p) \equiv \text{Forall}(\text{Implies}(\text{univalent_fm}(\text{univalent_Q1}(\text{incr_bv}(p)^2), \text{univalent_Q2}(\text{incr_bv}(p)^2), 0), \text{Exists}(\text{Forall}(\text{Iff}(\text{Member}(0, 1), \text{Exists}(\text{And}(\text{Member}(0, 3), \text{incr_bv}(\text{incr_bv}(p)^2)^2))))))$

lemma *rep_body_fm_type* [TC]: $p \in \text{formula} \Longrightarrow \text{rep_body_fm}(p) \in \text{formula}$
 $\langle \text{proof} \rangle$

lemmas *ZF_replacement_simps* = *formula_add_params1*[of φ 2 M $[_, _]$]
sats_incr_bv_iff[of $_$ M $[_]$] — simplifies iterates of $\lambda x. \text{incr_bv}(x)^2$
sats_incr_bv_iff[of $_$ M $[_, _]$] — simplifies $\lambda x. \text{incr_bv}(x)^2$
sats_incr_bv1_iff[of $_$ M] *sats_swap_vars* **for** φ M

lemma *sats_rep_body_fm*:
assumes
 $\varphi \in \text{formula}$ $ms \in \text{list}(M)$ $rest \in \text{list}(M)$
shows
 $(M, rest @ ms \models \text{rep_body_fm}(\varphi)) \longleftrightarrow$
 $\text{strong_replacement}(\#\#M, \lambda x y. M, [x, y] @ rest @ ms \models \varphi)$
 $\langle \text{proof} \rangle$

definition
 $\text{ZF_replacement_fm} :: i \Rightarrow i$ (\cdot *Replacement'* $(_)$ \cdot) **where**
 $\text{ZF_replacement_fm}(p) \equiv \text{Forall}(\text{pred}(\text{pred}(\text{arity}(p))))(\text{rep_body_fm}(p))$

lemma *ZF_replacement_fm_type* [TC]: $p \in \text{formula} \Longrightarrow \text{ZF_replacement_fm}(p) \in \text{formula}$
 $\langle \text{proof} \rangle$

lemma *sats_ZF_replacement_fm_iff*:
assumes
 $\varphi \in \text{formula}$
shows
 $(M, [] \models \cdot \text{Replacement}(\varphi) \cdot) \longleftrightarrow (\forall \text{env. replacement_assm}(M, \text{env}, \varphi))$

$\langle proof \rangle$

definition

$ZF_schemes :: i$ **where**
 $ZF_schemes \equiv \{ \cdot Separation(p) \cdot . p \in formula \} \cup \{ \cdot Replacement(p) \cdot . p \in formula \}$

lemma $Un_subset_formula$ [TC]: $A \subseteq formula \wedge B \subseteq formula \implies A \cup B \subseteq formula$
 $\langle proof \rangle$

lemma $ZF_schemes_subset_formula$ [TC]: $ZF_schemes \subseteq formula$
 $\langle proof \rangle$

lemma $ZF_fin_subset_formula$ [TC]: $ZF_fin \subseteq formula$
 $\langle proof \rangle$

definition

$ZF :: i$ **where**
 $ZF \equiv ZF_schemes \cup ZF_fin$

lemma $ZF_subset_formula$ [TC]: $ZF \subseteq formula$
 $\langle proof \rangle$

definition

$ZFC :: i$ **where**
 $ZFC \equiv ZF \cup \{ \cdot AC \cdot \}$

definition

$ZF_minus_P :: i$ **where**
 $ZF_minus_P \equiv ZF - \{ \cdot Powerset Ax \cdot \}$

definition

$Zermelo_fms :: i$ ($\langle Z \cdot \rangle$) **where**
 $Zermelo_fms \equiv ZF_fin \cup \{ \cdot Separation(p) \cdot . p \in formula \}$

definition

$ZC :: i$ **where**
 $ZC \equiv Zermelo_fms \cup \{ \cdot AC \cdot \}$

lemma $ZFC_subset_formula$: $ZFC \subseteq formula$
 $\langle proof \rangle$

Satisfaction of a set of sentences

definition

$satT :: [i, i] \Rightarrow o$ ($\langle _ \models _ \rangle$ [36,36] 60) **where**
 $A \models \Phi \equiv \forall \varphi \in \Phi. (A, [] \models \varphi)$

lemma $satTI$ [intro!]:

assumes $\bigwedge \varphi. \varphi \in \Phi \implies A, [] \models \varphi$

shows $A \models \Phi$
 $\langle proof \rangle$

lemma *satTD [dest]* : $A \models \Phi \implies \varphi \in \Phi \implies A, [] \models \varphi$
 $\langle proof \rangle$

lemma *satT_mono*: $A \models \Phi \implies \Psi \subseteq \Phi \implies A \models \Psi$
 $\langle proof \rangle$

lemma *satT_Un_iff*: $M \models \Phi \cup \Psi \iff M \models \Phi \wedge M \models \Psi$ $\langle proof \rangle$

lemma *sats_ZFC_iff_sats_ZF_AC*:
 $(N \models ZFC) \iff (N \models ZF) \wedge (N, [] \models \cdot AC \cdot)$
 $\langle proof \rangle$

lemma *satT_ZF_imp_satT_Z*: $M \models ZF \implies M \models \cdot Z \cdot$
 $\langle proof \rangle$

lemma *satT_ZFC_imp_satT_ZC*: $M \models ZFC \implies M \models ZC$
 $\langle proof \rangle$

lemma *satT_Z_ZF_replacement_imp_satT_ZF*: $N \models \cdot Z \cdot \implies N \models \{\cdot Replacement(x) \cdot$
 $\cdot x \in formula\} \implies N \models ZF$
 $\langle proof \rangle$

lemma *satT_ZC_ZF_replacement_imp_satT_ZFC*: $N \models ZC \implies N \models \{\cdot Replacement(x) \cdot$
 $\cdot x \in formula\} \implies N \models ZFC$
 $\langle proof \rangle$

lemma *ground_repl_fm_sub_ZF*: $\{\cdot Replacement(ground_repl_fm(\varphi)) \cdot \cdot \varphi \in formula\} \subseteq ZF$
 $\langle proof \rangle$

lemma *ZF_replacement_fms_sub_ZFC*: $\{\cdot Replacement(\varphi) \cdot \cdot \varphi \in formula\} \subseteq ZFC$
 $\langle proof \rangle$

lemma *ground_repl_fm_sub_ZFC*: $\{\cdot Replacement(ground_repl_fm(\varphi)) \cdot \cdot \varphi \in formula\} \subseteq ZFC$
 $\langle proof \rangle$

lemma *ZF_replacement_ground_repl_fm_type*: $\{\cdot Replacement(ground_repl_fm(\varphi)) \cdot \cdot \varphi \in formula\} \subseteq formula$
 $\langle proof \rangle$

end

7 Interface between set models and Constructibility

This theory provides an interface between Paulson's relativization results and set models of ZFC. In particular, it is used to prove that the locale *forcing_data* is a sublocale of all relevant locales in **ZF-Constructible** (*M_trivial*, *M_basic*, *M_eclose*, etc).

In order to interpret the locales in **ZF-Constructible** we introduce new locales, each stronger than the previous one, assuming only the instances of Replacement needed to interpret the subsequent locales of that session. From the start we assume Separation for every internalized formula (with one parameter, but this is not a problem since we can use pairing).

theory *Interface*

imports

Fm_Definitions

Transitive_Models.Cardinal_AC_Relative

begin

locale *M_Z_basic* =

fixes *M*

assumes

upair_ax: *upair_ax*(##*M*) **and**

Union_ax: *Union_ax*(##*M*) **and**

power_ax: *power_ax*(##*M*) **and**

extensionality:*extensionality*(##*M*) **and**

foundation_ax: *foundation_ax*(##*M*) **and**

infinity_ax: *infinity_ax*(##*M*) **and**

separation_ax: $\varphi \in \text{formula} \implies \text{env} \in \text{list}(M) \implies$

$\text{arity}(\varphi) \leq 1 +_{\omega} \text{length}(\text{env}) \implies$

$\text{separation}(\##M, \lambda x. (M, [x] @ \text{env} \models \varphi))$

locale *M_transset* =

fixes *M*

assumes

trans_M: *Transset*(*M*)

locale *M_Z_trans* = *M_Z_basic* + *M_transset*

locale *M_ZF1* = *M_Z_basic* +

assumes

replacement_ax1:

replacement_assm(*M*, *env*, *eclose_closed_fm*)

replacement_assm(*M*, *env*, *eclose_abs_fm*)

replacement_assm(*M*, *env*, *wfrec_rank_fm*)

replacement_assm(*M*, *env*, *transrec_VFrom_fm*)

definition *instances1_fms* **where** *instances1_fms* \equiv

```

{ eclose_closed_fm,
  eclose_abs_fm,
  wfrec_rank_fm,
  transrec_VFrom_fm
}

```

This set has 4 internalized formulas.

```

lemmas replacement_instances1_defs =
  list_repl1_intf_fm_def list_repl2_intf_fm_def
  formula_repl1_intf_fm_def formula_repl2_intf_fm_def
  eclose_closed_fm_def eclose_abs_fm_def
  wfrec_rank_fm_def transrec_VFrom_fm_def tl_repl_intf_fm_def

```

```

lemma instances1_fms_type[TC]: instances1_fms  $\subseteq$  formula
  <proof>

```

```

declare (in M_ZF1) replacement_instances1_defs[simp]

```

```

locale M_ZF1_trans = M_ZF1 + M_Z_trans

```

```

context M_Z_trans
begin

```

```

lemmas transitivity = Transset_intf[OF trans_M]

```

7.1 Interface with $M_{trivial}$

```

lemma zero_in_M:  $0 \in M$ 
  <proof>

```

```

lemma separation_in_ctm :
  assumes
     $\varphi \in \text{formula}$   $env \in \text{list}(M)$ 
     $\text{arity}(\varphi) \leq 1 + \omega \text{ length}(env)$  and
     $\text{sats}Q: \bigwedge x. x \in M \implies (M, [x]@env \models \varphi) \longleftrightarrow Q(x)$ 
  shows
     $\text{separation}(\#\#M, Q)$ 
  <proof>

```

```

end — M_Z_trans

```

```

locale M_ZC_basic = M_Z_basic + M_AC  $\#\#M$ 

```

```

locale M_ZFC1 = M_ZF1 + M_ZC_basic

```

```

locale M_ZFC1_trans = M_ZF1_trans + M_ZFC1

```

```

sublocale M_Z_trans  $\subseteq$  M_trans  $\#\#M$ 
  <proof>

```

sublocale $M_Z_trans \subseteq M_trivial \#\#M$
<proof>

7.2 Interface with M_basic

definition *Intersection where*

$$Intersection(N,B,x) \equiv (\forall y[N]. y \in B \longrightarrow x \in y)$$

<ML>

definition *CartProd where*

$$CartProd(N,B,C,z) \equiv (\exists x[N]. x \in B \wedge (\exists y[N]. y \in C \wedge pair(N,x,y,z)))$$

<ML>

definition *ImageSep where*

$$ImageSep(N,B,r,y) \equiv (\exists p[N]. p \in r \wedge (\exists x[N]. x \in B \wedge pair(N,x,y,p)))$$

<ML>

definition *Converse where*

$$Converse(N,R,z) \equiv \exists p[N]. p \in R \wedge (\exists x[N]. \exists y[N]. pair(N,x,y,p) \wedge pair(N,y,x,z))$$

<ML>

definition *Restrict where*

$$Restrict(N,A,z) \equiv \exists x[N]. x \in A \wedge (\exists y[N]. pair(N,x,y,z))$$

<ML>

definition *Comp where*

$$Comp(N,R,S,xz) \equiv \exists x[N]. \exists y[N]. \exists z[N]. \exists xy[N]. \exists yz[N]. \\ pair(N,x,z,xz) \wedge pair(N,x,y,xy) \wedge pair(N,y,z,yz) \wedge xy \in S \wedge yz \in R$$

<ML>

definition *Pred where*

$$Pred(N,R,X,y) \equiv \exists p[N]. p \in R \wedge pair(N,y,X,p)$$

<ML>

definition *is_Memrel where*

$$is_Memrel(N,z) \equiv \exists x[N]. \exists y[N]. pair(N,x,y,z) \wedge x \in y$$

<ML>

definition *RecFun where*

$$RecFun(N,r,f,g,a,b,x) \equiv \exists xa[N]. \exists xb[N].$$

$$\begin{aligned} & pair(N,x,a,xa) \wedge xa \in r \wedge pair(N,x,b,xb) \wedge xb \in r \wedge \\ & (\exists fx[N]. \exists gx[N]. fun_apply(N,f,x,fx) \wedge fun_apply(N,g,x,gx) \wedge \\ & \quad fx \neq gx) \end{aligned}$$

$\langle ML \rangle$

context M_Z_trans

begin

lemma $inter_sep_intf$:

assumes $A \in M$

shows $separation(\#\#M, \lambda x. \forall y \in M. y \in A \longrightarrow x \in y)$

$\langle proof \rangle$

lemma $diff_sep_intf$:

assumes $B \in M$

shows $separation(\#\#M, \lambda x. x \notin B)$

$\langle proof \rangle$

lemma $cartprod_sep_intf$:

assumes $A \in M$ **and** $B \in M$

shows $separation(\#\#M, \lambda z. \exists x \in M. x \in A \wedge (\exists y \in M. y \in B \wedge pair(\#\#M, x, y, z)))$

$\langle proof \rangle$

lemma $image_sep_intf$:

assumes $A \in M$ **and** $B \in M$

shows $separation(\#\#M, \lambda y. \exists p \in M. p \in B \wedge (\exists x \in M. x \in A \wedge pair(\#\#M, x, y, p)))$

$\langle proof \rangle$

lemma $converse_sep_intf$:

assumes $R \in M$

shows $separation(\#\#M, \lambda z. \exists p \in M. p \in R \wedge (\exists x \in M. \exists y \in M. pair(\#\#M, x, y, p) \wedge pair(\#\#M, y, x, z)))$

$\langle proof \rangle$

lemma $restrict_sep_intf$:

assumes $A \in M$

shows $separation(\#\#M, \lambda z. \exists x \in M. x \in A \wedge (\exists y \in M. pair(\#\#M, x, y, z)))$

$\langle proof \rangle$

lemma $comp_sep_intf$:

assumes $R \in M$ **and** $S \in M$

shows $separation(\#\#M, \lambda xz. \exists x \in M. \exists y \in M. \exists z \in M. \exists xy \in M. \exists yz \in M.$

$pair(\#\#M, x, z, xz) \wedge pair(\#\#M, x, y, xy) \wedge pair(\#\#M, y, z, yz) \wedge xy \in S \wedge$

$yz \in R)$

$\langle proof \rangle$

lemma $pred_sep_intf$:

assumes $R \in M$ **and** $X \in M$

shows $\text{separation}(\#\#M, \lambda y. \exists p \in M. p \in R \wedge \text{pair}(\#\#M, y, X, p))$
 $\langle \text{proof} \rangle$

lemma memrel_sep_intf :
 $\text{separation}(\#\#M, \lambda z. \exists x \in M. \exists y \in M. \text{pair}(\#\#M, x, y, z) \wedge x \in y)$
 $\langle \text{proof} \rangle$

lemma $\text{is_recfun_sep_intf}$:
assumes $r \in M \ f \in M \ g \in M \ a \in M \ b \in M$
shows $\text{separation}(\#\#M, \lambda x. \exists xa \in M. \exists xb \in M.$
 $\text{pair}(\#\#M, x, a, xa) \wedge xa \in r \wedge \text{pair}(\#\#M, x, b, xb) \wedge xb \in r \wedge$
 $(\exists fx \in M. \exists gx \in M. \text{fun_apply}(\#\#M, f, x, fx) \wedge \text{fun_apply}(\#\#M, g, x, gx)$
 \wedge
 $\text{fx} \neq \text{gx}))$
 $\langle \text{proof} \rangle$

lemmas $M_basic_sep_instances =$
 $\text{inter_sep_intf} \ \text{diff_sep_intf} \ \text{cartprod_sep_intf}$
 $\text{image_sep_intf} \ \text{converse_sep_intf} \ \text{restrict_sep_intf}$
 $\text{pred_sep_intf} \ \text{memrel_sep_intf} \ \text{comp_sep_intf} \ \text{is_recfun_sep_intf}$

end — M_Z_trans

sublocale $M_Z_trans \subseteq M_basic_no_repl \ \#\#M$
 $\langle \text{proof} \rangle$

lemma $\text{Replace_eq_Collect}$:
assumes $\bigwedge x \ y \ y'. x \in A \implies P(x, y) \implies P(x, y') \implies y = y' \ \{y . x \in A, P(x, y)\}$
 $\subseteq B$
shows $\{y . x \in A, P(x, y)\} = \{y \in B . \exists x \in A. P(x, y)\}$
 $\langle \text{proof} \rangle$

context M_Z_trans
begin

lemma $\text{Pow_inter_M_closed}$: **assumes** $A \in M$ **shows** $\text{Pow}(A) \cap M \in M$
 $\langle \text{proof} \rangle$

lemma $\text{Pow}'_inter_M_closed$: **assumes** $A \in M$ **shows** $\{a \in \text{Pow}(A) . a \in M\}$
 $\in M$
 $\langle \text{proof} \rangle$

end — M_Z_trans

context $M_basic_no_repl$
begin

lemma $\text{Replace_funspace_succ_rep_intf_sub}$:
assumes

$M(A) M(n)$
shows
 $\{z . p \in A, \text{funspace_succ_rep_intf_rel}(M,p,z,n)\}$
 $\subseteq \text{Pow}^M(\text{Pow}^M(\bigcup \text{domain}(A) \cup (\{n\} \times \text{range}(A)) \cup (\bigcup (\{n\} \times \text{range}(A))))))$
 $\langle \text{proof} \rangle$

lemma *funspace_succ_rep_intf_uniq*:

assumes

$\text{funspace_succ_rep_intf_rel}(M,p,z,n) \text{ funspace_succ_rep_intf_rel}(M,p,z',n)$

shows

$z = z'$

$\langle \text{proof} \rangle$

lemma *Replace_funspace_succ_rep_intf_eq*:

assumes

$M(A) M(n)$

shows

$\{z . p \in A, \text{funspace_succ_rep_intf_rel}(M,p,z,n)\} =$

$\{z \in \text{Pow}^M(\text{Pow}^M(\bigcup \text{domain}(A) \cup (\{n\} \times \text{range}(A)) \cup (\bigcup (\{n\} \times \text{range}(A))))))$

.

$\exists p \in A. \text{funspace_succ_rep_intf_rel}(M,p,z,n)\}$

$\langle \text{proof} \rangle$

end — *M_basic_no_repl*

definition *fsri* **where**

$\text{fsri}(N,A,B) \equiv \lambda z. \exists p \in A. \exists f[N]. \exists b[N]. p = \langle f, b \rangle \wedge z = \{\text{cons}(\langle B, b \rangle, f)\}$

$\langle \text{ML} \rangle$

context *M_Z_trans*

begin

lemma *separation_fsri*:

$(\#\#M)(A) \implies (\#\#M)(B) \implies \text{separation}(\#\#M, \text{is_fsri}(\#\#M,A,B))$

$\langle \text{proof} \rangle$

lemma *separation_funspace_succ_rep_intf_rel*:

$(\#\#M)(A) \implies (\#\#M)(B) \implies \text{separation}(\#\#M, \lambda z. \exists p \in A. \text{funspace_succ_rep_intf_rel}(\#\#M,p,z,B))$

$\langle \text{proof} \rangle$

lemma *Replace_funspace_succ_rep_intf_in_M*:

assumes

$A \in M \ n \in M$

shows

$\{z . p \in A, \text{funspace_succ_rep_intf_rel}(\#\#M,p,z,n)\} \in M$

$\langle \text{proof} \rangle$

lemma *funspace_succ_rep_intf*:
assumes $n \in M$
shows
strong_replacement($\#\#M$,
 $\lambda p z. \exists f \in M. \exists b \in M. \exists nb \in M. \exists cnbf \in M.$
 $pair(\#\#M, f, b, p) \wedge pair(\#\#M, n, b, nb) \wedge is_cons(\#\#M, nb, f, cnbf) \wedge$
 $upair(\#\#M, cnbf, cnbf, z)$)
 $\langle proof \rangle$

end — *M_Z_trans*

sublocale $M_Z_trans \subseteq M_basic \ \#\#M$
 $\langle proof \rangle$

7.3 Interface with *M_trancl*

context *M_ZF1_trans*
begin

lemma *rtrancl_separation_intf*:
assumes $r \in M \ A \in M$
shows *separation* ($\#\#M$, *rtran_closure_mem*($\#\#M, A, r$))
 $\langle proof \rangle$

lemma *wftrancl_separation_intf*:
assumes $r \in M$ **and** $Z \in M$
shows *separation* ($\#\#M$, *wellfounded_trancl*($\#\#M, Z, r$))
 $\langle proof \rangle$

To prove $\omega \in M$ we get an infinite set I from *infinity_ax* closed under θ and *succ*; that shows $\omega \subseteq I$. Then we can separate I with the predicate $\lambda x. x \in \omega$.

lemma *finite_sep_intf*: *separation*($\#\#M$, $\lambda x. x \in nat$)
 $\langle proof \rangle$

lemma *nat_subset_I*: $\exists I \in M. nat \subseteq I$
 $\langle proof \rangle$

lemma *nat_in_M*: $nat \in M$
 $\langle proof \rangle$

end — *M_ZF1_trans*

sublocale $M_ZF1_trans \subseteq M_trancl \ \#\#M$
 $\langle proof \rangle$

7.4 Interface with *M_eclose*

lemma *repl_sats*:

assumes
 $sat: \bigwedge x z. x \in M \implies z \in M \implies (M, Cons(x, Cons(z, env))) \models \varphi \iff P(x, z)$
shows
 $strong_replacement(\#\#M, \lambda x z. (M, Cons(x, Cons(z, env))) \models \varphi) \iff$
 $strong_replacement(\#\#M, P)$
 $\langle proof \rangle$

$\langle ML \rangle$

context M_ZF1_trans
begin

This lemma obtains *iterates_replacement* for predicates without parameters.

lemma *iterates_repl_intf* :

assumes
 $v \in M$ **and**
 $isfm: is_F_fm \in formula$ **and**
 $arty: arity(is_F_fm) = 2$ **and**
 $satsf: \bigwedge a b env'. \llbracket a \in M ; b \in M ; env' \in list(M) \rrbracket$
 $\implies is_F(a, b) \iff (M, [b, a] @ env' \models is_F_fm)$
and $is_F_fm_replacement:$
 $\bigwedge env. (\exists \cdot \langle 1, 0 \rangle is\ 2 \cdot \wedge is_wfrec_fm(iterates_MH_fm(is_F_fm, 9, 2, 1, 0), 3, 1, 0)$
 $\cdot) \in formula \implies env \in list(M) \implies$
 $arity((\exists \cdot \langle 1, 0 \rangle is\ 2 \cdot \wedge is_wfrec_fm(iterates_MH_fm(is_F_fm, 9, 2, 1, 0), 3, 1, 0)$
 $\cdot)) \leq 2 + \omega \ length(env) \implies$
 $strong_replacement(\#\#M, \lambda x y.$
 $M, [x, y] @ env \models (\exists \cdot \langle 1, 0 \rangle is\ 2 \cdot \wedge is_wfrec_fm(iterates_MH_fm(is_F_fm, 9, 2, 1, 0), 3, 1, 0)$
 $\cdot))$
shows
 $iterates_replacement(\#\#M, is_F, v)$
 $\langle proof \rangle$

lemma *eclose_repl1_intf*:

assumes $A \in M$
shows $iterates_replacement(\#\#M, big_union(\#\#M), A)$
 $\langle proof \rangle$

lemma *eclose_repl2_intf*:

assumes $A \in M$
shows $strong_replacement(\#\#M, \lambda n y. n \in nat \wedge is_iterates(\#\#M, big_union(\#\#M),$
 $A, n, y))$
 $\langle proof \rangle$

end — M_ZF1_trans

sublocale $M_ZF1_trans \subseteq M_eclose \#\#M$
 $\langle proof \rangle$

Interface with M_eclose .

```

schematic_goal sats_is_Vset_fm_auto:
  assumes
     $i \in \text{nat } v \in \text{nat } \text{env} \in \text{list}(A) \ 0 \in A$ 
     $i < \text{length}(\text{env}) \ v < \text{length}(\text{env})$ 
  shows
     $\text{is\_Vset}(\#\#A, \text{nth}(i, \text{env}), \text{nth}(v, \text{env})) \longleftrightarrow (A, \text{env} \models \text{?ivs\_fm}(i, v))$ 
  <proof>

```

<ML>

```

declare is_Hrank_fm_def[fm_definitions add]

```

```

context M_ZF1_trans
begin

```

```

lemma wfrec_rank :
  assumes  $X \in M$ 
  shows  $\text{wfrec\_replacement}(\#\#M, \text{is\_Hrank}(\#\#M), \text{rrank}(X))$ 
  <proof>

```

```

lemma trans_repl_HVFrom :
  assumes  $A \in M \ i \in M$ 
  shows  $\text{transrec\_replacement}(\#\#M, \text{is\_HVfrom}(\#\#M, A), i)$ 
  <proof>

```

```

end — M_ZF1_trans

```

7.5 Interface for proving Collects and Replace in M.

```

context M_ZF1_trans
begin

```

```

lemma Collect_in_M :
  assumes
     $\varphi \in \text{formula } \text{env} \in \text{list}(M)$ 
     $\text{arity}(\varphi) \leq 1 +_{\omega} \text{length}(\text{env}) \ A \in M$  and
     $\text{sats}Q: \bigwedge x. x \in M \implies (M, [x]@\text{env} \models \varphi) \longleftrightarrow Q(x)$ 
  shows
     $\{y \in A . Q(y)\} \in M$ 
  <proof>

```

```

lemma separation_in_M :
  assumes
     $\varphi \in \text{formula } \text{env} \in \text{list}(M)$ 
     $\text{arity}(\varphi) \leq 1 +_{\omega} \text{length}(\text{env}) \ A \in M$  and
     $\text{sats}Q: \bigwedge x. x \in A \implies (M, [x]@\text{env} \models \varphi) \longleftrightarrow Q(x)$ 
  shows
     $\{y \in A . Q(y)\} \in M$ 
  <proof>

```

end — M_ZF1_trans

context M_Z_trans

begin

lemma $strong_replacement_in_ctm$:

assumes

f_fm : $\varphi \in formula$ **and**

f_ar : $arity(\varphi) \leq 2 +_{\omega} length(env)$ **and**

$fsats$: $\bigwedge x y. x \in M \implies y \in M \implies (M, [x, y]@env \models \varphi) \longleftrightarrow y = f(x)$ **and**

$fclosed$: $\bigwedge x. x \in M \implies f(x) \in M$ **and**

$phi_replacement$: $replacement_assm(M, env, \varphi)$ **and**

$env \in list(M)$

shows $strong_replacement(\#\#M, \lambda x y. y = f(x))$

$\langle proof \rangle$

lemma $strong_replacement_rel_in_ctm$:

assumes

f_fm : $\varphi \in formula$ **and**

f_ar : $arity(\varphi) \leq 2 +_{\omega} length(env)$ **and**

$fsats$: $\bigwedge x y. x \in M \implies y \in M \implies (M, [x, y]@env \models \varphi) \longleftrightarrow f(x, y)$ **and**

$phi_replacement$: $replacement_assm(M, env, \varphi)$ **and**

$env \in list(M)$

shows $strong_replacement(\#\#M, f)$

$\langle proof \rangle$

lemma $Replace_in_M$:

assumes

f_fm : $\varphi \in formula$ **and**

f_ar : $arity(\varphi) \leq 2 +_{\omega} length(env)$ **and**

$fsats$: $\bigwedge x y. x \in A \implies y \in M \implies (M, [x, y]@env \models \varphi) \longleftrightarrow y = f(x)$ **and**

$fclosed$: $\bigwedge x. x \in A \implies f(x) \in M$ **and**

$A \in M$ $env \in list(M)$ **and**

$phi_replacement$: $replacement_assm(M, env@[A], \cdot\varphi \wedge \cdot 0 \in length(env) +_{\omega} 2 \cdot \cdot$

)

shows $\{f(x) . x \in A\} \in M$

$\langle proof \rangle$

lemma $Replace_relativized_in_M$:

assumes

f_fm : $\varphi \in formula$ **and**

f_ar : $arity(\varphi) \leq 2 +_{\omega} length(env)$ **and**

$fsats$: $\bigwedge x y. x \in A \implies y \in M \implies (M, [x, y]@env \models \varphi) \longleftrightarrow is_f(x, y)$ **and**

$fabs$: $\bigwedge x y. x \in A \implies y \in M \implies is_f(x, y) \longleftrightarrow y = f(x)$ **and**

$fclosed$: $\bigwedge x. x \in A \implies f(x) \in M$ **and**

$A \in M$ $env \in list(M)$ **and**

$phi_replacement$: $replacement_assm(M, env@[A], \cdot\varphi \wedge \cdot 0 \in length(env) +_{\omega} 2 \cdot \cdot$

)

shows $\{f(x) . x \in A\} \in M$

$\langle \text{proof} \rangle$

lemma *ren_action* :

assumes

$env \in list(M)$ $x \in M$ $y \in M$ $z \in M$

shows $\forall i . i < 2 + \omega \cdot length(env) \longrightarrow$

$nth(i, [x, z] @ env) = nth(\rho_repl(length(env))'i, [z, x, y] @ env)$

$\langle \text{proof} \rangle$

lemma *Lambda_in_M* :

assumes

f_fm : $\varphi \in \text{formula}$ **and**

f_ar : $arity(\varphi) \leq 2 + \omega \cdot length(env)$ **and**

$fsats$: $\bigwedge x y . x \in A \implies y \in M \implies (M, [x, y] @ env \models \varphi) \longleftrightarrow is_f(x, y)$ **and**

$fabs$: $\bigwedge x y . x \in A \implies y \in M \implies is_f(x, y) \longleftrightarrow y = f(x)$ **and**

$fclosed$: $\bigwedge x . x \in A \implies f(x) \in M$ **and**

$A \in M$ $env \in list(M)$ **and**

$phi'_replacement2$: $replacement_assm(M, env @ [A], Lambda_in_M_fm(\varphi, length(env)))$

shows $(\lambda x \in A . f(x)) \in M$

$\langle \text{proof} \rangle$

lemma *ren_action'* :

assumes

$env \in list(M)$ $x \in M$ $y \in M$ $z \in M$ $u \in M$

shows $\forall i . i < 3 + \omega \cdot length(env) \longrightarrow$

$nth(i, [x, z, u] @ env) = nth(\rho_pair_repl(length(env))'i, [x, z, y, u] @ env)$

$\langle \text{proof} \rangle$

lemma *LambdaPair_in_M* :

assumes

f_fm : $\varphi \in \text{formula}$ **and**

f_ar : $arity(\varphi) \leq 3 + \omega \cdot length(env)$ **and**

$fsats$: $\bigwedge x z r . x \in M \implies z \in M \implies r \in M \implies (M, [x, z, r] @ env \models \varphi) \longleftrightarrow is_f(x, z, r)$

and

$fabs$: $\bigwedge x z r . x \in M \implies z \in M \implies r \in M \implies is_f(x, z, r) \longleftrightarrow r = f(x, z)$ **and**

$fclosed$: $\bigwedge x z . x \in M \implies z \in M \implies f(x, z) \in M$ **and**

$A \in M$ $env \in list(M)$ **and**

$phi'_replacement3$: $replacement_assm(M, env @ [A], LambdaPair_in_M_fm(\varphi, length(env)))$

shows $(\lambda x \in A . f(fst(x), snd(x))) \in M$

$\langle \text{proof} \rangle$

lemma (*in M_ZF1_trans*) *lam_replacement2_in_ctm* :

assumes

f_fm : $\varphi \in \text{formula}$ **and**

f_ar : $arity(\varphi) \leq 3 + \omega \cdot length(env)$ **and**

$fsats$: $\bigwedge x z r . x \in M \implies z \in M \implies r \in M \implies (M, [x, z, r] @ env \models \varphi) \longleftrightarrow is_f(x, z, r)$

and

$fabs$: $\bigwedge x z r . x \in M \implies z \in M \implies r \in M \implies is_f(x, z, r) \longleftrightarrow r = f(x, z)$ **and**

$fclosed$: $\bigwedge x z . x \in M \implies z \in M \implies f(x, z) \in M$ **and**

$env \in list(M)$ **and**
 $phi_replacement3: \bigwedge A. A \in M \implies replacement_assm(M, env@[A], LambdaPair_in_M_fm(\varphi, length(env)))$
shows $lam_replacement(\#\#M, \lambda x. f(fst(x), snd(x)))$
 $\langle proof \rangle$

$\langle ML \rangle$

lemma $separation_sat_after_function_1:$

assumes $[a, b, c, d] \in list(M)$ **and** $\chi \in formula$ **and** $arity(\chi) \leq 6$
and
 $f_fm: f_fm \in formula$ **and**
 $f_ar: arity(f_fm) \leq 6$ **and**
 $fsats: \bigwedge fx x. fx \in M \implies x \in M \implies (M, [fx, x]@[a, b, c, d] \models f_fm) \longleftrightarrow fx = f(x)$
and
 $fclosed: \bigwedge x. x \in M \implies f(x) \in M$ **and**
 $g_fm: g_fm \in formula$ **and**
 $g_ar: arity(g_fm) \leq 7$ **and**
 $gsats: \bigwedge gx fx x. gx \in M \implies fx \in M \implies x \in M \implies (M, [gx, fx, x]@[a, b, c, d] \models g_fm) \longleftrightarrow gx = g(x)$ **and**
 $gclosed: \bigwedge x. x \in M \implies g(x) \in M$
shows $separation(\#\#M, \lambda r. M, [f(r), a, b, c, d, g(r)] \models \chi)$
 $\langle proof \rangle$

lemma $separation_sat_after_function3:$

assumes $[a, b, c, d] \in list(M)$ **and** $\chi \in formula$ **and** $arity(\chi) \leq 7$
and
 $f_fm: f_fm \in formula$ **and**
 $f_ar: arity(f_fm) \leq 6$ **and**
 $fsats: \bigwedge fx x. fx \in M \implies x \in M \implies (M, [fx, x]@[a, b, c, d] \models f_fm) \longleftrightarrow fx = f(x)$
and
 $fclosed: \bigwedge x. x \in M \implies f(x) \in M$ **and**
 $g_fm: g_fm \in formula$ **and**
 $g_ar: arity(g_fm) \leq 7$ **and**
 $gsats: \bigwedge gx fx x. gx \in M \implies fx \in M \implies x \in M \implies (M, [gx, fx, x]@[a, b, c, d] \models g_fm) \longleftrightarrow gx = g(x)$ **and**
 $gclosed: \bigwedge x. x \in M \implies g(x) \in M$ **and**
 $h_fm: h_fm \in formula$ **and**
 $h_ar: arity(h_fm) \leq 8$ **and**
 $hsats: \bigwedge hx gx fx x. hx \in M \implies gx \in M \implies fx \in M \implies x \in M \implies (M, [hx, gx, fx, x]@[a, b, c, d] \models h_fm) \longleftrightarrow hx = h(x)$ **and**
 $hclosed: \bigwedge x. x \in M \implies h(x) \in M$
shows $separation(\#\#M, \lambda r. M, [f(r), a, b, c, d, g(r), h(r)] \models \chi)$
 $\langle proof \rangle$

lemma $separation_sat_after_function:$

assumes $[a, b, c, d, \tau] \in list(M)$ **and** $\chi \in formula$ **and** $arity(\chi) \leq 7$
and
 $f_fm: f_fm \in formula$ **and**
 $f_ar: arity(f_fm) \leq 7$ **and**

fsats: $\bigwedge fx x. fx \in M \implies x \in M \implies (M, [fx, x]@[a, b, c, d, \tau] \models f_fm) \longleftrightarrow$
 $fx = f(x)$ **and**
fclosed: $\bigwedge x . x \in M \implies f(x) \in M$ **and**
g_fm: $g_fm \in \text{formula}$ **and**
g_ar: $\text{arity}(g_fm) \leq 8$ **and**
gsats: $\bigwedge gx fx x. gx \in M \implies fx \in M \implies x \in M \implies (M, [gx, fx, x]@[a, b, c, d, \tau]$
 $\models g_fm) \longleftrightarrow gx = g(x)$ **and**
gclosed: $\bigwedge x . x \in M \implies g(x) \in M$
shows $\text{separation}(\#\#M, \lambda r. M, [f(r), a, b, c, d, \tau, g(r)] \models \chi)$
 $\langle \text{proof} \rangle$
end

definition $\text{separation_assm_fm} :: [i, i, i] \Rightarrow i$

where

$\text{separation_assm_fm}(A, x, f_fm) \equiv (\exists (\exists \cdot 0 \in A +_{\omega} 2 \cdot \wedge \cdot \langle 0, 1 \rangle \text{ is } x +_{\omega} 2 \cdot \wedge$
 $f_fm \dots))$

lemma $\text{separation_assm_fm_type}[TC]:$

$A \in \omega \implies y \in \omega \implies f_fm \in \text{formula} \implies \text{separation_assm_fm}(A, y, f_fm) \in$
 formula
 $\langle \text{proof} \rangle$

lemma $\text{arity_separation_assm_fm} : A \in \omega \implies x \in \omega \implies f_fm \in \text{formula} \implies$
 $\text{arity}(\text{separation_assm_fm}(A, x, f_fm)) = \text{succ}(A) \cup \text{succ}(x) \cup \text{pred}(\text{pred}(\text{arity}(f_fm)))$
 $\langle \text{proof} \rangle$

definition $\text{separation_assm_bin_fm}$ **where**

$\text{separation_assm_bin_fm}(A, y, f_fm) \equiv$
 $(\exists (\exists (\exists (\exists (\exists \cdot 3 \in A +_{\omega} 4 \cdot \wedge \cdot \langle 3, 2 \rangle \text{ is } y +_{\omega} 4 \cdot) \wedge \cdot f_fm \wedge \cdot \text{fst}(3) \text{ is } 0 \cdot$
 $\wedge \cdot \text{snd}(3) \text{ is } 1 \dots) \cdot) \cdot) \cdot)$

lemma $\text{separation_assm_bin_fm_type}[TC]:$

$A \in \omega \implies y \in \omega \implies f_fm \in \text{formula} \implies \text{separation_assm_bin_fm}(A, y, f_fm)$
 $\in \text{formula}$
 $\langle \text{proof} \rangle$

lemma $\text{arity_separation_assm_bin_fm} : A \in \omega \implies x \in \omega \implies f_fm \in \text{formula}$
 \implies

$\text{arity}(\text{separation_assm_bin_fm}(A, x, f_fm)) = \text{succ}(A) \cup \text{succ}(x) \cup (\text{pred}^4(\text{arity}(f_fm)))$
 $\langle \text{proof} \rangle$

context M_Z_trans

begin

lemma $\text{separation_assm_sats} :$

assumes

$f_fm: \varphi \in \text{formula}$ **and**

$f_ar: \text{arity}(\varphi) = 2$ **and**

$\text{fsats}: \bigwedge env x y. env \in \text{list}(M) \implies x \in M \implies y \in M \implies (M, [x, y]@env \models \varphi) \longleftrightarrow$

is_f(x,y) **and**
fabs: $\bigwedge x y. x \in M \implies y \in M \implies is_f(x,y) \longleftrightarrow y = f(x)$ **and**
fclosed: $\bigwedge x. x \in M \implies f(x) \in M$ **and**
 $A \in M$
shows *separation*($\#\#M, \lambda y. \exists x \in M . x \in A \wedge y = \langle x, f(x) \rangle$)
 $\langle proof \rangle$

lemma *separation_assm_bin_sats* :

assumes
f_fm: $\varphi \in formula$ **and**
f_ar: $arity(\varphi) = 3$ **and**
fats: $\bigwedge env\ x\ z\ y. env \in list(M) \implies x \in M \implies z \in M \implies y \in M \implies (M, [x,z,y]@env \models \varphi) \longleftrightarrow is_f(x,z,y)$ **and**
fabs: $\bigwedge x\ z\ y. x \in M \implies z \in M \implies y \in M \implies is_f(x,z,y) \longleftrightarrow y = f(x,z)$ **and**
fclosed: $\bigwedge x\ z . x \in M \implies z \in M \implies f(x,z) \in M$ **and**
 $A \in M$
shows *separation*($\#\#M, \lambda y. \exists x \in M . x \in A \wedge y = \langle x, f(fst(x),snd(x)) \rangle$)
 $\langle proof \rangle$

lemma *separation_Union*: $A \in M \implies$

separation($\#\#M, \lambda y. \exists x \in M . x \in A \wedge y = \langle x, Union(x) \rangle$)
 $\langle proof \rangle$

lemma *lam_replacement_Union*: *lam_replacement*($\#\#M, Union$)

$\langle proof \rangle$

lemma *separation_fst*: $A \in M \implies$

separation($\#\#M, \lambda y. \exists x \in M . x \in A \wedge y = \langle x, fst(x) \rangle$)
 $\langle proof \rangle$

lemma *lam_replacement_fst*: *lam_replacement*($\#\#M, fst$)

$\langle proof \rangle$

lemma *separation_snd*: $A \in M \implies$

separation($\#\#M, \lambda y. \exists x \in M . x \in A \wedge y = \langle x, snd(x) \rangle$)
 $\langle proof \rangle$

lemma *lam_replacement_snd*: *lam_replacement*($\#\#M, snd$)

$\langle proof \rangle$

Binary lambda-replacements

lemma *separation_Image*: $A \in M \implies$

separation($\#\#M, \lambda y. \exists x \in M . x \in A \wedge y = \langle x, fst(x) \text{ “ } snd(x) \rangle$)
 $\langle proof \rangle$

lemma *lam_replacement_Image*: *lam_replacement*($\#\#M, \lambda x . fst(x) \text{ “ } snd(x)$)

$\langle proof \rangle$

lemma *separation_middle_del*: $A \in M \implies$

$separation(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, middle_del(fst(x), snd(x)) \rangle)$
 $\langle proof \rangle$

lemma $lam_replacement_middle_del$: $lam_replacement(\#\#M, \lambda r. middle_del(fst(r), snd(r)))$
 $\langle proof \rangle$

lemma $separation_prodRepl$: $A \in M \implies$
 $separation(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, prodRepl(fst(x), snd(x)) \rangle)$
 $\langle proof \rangle$

lemma $lam_replacement_prodRepl$: $lam_replacement(\#\#M, \lambda r. prodRepl(fst(r), snd(r)))$
 $\langle proof \rangle$

end — M_Z_trans

context $M_trivial$
begin

lemma $first_closed$:
 $M(B) \implies M(r) \implies first(u, r, B) \implies M(u)$
 $\langle proof \rangle$

$\langle ML \rangle$
 $\langle proof \rangle$

$\langle ML \rangle$
 $\langle proof \rangle$

end — $M_trivial$

context M_Z_trans
begin

lemma (**in** M_basic) $is_minimum_equivalence$:
 $M(R) \implies M(X) \implies M(u) \implies is_minimum(M, R, X, u) \longleftrightarrow is_minimum'(M, R, X, u)$
 $\langle proof \rangle$

lemma $separation_minimum$: $A \in M \implies$
 $separation(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, minimum(fst(x), snd(x)) \rangle)$
 $\langle proof \rangle$

lemma $lam_replacement_minimum$: $lam_replacement(\#\#M, \lambda x. minimum(fst(x), snd(x)))$
 $\langle proof \rangle$

end — M_Z_trans

end

7.6 More Instances of Separation

theory *Separation_Instances*

imports

Interface

begin

The following instances are mostly the same repetitive task; and we just copied and pasted, tweaking some lemmas if needed (for example, we might have needed to use some closure results).

definition *radd_body* :: $[i,i,i] \Rightarrow o$ **where**

$$\begin{aligned} \text{radd_body}(R,S) \equiv & \lambda z. (\exists x y. z = \langle \text{Inl}(x), \text{Inr}(y) \rangle) \vee \\ & (\exists x' x. z = \langle \text{Inl}(x'), \text{Inl}(x) \rangle \wedge \langle x', x \rangle \in R) \vee \\ & (\exists y' y. z = \langle \text{Inr}(y'), \text{Inr}(y) \rangle \wedge \langle y', y \rangle \in S) \end{aligned}$$

$\langle ML \rangle$

definition *rmult_body* :: $[i,i,i] \Rightarrow o$ **where**

$$\begin{aligned} \text{rmult_body}(b,d) \equiv & \lambda z. \exists x' y' x y. z = \langle \langle x', y' \rangle, x, y \rangle \wedge (\langle x', x \rangle \in b \vee \\ & x' = x \wedge \langle y', y \rangle \in d) \end{aligned}$$

$\langle ML \rangle$

lemma (in *M_replacement*) *separation_well_ord_iso*:

$$\begin{aligned} (M)(f) \implies (M)(r) \implies (M)(A) \implies & \text{separation} \\ (M, \lambda x. x \in A \longrightarrow (\exists y[M]. \exists p[M]. & \text{is_apply}(M, f, x, y) \wedge \text{pair}(M, y, x, p) \\ \wedge p \in r)) & \\ \langle \text{proof} \rangle & \end{aligned}$$

definition *is_obase_body* :: $[i \Rightarrow o, i, i, i] \Rightarrow o$ **where**

$$\begin{aligned} \text{is_obase_body}(N,A,r,x) \equiv & x \in A \longrightarrow \\ & \neg (\exists y[N]. \\ & \quad \exists g[N]. \\ & \quad \text{ordinal}(N, y) \wedge \\ & \quad (\exists my[N]. \\ & \quad \quad \exists pxr[N]. \\ & \quad \quad \text{membership}(N, y, my) \wedge \\ & \quad \quad \text{pred_set}(N, A, x, r, pxr) \wedge \\ & \quad \quad \text{order_isomorphism}(N, pxr, r, y, my, g))) \end{aligned}$$

$\langle ML \rangle$

definition *is_obase_equals* :: $[i \Rightarrow o, i, i, i] \Rightarrow o$ **where**

$$\begin{aligned} \text{is_obase_equals}(N,A,r,a) \equiv & \exists x[N]. \\ & \exists g[N]. \\ & \exists mx[N]. \\ & \exists par[N]. \\ & \text{ordinal}(N, x) \wedge \\ & \text{membership}(N, x, mx) \wedge \end{aligned}$$

$r, x, mx, g) \quad \text{pred_set}(N, A, a, r, \text{par}) \wedge \text{order_isomorphism}(N, \text{par},$

$\langle ML \rangle$

context M_ZF1_trans

begin

lemma radd_body_abs :

assumes $(\#\#M)(R) (\#\#M)(S) (\#\#M)(x)$
shows $\text{is_radd_body}(\#\#M, R, S, x) \longleftrightarrow \text{radd_body}(R, S, x)$
 $\langle \text{proof} \rangle$

lemma $\text{separation_radd_body}$:

$(\#\#M)(R) \implies (\#\#M)(S) \implies \text{separation}$
 $(\#\#M, \lambda z. (\exists x y. z = \langle \text{Inl}(x), \text{Inr}(y) \rangle) \vee$
 $(\exists x' x. z = \langle \text{Inl}(x'), \text{Inl}(x) \rangle \wedge \langle x', x \rangle \in R) \vee$
 $(\exists y' y. z = \langle \text{Inr}(y'), \text{Inr}(y) \rangle \wedge \langle y', y \rangle \in S))$
 $\langle \text{proof} \rangle$

lemma rmult_body_abs :

assumes $(\#\#M)(b) (\#\#M)(d) (\#\#M)(x)$
shows $\text{is_rmult_body}(\#\#M, b, d, x) \longleftrightarrow \text{rmult_body}(b, d, x)$
 $\langle \text{proof} \rangle$

lemma $\text{separation_rmult_body}$:

$(\#\#M)(b) \implies (\#\#M)(d) \implies \text{separation}$
 $(\#\#M, \lambda z. \exists x' y' x y. z = \langle \langle x', y' \rangle, x, y \rangle \wedge (\langle x', x \rangle \in b \vee x' = x \wedge \langle y', y \rangle$
 $\in d))$
 $\langle \text{proof} \rangle$

lemma $\text{separation_is_obase}$:

$(\#\#M)(f) \implies (\#\#M)(r) \implies (\#\#M)(A) \implies \text{separation}$
 $(\#\#M, \lambda x. x \in A \longrightarrow$
 $\neg (\exists y[\#\#M].$
 $\exists g[\#\#M].$
 $\text{ordinal}(\#\#M, y) \wedge$
 $(\exists my[\#\#M].$
 $\exists pxr[\#\#M].$
 $\text{membership}(\#\#M, y, my) \wedge$
 $\text{pred_set}(\#\#M, A, x, r, pxr) \wedge$
 $\text{order_isomorphism}(\#\#M, pxr, r, y, my, g)))$
 $\langle \text{proof} \rangle$

lemma $\text{separation_obase_equals}$:

$(\#\#M)(f) \implies (\#\#M)(r) \implies (\#\#M)(A) \implies \text{separation}$
 $(\#\#M, \lambda a. \exists x[\#\#M].$
 $\exists g[\#\#M].$

$\exists mx[\#\#M].$
 $\exists par[\#\#M].$
 $ordinal(\#\#M, x) \wedge$
 $membership(\#\#M, x, mx) \wedge$
 $pred_set(\#\#M, A, a, r, par) \wedge order_isomorphism(\#\#M,$
 $par, r, x, mx, g))$
 ⟨proof⟩

lemma separation_PiP_rel:
 $(\#\#M)(A) \implies separation(\#\#M, PiP_rel(\#\#M, A))$
 ⟨proof⟩

lemma separation_injP_rel:
 $(\#\#M)(A) \implies separation(\#\#M, injP_rel(\#\#M, A))$
 ⟨proof⟩

lemma separation_surjP_rel:
 $(\#\#M)(A) \implies (\#\#M)(B) \implies separation(\#\#M, surjP_rel(\#\#M, A, B))$
 ⟨proof⟩

lemma separation_is_function:
 $separation(\#\#M, is_function(\#\#M))$
 ⟨proof⟩

end — M_ZF1_trans

definition fstsnd_in_sndsnd :: $[i] \Rightarrow o$ **where**
 $fstsnd_in_sndsnd \equiv \lambda x. fst(snd(x)) \in snd(snd(x))$
 ⟨ML⟩

definition sndfst_eq_fstsnd :: $[i] \Rightarrow o$ **where**
 $sndfst_eq_fstsnd \equiv \lambda x. snd(fst(x)) = fst(snd(x))$
 ⟨ML⟩

context M_ZF1_trans
begin

lemma fstsnd_in_sndsnd_abs:
assumes $(\#\#M)(x)$
shows $is_fstsnd_in_sndsnd(\#\#M, x) \longleftrightarrow fstsnd_in_sndsnd(x)$
 ⟨proof⟩

lemma separation_fstsnd_in_sndsnd:
 $separation(\#\#M, \lambda x. fst(snd(x)) \in snd(snd(x)))$
 ⟨proof⟩

lemma sndfst_eq_fstsnd_abs:

```

assumes ( $\#\#M$ )( $x$ )
shows  $is\_sndfst\_eq\_fstsnd(\#\#M,x) \longleftrightarrow sndfst\_eq\_fstsnd(x)$ 
   $\langle proof \rangle$ 

lemma  $separation\_sndfst\_eq\_fstsnd$ :
   $separation(\#\#M, \lambda x. snd(fst(x)) = fst(snd(x)))$ 
   $\langle proof \rangle$ 

end —  $M\_ZF1\_trans$ 

end

```

8 More Instances of Replacement

theory $Replacement_Instances$

imports

$Separation_Instances$

$Transitive_Models.Pointed_DC_Relative$

begin

lemma $composition_fm_type[TC]$: $a0 \in \omega \implies a1 \in \omega \implies a2 \in \omega \implies$
 $composition_fm(a0,a1,a2) \in formula$
 $\langle proof \rangle$

$\langle ML \rangle$

definition $is_omega_funspace :: [i \Rightarrow o, i, i, i] \Rightarrow o$ **where**
 $is_omega_funspace(N, B, n, z) \equiv \exists o[N]. \omega(N, o) \wedge n \in o \wedge is_funspace(N, n,$
 $B, z)$

$\langle ML \rangle$

definition $HAleph_wfrec_repl_body$ **where**

$HAleph_wfrec_repl_body(N, mesa, x, z) \equiv \exists y[N].$

$pair(N, x, y, z) \wedge$

$(\exists g[N].$

$(\forall u[N].$

$u \in g \longleftrightarrow$

$(\exists a[N].$

$\exists y[N].$

$\exists ax[N].$

$\exists sx[N].$

$\exists r_sx[N].$

$\exists f_r_sx[N].$

$pair(N, a, y, u) \wedge$

$pair(N, a, x, ax) \wedge$

$upair(N, a, a, sx) \wedge$

$pre_image(N, mesa, sx, r_sx) \wedge$

$restriction(N, g, r_sx, f_r_sx) \wedge ax \in mesa \wedge is_HAleph(N, a, f_r_sx, y)))$

\wedge

$is_HAleph(N, x, g, y)$

$\langle ML \rangle$

definition $dcwit_repl_body$ **where**

$dcwit_repl_body(N, mesa, A, a, s, R) \equiv \lambda x z. \exists y[N]. pair(N, x, y, z) \wedge$
 is_wfrec
 $(N, \lambda n f. is_nat_case$
 $(N, a,$
 $\lambda m bmfm.$
 $\exists fm[N].$
 $\exists cp[N].$
 $is_apply(N, f, m, fm) \wedge$
 $is_Collect(N, A, \lambda x. \exists fmx[N]. (N(x)$
 $\wedge fmx \in R) \wedge pair(N, fm, x, fmx), cp) \wedge$
 $is_apply(N, s, cp, bmfm),$
 $n),$
 $mesa, x, y)$

$\langle ML \rangle$

$\langle proof \rangle$

$\langle ML \rangle$

definition $dcwit_aux_fm$ **where**

$dcwit_aux_fm(A, s, R) \equiv (\cdot \exists \cdot \cdot 4' 2 is 0 \cdot \wedge$
 $(\cdot \exists \cdot Collect_fm$
 $(succ(succ(succ(succ(succ(succ(succ(succ(A))))))))),$
 $(\cdot \exists \cdot \cdot 0 \in$
 $succ(succ(succ(succ(succ(succ(succ(succ(R))))))))))$
 $\cdot \wedge$
 $pair_fm(3, 1, 0) \cdot \cdot),$
 $0) \wedge$
 $\cdot succ(succ(succ(succ(succ(succ(succ(succ(s))))))))' 0 is$
 $2 \cdot \dots \cdot \cdot)$

$\langle ML \rangle$

lemma $dcwit_aux_fm_type[TC]: A \in \omega \implies s \in \omega \implies R \in \omega \implies dcwit_aux_fm(A, s, R)$

$\in formula$

$\langle proof \rangle$

definition $is_nat_case_dcwit_aux_fm$ **where**

$is_nat_case_dcwit_aux_fm(A, a, s, R) \equiv is_nat_case_fm$
 $(succ(succ(succ(succ(succ(a))))), dcwit_aux_fm(A, s, R),$
 $2, 0)$

lemma *is_nat_case_dcwit_aux_fm_type*[TC]: $A \in \omega \implies a \in \omega \implies s \in \omega \implies R \in \omega \implies \text{is_nat_case_dcwit_aux_fm}(A, a, s, R) \in \text{formula}$

<proof>

<ML>

<proof>

<ML>

<proof>

lemma *arity_dcwit_repl_body*: $\text{arity}(\text{dcwit_repl_body_fm}(6, 5, 4, 3, 2, 0, 1)) = 7$

<proof>

definition *fst2_snd2*

where $\text{fst2_snd2}(x) \equiv \langle \text{fst}(\text{fst}(x)), \text{snd}(\text{snd}(x)) \rangle$

<ML>

lemma (**in** *M_trivial*) *fst2_snd2_abs*:

assumes $M(x) \ M(\text{res})$

shows $\text{is_fst2_snd2}(M, x, \text{res}) \longleftrightarrow \text{res} = \text{fst2_snd2}(x)$

<proof>

<ML>

definition *sndfst_fst2_snd2*

where $\text{sndfst_fst2_snd2}(x) \equiv \langle \text{snd}(\text{fst}(x)), \text{fst}(\text{fst}(x)), \text{snd}(\text{snd}(x)) \rangle$

<ML>

definition *order_eq_map* **where**

$\text{order_eq_map}(M, A, r, a, z) \equiv \exists x[M]. \exists g[M]. \exists mx[M]. \exists par[M].$

$\text{ordinal}(M, x) \ \& \ \text{pair}(M, a, x, z) \ \& \ \text{membership}(M, x, mx) \ \&$

$\text{pred_set}(M, A, a, r, par) \ \& \ \text{order_isomorphism}(M, par, r, x, mx, g)$

<ML>

definition *banach_body_iterates* **where**

$\text{banach_body_iterates}(M, X, Y, f, g, W, n, x, z) \equiv \exists y[M].$

$\text{pair}(M, x, y, z) \wedge$

$(\exists fa[M].$

$(\forall z[M].$

$z \in fa \longleftrightarrow$

$(\exists xa[M].$

$\exists y[M].$

$\exists xaa[M].$

$\exists sx[M].$

$\exists r_sx[M].$

$$\begin{aligned}
& \exists f_r_sx[M]. \exists sn[M]. \exists msn[M]. \text{successor}(M, n, sn) \\
\wedge & \\
& \text{membership}(M, sn, msn) \wedge \\
& \text{pair}(M, xa, y, z) \wedge \\
& \text{pair}(M, xa, x, xaa) \wedge \\
& \text{upair}(M, xa, xa, sx) \wedge \\
& \text{pre_image}(M, msn, sx, r_sx) \wedge \\
& \text{restriction}(M, fa, r_sx, f_r_sx) \wedge \\
& xaa \in msn \wedge \\
& (\text{empty}(M, xa) \longrightarrow y = W) \wedge \\
& (\forall m[M]. \\
& \quad \text{successor}(M, m, xa) \longrightarrow \\
& \quad (\exists gm[M]. \\
& \quad \quad \text{is_apply}(M, f_r_sx, m, gm) \wedge \\
& \quad \quad \text{is_banach_functor}(M, X, Y, f, g, gm, y))) \wedge \\
& \quad (\text{is_quasimat}(M, xa) \vee \text{empty}(M, y))) \wedge \\
& (\text{empty}(M, x) \longrightarrow y = W) \wedge \\
& (\forall m[M]. \\
& \quad \text{successor}(M, m, x) \longrightarrow \\
& \quad (\exists gm[M]. \text{is_apply}(M, fa, m, gm) \wedge \text{is_banach_functor}(M, \\
& X, Y, f, g, gm, y))) \wedge \\
& \quad (\text{is_quasimat}(M, x) \vee \text{empty}(M, y)))
\end{aligned}$$

$\langle ML \rangle$

definition *banach_is_iterates_body* where

banach_is_iterates_body(M, X, Y, f, g, W, n, y) $\equiv \exists om[M]. \text{omega}(M, om) \wedge n \in om \wedge$

$$\begin{aligned}
& (\exists sn[M]. \\
& \quad \exists msn[M]. \\
& \quad \quad \text{successor}(M, n, sn) \wedge \\
& \quad \quad \text{membership}(M, sn, msn) \wedge \\
& \quad (\exists fa[M]. \\
& \quad \quad (\forall z[M]. \\
& \quad \quad \quad z \in fa \longleftrightarrow \\
& \quad \quad \quad (\exists x[M]. \\
& \quad \quad \quad \quad \exists y[M]. \\
& \quad \quad \quad \quad \quad \exists xa[M]. \\
& \quad \quad \quad \quad \quad \quad \exists sx[M]. \\
& \quad \quad \quad \quad \quad \quad \quad \exists r_sx[M]. \\
& \quad \quad \quad \quad \quad \quad \quad \quad \exists f_r_sx[M]. \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{pair}(M, x, y, z) \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{pair}(M, x, n, xa) \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{upair}(M, x, x, sx) \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{pre_image}(M, msn, sx, r_sx) \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{restriction}(M, fa, r_sx, f_r_sx) \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad xa \in msn \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (\text{empty}(M, x) \longrightarrow y = W) \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (\forall m[M].
\end{aligned}$$

$$\begin{aligned}
& \text{successor}(M, m, x) \longrightarrow \\
& (\exists gm[M]. \\
& \quad \text{fun_apply}(M, f_r_sx, m, gm) \wedge \\
\text{is_banach_functor}(M, X, Y, f, g, gm, y))) \wedge \\
& \quad (\text{is_quasinat}(M, x) \vee \text{empty}(M, y))) \wedge \\
& (\text{empty}(M, n) \longrightarrow y = W) \wedge \\
& (\forall m[M]. \\
& \quad \text{successor}(M, m, n) \longrightarrow \\
& \quad (\exists gm[M]. \text{fun_apply}(M, fa, m, gm) \wedge \text{is_banach_functor}(M, \\
& X, Y, f, g, gm, y))) \wedge \\
& \quad (\text{is_quasinat}(M, n) \vee \text{empty}(M, y)))
\end{aligned}$$

$\langle ML \rangle$

definition *trans_apply_image* **where**

$$\text{trans_apply_image}(f) \equiv \lambda a. g. f \text{ ' } (g \text{ ' } a)$$

$\langle ML \rangle$

schematic_goal *arity_is_recfun_fm*[arity]:

$$\begin{aligned}
& p \in \text{formula} \implies a \in \omega \implies z \in \omega \implies r \in \omega \implies \text{arity}(\text{is_recfun_fm}(p, a, z, r)) \\
& = ?ar \\
& \langle \text{proof} \rangle
\end{aligned}$$

schematic_goal *arity_is_wfrec_fm*[arity]:

$$\begin{aligned}
& p \in \text{formula} \implies a \in \omega \implies z \in \omega \implies r \in \omega \implies \text{arity}(\text{is_wfrec_fm}(p, a, z, r)) \\
& = ?ar \\
& \langle \text{proof} \rangle
\end{aligned}$$

schematic_goal *arity_is_transrec_fm*[arity]:

$$\begin{aligned}
& p \in \text{formula} \implies a \in \omega \implies z \in \omega \implies \text{arity}(\text{is_transrec_fm}(p, a, z)) = ?ar \\
& \langle \text{proof} \rangle
\end{aligned}$$

$\langle ML \rangle$

definition *transrec_apply_image_body* **where**

$$\begin{aligned}
\text{transrec_apply_image_body}(M, f, mesa, x, z) \equiv & \exists y[M]. \text{pair}(M, x, y, z) \wedge \\
& (\exists fa[M]. \\
& \quad (\forall z[M]. \\
& \quad \quad z \in fa \longleftrightarrow \\
& \quad \quad (\exists xa[M]. \\
& \quad \quad \quad \exists y[M]. \\
& \quad \quad \quad \quad \exists xaa[M]. \\
& \quad \quad \quad \quad \quad \exists sx[M]. \\
& \quad \quad \quad \quad \quad \quad \exists r_sx[M]. \\
& \quad \quad \quad \quad \quad \quad \quad \exists f_r_sx[M].
\end{aligned}$$

$$\begin{aligned}
& \text{pair}(M, xa, y, z) \wedge \\
& \text{pair}(M, xa, x, xaa) \wedge \\
& \text{upair}(M, xa, xa, sx) \wedge \\
& \text{pre_image}(M, mesa, sx, r_sx) \wedge \\
& \text{restriction}(M, fa, r_sx, f_r_sx) \wedge \\
& xaa \in mesa \wedge \text{is_trans_apply_image}(M, \\
& f, xa, f_r_sx, y))) \wedge \\
& \text{is_trans_apply_image}(M, f, x, fa, y))
\end{aligned}$$

$\langle ML \rangle$

definition *is_trans_apply_image_body* **where**

$$\begin{aligned}
\text{is_trans_apply_image_body}(M, f, \beta, a, w) \equiv & \exists z[M]. \text{pair}(M, a, z, w) \wedge a \in \beta \wedge (\exists sa[M]. \\
& \exists esa[M]. \\
& \exists mesa[M]. \\
& \text{upair}(M, a, a, sa) \wedge \\
& \text{is_eclose}(M, sa, esa) \wedge \\
& \text{membership}(M, esa, mesa) \wedge \\
& (\exists fa[M]. \\
& (\forall z[M]. \\
& z \in fa \longleftrightarrow \\
& (\exists x[M]. \\
& \exists y[M]. \\
& \exists xa[M]. \\
& \exists sx[M]. \\
& \exists r_sx[M]. \\
& \exists f_r_sx[M]. \\
& \text{pair}(M, x, y, z) \wedge \\
& \text{pair}(M, x, a, xa) \wedge \\
& \text{upair}(M, x, x, sx) \wedge \\
& \text{pre_image}(M, mesa, sx, r_sx) \wedge \\
& \text{restriction}(M, fa, r_sx, f_r_sx) \wedge \\
& xa \in mesa \wedge \text{is_trans_apply_image}(M, f, \\
& x, f_r_sx, y))) \wedge \\
& \text{is_trans_apply_image}(M, f, a, fa, z)))
\end{aligned}$$

$\langle ML \rangle$

definition *replacement_is_omega_funspace_fm* **where** *replacement_is_omega_funspace_fm*
 $\equiv \text{omega_funspace_fm}(2, 0, 1)$

definition *wfrec_Aleph_fm* **where** *wfrec_Aleph_fm* $\equiv \text{HAleph_wfrec_repl_body_fm}(2, 0, 1)$

definition *replacement_is_fst2_snd2_fm* **where** *replacement_is_fst2_snd2_fm*
 $\equiv \text{is_fst2_snd2_fm}(0, 1)$

definition *replacement_is_sndfst_fst2_snd2_fm* **where** *replacement_is_sndfst_fst2_snd2_fm*
 $\equiv \text{is_sndfst_fst2_snd2_fm}(0, 1)$

definition *omap_replacement_fm* **where** *omap_replacement_fm* $\equiv \text{order_eq_map_fm}(2, 3, 0, 1)$

definition *rec_constr_abs_fm* **where** *rec_constr_abs_fm* $\equiv \text{transrec_apply_image_body_fm}(3, 2, 0, 1)$

definition *banach_replacement_iterates_fm* **where** *banach_replacement_iterates_fm*

\equiv *banach_is_iterates_body_fm*(6,5,4,3,2,0,1)

definition *rec_constr_fm* **where** *rec_constr_fm* \equiv *is_trans_apply_image_body_fm*(3,2,0,1)

definition *dc_abs_fm* **where** *dc_abs_fm* \equiv *dcwit_repl_body_fm*(6,5,4,3,2,0,1)

definition *lam_replacement_check_fm* **where** *lam_replacement_check_fm* \equiv *Lambda_in_M_fm*(*check_fm*)

The following instances are needed only on the ground model. The first one corresponds to the recursive definition of forces for atomic formulas; the next two corresponds to *PHcheck*; the following is used to get a generic filter using some form of choice.

locale *M_ZF_ground* = *M_ZF1* +
assumes

ZF_ground_replacements:
replacement_assm(*M*,*env*,*wfrec_Hfrc_at_fm*)
replacement_assm(*M*,*env*,*wfrec_Hcheck_fm*)
replacement_assm(*M*,*env*,*lam_replacement_check_fm*)

locale *M_ZF_ground_trans* = *M_ZF1_trans* + *M_ZF_ground*

definition *instances_ground_fms* **where** *instances_ground_fms* \equiv
{ *wfrec_Hfrc_at_fm*,
wfrec_Hcheck_fm,
lam_replacement_check_fm }

lemmas *replacement_instances_ground_defs* =
wfrec_Hfrc_at_fm_def *wfrec_Hcheck_fm_def* *lam_replacement_check_fm_def*

declare (**in** *M_ZF_ground*) *replacement_instances_ground_defs* [*simp*]

lemma *instances_ground_fms_type*[*TC*]: *instances_ground_fms* \subseteq *formula*
{*proof*}

locale *M_ZF_ground_notCH* = *M_ZF_ground* +
assumes

ZF_ground_notCH_replacements:
replacement_assm(*M*,*env*,*rec_constr_abs_fm*)
replacement_assm(*M*,*env*,*rec_constr_fm*)

definition *instances_ground_notCH_fms* **where** *instances_ground_notCH_fms*
 \equiv
{ *rec_constr_abs_fm*,
rec_constr_fm }

lemma *instances_ground_notCH_fms_type*[*TC*]: *instances_ground_notCH_fms*
 \subseteq *formula*
{*proof*}

declare (**in** *M_ZF_ground_notCH*) *rec_constr_abs_fm_def*[*simp*]
rec_constr_fm_def[*simp*]

```

locale  $M\_ZF\_ground\_notCH\_trans = M\_ZF\_ground\_trans + M\_ZF\_ground\_notCH$ 

locale  $M\_ZF\_ground\_CH = M\_ZF\_ground\_notCH +$ 
  assumes
     $dcwit\_replacement: replacement\_assm(M, env, dc\_abs\_fm)$ 

declare (in  $M\_ZF\_ground\_CH$ )  $dc\_abs\_fm\_def [simp]$ 

locale  $M\_ZF\_ground\_CH\_trans = M\_ZF\_ground\_notCH\_trans + M\_ZF\_ground\_CH$ 

locale  $M\_ctm1 = M\_ZF1\_trans + M\_ZF\_ground\_trans +$ 
  fixes  $enum$ 
  assumes  $M\_countable: enum \in bij(nat, M)$ 

locale  $M\_ctm1\_AC = M\_ctm1 + M\_ZFC1\_trans$ 

context  $M\_ZF\_ground\_CH\_trans$ 
begin

lemma  $replacement\_dcwit\_repl\_body:$ 
   $(\#\#M)(mesa) \implies (\#\#M)(A) \implies (\#\#M)(a) \implies (\#\#M)(s) \implies (\#\#M)(R)$ 
 $\implies$ 
   $strong\_replacement(\#\#M, dcwit\_repl\_body(\#\#M, mesa, A, a, s, R))$ 
   $\langle proof \rangle$ 

lemma  $dcwit\_repl:$ 
   $(\#\#M)(sa) \implies$ 
   $(\#\#M)(esa) \implies$ 
   $(\#\#M)(mesa) \implies (\#\#M)(A) \implies (\#\#M)(a) \implies (\#\#M)(s) \implies$ 
 $(\#\#M)(R) \implies$ 
   $strong\_replacement$ 
   $((\#\#M), \lambda x z. \exists y[(\#\#M)]. pair((\#\#M), x, y, z) \wedge$ 
   $is\_wfrec$ 
   $((\#\#M), \lambda n f. is\_nat\_case$ 
   $((\#\#M), a,$ 
   $\lambda m bmfm.$ 
   $\exists fm[(\#\#M)].$ 
   $\exists cp[(\#\#M)].$ 
   $is\_apply((\#\#M), f, m, fm) \wedge$ 
   $is\_Collect((\#\#M), A, \lambda x. \exists fmx[(\#\#M)].$ 
 $((\#\#M)(x) \wedge fmx \in R) \wedge pair((\#\#M), fm, x, fmx), cp) \wedge$ 
   $is\_apply((\#\#M), s, cp, bmfm),$ 
   $n),$ 
   $mesa, x, y))$ 
   $\langle proof \rangle$ 

end —  $M\_ZF\_ground\_CH\_trans$ 

```

context M_ZF1_trans
begin

lemmas $M_replacement_ZF_instances = lam_replacement_fst lam_replacement_snd$
 $lam_replacement_Union lam_replacement_Image$
 $lam_replacement_middle_del lam_replacement_prodRepl$

lemmas $M_separation_ZF_instances = separation_fstsnd_in_sndsnd separation_sndfst_eq_fstsnd$

lemma $separation_is_dcwit_body$:
assumes $(\#\#M)(A) (\#\#M)(a) (\#\#M)(g) (\#\#M)(R)$
shows $separation(\#\#M, is_dcwit_body(\#\#M, A, a, g, R))$
 $\langle proof \rangle$

end — M_ZF1_trans

sublocale $M_ZF1_trans \subseteq M_replacement \#\#M$
 $\langle proof \rangle$

context M_ZF1_trans
begin

lemma $separation_Pow_rel$: $A \in M \implies$
 $separation(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, Pow^{\#\#M}(x) \rangle)$
 $\langle proof \rangle$

lemma $strong_replacement_Powapply_rel$:
 $f \in M \implies strong_replacement(\#\#M, \lambda x y. y = Powapply^{\#\#M}(f, x))$
 $\langle proof \rangle$

end — M_ZF1_trans

sublocale $M_ZF1_trans \subseteq M_Vfrom \#\#M$
 $\langle proof \rangle$

sublocale $M_ZF1_trans \subseteq M_Perm \#\#M$
 $\langle proof \rangle$

sublocale $M_ZF1_trans \subseteq M_pre_seqspace \#\#M$
 $\langle proof \rangle$

context M_ZF1_trans
begin

lemma $separation_inj_rel$: $A \in M \implies$
 $separation(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, inj_rel(\#\#M, fst(x), snd(x)) \rangle)$
 $\langle proof \rangle$

lemma $lam_replacement_inj_rel$: $lam_replacement(\#\#M, \lambda x. inj_rel(\#\#M, fst(x), snd(x)))$

<proof>

end — *M_ZF1_trans*

lemma (in *M_basic*) *rel2_trans_apply*:

$M(f) \implies \text{relation2}(M, \text{is_trans_apply_image}(M, f), \text{trans_apply_image}(f))$
<proof>

lemma (in *M_basic*) *apply_image_closed*:

shows $M(f) \implies \forall x[M]. \forall g[M]. M(\text{trans_apply_image}(f, x, g))$
<proof>

context *M_ZF_ground_notCH_trans*

begin

lemma *replacement_transrec_apply_image_body* :

$(\#\#M)(f) \implies (\#\#M)(\text{mesa}) \implies \text{strong_replacement}(\#\#M, \text{transrec_apply_image_body}(\#\#M, f, \text{mesa}))$
<proof>

lemma *transrec_replacement_apply_image*:

assumes $(\#\#M)(f) (\#\#M)(\alpha)$
shows $\text{transrec_replacement}(\#\#M, \text{is_trans_apply_image}(\#\#M, f), \alpha)$
<proof>

lemma *rec_trans_apply_image_abs*:

assumes $(\#\#M)(f) (\#\#M)(x) (\#\#M)(y) \text{Ord}(x)$
shows $\text{is_transrec}(\#\#M, \text{is_trans_apply_image}(\#\#M, f), x, y) \longleftrightarrow y = \text{transrec}(x, \text{trans_apply_image}(f))$
<proof>

lemma *replacement_is_trans_apply_image*:

$(\#\#M)(f) \implies (\#\#M)(\beta) \implies \text{strong_replacement}(\#\#M, \lambda x z . \exists y[\#\#M]. \text{pair}(\#\#M, x, y, z) \wedge x \in \beta \wedge (\text{is_transrec}(\#\#M, \text{is_trans_apply_image}(\#\#M, f), x, y)))$
<proof>

lemma *trans_apply_abs*:

$(\#\#M)(f) \implies (\#\#M)(\beta) \implies \text{Ord}(\beta) \implies (\#\#M)(x) \implies (\#\#M)(z) \implies (x \in \beta \wedge z = \langle x, \text{transrec}(x, \lambda a g. f \text{ ‘ ‘ } (g \text{ ‘ ‘ } a) \rangle) \longleftrightarrow (\exists y[\#\#M]. \text{pair}(\#\#M, x, y, z) \wedge x \in \beta \wedge (\text{is_transrec}(\#\#M, \text{is_trans_apply_image}(\#\#M, f), x, y)))$
<proof>

lemma *replacement_trans_apply_image*:

$(\#\#M)(f) \implies (\#\#M)(\beta) \implies \text{Ord}(\beta) \implies \text{strong_replacement}(\#\#M, \lambda x y. x \in \beta \wedge y = \langle x, \text{transrec}(x, \lambda a g. f \text{ ‘ ‘ } (g \text{ ‘ ‘ } a) \rangle)$
<proof>

end — $M_ZF_ground_notCH_trans$

definition $ifrFb_body$ **where**

$ifrFb_body(M,b,f,x,i) \equiv x \in$

(if $b = 0$ then if $i \in range(f)$ then

if $M(\text{converse}(f) \text{ ' } i)$ then $\text{converse}(f) \text{ ' } i$ else 0 else 0 else if $M(i)$ then i else 0)

$\langle ML \rangle$

definition $ifrangeF_body :: [i \Rightarrow o, i, i, i, i] \Rightarrow o$ **where**

$ifrangeF_body(M,A,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb_body(M,b,f,x,i) \rangle$

$\langle ML \rangle$

lemma (in M_Z_trans) $separation_is_ifrangeF_body$:

$(\#\#M)(A) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(s) \Longrightarrow separation(\#\#M, is_ifrangeF_body(\#\#M,A,r,s))$

$\langle proof \rangle$

lemma (in M_basic) $is_ifrFb_body_closed: M(r) \Longrightarrow M(s) \Longrightarrow is_ifrFb_body(M, r, s, x, i) \Longrightarrow M(i)$

$\langle proof \rangle$

lemma (in M_ZF1_trans) $ifrangeF_body_abs$:

assumes $(\#\#M)(A) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$

shows $is_ifrangeF_body(\#\#M,A,r,s,x) \longleftrightarrow ifrangeF_body(\#\#M,A,r,s,x)$

$\langle proof \rangle$

lemma (in M_ZF1_trans) $separation_ifrangeF_body$:

$(\#\#M)(A) \Longrightarrow (\#\#M)(b) \Longrightarrow (\#\#M)(f) \Longrightarrow separation$

$(\#\#M, \lambda y. \exists x \in A. y = \langle x, \mu i. x \in if_range_F_else_F(\lambda x. if (\#\#M)(x) \text{ then } x \text{ else } 0, b, f, i) \rangle)$

$\langle proof \rangle$

definition $ifrFb_body2$ **where**

$ifrFb_body2(M,G,b,f,x,i) \equiv x \in$

(if $b = 0$ then if $i \in range(f)$ then

if $M(\text{converse}(f) \text{ ' } i)$ then $G(\text{converse}(f) \text{ ' } i)$ else 0 else 0 else if $M(i)$ then $G i$ else 0)

$\langle ML \rangle$

definition $ifrangeF_body2 :: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$ **where**

$ifrangeF_body2(M,A,G,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb_body2(M,G,b,f,x,i) \rangle$

$\langle ML \rangle$

lemma (in M_Z_trans) *separation_is_ifrangeF_body2*:

$(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(r) \implies (\#\#M)(s) \implies separation(\#\#M, is_ifrangeF_body2(\#\#M,A,G,r,s))$
 ⟨proof⟩

lemma (in M_basic) *is_ifrFb_body2_closed*: $M(G) \implies M(r) \implies M(s) \implies is_ifrFb_body2(M, G, r, s, x, i) \implies M(i)$

⟨proof⟩

lemma (in M_ZF1_trans) *ifrangeF_body2_abs*:

assumes $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$

shows $is_ifrangeF_body2(\#\#M,A,G,r,s,x) \longleftrightarrow ifrangeF_body2(\#\#M,A,G,r,s,x)$
 ⟨proof⟩

lemma (in M_ZF1_trans) *separation_ifrangeF_body2*:

$(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(b) \implies (\#\#M)(f) \implies$

separation

$(\#\#M,$

$\lambda y. \exists x \in A.$

$y =$

$\langle x, \mu i. x \in$

$if_range_F_else_F(\lambda a. if (\#\#M)(a) then G \text{ ‘ } a \text{ else } 0, b, f,$

$i \rangle \rangle)$

⟨proof⟩

definition *ifrFb_body3* **where**

$ifrFb_body3(M,G,b,f,x,i) \equiv x \in$

$(if\ b = 0\ then\ if\ i \in range(f)\ then$

$if\ M(converse(f)\ \text{‘}\ i)\ then\ G\ \text{‘}\ \{converse(f)\ \text{‘}\ i\}\ else\ 0\ else\ 0\ else\ if\ M(i)\ then\ G\ \text{‘}\ \{i\}\ else\ 0)$

⟨ML⟩

definition *ifrangeF_body3* :: $[i \Rightarrow o, i, i, i, i, i] \Rightarrow o$ **where**

$ifrangeF_body3(M,A,G,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb_body3(M,G,b,f,x,i) \rangle$

⟨ML⟩

lemma (in M_Z_trans) *separation_is_ifrangeF_body3*:

$(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(r) \implies (\#\#M)(s) \implies separation(\#\#M, is_ifrangeF_body3(\#\#M,A,G,r,s))$

⟨proof⟩

lemma (in M_basic) *is_ifrFb_body3_closed*: $M(G) \implies M(r) \implies M(s) \implies is_ifrFb_body3(M, G, r, s, x, i) \implies M(i)$

⟨proof⟩

lemma (in M_ZF1_trans) *ifrangeF_body3_abs*:
assumes $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$
shows $is_ifrangeF_body3(\#\#M,A,G,r,s,x) \longleftrightarrow ifrangeF_body3(\#\#M,A,G,r,s,x)$
 $\langle proof \rangle$

lemma (in M_ZF1_trans) *separation_ifrangeF_body3*:
 $(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(b) \implies (\#\#M)(f) \implies$
separation
 $(\#\#M,$
 $\lambda y. \exists x \in A.$
 $y =$
 $\langle x, \mu i. x \in$
 $if_range_F_else_F(\lambda a. if (\#\#M)(a) then G^{-}\{a\} else 0, b,$
 $f, i) \rangle)$
 $\langle proof \rangle$

definition *ifrFb_body4* **where**
 $ifrFb_body4(G,b,f,x,i) \equiv x \in$
 $(if b = 0 then if i \in range(f) then G^{-}(converse(f) \text{ ` } i) else 0 else G^{-}i)$

$\langle ML \rangle$

definition *ifrangeF_body4* $:: [i \Rightarrow o, i, i, i, i] \Rightarrow o$ **where**
 $ifrangeF_body4(M,A,G,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb_body4(G,b,f,x,i) \rangle$

$\langle ML \rangle$

lemma (in M_Z_trans) *separation_is_ifrangeF_body4*:
 $(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(r) \implies (\#\#M)(s) \implies separation(\#\#M,$
 $is_ifrangeF_body4(\#\#M,A,G,r,s))$
 $\langle proof \rangle$

lemma (in M_basic) *is_ifrFb_body4_closed*: $M(G) \implies M(r) \implies M(s) \implies$
 $is_ifrFb_body4(M, G, r, s, x, i) \implies M(i)$
 $\langle proof \rangle$

lemma (in M_ZF1_trans) *ifrangeF_body4_abs*:
assumes $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$
shows $is_ifrangeF_body4(\#\#M,A,G,r,s,x) \longleftrightarrow ifrangeF_body4(\#\#M,A,G,r,s,x)$
 $\langle proof \rangle$

lemma (in M_ZF1_trans) *separation_ifrangeF_body4*:
 $(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(b) \implies (\#\#M)(f) \implies$
 $separation(\#\#M, \lambda y. \exists x \in A. y = \langle x, \mu i. x \in if_range_F_else_F((\text{ ` })(G),$
 $b, f, i) \rangle)$
 $\langle proof \rangle$

definition *ifrFb_body5* **where**

$ifrFb_body5(G,b,f,x,i) \equiv x \in$
(if $b = 0$ *then if* $i \in \text{range}(f)$ *then* $\{xa \in G . \text{converse}(f) \text{ ' } i \in xa\}$ *else* 0 *else* $\{xa \in G . i \in xa\}$ *)*

$\langle ML \rangle$

definition *ifrangeF_body5* $:: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$ **where**

$ifrangeF_body5(M,A,G,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb_body5(G,b,f,x,i) \rangle$

$\langle ML \rangle$

lemma (in *M_Z_trans*) *separation_is_ifrangeF_body5*:

$(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(s) \Longrightarrow \text{separation}(\#\#M,$
is_ifrangeF_body5 $(\#\#M,A,G,r,s))$
 $\langle proof \rangle$

lemma (in *M_basic*) *is_ifrFb_body5_closed*: $M(G) \Longrightarrow M(r) \Longrightarrow M(s) \Longrightarrow$
is_ifrFb_body5 $(M, G, r, s, x, i) \Longrightarrow M(i)$

$\langle proof \rangle$

lemma (in *M_ZF1_trans*) *ifrangeF_body5_abs*:

assumes $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$

shows $is_ifrangeF_body5(\#\#M,A,G,r,s,x) \longleftrightarrow ifrangeF_body5(\#\#M,A,G,r,s,x)$
 $\langle proof \rangle$

lemma (in *M_ZF1_trans*) *separation_ifrangeF_body5*:

$(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(b) \Longrightarrow (\#\#M)(f) \Longrightarrow$
 $\text{separation}(\#\#M, \lambda y. \exists x \in A. y = \langle x, \mu i. x \in if_range_F_else_F(\lambda x. \{xa \in G . x \in xa\}, b, f, i) \rangle)$
 $\langle proof \rangle$

definition *ifrFb_body6* **where**

$ifrFb_body6(G,b,f,x,i) \equiv x \in$
(if $b = 0$ *then if* $i \in \text{range}(f)$ *then* $\{p \in G . \text{domain}(p) = \text{converse}(f) \text{ ' } i\}$ *else* 0 *else* $\{p \in G . \text{domain}(p) = i\}$ *)*

$\langle ML \rangle$

definition *ifrangeF_body6* $:: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$ **where**

$ifrangeF_body6(M,A,G,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb_body6(G,b,f,x,i) \rangle$

$\langle ML \rangle$

lemma (in *M_Z_trans*) *separation_is_ifrangeF_body6*:

$(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(r) \implies (\#\#M)(s) \implies \text{separation}(\#\#M,$
 $\text{is_ifrangeF_body6}(\#\#M,A,G,r,s))$
 ⟨proof⟩

lemma (in M_basic) $\text{ifrFb_body6_closed}: M(G) \implies M(r) \implies M(s) \implies \text{ifrFb_body6}(G,$
 $r, s, x, i) \longleftrightarrow M(i) \wedge \text{ifrFb_body6}(G, r, s, x, i)$
 ⟨proof⟩

lemma (in M_basic) $\text{is_ifrFb_body6_closed}: M(G) \implies M(r) \implies M(s) \implies$
 $\text{is_ifrFb_body6}(M, G, r, s, x, i) \implies M(i)$
 ⟨proof⟩

lemma (in M_ZF1_trans) $\text{ifrangeF_body6_abs}$:
assumes $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$
shows $\text{is_ifrangeF_body6}(\#\#M,A,G,r,s,x) \longleftrightarrow \text{ifrangeF_body6}(\#\#M,A,G,r,s,x)$
 ⟨proof⟩

lemma (in M_ZF1_trans) $\text{separation_ifrangeF_body6}$:
 $(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(b) \implies (\#\#M)(f) \implies$
 $\text{separation}(\#\#M,$
 $\lambda y. \exists x \in A. y = \langle x, \mu i. x \in \text{if_range_F_else_F}(\lambda a. \{p \in G . \text{domain}(p) =$
 $a\}, b, f, i))$
 ⟨proof⟩

definition ifrFb_body7 **where**

$\text{ifrFb_body7}(B,D,A,b,f,x,i) \equiv x \in$
 $(\text{if } b = 0 \text{ then if } i \in \text{range}(f) \text{ then}$
 $\{d \in D . \exists r \in A. \text{restrict}(r, B) = \text{converse}(f) \text{ ' } i \wedge d = \text{domain}(r)\} \text{ else } 0$
 $\text{else } \{d \in D . \exists r \in A. \text{restrict}(r, B) = i \wedge d = \text{domain}(r)\})$

⟨ML⟩

definition $\text{ifrangeF_body7} :: [i \Rightarrow o, i, i, i, i, i, i] \Rightarrow o$ **where**

$\text{ifrangeF_body7}(M,A,B,D,G,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. \text{ifrFb_body7}(B,D,G,b,f,x,i) \rangle$

⟨ML⟩

lemma (in M_Z_trans) $\text{separation_is_ifrangeF_body7}$:

$(\#\#M)(A) \implies (\#\#M)(B) \implies (\#\#M)(D) \implies (\#\#M)(G) \implies (\#\#M)(r)$
 $\implies (\#\#M)(s) \implies \text{separation}(\#\#M, \text{is_ifrangeF_body7}(\#\#M,A,B,D,G,r,s))$
 ⟨proof⟩

lemma (in M_basic) $\text{ifrFb_body7_closed}: M(B) \implies M(D) \implies M(G) \implies M(r)$
 $\implies M(s) \implies$
 $\text{ifrFb_body7}(B,D,G, r, s, x, i) \longleftrightarrow M(i) \wedge \text{ifrFb_body7}(B,D,G, r, s, x, i)$

<proof>

lemma (in *M_basic*) *is_ifrFb_body7_closed*: $M(B) \implies M(D) \implies M(G) \implies M(r) \implies M(s) \implies$
 $is_ifrFb_body7(M, B, D, G, r, s, x, i) \implies M(i)$
<proof>

lemma (in *M_ZF1_trans*) *ifrangeF_body7_abs*:
assumes $(\#\#M)(A) (\#\#M)(B) (\#\#M)(D) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s)$
 $(\#\#M)(x)$
shows $is_ifrangeF_body7(\#\#M, A, B, D, G, r, s, x) \longleftrightarrow ifrangeF_body7(\#\#M, A, B, D, G, r, s, x)$
<proof>

lemma (in *M_ZF1_trans*) *separation_ifrangeF_body7*:
 $(\#\#M)(A) \implies (\#\#M)(B) \implies (\#\#M)(D) \implies (\#\#M)(G) \implies (\#\#M)(b) \implies$
 $(\#\#M)(f) \implies$
 $separation(\#\#M,$
 $\lambda y. \exists x \in A. y = \langle x, \mu i. x \in if_range_F_else_F(drSR_Y(B, D, G), b, f, i) \rangle)$
<proof>

definition *omfunspace* :: $[i, i] \Rightarrow o$ **where**
 $omfunspace(B) \equiv \lambda z. \exists x. \exists n \in \omega. z \in x \wedge x = n \rightarrow B$
<ML>

context *M_pre_seqspace*
begin

<ML>
<proof>

end — *M_pre_seqspace*

context *M_ZF1_trans*
begin

lemma *separation_omfunspace*:
assumes $(\#\#M)(B)$
shows $separation(\#\#M, \lambda z. \exists x[\#\#M]. \exists n[\#\#M]. n \in \omega \wedge z \in x \wedge x = n \rightarrow^M B)$
<proof>

end — *M_ZF1_trans*

sublocale $M_ZF1_trans \subseteq M_seqspace \#\#M$
<proof>

definition *cdltgamma* :: $[i, i] \Rightarrow o$ **where**
 $cdltgamma(\gamma) \equiv \lambda Z. |Z| < \gamma$
<ML>

```

definition cdeggamma :: [i] ⇒ o where
  cdeggamma ≡ λZ . |fst(Z)| = snd(Z)
⟨ML⟩

context M_Perm
begin

⟨ML⟩
  ⟨proof⟩

⟨ML⟩
  ⟨proof⟩

lemma is_cdeggamma_iff_split: M(Z) ⇒ cdeggamma_rel(M, Z) ↔ (λ⟨x,y⟩.
|x|M = y)(Z)
  ⟨proof⟩

end

context M_ZF1_trans
begin

lemma separation_cdltgamma:
  assumes (##M)(γ)
  shows separation(##M, λZ . cardinal_rel(##M,Z) < γ)
  ⟨proof⟩

lemma separation_cdeggamma:
  shows separation(##M, λZ. (λ⟨x,y⟩ . cardinal_rel(##M,x) = y)(Z))
  ⟨proof⟩

end — M_ZF1_trans

end

```

9 Further instances of axiom-schemes

```

theory ZF_Trans_Interpretations
  imports
    Internal_ZFC_Axioms
    Replacement_Instances

begin

locale M_ZF2 = M_ZF1 +
  assumes
    replacement_ax2:
    replacement_assm(M, env, ordtype_replacement_fm)

```

```

replacement_assm(M,env,wfrec_ordertype_fm)
replacement_assm(M,env,wfrec_Aleph_fm)
replacement_assm(M,env,omap_replacement_fm)

```

definition *instances2_fms* **where** *instances2_fms* \equiv
{ *ordtype_replacement_fm*,
wfrec_ordertype_fm,
wfrec_Aleph_fm,
omap_replacement_fm }

lemmas *replacement_instances2_defs* =
ordtype_replacement_fm_def wfrec_ordertype_fm_def
wfrec_Aleph_fm_def omap_replacement_fm_def

declare (in *M_ZF2*) *replacement_instances2_defs* [*simp*]

locale *M_ZF2_trans* = *M_ZF1_trans* + *M_ZF2*

locale *M_ZFC2* = *M_ZFC1* + *M_ZF2*

locale *M_ZFC2_trans* = *M_ZFC1_trans* + *M_ZF2_trans* + *M_ZFC2*

locale *M_ZF2_ground_notCH* = *M_ZF2* + *M_ZF_ground_notCH*

locale *M_ZF2_ground_notCH_trans* = *M_ZF2_trans* + *M_ZF2_ground_notCH*
+ *M_ZF_ground_notCH_trans*

locale *M_ZFC2_ground_notCH* = *M_ZFC2* + *M_ZF2_ground_notCH*

locale *M_ZFC2_ground_notCH_trans* = *M_ZFC2_trans* + *M_ZFC2_ground_notCH*
+ *M_ZF2_ground_notCH_trans*

locale *M_ZFC2_ground_CH_trans* = *M_ZFC2_ground_notCH_trans* + *M_ZF_ground_CH_trans*

locale *M_ctm2* = *M_ctm1* + *M_ZF2_ground_notCH_trans*

locale *M_ctm2_AC* = *M_ctm2* + *M_ctm1_AC* + *M_ZFC2_ground_notCH_trans*

locale *M_ctm2_AC_CH* = *M_ctm2_AC* + *M_ZFC2_ground_CH_trans*

lemmas (in *M_ZF1_trans*) *separation_instances* =
separation_well_ord_iso
separation_obase_equals_separation_is_obase
separation_PiP_rel separation_surjP_rel
separation_radd_body separation_rmult_body

context *M_ZF2_trans*
begin

lemma *replacement_HAleph_wfrec_repl_body*:

$B \in M \implies \text{strong_replacement}(\#\#M, \text{HAleph_wfrec_repl_body}(\#\#M, B))$

<proof>

lemma *HAleph_wfrec_repl*:

$(\#\#M)(sa) \implies$

$(\#\#M)(esa) \implies$

$(\#\#M)(mesa) \implies$

strong_replacement

$(\#\#M,$

$\lambda x z. \exists y[\#\#M].$

$\text{pair}(\#\#M, x, y, z) \wedge$

$(\exists f[\#\#M].$

$(\forall z[\#\#M].$

$z \in f \longleftrightarrow$

$(\exists xa[\#\#M].$

$\exists y[\#\#M].$

$\exists xaa[\#\#M].$

$\exists sx[\#\#M].$

$\exists r_sx[\#\#M].$

$\exists f_r_sx[\#\#M].$

$\text{pair}(\#\#M, xa, y, z) \wedge$

$\text{pair}(\#\#M, xa, x, xaa) \wedge$

$\text{upair}(\#\#M, xa, xa, sx) \wedge$

$\text{pre_image}(\#\#M, mesa, sx, r_sx) \wedge$

$\text{restriction}(\#\#M, f, r_sx, f_r_sx) \wedge xaa \in mesa \wedge \text{is_HAleph}(\#\#M, xa, f_r_sx, y)) \wedge$

$\text{is_HAleph}(\#\#M, x, f, y))$

<proof>

lemma *replacement_is_order_eq_map*:

$A \in M \implies r \in M \implies \text{strong_replacement}(\#\#M, \text{order_eq_map}(\#\#M, A, r))$

<proof>

end — *M_ZF2_trans*

definition *omap_wfrec_body* **where**

$\text{omap_wfrec_body}(A, r) \equiv (\cdot \exists \cdot \text{image_fm}(2, 0, 1) \wedge \text{pred_set_fm}(A \# + 9, 3, r \# + 9, 0) \cdot \cdot)$

lemma *type_omap_wfrec_body_fm* : $A \in \text{nat} \implies r \in \text{nat} \implies \text{omap_wfrec_body}(A, r) \in \text{formula}$

<proof>

lemma *arity_aux* : $A \in \text{nat} \implies r \in \text{nat} \implies \text{arity}(\text{omap_wfrec_body}(A, r)) = (9 +_\omega A)$

$\cup (9 +_\omega r)$

<proof>

lemma *arity_omap_wfrec* : $A \in \text{nat} \implies r \in \text{nat} \implies$

$\text{arity}(\text{is_wfrec_fm}(\text{omap_wfrec_body}(A, r), \text{succ}(\text{succ}(\text{succ}(r))), 1, 0)) =$

$(4+\omega A) \cup (4+\omega r)$
 $\langle proof \rangle$

lemma *arity_isordermap*: $A \in nat \implies r \in nat \implies d \in nat \implies$
 $arity(is_ordermap_fm(A,r,d)) = succ(d) \cup (succ(A) \cup succ(r))$
 $\langle proof \rangle$

lemma *arity_is_ordertype*: $A \in nat \implies r \in nat \implies d \in nat \implies$
 $arity(is_ordertype_fm(A,r,d)) = succ(d) \cup (succ(A) \cup succ(r))$
 $\langle proof \rangle$

lemma *arity_is_order_body*: $arity(is_order_body_fm(1,0)) = 2$
 $\langle proof \rangle$

lemma (in *M_ZF2_trans*) *replacement_is_order_body*:
 $strong_replacement(\#\#M, \lambda x z . \exists y[\#\#M]. is_order_body(\#\#M,x,y) \wedge z =$
 $\langle x,y \rangle)$
 $\langle proof \rangle$

definition *H_order_pred* **where**
 $H_order_pred(A,r) \equiv \lambda x f . f \text{ `` } Order.pred(A, x, r)$

$\langle ML \rangle$

lemma (in *M_basic*) *H_order_pred_abs* :
 $M(A) \implies M(r) \implies M(x) \implies M(f) \implies M(z) \implies$
 $is_H_order_pred(M,A,r,x,f,z) \longleftrightarrow z = H_order_pred(A,r,x,f)$
 $\langle proof \rangle$

$\langle ML \rangle$

lemma (in *M_ZF2_trans*) *wfrec_replacement_order_pred*:
 $A \in M \implies r \in M \implies wfrec_replacement(\#\#M, \lambda x g z . is_H_order_pred(\#\#M,A,r,x,g,z)$
 $, r)$
 $\langle proof \rangle$

lemma (in *M_ZF2_trans*) *wfrec_replacement_order_pred'*:
 $A \in M \implies r \in M \implies wfrec_replacement(\#\#M, \lambda x g z . z = H_order_pred(A,r,x,g)$
 $, r)$
 $\langle proof \rangle$

sublocale *M_ZF2_trans* \subseteq *M_pre_cardinal_arith* $\#\#M$
 $\langle proof \rangle$

definition *is_well_ord_fst_snd* **where**
 $is_well_ord_fst_snd(A,x) \equiv (\exists a[A]. \exists b[A]. is_well_ord(A,a,b) \wedge is_snd(A, x,$
 $b) \wedge is_fst(A, x, a))$

$\langle ML \rangle$

lemma (in M_ZF2_trans) *separation_well_ord*: $separation(\#\#M, \lambda x. is_well_ord(\#\#M, fst(x), snd(x)))$
 $\langle proof \rangle$

sublocale $M_ZF2_trans \subseteq M_pre_aleph \ \#\#M$
 $\langle proof \rangle$

$\langle ML \rangle$

lemma *arity_is_HAleph_fm*: $arity(is_HAleph_fm(2, 1, 0)) = 3$
 $\langle proof \rangle$

lemma *arity_is_Aleph*[*arity*]: $arity(is_Aleph_fm(0, 1)) = 2$
 $\langle proof \rangle$

definition *bex_Aleph_rel* :: $[i \Rightarrow o, i, i] \Rightarrow o$ **where**
 $bex_Aleph_rel(M, x) \equiv \lambda y. \exists z \in x. y = \aleph_z^M$

$\langle ML \rangle$

schematic_goal *sats_is_bex_Aleph_fm_auto*:
 $a \in nat \Longrightarrow c \in nat \Longrightarrow env \in list(A) \Longrightarrow$
 $a < length(env) \Longrightarrow c < length(env) \Longrightarrow 0 \in A \Longrightarrow$
 $is_bex_Aleph(\#\#A, nth(a, env), nth(c, env)) \longleftrightarrow A, env \models ?fm(a, c)$
 $\langle proof \rangle$

$\langle ML \rangle$

lemma *is_bex_Aleph_fm_type* [TC]:
 $x \in \omega \Longrightarrow z \in \omega \Longrightarrow is_bex_Aleph_fm(x, z) \in formula$
 $\langle proof \rangle$

lemma *sats_is_bex_Aleph_fm*:
 $x \in \omega \Longrightarrow$
 $z \in \omega \Longrightarrow x < length(env) \Longrightarrow z < length(env) \Longrightarrow$
 $env \in list(Aa) \Longrightarrow$
 $0 \in Aa \Longrightarrow$
 $(Aa, env \models is_bex_Aleph_fm(x, z)) \longleftrightarrow$
 $is_bex_Aleph(\#\#Aa, nth(x, env), nth(z, env))$
 $\langle proof \rangle$

lemma *is_bex_Aleph_iff_sats* [*iff_sats*]:
 $nth(x, env) = xa \Longrightarrow$
 $nth(z, env) = za \Longrightarrow$
 $x \in \omega \Longrightarrow$
 $z \in \omega \Longrightarrow x < length(env) \Longrightarrow z < length(env) \Longrightarrow$

$env \in list(Aa) \implies$
 $0 \in Aa \implies$
 $is_bex_Aleph(\#\#Aa, xa, za) \longleftrightarrow$
 $Aa, env \models is_bex_Aleph_fm(x, z)$
 <proof>

<ML>

lemma (in M_ZF1_trans) *separation_is_bex_Aleph*:
assumes $(\#\#M)(A)$
shows $separation(\#\#M, is_bex_Aleph(\#\#M, A))$
 <proof>

lemma (in M_pre_aleph) *bex_Aleph_rel_abs*:
assumes $Ord(u) M(u) M(v)$
shows $is_bex_Aleph(M, u, v) \longleftrightarrow bex_Aleph_rel(M, u, v)$
 <proof>

lemma (in M_ZF2_trans) *separation_bex_Aleph_rel*:
 $Ord(x) \implies (\#\#M)(x) \implies separation(\#\#M, bex_Aleph_rel(\#\#M, x))$
 <proof>

sublocale $M_ZF2_trans \subseteq M_aleph \#\#M$
 <proof>

sublocale $M_ZF1_trans \subseteq M_FiniteFun \#\#M$
 <proof>

sublocale $M_ZFC2_trans \subseteq M_cardinal_AC \#\#M$
 <proof>

lemma (in M_ZF1_trans) *separation_cardinal_rel_lespoll_rel*:
 $(\#\#M)(\kappa) \implies separation(\#\#M, \lambda x. x \prec^M \kappa)$
 <proof>

sublocale $M_ZFC2_trans \subseteq M_library \#\#M$
 <proof>

locale $M_ZF3 = M_ZF2 +$
assumes
 $ground_replacements3:$
 $ground_replacement_assm(M, env, ordtype_replacement_fm)$
 $ground_replacement_assm(M, env, wfrec_ordertype_fm)$
 $ground_replacement_assm(M, env, eclose_abs_fm)$
 $ground_replacement_assm(M, env, wfrec_rank_fm)$
 $ground_replacement_assm(M, env, transrec_VFrom_fm)$
 $ground_replacement_assm(M, env, eclose_closed_fm)$

$ground_replacement_assm(M, env, wfrec_Aleph_fm)$
 $ground_replacement_assm(M, env, omap_replacement_fm)$

definition $instances3_fms$ **where** $instances3_fms \equiv$
 $\{ ground_repl_fm(ordtype_replacement_fm),$
 $ground_repl_fm(wfrec_ordertype_fm),$
 $ground_repl_fm(eclose_abs_fm),$
 $ground_repl_fm(wfrec_rank_fm),$
 $ground_repl_fm(transrec_VFrom_fm),$
 $ground_repl_fm(eclose_closed_fm),$
 $ground_repl_fm(wfrec_Aleph_fm),$
 $ground_repl_fm(omap_replacement_fm) \}$

This set has 8 internalized formulas, corresponding to the total count of previous replacement instances (apart from those 5 in $instances_ground_fms$ and $instances_ground_notCH_fms$, and dc_abs_fm).

definition $overhead$ **where**
 $overhead \equiv instances1_fms \cup instances_ground_fms$

definition $overhead_notCH$ **where**
 $overhead_notCH \equiv overhead \cup instances2_fms \cup$
 $instances3_fms \cup instances_ground_notCH_fms$

definition $overhead_CH$ **where**
 $overhead_CH \equiv overhead_notCH \cup \{ dc_abs_fm \}$

Hence, the “overhead” to create a proper extension of a ctm by forcing consists of 7 replacement instances. To force $\neg CH$, 21 instances are need, and one further instance is required to force CH .

lemma $instances2_fms_type[TC] : instances2_fms \subseteq formula$
 $\langle proof \rangle$

lemma $overhead_type : overhead \subseteq formula$
 $\langle proof \rangle$

lemma $overhead_notCH_type : overhead_notCH \subseteq formula$
 $\langle proof \rangle$

lemma $overhead_CH_type : overhead_CH \subseteq formula$
 $\langle proof \rangle$

locale $M_ZF3_trans = M_ZF2_trans + M_ZF3$

locale $M_ZFC3 = M_ZFC2 + M_ZF3$

locale $M_ZFC3_trans = M_ZFC2_trans + M_ZF3_trans + M_ZFC3$

locale $M_ctm3 = M_ctm2 + M_ZF3_trans$

locale $M_ctm3_AC = M_ctm3 + M_ctm1_AC + M_ZFC3_trans$

lemma $M_satT_imp_M_ZF2$: $(M \models ZF) \implies M_ZF1(M)$
<proof>

lemma $M_satT_imp_M_ZFC1$:
shows $(M \models ZFC) \longrightarrow M_ZFC1(M)$
<proof>

lemma $M_satT_instances1_imp_M_ZF1$:
assumes $(M \models \cdot Z \cdot \cup \{\cdot Replacement(p) \cdot \cdot p \in instances1_fms \})$
shows $M_ZF1(M)$
<proof>

theorem $M_satT_imp_M_ZF_ground_trans$:
assumes $Transset(M) \ M \models \cdot Z \cdot \cup \{\cdot Replacement(p) \cdot \cdot p \in overhead\}$
shows $M_ZF_ground_trans(M)$
<proof>

theorem $M_satT_imp_M_ZF_ground_notCH_trans$:
assumes
 $Transset(M)$
 $M \models \cdot Z \cdot \cup \{\cdot Replacement(p) \cdot \cdot p \in overhead_notCH\}$
shows $M_ZF_ground_notCH_trans(M)$
<proof>

theorem $M_satT_imp_M_ZF_ground_CH_trans$:
assumes
 $Transset(M)$
 $M \models \cdot Z \cdot \cup \{\cdot Replacement(p) \cdot \cdot p \in overhead_CH \}$
shows $M_ZF_ground_CH_trans(M)$
<proof>

lemma (in M_Z_basic) $M_satT_Zermelo_fms$: $M \models \cdot Z \cdot$
<proof>

lemma (in M_ZFC1) M_satT_ZC : $M \models ZC$
<proof>

locale $M_ZF = M_Z_basic +$
assumes
 $replacement_ax: replacement_assm(M, env, \varphi)$

sublocale $M_ZF \subseteq M_ZF3$
<proof>

lemma $M_satT_imp_M_ZF$: $M \models ZF \implies M_ZF(M)$
<proof>

```

lemma (in  $M\_ZF$ )  $M\_satT\_ZF: M \models ZF$ 
  <proof>

lemma  $M\_ZF\_iff\_M\_satT: M\_ZF(M) \longleftrightarrow (M \models ZF)$ 
  <proof>

locale  $M\_ZFC = M\_ZF + M\_ZC\_basic$ 

sublocale  $M\_ZFC \subseteq M\_ZFC3$ 
  <proof>

lemma  $M\_ZFC\_iff\_M\_satT:$ 
  notes  $iff\_trans[trans]$ 
  shows  $M\_ZFC(M) \longleftrightarrow (M \models ZFC)$ 
  <proof>

lemma  $M\_satT\_imp\_M\_ZF3: (M \models ZF) \longrightarrow M\_ZF3(M)$ 
  <proof>

lemma  $M\_satT\_imp\_M\_ZFC3:$ 
  shows  $(M \models ZFC) \longrightarrow M\_ZFC3(M)$ 
  <proof>

lemma  $M\_satT\_overhead\_imp\_M\_ZF3:$ 
   $(M \models ZC \cup \{\cdot Replacement(p) \cdot \mid p \in overhead\_notCH\}) \longrightarrow M\_ZFC3(M)$ 
  <proof>

end

```

10 Transitive set models of ZF

This theory defines locales for countable transitive models of ZF , and on top of that, one that includes a forcing notion. Weakened versions of both locales are included, that only assume finitely many replacement instances.

```

theory Forcing_Data
  imports
    Forcing_Notions
    Cohen_Posets_Relative
    ZF_Trans_Interpretations
begin

```

```

no_notation Aleph ( $\aleph_{\_}$ ) [90] 90)

```

10.1 A forcing locale and generic filters

Ideally, countability should be separated from the assumption of this locale. The fact is that our present proofs of the “definition of forces” (and many

consequences) and of the lemma for “forcing a value” of function unnecessarily depend on the countability of the ground model.

```

locale forcing_data1 = forcing_notion + M_ctm1 +
  assumes P_in_M:       $\mathbb{P} \in M$ 
  and leq_in_M:       $leq \in M$ 

```

```

locale forcing_data2 = forcing_data1 + M_ctm2_AC

```

```

locale forcing_data3 = forcing_data2 + M_ctm3_AC

```

```

context forcing_data1
begin

```

```

lemma P_sub_M :  $\mathbb{P} \subseteq M$ 
  <proof>

```

definition

```

M_generic ::  $i \Rightarrow o$  where
M_generic(G)  $\equiv$  filter(G)  $\wedge$  ( $\forall D \in M. D \subseteq \mathbb{P} \wedge dense(D) \longrightarrow D \cap G \neq 0$ )

```

```

declare iff_trans [trans]

```

```

lemma M_generic_imp_filter[dest]:  $M\_generic(G) \Longrightarrow filter(G)$ 
  <proof>

```

```

lemma generic_filter_existence:
   $p \in \mathbb{P} \Longrightarrow \exists G. p \in G \wedge M\_generic(G)$ 
  <proof>

```

```

lemma one_in_M:  $\mathbf{1} \in M$ 
  <proof>

```

```

declare P_in_M [simp,intro]
declare one_in_M [simp,intro]
declare leq_in_M [simp,intro]
declare one_in_P [intro]

```

```

end — forcing_data1

```

```

locale G_generic1 = forcing_data1 +
  fixes G ::  $i$ 
  assumes generic : M_generic(G)
begin

```

```

lemma G_nonempty:  $G \neq 0$ 
  <proof>

```

```

lemma M_genericD [dest]:  $x \in G \Longrightarrow x \in \mathbb{P}$ 
  <proof>

```

lemma *M_generic_leqD* [*dest*]: $p \in G \implies q \in \mathbb{P} \implies p \preceq q \implies q \in G$
 ⟨*proof*⟩

lemma *M_generic_compatD* [*dest*]: $p \in G \implies r \in G \implies \exists q \in G. q \preceq p \wedge q \preceq r$
 ⟨*proof*⟩

lemma *M_generic_denseD* [*dest*]: $\text{dense}(D) \implies D \subseteq \mathbb{P} \implies D \in M \implies \exists q \in G. q \in D$
 ⟨*proof*⟩

lemma *G_subset_P*: $G \subseteq \mathbb{P}$
 ⟨*proof*⟩

lemma *one_in_G* : $\mathbf{1} \in G$
 ⟨*proof*⟩

lemma *G_subset_M*: $G \subseteq M$
 ⟨*proof*⟩

end — *G_generic1*

locale *G_generic1_AC* = *G_generic1* + *M_ctm1_AC*

end

11 The definition of forces

theory *Forces_Definition*

imports

Forcing_Data

begin

This is the core of our development.

11.1 The relation *frecrel*

lemma *names_belowsD*:

assumes $x \in \text{names_below}(P, z)$

obtains $f \ n1 \ n2 \ p$ **where**

$x = \langle f, n1, n2, p \rangle \ f \in 2 \ n1 \in \text{eclose}N(z) \ n2 \in \text{eclose}N(z) \ p \in P$

⟨*proof*⟩

context *forcing_data1*

begin

lemma *f_type_abs*:

$\llbracket x \in M; y \in M \rrbracket \implies \text{is_f_type}(\#\#M, x, y) \longleftrightarrow y = \text{f_type}(x)$

⟨*proof*⟩

lemma *name1_abs*:

$\llbracket x \in M; y \in M \rrbracket \implies is_name1(\#\#M, x, y) \longleftrightarrow y = name1(x)$
<proof>

lemma *snd_snd_abs*:

$\llbracket x \in M; y \in M \rrbracket \implies is_snd_snd(\#\#M, x, y) \longleftrightarrow y = snd(snd(x))$
<proof>

lemma *name2_abs*:

$\llbracket x \in M; y \in M \rrbracket \implies is_name2(\#\#M, x, y) \longleftrightarrow y = name2(x)$
<proof>

lemma *cond_of_abs*:

$\llbracket x \in M; y \in M \rrbracket \implies is_cond_of(\#\#M, x, y) \longleftrightarrow y = cond_of(x)$
<proof>

lemma *tuple_abs*:

$\llbracket z \in M; t1 \in M; t2 \in M; p \in M; t \in M \rrbracket \implies$
 $is_tuple(\#\#M, z, t1, t2, p, t) \longleftrightarrow t = \langle z, t1, t2, p \rangle$
<proof>

lemmas *components_abs = ftype_abs name1_abs name2_abs cond_of_abs tuple_abs*

lemma *comp_in_M*:

$p \preceq q \implies p \in M$
 $p \preceq q \implies q \in M$
<proof>

lemma *eq_case_abs [simp]*:

assumes $t1 \in M$ $t2 \in M$ $p \in M$ $f \in M$
shows $is_eq_case(\#\#M, t1, t2, p, \mathbb{P}, leq, f) \longleftrightarrow eq_case(t1, t2, p, \mathbb{P}, leq, f)$
<proof>

lemma *mem_case_abs [simp]*:

assumes $t1 \in M$ $t2 \in M$ $p \in M$ $f \in M$
shows $is_mem_case(\#\#M, t1, t2, p, \mathbb{P}, leq, f) \longleftrightarrow mem_case(t1, t2, p, \mathbb{P}, leq, f)$
<proof>

lemma *Hfrc_abs*:

$\llbracket fnnc \in M; f \in M \rrbracket \implies$
 $is_Hfrc(\#\#M, \mathbb{P}, leq, fnnc, f) \longleftrightarrow Hfrc(\mathbb{P}, leq, fnnc, f)$
<proof>

lemma *Hfrc_at_abs*:

$\llbracket fnnc \in M; f \in M ; z \in M \rrbracket \implies$

$is_Hfrc_at(\#\#M, \mathbb{P}, leq, fnnc, f, z) \longleftrightarrow z = bool_of_o(Hfrc(\mathbb{P}, leq, fnnc, f))$
 ⟨proof⟩

lemma *components_closed* :

$x \in M \implies (\#\#M)(ftype(x))$
 $x \in M \implies (\#\#M)(name1(x))$
 $x \in M \implies (\#\#M)(name2(x))$
 $x \in M \implies (\#\#M)(cond_of(x))$
 ⟨proof⟩

lemma *ecloseN_closed*:

$(\#\#M)(A) \implies (\#\#M)(ecloseN(A))$
 $(\#\#M)(A) \implies (\#\#M)(eclose_n(name1, A))$
 $(\#\#M)(A) \implies (\#\#M)(eclose_n(name2, A))$
 ⟨proof⟩

lemma *eclose_n_abs* :

assumes $x \in M \ ec \in M$
shows $is_eclose_n(\#\#M, is_name1, ec, x) \longleftrightarrow ec = eclose_n(name1, x)$
 $is_eclose_n(\#\#M, is_name2, ec, x) \longleftrightarrow ec = eclose_n(name2, x)$
 ⟨proof⟩

lemma *ecloseN_abs* :

$\llbracket x \in M; ec \in M \rrbracket \implies is_ecloseN(\#\#M, x, ec) \longleftrightarrow ec = ecloseN(x)$
 ⟨proof⟩

lemma *frecR_abs* :

$x \in M \implies y \in M \implies frecR(x, y) \longleftrightarrow is_frecR(\#\#M, x, y)$
 ⟨proof⟩

lemma *frecrelP_abs* :

$z \in M \implies frecrelP(\#\#M, z) \longleftrightarrow (\exists x y. z = \langle x, y \rangle \wedge frecR(x, y))$
 ⟨proof⟩

lemma *frecrel_abs*:

assumes $A \in M \ r \in M$
shows $is_frecrel(\#\#M, A, r) \longleftrightarrow r = frecrel(A)$
 ⟨proof⟩

lemma *frecrel_closed*:

assumes $x \in M$
shows $frecrel(x) \in M$
 ⟨proof⟩

lemma *field_frecrel* : $field(frecrel(names_below(\mathbb{P}, x))) \subseteq names_below(\mathbb{P}, x)$

⟨proof⟩

lemma *forcerelD* : $uv \in forcerel(\mathbb{P}, x) \implies uv \in names_below(\mathbb{P}, x) \times names_below(\mathbb{P}, x)$

<proof>

lemma *wf_forcerel* :

wf(*forcerel*(\mathbb{P}, x))

<proof>

lemma *restrict_trancl_forcerel*:

assumes *frecR*(w, y)

shows $\text{restrict}(f, \text{frecrel}(\text{names_below}(\mathbb{P}, x)) - \{\!-\{y\}\}) \text{'}w$
 $= \text{restrict}(f, \text{forcerel}(\mathbb{P}, x) - \{\!-\{y\}\}) \text{'}w$

<proof>

lemma *names_belowI* :

assumes *frecR*($\langle ft, n1, n2, p \rangle, \langle a, b, c, d \rangle$) $p \in \mathbb{P}$

shows $\langle ft, n1, n2, p \rangle \in \text{names_below}(\mathbb{P}, \langle a, b, c, d \rangle)$ (**is** $?x \in \text{names_below}(_, ?y)$)

<proof>

lemma *names_below_tr* :

assumes $x \in \text{names_below}(\mathbb{P}, y)$ $y \in \text{names_below}(\mathbb{P}, z)$

shows $x \in \text{names_below}(\mathbb{P}, z)$

<proof>

lemma *arg_into_names_below2* :

assumes $\langle x, y \rangle \in \text{frecrel}(\text{names_below}(\mathbb{P}, z))$

shows $x \in \text{names_below}(\mathbb{P}, y)$

<proof>

lemma *arg_into_names_below* :

assumes $\langle x, y \rangle \in \text{frecrel}(\text{names_below}(\mathbb{P}, z))$

shows $x \in \text{names_below}(\mathbb{P}, x)$

<proof>

lemma *forcerel_arg_into_names_below* :

assumes $\langle x, y \rangle \in \text{forcerel}(\mathbb{P}, z)$

shows $x \in \text{names_below}(\mathbb{P}, x)$

<proof>

lemma *names_below_mono* :

assumes $\langle x, y \rangle \in \text{frecrel}(\text{names_below}(\mathbb{P}, z))$

shows $\text{names_below}(\mathbb{P}, x) \subseteq \text{names_below}(\mathbb{P}, y)$

<proof>

lemma *frecrel_mono* :

assumes $\langle x, y \rangle \in \text{frecrel}(\text{names_below}(\mathbb{P}, z))$

shows $\text{frecrel}(\text{names_below}(\mathbb{P}, x)) \subseteq \text{frecrel}(\text{names_below}(\mathbb{P}, y))$

<proof>

lemma *forcerel_mono2* :

assumes $\langle x, y \rangle \in \text{frecrel}(\text{names_below}(\mathbb{P}, z))$

shows $\text{forcere}(\mathbb{P},x) \subseteq \text{forcere}(\mathbb{P},y)$
 ⟨proof⟩

lemma forcere_mono_aux :
assumes $\langle x,y \rangle \in \text{frecr}(\text{names_below}(\mathbb{P}, w)) \hat{+}$
shows $\text{forcere}(\mathbb{P},x) \subseteq \text{forcere}(\mathbb{P},y)$
 ⟨proof⟩

lemma forcere_mono :
assumes $\langle x,y \rangle \in \text{forcere}(\mathbb{P},z)$
shows $\text{forcere}(\mathbb{P},x) \subseteq \text{forcere}(\mathbb{P},y)$
 ⟨proof⟩

lemma forcere_eq_aux: $x \in \text{names_below}(\mathbb{P}, w) \implies \langle x,y \rangle \in \text{forcere}(\mathbb{P},z) \implies$
 $(y \in \text{names_below}(\mathbb{P}, w) \longrightarrow \langle x,y \rangle \in \text{forcere}(\mathbb{P},w))$
 ⟨proof⟩

lemma forcere_eq :
assumes $\langle z,x \rangle \in \text{forcere}(\mathbb{P},x)$
shows $\text{forcere}(\mathbb{P},z) = \text{forcere}(\mathbb{P},x) \cap \text{names_below}(\mathbb{P},z) \times \text{names_below}(\mathbb{P},z)$
 ⟨proof⟩

lemma forcere_below_aux :
assumes $\langle z,x \rangle \in \text{forcere}(\mathbb{P},x)$ $\langle u,z \rangle \in \text{forcere}(\mathbb{P},x)$
shows $u \in \text{names_below}(\mathbb{P},z)$
 ⟨proof⟩

lemma forcere_below :
assumes $\langle z,x \rangle \in \text{forcere}(\mathbb{P},x)$
shows $\text{forcere}(\mathbb{P},x) - \{z\} \subseteq \text{names_below}(\mathbb{P},z)$
 ⟨proof⟩

lemma relation_forcere :
shows $\text{relation}(\text{forcere}(\mathbb{P},z)) \text{ trans}(\text{forcere}(\mathbb{P},z))$
 ⟨proof⟩

lemma Hfrc_restrict_trancl: $\text{bool_of_o}(Hfrc(\mathbb{P}, \text{leq}, y, \text{restrict}(f, \text{frecr}(\text{names_below}(\mathbb{P},x) - \{y\})))$
 $= \text{bool_of_o}(Hfrc(\mathbb{P}, \text{leq}, y, \text{restrict}(f, (\text{frecr}(\text{names_below}(\mathbb{P},x)) \hat{+} - \{y\})))$
 ⟨proof⟩

lemma frc_at_trancl: $\text{frc_at}(\mathbb{P}, \text{leq}, z) = \text{wfrec}(\text{forcere}(\mathbb{P},z), z, \lambda x f. \text{bool_of_o}(Hfrc(\mathbb{P}, \text{leq}, x, f)))$
 ⟨proof⟩

lemma forcereI1 :
assumes $n1 \in \text{domain}(b) \vee n1 \in \text{domain}(c)$ $p \in \mathbb{P}$ $d \in \mathbb{P}$
shows $\langle \langle 1, n1, b, p \rangle, \langle 0, b, c, d \rangle \rangle \in \text{forcere}(\mathbb{P}, \langle 0, b, c, d \rangle)$
 ⟨proof⟩

lemma *forcerelI2* :

assumes $n1 \in \text{domain}(b) \vee n1 \in \text{domain}(c) \ p \in \mathbb{P} \ d \in \mathbb{P}$

shows $\langle \langle 1, n1, c, p \rangle, \langle 0, b, c, d \rangle \rangle \in \text{forcerel}(\mathbb{P}, \langle 0, b, c, d \rangle)$

<proof>

lemma *forcerelI3* :

assumes $\langle n2, r \rangle \in c \ p \in \mathbb{P} \ d \in \mathbb{P} \ r \in \mathbb{P}$

shows $\langle \langle 0, b, n2, p \rangle, \langle 1, b, c, d \rangle \rangle \in \text{forcerel}(\mathbb{P}, \langle 1, b, c, d \rangle)$

<proof>

lemmas *forcerelI* = *forcerelI1*[*THEN vimage_singleton_iff*[*THEN iffD2*]]

forcerelI2[*THEN vimage_singleton_iff*[*THEN iffD2*]]

forcerelI3[*THEN vimage_singleton_iff*[*THEN iffD2*]]

lemma *aux_def_frc_at*:

assumes $z \in \text{forcerel}(\mathbb{P}, x) \ -'' \ \{x\}$

shows $\text{wfrec}(\text{forcerel}(\mathbb{P}, x), z, H) = \text{wfrec}(\text{forcerel}(\mathbb{P}, z), z, H)$

<proof>

11.2 Recursive expression of *frc_at*

lemma *def_frc_at* :

assumes $p \in \mathbb{P}$

shows

$\text{frc_at}(\mathbb{P}, \text{leq}, \langle \text{ft}, n1, n2, p \rangle) =$

$\text{bool_of_o}(p \in \mathbb{P} \wedge$

$(\text{ft} = 0 \wedge (\forall s. s \in \text{domain}(n1) \cup \text{domain}(n2) \longrightarrow$

$(\forall q. q \in \mathbb{P} \wedge q \preceq p \longrightarrow (\text{frc_at}(\mathbb{P}, \text{leq}, \langle 1, s, n1, q \rangle) = 1 \longleftrightarrow \text{frc_at}(\mathbb{P}, \text{leq}, \langle 1, s, n2, q \rangle)$

$= 1)))$

$\vee \text{ft} = 1 \wedge (\forall v \in \mathbb{P}. v \preceq p \longrightarrow$

$(\exists q. \exists s. \exists r. r \in \mathbb{P} \wedge q \in \mathbb{P} \wedge q \preceq v \wedge \langle s, r \rangle \in n2 \wedge q \preceq r \wedge \text{frc_at}(\mathbb{P}, \text{leq}, \langle 0, n1, s, q \rangle)$

$= 1))))$

<proof>

11.3 Absoluteness of *frc_at*

lemma *forcerel_in_M* :

assumes $x \in M$

shows $\text{forcerel}(\mathbb{P}, x) \in M$

<proof>

lemma *relation2_Hfrc_at_abs*:

$\text{relation2}(\#\#M, \text{is_Hfrc_at}(\#\#M, \mathbb{P}, \text{leq}), \lambda x f. \text{bool_of_o}(\text{Hfrc}(\mathbb{P}, \text{leq}, x, f)))$

<proof>

lemma *Hfrc_at_closed* :

$\forall x \in M. \forall g \in M. \text{function}(g) \longrightarrow \text{bool_of_o}(\text{Hfrc}(\mathbb{P}, \text{leq}, x, g)) \in M$

<proof>

lemma *wfrec_Hfrc_at* :

assumes $X \in M$
shows $wfrec_replacement(\#\#M, is_Hfrc_at(\#\#M, \mathbb{P}, leq), forcereI(\mathbb{P}, X))$
 $\langle proof \rangle$

lemma $names_below_abs :$
 $\llbracket Q \in M ; x \in M ; nb \in M \rrbracket \implies is_names_below(\#\#M, Q, x, nb) \longleftrightarrow nb = names_below(Q, x)$
 $\langle proof \rangle$

lemma $names_below_closed :$
 $\llbracket Q \in M ; x \in M \rrbracket \implies names_below(Q, x) \in M$
 $\langle proof \rangle$

lemma $names_below_productE :$
assumes $Q \in M \ x \in M$
 $\bigwedge A1 \ A2 \ A3 \ A4. A1 \in M \implies A2 \in M \implies A3 \in M \implies A4 \in M \implies R(A1$
 $\times A2 \times A3 \times A4)$
shows $R(names_below(Q, x))$
 $\langle proof \rangle$

lemma $forcereI_abs :$
 $\llbracket x \in M ; z \in M \rrbracket \implies is_forcereI(\#\#M, \mathbb{P}, x, z) \longleftrightarrow z = forcereI(\mathbb{P}, x)$
 $\langle proof \rangle$

lemma $frc_at_abs :$
assumes $fnc \in M \ z \in M$
shows $is_frc_at(\#\#M, \mathbb{P}, leq, fnc, z) \longleftrightarrow z = frc_at(\mathbb{P}, leq, fnc)$
 $\langle proof \rangle$

lemma $forces_eq'_abs :$
 $\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \implies is_forces_eq'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces_eq'(\mathbb{P}, leq, p, t1, t2)$
 $\langle proof \rangle$

lemma $forces_mem'_abs :$
 $\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \implies is_forces_mem'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces_mem'(\mathbb{P}, leq, p, t1, t2)$
 $\langle proof \rangle$

lemma $forces_neq'_abs :$
assumes $p \in M \ t1 \in M \ t2 \in M$
shows $is_forces_neq'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces_neq'(\mathbb{P}, leq, p, t1, t2)$
 $\langle proof \rangle$

lemma $forces_nmem'_abs :$
assumes $p \in M \ t1 \in M \ t2 \in M$
shows $is_forces_nmem'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces_nmem'(\mathbb{P}, leq, p, t1, t2)$
 $\langle proof \rangle$

lemma $leq_abs :$
 $\llbracket l \in M ; q \in M ; p \in M \rrbracket \implies is_leq(\#\#M, l, q, p) \longleftrightarrow \langle q, p \rangle \in l$
 $\langle proof \rangle$

11.4 Forcing for atomic formulas in context

definition

$forces_eq :: [i,i,i] \Rightarrow o (\langle _ forces_a '(_ = _)' \rangle [36,1,1] 60)$ **where**
 $forces_eq \equiv forces_eq'(\mathbb{P},leq)$

definition

$forces_mem :: [i,i,i] \Rightarrow o (\langle _ forces_a '(_ \in _)' \rangle [36,1,1] 60)$ **where**
 $forces_mem \equiv forces_mem'(\mathbb{P},leq)$

abbreviation is_forces_eq

where $is_forces_eq \equiv is_forces_eq'(\#\#M,\mathbb{P},leq)$

abbreviation

$is_forces_mem :: [i,i,i] \Rightarrow o$ **where**
 $is_forces_mem \equiv is_forces_mem'(\#\#M,\mathbb{P},leq)$

lemma $def_forces_eq: p \in \mathbb{P} \Longrightarrow p forces_a (t1 = t2) \longleftrightarrow$
 $(\forall s \in domain(t1) \cup domain(t2). \forall q. q \in \mathbb{P} \wedge q \preceq p \longrightarrow$
 $(q forces_a (s \in t1) \longleftrightarrow q forces_a (s \in t2)))$
 $\langle proof \rangle$

lemma $def_forces_mem: p \in \mathbb{P} \Longrightarrow p forces_a (t1 \in t2) \longleftrightarrow$
 $(\forall v \in \mathbb{P}. v \preceq p \longrightarrow$
 $(\exists q. \exists s. \exists r. r \in \mathbb{P} \wedge q \in \mathbb{P} \wedge q \preceq v \wedge \langle s,r \rangle \in t2 \wedge q \preceq r \wedge q forces_a (t1 = s)))$
 $\langle proof \rangle$

lemma $forces_eq_abs :$

$\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \Longrightarrow is_forces_eq(p,t1,t2) \longleftrightarrow p forces_a (t1 = t2)$
 $\langle proof \rangle$

lemma $forces_mem_abs :$

$\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \Longrightarrow is_forces_mem(p,t1,t2) \longleftrightarrow p forces_a (t1 \in t2)$
 $\langle proof \rangle$

definition

$forces_neq :: [i,i,i] \Rightarrow o (\langle _ forces_a '(_ \neq _)' \rangle [36,1,1] 60)$ **where**
 $p forces_a (t1 \neq t2) \equiv \neg (\exists q \in \mathbb{P}. q \preceq p \wedge q forces_a (t1 = t2))$

definition

$forces_nmem :: [i,i,i] \Rightarrow o (\langle _ forces_a '(_ \notin _)' \rangle [36,1,1] 60)$ **where**
 $p forces_a (t1 \notin t2) \equiv \neg (\exists q \in \mathbb{P}. q \preceq p \wedge q forces_a (t1 \in t2))$

lemma $forces_neq :$

$p forces_a (t1 \neq t2) \longleftrightarrow forces_neq'(\mathbb{P},leq,p,t1,t2)$
 $\langle proof \rangle$

lemma $forces_nmem :$

$p \text{ forces}_a (t1 \notin t2) \longleftrightarrow \text{forces_nmem}'(\mathbb{P}, \text{leq}, p, t1, t2)$
 $\langle \text{proof} \rangle$

abbreviation $\text{Forces} :: [i, i, i] \Rightarrow o \ (\langle _ \Vdash _ \rangle \text{ [36,36,36] } 60)$ **where**
 $p \Vdash \varphi \text{ env} \equiv M, ([p, \mathbb{P}, \text{leq}, \mathbf{1}] @ \text{env}) \models \text{forces}(\varphi)$

lemma $\text{sats_forces_Member} :$
assumes $x \in \text{nat } y \in \text{nat } \text{env} \in \text{list}(M)$
 $\text{nth}(x, \text{env}) = xx \ \text{nth}(y, \text{env}) = yy \ q \in M$
shows $q \Vdash \cdot x \in y \cdot \text{env} \longleftrightarrow q \in \mathbb{P} \wedge \text{is_forces_mem}(q, xx, yy)$
 $\langle \text{proof} \rangle$

lemma $\text{sats_forces_Equal} :$
assumes $a \in \text{nat } b \in \text{nat } \text{env} \in \text{list}(M) \ \text{nth}(a, \text{env}) = x \ \text{nth}(b, \text{env}) = y \ q \in M$
shows $q \Vdash \cdot a = b \cdot \text{env} \longleftrightarrow q \in \mathbb{P} \wedge \text{is_forces_eq}(q, x, y)$
 $\langle \text{proof} \rangle$

lemma $\text{sats_forces_Nand} :$
assumes $\varphi \in \text{formula } \psi \in \text{formula } \text{env} \in \text{list}(M) \ p \in M$
shows $p \Vdash \cdot \neg(\varphi \wedge \psi) \cdot \text{env} \longleftrightarrow$
 $p \in \mathbb{P} \wedge \neg(\exists q \in M. q \in \mathbb{P} \wedge \text{is_leq}(\#\#M, \text{leq}, q, p) \wedge (q \Vdash \varphi \text{ env}) \wedge (q \Vdash \psi \text{ env}))$
 $\langle \text{proof} \rangle$

lemma $\text{sats_forces_Neg} :$
assumes $\varphi \in \text{formula } \text{env} \in \text{list}(M) \ p \in M$
shows $p \Vdash \cdot \neg \varphi \cdot \text{env} \longleftrightarrow$
 $(p \in \mathbb{P} \wedge \neg(\exists q \in M. q \in \mathbb{P} \wedge \text{is_leq}(\#\#M, \text{leq}, q, p) \wedge (q \Vdash \varphi \text{ env})))$
 $\langle \text{proof} \rangle$

lemma $\text{sats_forces_Forall} :$
assumes $\varphi \in \text{formula } \text{env} \in \text{list}(M) \ p \in M$
shows $p \Vdash \cdot (\forall \varphi) \cdot \text{env} \longleftrightarrow p \in \mathbb{P} \wedge (\forall x \in M. p \Vdash \varphi ([x] @ \text{env}))$
 $\langle \text{proof} \rangle$

end — *forcing_data1*

end

12 Names and generic extensions

theory *Names*

imports

Forcing_Data

FrecR_Arities

ZF_Trans_Interpretations

begin

definition

$Hv :: [i, i, i] \Rightarrow i$ **where**

$$Hv(G,x,f) \equiv \{ z . y \in \text{domain}(x), (\exists p \in G. \langle y,p \rangle \in x) \wedge z=f'y \}$$

The function *val* interprets a name in *M* according to a (generic) filter *G*. Note the definition in terms of the well-founded recursor.

definition

$$\begin{aligned} \text{val} &:: [i,i] \Rightarrow i \text{ where} \\ \text{val}(G,\tau) &\equiv \text{wfrec}(\text{edrel}(\text{eclose}(\{\tau\})), \tau, Hv(G)) \end{aligned}$$

definition

$$\begin{aligned} \text{GenExt} &:: [i,i] \Rightarrow i \quad (\langle _ \rangle [71,1]) \\ \text{where } M[G] &\equiv \{ \text{val}(G,\tau) . \tau \in M \} \end{aligned}$$

lemma *map_val_in_MG*:

$$\begin{aligned} \text{assumes} & \\ \text{env} &\in \text{list}(M) \\ \text{shows} & \\ \text{map}(\text{val}(G), \text{env}) &\in \text{list}(M[G]) \\ \langle \text{proof} \rangle & \end{aligned}$$

12.1 Values and check-names

context *forcing_data1*

begin

lemma *name_components_in_M*:

$$\begin{aligned} \text{assumes} & \langle \sigma,p \rangle \in \vartheta \quad \vartheta \in M \\ \text{shows} & \sigma \in M \quad p \in M \\ \langle \text{proof} \rangle & \end{aligned}$$

definition

$$\begin{aligned} \text{Hcheck} &:: [i,i] \Rightarrow i \text{ where} \\ \text{Hcheck}(z,f) &\equiv \{ \langle f'y,1 \rangle . y \in z \} \end{aligned}$$

definition

$$\begin{aligned} \text{check} &:: i \Rightarrow i \text{ where} \\ \text{check}(x) &\equiv \text{transrec}(x, \text{Hcheck}) \end{aligned}$$

lemma *checkD*:

$$\begin{aligned} \text{check}(x) &= \text{wfrec}(\text{Memrel}(\text{eclose}(\{x\})), x, \text{Hcheck}) \\ \langle \text{proof} \rangle & \end{aligned}$$

lemma *Hcheck_trancl*: $\text{Hcheck}(y, \text{restrict}(f, \text{Memrel}(\text{eclose}(\{x\}))-'\{y\}))$

$$= \text{Hcheck}(y, \text{restrict}(f, (\text{Memrel}(\text{eclose}(\{x\}))^{\wedge+})-'\{y\}))$$

$\langle \text{proof} \rangle$

lemma *check_trancl*: $\text{check}(x) = \text{wfrec}(\text{rcheck}(x), x, \text{Hcheck})$

$\langle \text{proof} \rangle$

lemma *rcheck_in_M* : $x \in M \implies \text{rcheck}(x) \in M$

<proof>

lemma *rcheck_subset_M* : $x \in M \implies \text{field}(\text{rcheck}(x)) \subseteq \text{eclose}(\{x\})$
<proof>

lemma *aux_def_check*: $x \in y \implies$
 $\text{wfrec}(\text{Memrel}(\text{eclose}(\{y\})), x, H\text{check}) =$
 $\text{wfrec}(\text{Memrel}(\text{eclose}(\{x\})), x, H\text{check})$
<proof>

lemma *def_check* : $\text{check}(y) = \{ \langle \text{check}(w), \mathbf{1} \rangle . w \in y \}$
<proof>

lemma *def_checkS* :
 fixes n
 assumes $n \in \text{nat}$
 shows $\text{check}(\text{succ}(n)) = \text{check}(n) \cup \{ \langle \text{check}(n), \mathbf{1} \rangle \}$
<proof>

lemma *field_Memrel2* :
 assumes $x \in M$
 shows $\text{field}(\text{Memrel}(\text{eclose}(\{x\}))) \subseteq M$
<proof>

lemma *aux_def_val*:
 assumes $z \in \text{domain}(x)$
 shows $\text{wfrec}(\text{edrel}(\text{eclose}(\{x\})), z, Hv(G)) = \text{wfrec}(\text{edrel}(\text{eclose}(\{z\})), z, Hv(G))$
<proof>

The next lemma provides the usual recursive expression for the definition of *val*.

lemma *def_val*: $\text{val}(G, x) = \{ z . t \in \text{domain}(x) , (\exists p \in G . \langle t, p \rangle \in x) \wedge z = \text{val}(G, t) \}$
<proof>

lemma *val_mono* : $x \subseteq y \implies \text{val}(G, x) \subseteq \text{val}(G, y)$
<proof>

Check-names are the canonical names for elements of the ground model. Here we show that this is the case.

lemma *val_check* : $\mathbf{1} \in G \implies \mathbf{1} \in \mathbb{P} \implies \text{val}(G, \text{check}(y)) = y$
<proof>

lemma *val_of_name* :
 $\text{val}(G, \{ x \in A \times \mathbb{P} . Q(x) \}) = \{ z . t \in A , (\exists p \in \mathbb{P} . Q(\langle t, p \rangle)) \wedge p \in G \wedge z = \text{val}(G, t) \}$
<proof>

lemma *val_of_name_alt* :
 $\text{val}(G, \{ x \in A \times \mathbb{P} . Q(x) \}) = \{ z . t \in A , (\exists p \in \mathbb{P} \cap G . Q(\langle t, p \rangle)) \wedge z = \text{val}(G, t) \}$
<proof>

lemma *val_only_names*: $val(F,\tau) = val(F,\{x \in \tau. \exists t \in domain(\tau). \exists p \in F. x = \langle t,p \rangle\})$
 (is $_ = val(F,?name)$)
 $\langle proof \rangle$

lemma *val_only_pairs*: $val(F,\tau) = val(F,\{x \in \tau. \exists t p. x = \langle t,p \rangle\})$
 $\langle proof \rangle$

lemma *val_subset_domain_times_range*: $val(F,\tau) \subseteq val(F, domain(\tau) \times range(\tau))$
 $\langle proof \rangle$

lemma *val_of_elem*: $\langle \vartheta,p \rangle \in \pi \implies p \in G \implies val(G,\vartheta) \in val(G,\pi)$
 $\langle proof \rangle$

lemma *elem_of_val*: $x \in val(G,\pi) \implies \exists \vartheta \in domain(\pi). val(G,\vartheta) = x$
 $\langle proof \rangle$

lemma *elem_of_val_pair*: $x \in val(G,\pi) \implies \exists \vartheta. \exists p \in G. \langle \vartheta,p \rangle \in \pi \wedge val(G,\vartheta) = x$
 $\langle proof \rangle$

lemma *elem_of_val_pair'*:
assumes $\pi \in M \ x \in val(G,\pi)$
shows $\exists \vartheta \in M. \exists p \in G. \langle \vartheta,p \rangle \in \pi \wedge val(G,\vartheta) = x$
 $\langle proof \rangle$

lemma *GenExtD*: $x \in M[G] \implies \exists \tau \in M. x = val(G,\tau)$
 $\langle proof \rangle$

lemma *GenExtI*: $x \in M \implies val(G,x) \in M[G]$
 $\langle proof \rangle$

lemma *Transset_MG* : $Transset(M[G])$
 $\langle proof \rangle$

lemmas *transitivity_MG* = $Transset_intf[OF\ Transset_MG]$

This lemma can be proved before having *check_in_M*. At some point Miguel naïvely thought that the *check_in_M* could be proved using this argument.

lemma *check_nat_M* :
assumes $n \in nat$
shows $check(n) \in M$
 $\langle proof \rangle$

lemma *def_PHcheck*:
assumes
 $z \in M \ f \in M$
shows
 $Hcheck(z,f) = Replace(z, PHcheck(\#\#M, \mathbf{1}, f))$
 $\langle proof \rangle$

lemma *wfrec_Hcheck* :
assumes $X \in M$
shows $wfrec_replacement(\#\#M, is_Hcheck(\#\#M, \mathbf{1}), rcheck(X))$
<proof>

lemma *Hcheck_closed'* : $f \in M \implies z \in M \implies \{f \cdot x \mid x \in z\} \in M$
<proof>

lemma *repl_PHcheck* :
assumes $f \in M$
shows $lam_replacement(\#\#M, \lambda x. Hcheck(x, f))$
<proof>

lemma *univ_PHcheck* : $\llbracket z \in M ; f \in M \rrbracket \implies univalent(\#\#M, z, PHcheck(\#\#M, \mathbf{1}, f))$
<proof>

lemma *PHcheck_closed* : $\llbracket z \in M ; f \in M ; x \in z ; PHcheck(\#\#M, \mathbf{1}, f, x, y) \rrbracket \implies (\#\#M)(y)$
<proof>

lemma *relation2_Hcheck* : $relation2(\#\#M, is_Hcheck(\#\#M, \mathbf{1}), Hcheck)$
<proof>

lemma *Hcheck_closed* : $\forall y \in M. \forall g \in M. Hcheck(y, g) \in M$
<proof>

lemma *wf_rcheck* : $x \in M \implies wf(rcheck(x))$
<proof>

lemma *trans_rcheck* : $x \in M \implies trans(rcheck(x))$
<proof>

lemma *relation_rcheck* : $x \in M \implies relation(rcheck(x))$
<proof>

lemma *check_in_M* : $x \in M \implies check(x) \in M$
<proof>

lemma *rcheck_abs[Rel]* : $\llbracket x \in M ; r \in M \rrbracket \implies is_rcheck(\#\#M, x, r) \longleftrightarrow r = rcheck(x)$
<proof>

lemma *check_abs[Rel]* :
assumes $x \in M \ z \in M$
shows $is_check(\#\#M, \mathbf{1}, x, z) \longleftrightarrow z = check(x)$
<proof>

lemma *check_lam_replacement*: *lam_replacement*(##*M*,*check*)
⟨*proof*⟩

lemma *check_replacement*: {*check*(*x*). *x* ∈ \mathbb{P} } ∈ *M*
⟨*proof*⟩

lemma *M_subset_MG* : $\mathbf{1} \in G \implies M \subseteq M[G]$
⟨*proof*⟩

The name for the generic filter

definition

G_dot :: *i* **where**
G_dot ≡ {⟨*check*(*p*),*p*⟩ . *p* ∈ \mathbb{P} }

lemma *G_dot_in_M* : *G_dot* ∈ *M*
⟨*proof*⟩

lemma *zero_in_MG* : $0 \in M[G]$
⟨*proof*⟩

declare *check_in_M* [*simp*,*intro*]

end — *forcing_data1*

context *G_generic1*
begin

lemma *val_G_dot* : *val*(*G*,*G_dot*) = *G*
⟨*proof*⟩

lemma *G_in_Gen_Ext* : *G* ∈ *M*[*G*]
⟨*proof*⟩

lemmas *generic_simps* = *val_check*[*OF one_in_G one_in_P*]
M_subset_MG[*OF one_in_G, THEN subsetD*]
GenExtI P_in_M

lemmas *generic_dests* = *M_genericD M_generic_compatD*

bundle *G_generic1_lemmas* = *generic_simps*[*simp*] *generic_dests*[*dest*]

end — *G_generic1*

sublocale *G_generic1* ⊆ *ext*: *M_trans* ##*M*[*G*]
⟨*proof*⟩

end

13 The Forcing Theorems

```
theory Forcing_Theorems
  imports
    Cohen_Posets_Relative
    Forces_Definition
    Names
```

```
begin
```

```
context forcing_data1
```

```
begin
```

13.1 The forcing relation in context

```
lemma separation_forces :
  assumes
    fty:  $\varphi \in \text{formula}$  and
    far:  $\text{arity}(\varphi) \leq \text{length}(\text{env})$  and
    envty:  $\text{env} \in \text{list}(M)$ 
  shows
    separation( $\#\#M, \lambda p. (p \Vdash \varphi \text{ env})$ )
  <proof>
```

```
lemma Collect_forces :
  assumes
     $\varphi \in \text{formula}$  and
     $\text{arity}(\varphi) \leq \text{length}(\text{env})$  and
     $\text{env} \in \text{list}(M)$ 
  shows
     $\{p \in \mathbb{P} . p \Vdash \varphi \text{ env}\} \in M$ 
  <proof>
```

```
lemma forces_mem_iff_dense_below:  $p \in \mathbb{P} \implies p \text{ forces}_a (t1 \in t2) \iff \text{dense\_below}(\{q \in \mathbb{P} . \exists s. \exists r. r \in \mathbb{P} \wedge \langle s, r \rangle \in t2 \wedge q \preceq r \wedge q \text{ forces}_a (t1 = s)\}, p)$ 
  <proof>
```

13.2 Kunen 2013, Lemma IV.2.37(a)

```
lemma strengthening_eq:
  assumes  $p \in \mathbb{P} \ r \in \mathbb{P} \ r \preceq p \ p \text{ forces}_a (t1 = t2)$ 
  shows  $r \text{ forces}_a (t1 = t2)$ 
  <proof>
```

13.3 Kunen 2013, Lemma IV.2.37(a)

```
lemma strengthening_mem:
  assumes  $p \in \mathbb{P} \ r \in \mathbb{P} \ r \preceq p \ p \text{ forces}_a (t1 \in t2)$ 
  shows  $r \text{ forces}_a (t1 \in t2)$ 
```

<proof>

13.4 Kunen 2013, Lemma IV.2.37(b)

lemma *density_mem*:

assumes $p \in \mathbb{P}$

shows $p \text{ forces}_a (t1 \in t2) \longleftrightarrow \text{dense_below}(\{q \in \mathbb{P}. q \text{ forces}_a (t1 \in t2)\}, p)$

<proof>

lemma *aux_density_eq*:

assumes

$\text{dense_below}(\{q' \in \mathbb{P}. \forall q. q \in \mathbb{P} \wedge q \preceq q' \longrightarrow q \text{ forces}_a (s \in t1) \longleftrightarrow q \text{ forces}_a (s \in t2)\}, p)$

$q \text{ forces}_a (s \in t1) \quad q \in \mathbb{P} \quad p \in \mathbb{P} \quad q \preceq p$

shows

$\text{dense_below}(\{r \in \mathbb{P}. r \text{ forces}_a (s \in t2)\}, q)$

<proof>

lemma *density_eq*:

assumes $p \in \mathbb{P}$

shows $p \text{ forces}_a (t1 = t2) \longleftrightarrow \text{dense_below}(\{q \in \mathbb{P}. q \text{ forces}_a (t1 = t2)\}, p)$

<proof>

13.5 Kunen 2013, Lemma IV.2.38

lemma *not_forces_neq*:

assumes $p \in \mathbb{P}$

shows $p \text{ forces}_a (t1 = t2) \longleftrightarrow \neg (\exists q \in \mathbb{P}. q \preceq p \wedge q \text{ forces}_a (t1 \neq t2))$

<proof>

lemma *not_forces_nmem*:

assumes $p \in \mathbb{P}$

shows $p \text{ forces}_a (t1 \in t2) \longleftrightarrow \neg (\exists q \in \mathbb{P}. q \preceq p \wedge q \text{ forces}_a (t1 \notin t2))$

<proof>

13.6 The relation of forcing and atomic formulas

lemma *Forces_Equal*:

assumes

$p \in \mathbb{P} \quad t1 \in M \quad t2 \in M \quad env \in \text{list}(M) \quad nth(n, env) = t1 \quad nth(m, env) = t2 \quad n \in \text{nat} \quad m \in \text{nat}$

shows

$(p \Vdash \text{Equal}(n, m) \text{ env}) \longleftrightarrow p \text{ forces}_a (t1 = t2)$

<proof>

lemma *Forces_Member*:

assumes

$p \in \mathbb{P} \quad t1 \in M \quad t2 \in M \quad env \in \text{list}(M) \quad nth(n, env) = t1 \quad nth(m, env) = t2 \quad n \in \text{nat} \quad m \in \text{nat}$

shows

$(p \Vdash \text{Member}(n,m) \text{ env}) \longleftrightarrow p \text{ forces}_a (t1 \in t2)$
 $\langle \text{proof} \rangle$

lemma *Forces_Neg*:

assumes

$p \in \mathbb{P} \text{ env} \in \text{list}(M) \varphi \in \text{formula}$

shows

$(p \Vdash \text{Neg}(\varphi) \text{ env}) \longleftrightarrow \neg(\exists q \in M. q \in \mathbb{P} \wedge q \preceq p \wedge (q \Vdash \varphi \text{ env}))$

$\langle \text{proof} \rangle$

13.7 The relation of forcing and connectives

lemma *Forces_Nand*:

assumes

$p \in \mathbb{P} \text{ env} \in \text{list}(M) \varphi \in \text{formula} \psi \in \text{formula}$

shows

$(p \Vdash \text{Nand}(\varphi, \psi) \text{ env}) \longleftrightarrow \neg(\exists q \in M. q \in \mathbb{P} \wedge q \preceq p \wedge (q \Vdash \varphi \text{ env}) \wedge (q \Vdash \psi \text{ env}))$

$\langle \text{proof} \rangle$

lemma *Forces_And_aux*:

assumes

$p \in \mathbb{P} \text{ env} \in \text{list}(M) \varphi \in \text{formula} \psi \in \text{formula}$

shows

$p \Vdash \text{And}(\varphi, \psi) \text{ env} \longleftrightarrow$

$(\forall q \in M. q \in \mathbb{P} \wedge q \preceq p \longrightarrow (\exists r \in M. r \in \mathbb{P} \wedge r \preceq q \wedge (r \Vdash \varphi \text{ env}) \wedge (r \Vdash \psi \text{ env})))$

$\langle \text{proof} \rangle$

lemma *Forces_And_iff_dense_below*:

assumes

$p \in \mathbb{P} \text{ env} \in \text{list}(M) \varphi \in \text{formula} \psi \in \text{formula}$

shows

$(p \Vdash \text{And}(\varphi, \psi) \text{ env}) \longleftrightarrow \text{dense_below}(\{r \in \mathbb{P}. (r \Vdash \varphi \text{ env}) \wedge (r \Vdash \psi \text{ env})\}, p)$

$\langle \text{proof} \rangle$

lemma *Forces_Forall*:

assumes

$p \in \mathbb{P} \text{ env} \in \text{list}(M) \varphi \in \text{formula}$

shows

$(p \Vdash \text{Forall}(\varphi) \text{ env}) \longleftrightarrow (\forall x \in M. (p \Vdash \varphi ([x] @ \text{env})))$

$\langle \text{proof} \rangle$

bundle *some_rules* = *elem_of_val_pair* [*dest*]

context

includes *some_rules*

begin

lemma *elem_of_valI*: $\exists \vartheta. \exists p \in \mathbb{P}. p \in G \wedge \langle \vartheta, p \rangle \in \pi \wedge \text{val}(G, \vartheta) = x \implies x \in \text{val}(G, \pi)$

<proof>

lemma *GenExt_iff*: $x \in M[G] \longleftrightarrow (\exists \tau \in M. x = \text{val}(G, \tau))$
<proof>
end

end

context *G_generic1*

begin

13.8 Kunen 2013, Lemma IV.2.29

lemma *generic_inter_dense_below*:
assumes $D \in M$ *dense_below*(D, p) $p \in G$
shows $D \cap G \neq \emptyset$
<proof>

13.9 Auxiliary results for Lemma IV.2.40(a)

lemma (*in forcing_data1*) *IV240a_mem_Collect*:
assumes
 $\pi \in M$ $\tau \in M$
shows
 $\{q \in \mathbb{P}. \exists \sigma. \exists r. r \in \mathbb{P} \wedge \langle \sigma, r \rangle \in \tau \wedge q \leq r \wedge q \text{ forces}_a (\pi = \sigma)\} \in M$
<proof>

lemma *IV240a_mem*:
assumes
 $p \in G$ $\pi \in M$ $\tau \in M$ $p \text{ forces}_a (\pi \in \tau)$
 $\bigwedge q \sigma. q \in \mathbb{P} \implies q \in G \implies \sigma \in \text{domain}(\tau) \implies q \text{ forces}_a (\pi = \sigma) \implies$
 $\text{val}(G, \pi) = \text{val}(G, \sigma)$
shows
 $\text{val}(G, \pi) \in \text{val}(G, \tau)$
<proof>

lemma *refl_forces_eq*: $p \in \mathbb{P} \implies p \text{ forces}_a (x = x)$
<proof>

lemma *forces_memI*: $\langle \sigma, r \rangle \in \tau \implies p \in \mathbb{P} \implies r \in \mathbb{P} \implies p \leq r \implies p \text{ forces}_a (\sigma \in \tau)$
<proof>

lemma *IV240a_eq_1st_incl*:
includes *some_rules*
assumes
 $p \in G$ $p \text{ forces}_a (\tau = \emptyset)$
and

IH: $\bigwedge q \sigma. q \in \mathbb{P} \implies q \in G \implies \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies$
 $(q \text{ forces}_a (\sigma \in \tau) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \tau)) \wedge$
 $(q \text{ forces}_a (\sigma \in \vartheta) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \vartheta))$

shows

$\text{val}(G, \tau) \subseteq \text{val}(G, \vartheta)$

<proof>

lemma *IV240a_eq_2nd_incl*:

includes *some_rules*

assumes

$p \in G \ p \text{ forces}_a (\tau = \vartheta)$

and

IH: $\bigwedge q \sigma. q \in \mathbb{P} \implies q \in G \implies \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies$
 $(q \text{ forces}_a (\sigma \in \tau) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \tau)) \wedge$
 $(q \text{ forces}_a (\sigma \in \vartheta) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \vartheta))$

shows

$\text{val}(G, \vartheta) \subseteq \text{val}(G, \tau)$

<proof>

lemma *IV240a_eq*:

includes *some_rules*

assumes

$p \in G \ p \text{ forces}_a (\tau = \vartheta)$

and

IH: $\bigwedge q \sigma. q \in \mathbb{P} \implies q \in G \implies \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies$
 $(q \text{ forces}_a (\sigma \in \tau) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \tau)) \wedge$
 $(q \text{ forces}_a (\sigma \in \vartheta) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \vartheta))$

shows

$\text{val}(G, \tau) = \text{val}(G, \vartheta)$

<proof>

13.10 Induction on names

lemma (*in forcing_data1*) *core_induction*:

assumes

$\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in \text{domain}(\vartheta) \rrbracket \implies Q(0, \tau, \sigma, q) \rrbracket \implies Q(1, \tau, \vartheta, p)$
 $\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \rrbracket \implies Q(1, \sigma, \tau, q)$
 $\wedge Q(1, \sigma, \vartheta, q) \rrbracket \implies Q(0, \tau, \vartheta, p)$
 $ft \in 2 \ p \in \mathbb{P}$

shows

$Q(ft, \tau, \vartheta, p)$

<proof>

lemma (*in forcing_data1*) *forces_induction_with_conds*:

assumes

$\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in \text{domain}(\vartheta) \rrbracket \implies Q(q, \tau, \sigma) \rrbracket \implies R(p, \tau, \vartheta)$
 $\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \rrbracket \implies R(q, \sigma, \tau) \wedge R(q, \sigma, \vartheta) \rrbracket \implies Q(p, \tau, \vartheta)$
 $p \in \mathbb{P}$
shows
 $Q(p, \tau, \vartheta) \wedge R(p, \tau, \vartheta)$
 $\langle \text{proof} \rangle$

lemma (in *forcing_data1*) *forces_induction*:

assumes
 $\bigwedge \tau \vartheta. \llbracket \bigwedge \sigma. \sigma \in \text{domain}(\vartheta) \implies Q(\tau, \sigma) \rrbracket \implies R(\tau, \vartheta)$
 $\bigwedge \tau \vartheta. \llbracket \bigwedge \sigma. \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies R(\sigma, \tau) \wedge R(\sigma, \vartheta) \rrbracket \implies Q(\tau, \vartheta)$
shows
 $Q(\tau, \vartheta) \wedge R(\tau, \vartheta)$
 $\langle \text{proof} \rangle$

13.11 Lemma IV.2.40(a), in full

lemma *IV240a*:

shows
 $(\tau \in M \longrightarrow \vartheta \in M \longrightarrow (\forall p \in G. p \text{ forces}_a (\tau = \vartheta) \longrightarrow \text{val}(G, \tau) = \text{val}(G, \vartheta))) \wedge$
 $(\tau \in M \longrightarrow \vartheta \in M \longrightarrow (\forall p \in G. p \text{ forces}_a (\tau \in \vartheta) \longrightarrow \text{val}(G, \tau) \in \text{val}(G, \vartheta)))$
(is $?Q(\tau, \vartheta) \wedge ?R(\tau, \vartheta)$
 $\langle \text{proof} \rangle$

13.12 Lemma IV.2.40(b)

lemma *IV240b_mem*:

includes *some_rules*
assumes
 $\text{val}(G, \pi) \in \text{val}(G, \tau) \quad \pi \in M \quad \tau \in M$
and
 $IH: \bigwedge \sigma. \sigma \in \text{domain}(\tau) \implies \text{val}(G, \pi) = \text{val}(G, \sigma) \implies$
 $\exists p \in G. p \text{ forces}_a (\pi = \sigma)$
shows
 $\exists p \in G. p \text{ forces}_a (\pi \in \tau)$
 $\langle \text{proof} \rangle$

end — *G_generic1*

context *forcing_data1*

begin

lemma *Collect_forces_eq_in_M*:

assumes $\tau \in M \quad \vartheta \in M$
shows $\{p \in \mathbb{P}. p \text{ forces}_a (\tau = \vartheta)\} \in M$
 $\langle \text{proof} \rangle$

lemma *IV240b_eq_Collects*:

assumes $\tau \in M \quad \vartheta \in M$

shows $\{p \in \mathbb{P}. \exists \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta). p \text{ forces}_a (\sigma \in \tau) \wedge p \text{ forces}_a (\sigma \notin \vartheta)\} \in M$ **and**
 $\{p \in \mathbb{P}. \exists \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta). p \text{ forces}_a (\sigma \notin \tau) \wedge p \text{ forces}_a (\sigma \in \vartheta)\} \in M$
 $\langle \text{proof} \rangle$

end — *forcing_data1*

context *G_generic1*
begin

lemma *IV240b_eq*:

includes *some_rules*

assumes

$\text{val}(G, \tau) = \text{val}(G, \vartheta) \quad \tau \in M \quad \vartheta \in M$

and

$IH: \bigwedge \sigma. \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies$

$(\text{val}(G, \sigma) \in \text{val}(G, \tau) \longrightarrow (\exists q \in G. q \text{ forces}_a (\sigma \in \tau))) \wedge$

$(\text{val}(G, \sigma) \in \text{val}(G, \vartheta) \longrightarrow (\exists q \in G. q \text{ forces}_a (\sigma \in \vartheta)))$

shows

$\exists p \in G. p \text{ forces}_a (\tau = \vartheta)$

$\langle \text{proof} \rangle$

lemma *IV240b*:

$(\tau \in M \longrightarrow \vartheta \in M \longrightarrow \text{val}(G, \tau) = \text{val}(G, \vartheta) \longrightarrow (\exists p \in G. p \text{ forces}_a (\tau = \vartheta))) \wedge$

$(\tau \in M \longrightarrow \vartheta \in M \longrightarrow \text{val}(G, \tau) \in \text{val}(G, \vartheta) \longrightarrow (\exists p \in G. p \text{ forces}_a (\tau \in \vartheta)))$

(is $?Q(\tau, \vartheta) \wedge ?R(\tau, \vartheta)$

$\langle \text{proof} \rangle$

lemma *truth_lemma_mem*:

assumes

$\text{env} \in \text{list}(M)$

$n \in \text{nat} \quad m \in \text{nat} \quad n < \text{length}(\text{env}) \quad m < \text{length}(\text{env})$

shows

$(\exists p \in G. p \Vdash \text{Member}(n, m) \text{ env}) \longleftrightarrow M[G], \text{map}(\text{val}(G), \text{env}) \models \text{Member}(n, m)$

$\langle \text{proof} \rangle$

lemma *truth_lemma_eq*:

assumes

$\text{env} \in \text{list}(M)$

$n \in \text{nat} \quad m \in \text{nat} \quad n < \text{length}(\text{env}) \quad m < \text{length}(\text{env})$

shows

$(\exists p \in G. p \Vdash \text{Equal}(n, m) \text{ env}) \longleftrightarrow M[G], \text{map}(\text{val}(G), \text{env}) \models \text{Equal}(n, m)$

$\langle \text{proof} \rangle$

end — *G_generic1*

lemma *arities_at_aux*:

assumes

$n \in \text{nat } m \in \text{nat } \text{env} \in \text{list}(M) \text{ succ}(n) \cup \text{succ}(m) \leq \text{length}(\text{env})$

shows

$n < \text{length}(\text{env}) \ m < \text{length}(\text{env})$

$\langle \text{proof} \rangle$

13.13 The Strengthening Lemma

context *forcing_data1*

begin

lemma *strengthening_lemma*:

assumes

$p \in \mathbb{P} \ \varphi \in \text{formula} \ r \in \mathbb{P} \ r \preceq p$

$\text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{length}(\text{env})$

shows

$p \Vdash \varphi \ \text{env} \implies r \Vdash \varphi \ \text{env}$

$\langle \text{proof} \rangle$

13.14 The Density Lemma

lemma *arity_Nand_le*:

assumes $\varphi \in \text{formula} \ \psi \in \text{formula} \ \text{arity}(\text{Nand}(\varphi, \psi)) \leq \text{length}(\text{env}) \ \text{env} \in \text{list}(A)$

shows $\text{arity}(\varphi) \leq \text{length}(\text{env}) \ \text{arity}(\psi) \leq \text{length}(\text{env})$

$\langle \text{proof} \rangle$

lemma *dense_below_imp_forces*:

assumes

$p \in \mathbb{P} \ \varphi \in \text{formula}$

$\text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{length}(\text{env})$

shows

$\text{dense_below}(\{q \in \mathbb{P}. (q \Vdash \varphi \ \text{env})\}, p) \implies (p \Vdash \varphi \ \text{env})$

$\langle \text{proof} \rangle$

lemma *density_lemma*:

assumes

$p \in \mathbb{P} \ \varphi \in \text{formula} \ \text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{length}(\text{env})$

shows

$p \Vdash \varphi \ \text{env} \iff \text{dense_below}(\{q \in \mathbb{P}. (q \Vdash \varphi \ \text{env})\}, p)$

$\langle \text{proof} \rangle$

13.15 The Truth Lemma

lemma *Forces_And*:

assumes

$p \in \mathbb{P} \ \text{env} \in \text{list}(M) \ \varphi \in \text{formula} \ \psi \in \text{formula}$

$\text{arity}(\varphi) \leq \text{length}(\text{env}) \ \text{arity}(\psi) \leq \text{length}(\text{env})$

shows

$p \Vdash \text{And}(\varphi, \psi) \ \text{env} \iff (p \Vdash \varphi \ \text{env}) \wedge (p \Vdash \psi \ \text{env})$

$\langle proof \rangle$

lemma *Forces_Nand_alt*:

assumes

$p \in \mathbb{P}$ $env \in list(M)$ $\varphi \in formula$ $\psi \in formula$
 $arity(\varphi) \leq length(env)$ $arity(\psi) \leq length(env)$

shows

$(p \Vdash Nand(\varphi, \psi) env) \longleftrightarrow (p \Vdash Neg(And(\varphi, \psi)) env)$

$\langle proof \rangle$

end

context *G_generic1*

begin

lemma *truth_lemma_Neg*:

assumes

$\varphi \in formula$ $env \in list(M)$ $arity(\varphi) \leq length(env)$ **and**
 $IH: (\exists p \in G. p \Vdash \varphi env) \longleftrightarrow M[G], map(val(G), env) \models \varphi$

shows

$(\exists p \in G. p \Vdash Neg(\varphi) env) \longleftrightarrow M[G], map(val(G), env) \models Neg(\varphi)$

$\langle proof \rangle$

lemma *truth_lemma_And*:

assumes

$env \in list(M)$ $\varphi \in formula$ $\psi \in formula$
 $arity(\varphi) \leq length(env)$ $arity(\psi) \leq length(env)$

and

$IH: (\exists p \in G. p \Vdash \varphi env) \longleftrightarrow M[G], map(val(G), env) \models \varphi$
 $(\exists p \in G. p \Vdash \psi env) \longleftrightarrow M[G], map(val(G), env) \models \psi$

shows

$(\exists p \in G. (p \Vdash And(\varphi, \psi) env)) \longleftrightarrow M[G], map(val(G), env) \models And(\varphi, \psi)$

$\langle proof \rangle$

end

definition

ren_truth_lemma :: $i \Rightarrow i$ **where**

ren_truth_lemma(φ) \equiv

$Exists(Exists(Exists(Exists(Exists($
 $And(Equal(0, 5), And(Equal(1, 8), And(Equal(2, 9), And(Equal(3, 10), And(Equal(4, 6),$
 $iterates(\lambda p. incr_bv(p) '5 , 6, \varphi))))))))))$

lemma *ren_truth_lemma_type*[*TC*] :

$\varphi \in formula \implies ren_truth_lemma(\varphi) \in formula$

$\langle proof \rangle$

lemma *arity_ren_truth* :

assumes $\varphi \in formula$

shows $\text{arity}(\text{ren_truth_lemma}(\varphi)) \leq 6 \cup \text{succ}(\text{arity}(\varphi))$
 <proof>

lemma *sats_ren_truth_lemma*:
 $[q,b,d,a1,a2,a3] @ \text{env} \in \text{list}(M) \implies \varphi \in \text{formula} \implies$
 $(M, [q,b,d,a1,a2,a3] @ \text{env} \models \text{ren_truth_lemma}(\varphi)) \longleftrightarrow$
 $(M, [q,a1,a2,a3,b] @ \text{env} \models \varphi)$
 <proof>

context *forcing_data1*
begin

lemma *truth_lemma'* :
assumes
 $\varphi \in \text{formula} \ \text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{succ}(\text{length}(\text{env}))$
shows
 $\text{separation}(\#\#M, \lambda d. \exists b \in M. \forall q \in \mathbb{P}. q \preceq d \longrightarrow \neg(q \Vdash \varphi ([b]@\text{env})))$
 <proof>

end

context *G_generic1*
begin

lemma *truth_lemma*:
assumes
 $\varphi \in \text{formula}$
 $\text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{length}(\text{env})$
shows
 $(\exists p \in G. p \Vdash \varphi \ \text{env}) \longleftrightarrow M[G], \text{map}(\text{val}(G), \text{env}) \models \varphi$
 <proof>

end

context *forcing_data1*
begin

13.16 The “Definition of forcing”

lemma *definition_of_forcing*:
assumes
 $p \in \mathbb{P} \ \varphi \in \text{formula} \ \text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{length}(\text{env})$
shows
 $(p \Vdash \varphi \ \text{env}) \longleftrightarrow$
 $(\forall G. M_generic(G) \wedge p \in G \longrightarrow M[G], \text{map}(\text{val}(G), \text{env}) \models \varphi)$
 <proof>

lemmas *definability = forces_type*

end — *forcing_data1*

end

14 Ordinals in generic extensions

theory *Ordinals_In_MG*

imports

Forcing_Theorems

begin

context *G_generic1*

begin

lemma *rank_val*: $\text{rank}(\text{val}(G,x)) \leq \text{rank}(x)$ (**is** $?Q(x)$)

<proof>

lemma *Ord_MG_iff*:

assumes $\text{Ord}(\alpha)$

shows $\alpha \in M \longleftrightarrow \alpha \in M[G]$

<proof>

end — *G_generic1*

end

15 Auxiliary renamings for Separation

theory *Separation_Rename*

imports

Interface

begin

no_notation *Aleph* ($\aleph_{[90]}$ *90*)

lemmas *apply_fun* = *apply_iff*[*THEN iffD1*]

lemma *nth_concat* : $[p,t] \in \text{list}(A) \implies \text{env} \in \text{list}(A) \implies \text{nth}(1 +_{\omega} \text{length}(\text{env}), [p]@$

$\text{env} @ [t]) = t$

<proof>

lemma *nth_concat2* : $\text{env} \in \text{list}(A) \implies \text{nth}(\text{length}(\text{env}), \text{env} @ [p,t]) = p$

<proof>

lemma *nth_concat3* : $\text{env} \in \text{list}(A) \implies u = \text{nth}(\text{succ}(\text{length}(\text{env})), \text{env} @ [pi, u])$

<proof>

definition

$sep_var :: i \Rightarrow i$ **where**
 $sep_var(n) \equiv \{\langle 0,1 \rangle, \langle 1,3 \rangle, \langle 2,4 \rangle, \langle 3,5 \rangle, \langle 4,0 \rangle, \langle 5+\omega n,6 \rangle, \langle 6+\omega n,2 \rangle\}$

definition

$sep_env :: i \Rightarrow i$ **where**
 $sep_env(n) \equiv \lambda i \in (5+\omega n)-5 . i+\omega 2$

definition $weak :: [i, i] \Rightarrow i$ **where**

$weak(n,m) \equiv \{i+\omega m . i \in n\}$

lemma $weakD$:

assumes $n \in nat$ $k \in nat$ $x \in weak(n,k)$
shows $\exists i \in n . x = i+\omega k$
 $\langle proof \rangle$

lemma $weak_equal$:

assumes $n \in nat$ $m \in nat$
shows $weak(n,m) = (m+\omega n) - m$
 $\langle proof \rangle$

lemma $weak_zero$:

shows $weak(0,n) = 0$
 $\langle proof \rangle$

lemma $weakening_diff$:

assumes $n \in nat$
shows $weak(n,7) - weak(n,5) \subseteq \{5+\omega n, 6+\omega n\}$
 $\langle proof \rangle$

lemma in_add_del :

assumes $x \in j+\omega n$ $n \in nat$ $j \in nat$
shows $x < j \vee x \in weak(n,j)$
 $\langle proof \rangle$

lemma sep_env_action :

assumes
 $[t,p,u,P,leg,o,pi] \in list(M)$
 $env \in list(M)$
shows $\forall i . i \in weak(length(env),5) \longrightarrow$
 $nth(sep_env(length(env)))'i,[t,p,u,P,leg,o,pi]@env) = nth(i,[p,P,leg,o,t] @ env$
 $@ [pi,u])$
 $\langle proof \rangle$

lemma sep_env_type :

assumes $n \in nat$
shows $sep_env(n) : (5+\omega n)-5 \rightarrow (7+\omega n)-7$
 $\langle proof \rangle$

lemma *sep_var_fin_type* :

assumes $n \in \text{nat}$

shows $\text{sep_var}(n) : \gamma_{+\omega} n - || > \gamma_{+\omega} n$

$\langle \text{proof} \rangle$

lemma *sep_var_domain* :

assumes $n \in \text{nat}$

shows $\text{domain}(\text{sep_var}(n)) = \gamma_{+\omega} n - \text{weak}(n, 5)$

$\langle \text{proof} \rangle$

lemma *sep_var_type* :

assumes $n \in \text{nat}$

shows $\text{sep_var}(n) : (\gamma_{+\omega} n) - \text{weak}(n, 5) \rightarrow \gamma_{+\omega} n$

$\langle \text{proof} \rangle$

lemma *sep_var_action* :

assumes

$[t, p, u, P, \text{leq}, o, pi] \in \text{list}(M)$

$\text{env} \in \text{list}(M)$

shows $\forall i . i \in (\gamma_{+\omega} \text{length}(\text{env})) - \text{weak}(\text{length}(\text{env}), 5) \longrightarrow$

$\text{nth}(\text{sep_var}(\text{length}(\text{env})) 'i, [t, p, u, P, \text{leq}, o, pi] @ \text{env}) = \text{nth}(i, [p, P, \text{leq}, o, t] @ \text{env}$

$@ [pi, u]$

$\langle \text{proof} \rangle$

definition

$\text{rensep} :: i \Rightarrow i$ **where**

$\text{rensep}(n) \equiv \text{union_fun}(\text{sep_var}(n), \text{sep_env}(n), \gamma_{+\omega} n - \text{weak}(n, 5), \text{weak}(n, 5))$

lemma *rensep_aux* :

assumes $n \in \text{nat}$

shows $(\gamma_{+\omega} n - \text{weak}(n, 5)) \cup \text{weak}(n, 5) = \gamma_{+\omega} n \cap \gamma_{+\omega} n \cup (\gamma_{+\omega} n - \gamma) = \gamma_{+\omega} n$

$\langle \text{proof} \rangle$

lemma *rensep_type* :

assumes $n \in \text{nat}$

shows $\text{rensep}(n) \in \gamma_{+\omega} n \rightarrow \gamma_{+\omega} n$

$\langle \text{proof} \rangle$

lemma *rensep_action* :

assumes $[t, p, u, P, \text{leq}, o, pi] @ \text{env} \in \text{list}(M)$

shows $\forall i . i < \gamma_{+\omega} \text{length}(\text{env}) \longrightarrow \text{nth}(\text{rensep}(\text{length}(\text{env})) 'i, [t, p, u, P, \text{leq}, o, pi] @ \text{env})$

$= \text{nth}(i, [p, P, \text{leq}, o, t] @ \text{env} @ [pi, u])$

$\langle \text{proof} \rangle$

definition *sep_ren* :: $[i, i] \Rightarrow i$ **where**

$\text{sep_ren}(n, \varphi) \equiv \text{ren}(\varphi) '(\gamma_{+\omega} n) '(\gamma_{+\omega} n) ' \text{rensep}(n)$

lemma *arity_rensep*: **assumes** $\varphi \in \text{formula}$ $\text{env} \in \text{list}(M)$

$\text{arity}(\varphi) \leq \gamma_{+\omega} \text{length}(\text{env})$

shows $\text{arity}(\text{sep_ren}(\text{length}(\text{env}),\varphi)) \leq 7 +_{\omega} \text{length}(\text{env})$
 ⟨proof⟩

lemma *type_rensep* [TC]:
assumes $\varphi \in \text{formula}$ $\text{env} \in \text{list}(M)$
shows $\text{sep_ren}(\text{length}(\text{env}),\varphi) \in \text{formula}$
 ⟨proof⟩

lemma *sepren_action*:
assumes $\text{arity}(\varphi) \leq 7 +_{\omega} \text{length}(\text{env})$
 $[t,p,u,P,\text{leq},o,\text{pi}] \in \text{list}(M)$
 $\text{env} \in \text{list}(M)$
 $\varphi \in \text{formula}$
shows $\text{sats}(M, \text{sep_ren}(\text{length}(\text{env}),\varphi), [t,p,u,P,\text{leq},o,\text{pi}] @ \text{env}) \longleftrightarrow \text{sats}(M,$
 $\varphi, [p,P,\text{leq},o,t] @ \text{env} @ [pi,u])$
 ⟨proof⟩

end

16 The Axiom of Separation in $M[G]$

theory *Separation_Axiom*
imports *Forcing_Theorems Separation_Rename*
begin

context *G_generic1*
begin

lemma *map_val* :
assumes $\text{env} \in \text{list}(M[G])$
shows $\exists \text{nenv} \in \text{list}(M). \text{env} = \text{map}(\text{val}(G), \text{nenv})$
 ⟨proof⟩

lemma *Collect_sats_in_MG* :
assumes
 $A \in M[G]$
 $\varphi \in \text{formula}$ $\text{env} \in \text{list}(M[G])$ $\text{arity}(\varphi) \leq 1 +_{\omega} \text{length}(\text{env})$
shows
 $\{x \in A . (M[G], [x] @ \text{env} \models \varphi)\} \in M[G]$
 ⟨proof⟩

theorem *separation_in_MG*:
assumes
 $\varphi \in \text{formula}$ **and** $\text{arity}(\varphi) \leq 1 +_{\omega} \text{length}(\text{env})$ **and** $\text{env} \in \text{list}(M[G])$
shows
 $\text{separation}(\#\#M[G], \lambda x. (M[G], [x] @ \text{env} \models \varphi))$
 ⟨proof⟩

end — *G_generic1*

end

17 The Axiom of Pairing in $M[G]$

theory *Pairing_Axiom*

imports

Names

begin

context *G_generic1*

begin

lemma *val_Upair* :

$\mathbf{1} \in G \implies \text{val}(G, \{\langle \tau, \mathbf{1} \rangle, \langle \rho, \mathbf{1} \rangle\}) = \{\text{val}(G, \tau), \text{val}(G, \rho)\}$

<proof>

lemma *pairing_in_MG* : *upair_ax*($\#\#M[G]$)

<proof>

end — *G_generic1*

end

18 The Axiom of Unions in $M[G]$

theory *Union_Axiom*

imports *Names*

begin

definition *Union_name_body* :: $[i, i, i, i] \Rightarrow o$ **where**

$\text{Union_name_body}(P, \text{leq}, \tau, x) \equiv \exists \sigma \in \text{domain}(\tau) . \exists q \in P . \exists r \in P .$

$\langle \sigma, q \rangle \in \tau \wedge \langle \text{fst}(x), r \rangle \in \sigma \wedge \langle \text{snd}(x), r \rangle \in \text{leq} \wedge \langle \text{snd}(x), q \rangle \in \text{leq}$

definition *Union_name* :: $[i, i, i] \Rightarrow i$ **where**

$\text{Union_name}(P, \text{leq}, \tau) \equiv \{u \in \text{domain}(\bigcup (\text{domain}(\tau))) \times P . \text{Union_name_body}(P, \text{leq}, \tau, u)\}$

context *forcing_data1*

begin

lemma *Union_name_closed* :

assumes $\tau \in M$

shows $\text{Union_name}(\mathbb{P}, \text{leq}, \tau) \in M$

<proof>

lemma *Union_MG_Eq* :

assumes $a \in M[G]$ **and** $a = \text{val}(G, \tau)$ **and** *filter*(G) **and** $\tau \in M$

shows $\bigcup a = \text{val}(G, \text{Union_name}(\mathbb{P}, \text{leq}, \tau))$

<proof>

lemma *union_in_MG* :
 assumes *filter*(*G*)
 shows *Union_ax*($\#\#M[G]$)
 <proof>

theorem *Union_MG* : $M_generic(G) \implies Union_ax(\#\#M[G])$
 <proof>

end — *forcing_data1*

end

19 The Powerset Axiom in $M[G]$

theory *Powerset_Axiom*
 imports
 Separation_Axiom Pairing_Axiom Union_Axiom
begin

<ML>

context *G_generic1*
begin

lemma *satsfst_snd_in_M*:
 assumes
 $A \in M \ B \in M \ \varphi \in \text{formula} \ p \in M \ l \in M \ o \in M \ \chi \in M \ \text{arity}(\varphi) \leq 6$
 shows $\{\langle s, q \rangle \in A \times B \ . \ M, [q, p, l, o, s, \chi] \models \varphi\} \in M$ (**is** $\emptyset \in M$)
 <proof>

declare *nat_into_M* [*rule del, simplified setclass_iff, intro*]
lemmas *ssimps* = *domain_closed cartprod_closed cons_closed Pow_rel_closed*
declare *ssimps* [*simp del, simplified setclass_iff, simp, intro*]

— We keep $Pow(a) \cap M[G]$ to be consistent with Kunen.

lemma *Pow_inter_MG*:
 assumes $a \in M[G]$
 shows $Pow(a) \cap M[G] \in M[G]$
 <proof>

end — *G_generic1*

sublocale $G_generic1 \subseteq \text{ext}: M_trivial \ \#\#M[G]$
 <proof>

context *G_generic1* **begin**

```

theorem power_in_MG : power_ax(##(M[G]))
  <proof>

end — G_generic1

end

```

20 The Axiom of Extensionality in $M[G]$

```

theory Extensionality_Axiom
  imports
    Names
begin

context forcing_data1
begin

lemma extensionality_in_MG : extensionality(##(M[G]))
  <proof>

end — forcing_data1

end

```

21 The Axiom of Foundation in $M[G]$

```

theory Foundation_Axiom
  imports
    Names
begin

context forcing_data1
begin

lemma foundation_in_MG : foundation_ax(##(M[G]))
  <proof>

lemma foundation_ax(##(M[G]))
  <proof>

end — forcing_data1

end

```

22 The Axiom of Replacement in $M[G]$

theory *Replacement_Axiom*

imports

Separation_Axiom

begin

context *forcing_data1*

begin

bundle *sharp_simps1* = *snd_abs[simp]* *fst_abs[simp]* *fst_closed[simp del, simplified, simp]*

snd_closed[simp del, simplified, simp] *M_inhabited[simplified, simp]*

pair_in_M_iff[simp del, simplified, simp]

lemma *sats_body_ground_repl_fm*:

includes *sharp_simps1*

assumes

$\exists t p. x = \langle t, p \rangle$ $[x, \alpha, m, \mathbb{P}, \text{leq}, \mathbf{1}] @ \text{nenv} \in \text{list}(M)$

$\varphi \in \text{formula}$

shows

$(\exists \tau \in M. \exists V \in M. \text{is_Vset}(\lambda a. (\#\#M)(a), \alpha, V) \wedge \tau \in V \wedge (\text{snd}(x) \Vdash \varphi$
 $([\text{fst}(x), \tau] @ \text{nenv})))$

$\longleftrightarrow M, [\alpha, x, m, \mathbb{P}, \text{leq}, \mathbf{1}] @ \text{nenv} \models \text{body_ground_repl_fm}(\varphi)$

$\langle \text{proof} \rangle$

end — *forcing_data1*

context *G_generic1*

begin

lemma *Replace_sats_in_MG*:

assumes

$c \in M[G]$ $\text{env} \in \text{list}(M[G])$

$\varphi \in \text{formula}$ $\text{arity}(\varphi) \leq 2 + \omega \text{ length}(\text{env})$

$\text{univalent}(\#\#M[G], c, \lambda x v. (M[G], [x, v] @ \text{env} \models \varphi))$

and

ground_replacement:

$\bigwedge \text{nenv}. \text{ground_replacement_assm}(M, [\mathbb{P}, \text{leq}, \mathbf{1}] @ \text{nenv}, \varphi)$

shows

$\{v. x \in c, v \in M[G] \wedge (M[G], [x, v] @ \text{env} \models \varphi)\} \in M[G]$

$\langle \text{proof} \rangle$

theorem *strong_replacement_in_MG*:

assumes

$\varphi \in \text{formula}$ **and** $\text{arity}(\varphi) \leq 2 + \omega \text{ length}(\text{env})$ $\text{env} \in \text{list}(M[G])$

and

ground_replacement:

$\bigwedge \text{nenv}. \text{ground_replacement_assm}(M, [\mathbb{P}, \text{leq}, \mathbf{1}] @ \text{nenv}, \varphi)$

shows
 $strong_replacement(\#\#M[G], \lambda x v. M[G], [x, v] @ env \models \varphi)$
 $\langle proof \rangle$

lemma *replacement_assm_MG*:

assumes

ground_replacement:

$\bigwedge nenv. ground_replacement_assm(M, [\mathbb{P}, leq, 1] @ nenv, \varphi)$

shows

$replacement_assm(M[G], env, \varphi)$

$\langle proof \rangle$

end — *G_generic1*

end

23 The Axiom of Infinity in $M[G]$

theory *Infinity_Axiom*

imports *Union_Axiom Pairing_Axiom*

begin

context *G_generic1* **begin**

interpretation *mg_triv*: $M_trivial\#\#M[G]$

$\langle proof \rangle$

lemma *infinity_in_MG* : $infinity_ax(\#\#M[G])$

$\langle proof \rangle$

end — *G_generic1*

end

24 The Axiom of Choice in $M[G]$

theory *Choice_Axiom*

imports

Powerset_Axiom

Extensionality_Axiom

Foundation_Axiom

Replacement_Axiom

Infinity_Axiom

begin

definition

upair_name :: $i \Rightarrow i \Rightarrow i \Rightarrow i$ **where**

$upair_name(\tau, \rho, on) \equiv Upair(\langle \tau, on \rangle, \langle \rho, on \rangle)$

definition

$opair_name :: i \Rightarrow i \Rightarrow i \Rightarrow i$ **where**
 $opair_name(\tau, \varrho, on) \equiv upair_name(upair_name(\tau, \tau, on), upair_name(\tau, \varrho, on), on)$

definition

$induced_surj :: i \Rightarrow i \Rightarrow i \Rightarrow i$ **where**
 $induced_surj(f, a, e) \equiv f^{-1}((range(f)-a) \times \{e\} \cup restrict(f, f^{-1}a))$

lemma $domain_induced_surj$: $domain(induced_surj(f, a, e)) = domain(f)$
 $\langle proof \rangle$

lemma $range_restrict_vimage$:
assumes $function(f)$
shows $range(restrict(f, f^{-1}a)) \subseteq a$
 $\langle proof \rangle$

lemma $induced_surj_type$:
assumes $function(f)$
shows
 $induced_surj(f, a, e) : domain(f) \rightarrow \{e\} \cup a$
and
 $x \in f^{-1}a \implies induced_surj(f, a, e) x = f x$
 $\langle proof \rangle$

lemma $induced_surj_is_surj$:
assumes
 $e \in a \ function(f) \ domain(f) = \alpha \ \wedge \ y. y \in a \implies \exists x \in \alpha. f x = y$
shows $induced_surj(f, a, e) \in surj(\alpha, a)$
 $\langle proof \rangle$

lemma (**in** M_ZF1_trans) $upair_name_closed$:
 $\llbracket x \in M; y \in M; o \in M \rrbracket \implies upair_name(x, y, o) \in M$
 $\langle proof \rangle$

context $G_generic1$
begin

lemma val_upair_name : $val(G, upair_name(\tau, \varrho, 1)) = \{val(G, \tau), val(G, \varrho)\}$
 $\langle proof \rangle$

lemma val_opair_name : $val(G, opair_name(\tau, \varrho, 1)) = \langle val(G, \tau), val(G, \varrho) \rangle$
 $\langle proof \rangle$

lemma val_RepFun_one : $val(G, \{\langle f(x), 1 \rangle . x \in a\}) = \{val(G, f(x)) . x \in a\}$
 $\langle proof \rangle$

end— $G_generic1$

24.1 $M[G]$ is a transitive model of ZF

sublocale $G_generic1 \subseteq ext:M_Z_trans M[G]$
<proof>

lemma (in $M_replacement$) $upair_name_lam_replacement$:
 $M(z) \implies lam_replacement(M, \lambda x . upair_name(fst(x), snd(x), z))$
<proof>

lemma (in $forcing_data1$) $repl_opname_check$:
assumes $A \in M \ f \in M$
shows $\{opair_name(check(x), f'x, \mathbf{1}). x \in A\} \in M$
<proof>

theorem (in $G_generic1$) $choice_in_MG$:
assumes $choice_ax(\#\#M)$
shows $choice_ax(\#\#M[G])$
<proof>

sublocale $G_generic1_AC \subseteq ext:M_ZC_basic M[G]$
<proof>

end

25 Separative notions and proper extensions

theory $Proper_Extension$
imports
 $Names$

begin

The key ingredient to obtain a proper extension is to have a *separative preorder*:

locale $separative_notion = forcing_notion +$
assumes $separative: p \in \mathbb{P} \implies \exists q \in \mathbb{P}. \exists r \in \mathbb{P}. q \preceq p \wedge r \preceq p \wedge q \perp r$
begin

For separative preorders, the complement of every filter is dense. Hence an M -generic filter cannot belong to the ground model.

lemma $filter_complement_dense$:
assumes $filter(G)$
shows $dense(\mathbb{P} - G)$
<proof>

end — $separative_notion$

locale $ctm_separative = forcing_data1 + separative_notion$
begin

```

context
  fixes  $G$ 
  assumes  $generic: M\_generic(G)$ 
begin

interpretation  $G\_generic1 \mathbb{P} leq \mathbf{1} M enum G$ 
   $\langle proof \rangle$ 

lemma  $generic\_not\_in\_M:$ 
  shows  $G \notin M$ 
   $\langle proof \rangle$ 

theorem  $proper\_extension: M \neq M[G]$ 
   $\langle proof \rangle$ 
end
end —  $ctm\_separative$ 

end

```

26 A poset of successions

```

theory  $Succession\_Poset$ 
  imports
     $ZF\_Trans\_Interpretations$ 
     $Proper\_Extension$ 
begin

```

In this theory we define a separative poset. Its underlying set is the set of finite binary sequences (that is, with codomain $2 = \{0, 1\}$); of course, one can see that set as the set $\omega \multimap 2$ or equivalently as the set of partial functions $Fn(\omega, \omega, 2)$, i.e. the set of partial functions bounded by ω .

The order relation of the poset is that of being less defined as functions (cf. $Fnlerel(A^{<\omega})$), so it could be surprising that we have not used $Fn(\omega, \omega, 2)$ for the set. The only reason why we keep this alternative definition is because we can prove $A^{<\omega} \in M$ (and therefore $Fnlerel(A^{<\omega}) \in M$) using only one instance of separation.

```

definition  $seq\_upd :: i \Rightarrow i \Rightarrow i$  where
   $seq\_upd(f, a) \equiv \lambda j \in succ(domain(f)) . if j < domain(f) then f`j else a$ 

```

```

lemma  $seq\_upd\_succ\_type :$ 
  assumes  $n \in nat \ f \in n \rightarrow A \ a \in A$ 
  shows  $seq\_upd(f, a) \in succ(n) \rightarrow A$ 
   $\langle proof \rangle$ 

```

```

lemma  $seq\_upd\_type :$ 
  assumes  $f \in A^{<\omega} \ a \in A$ 

```

shows $seq_upd(f,a) \in A^{<\omega}$
<proof>

lemma $seq_upd_apply_domain$ [*simp*]:
assumes $f:n \rightarrow A$ $n \in nat$
shows $seq_upd(f,a) \cdot n = a$
<proof>

lemma $zero_in_seqspace$:
shows $0 \in A^{<\omega}$
<proof>

definition
 $seqlerel :: i \Rightarrow i$ **where**
 $seqlerel(A) \equiv Fnlerel(A^{<\omega})$

definition
 $seqle :: i$ **where**
 $seqle \equiv seqlerel(2)$

lemma $seqleI$ [*intro!*]:
 $\langle f,g \rangle \in 2^{<\omega} \times 2^{<\omega} \implies g \subseteq f \implies \langle f,g \rangle \in seqle$
<proof>

lemma $seqleD$ [*dest!*]:
 $z \in seqle \implies \exists x y. \langle x,y \rangle \in 2^{<\omega} \times 2^{<\omega} \wedge y \subseteq x \wedge z = \langle x,y \rangle$
<proof>

lemma upd_leI :
assumes $f \in 2^{<\omega}$ $a \in 2$
shows $\langle seq_upd(f,a), f \rangle \in seqle$ (**is** $\langle ?f, _ \rangle \in _$)
<proof>

lemma $preorder_on_seqle$: $preorder_on(2^{<\omega}, seqle)$
<proof>

lemma $zero_seqle_max$: $x \in 2^{<\omega} \implies \langle x, 0 \rangle \in seqle$
<proof>

interpretation sp : $forcing_notion$ $2^{<\omega}$ $seqle$ 0
<proof>

notation sp . Leq (**infixl** $\langle \preceq_s \rangle$ 50)
notation sp . $Incompatible$ (**infixl** $\langle \perp_s \rangle$ 50)

lemma $seqspace_separative$:
assumes $f \in 2^{<\omega}$
shows $seq_upd(f,0) \perp_s seq_upd(f,1)$ (**is** $?f \perp_s ?g$)
<proof>

definition $seqleR_fm :: i \Rightarrow i$ **where**
 $seqleR_fm(fg) \equiv Exists(Exists(And(pair_fm(0,1,fg+\omega 2),subset_fm(1,0))))$

lemma $type_seqleR_fm : fg \in nat \Longrightarrow seqleR_fm(fg) \in formula$
 $\langle proof \rangle$

$\langle ML \rangle$

lemma (in M_ctm1) $seqleR_fm_sats :$
assumes $fg \in nat \ env \in list(M)$
shows $(M, env \models seqleR_fm(fg)) \longleftrightarrow (\exists f[\#\#M]. \exists g[\#\#M]. pair(\#\#M, f, g, nth(fg, env))$
 $\wedge f \supseteq g)$
 $\langle proof \rangle$

context M_ctm1
begin

lemma $seqle_in_M: seqle \in M$
 $\langle proof \rangle$

26.1 Cohen extension is proper

interpretation $ctm_separative \ 2^{<\omega} \ seqle \ 0$
 $\langle proof \rangle$

lemma $cohen_extension_is_proper: \exists G. M_generic(G) \wedge M \neq M[G]$
 $\langle proof \rangle$

end — M_ctm1

end

27 The existence of generic extensions

theory $Forcing_Main$

imports

$Ordinals_In_MG$

$Choice_Axiom$

$Succession_Poset$

begin

27.1 The generic extension is countable

lemma (in $forcing_data1$) $surj_nat_MG : \exists f. f \in surj(\omega, M[G])$
 $\langle proof \rangle$

lemma (in $G_generic1$) $MG_eqpoll_nat: M[G] \approx \omega$

<proof>

27.2 Extensions of ctms of fragments of ZFC

context $G_generic1$

begin

lemma $sats_ground_repl_fm_imp_sats_ZF_replacement_fm$:

assumes

$\varphi \in formula\ M, \ [] \models \cdot Replacement(ground_repl_fm(\varphi)) \cdot$

shows

$M[G], \ [] \models \cdot Replacement(\varphi) \cdot$

<proof>

lemma $satT_ground_repl_fm_imp_satT_ZF_replacement_fm$:

assumes

$\Phi \subseteq formula\ M \models \{ \cdot Replacement(ground_repl_fm(\varphi)) \cdot \ . \ \varphi \in \Phi \}$

shows

$M[G] \models \{ \cdot Replacement(\varphi) \cdot \ . \ \varphi \in \Phi \}$

<proof>

end — $G_generic1$

theorem $extensions_of_ctms$:

assumes

$M \approx \omega\ Transset(M)$

$M \models \cdot Z \cup \{ \cdot Replacement(p) \cdot \ . \ p \in overhead \}$

$\Phi \subseteq formula\ M \models \{ \cdot Replacement(ground_repl_fm(\varphi)) \cdot \ . \ \varphi \in \Phi \}$

shows

$\exists N.$

$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge M \neq N \wedge$

$(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N)) \wedge$

$((M, [] \models \cdot AC \cdot) \longrightarrow N, [] \models \cdot AC \cdot) \wedge N \models \cdot Z \cup \{ \cdot Replacement(\varphi) \cdot \ . \ \varphi \in \Phi \}$

<proof>

lemma $ZF_replacement_overhead_sub_ZF$: $\{ \cdot Replacement(p) \cdot \ . \ p \in overhead \} \subseteq ZF$

<proof>

theorem $extensions_of_ctms_ZF$:

assumes

$M \approx \omega\ Transset(M) \ M \models ZF$

shows

$\exists N.$

$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZF \wedge M \neq N \wedge$

$(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N)) \wedge$

$((M, [] \models \cdot AC \cdot) \longrightarrow N \models ZFC)$

<proof>

end

28 Preservation of cardinals in generic extensions

theory *Cardinal_Preservation*

imports

Forcing_Main

begin

context *forcing_data1*

begin

lemma *antichain_abs'* [*absolut*]:

$\llbracket A \in M \rrbracket \implies \text{antichain}^M(\mathbb{P}, \text{leq}, A) \longleftrightarrow \text{antichain}(\mathbb{P}, \text{leq}, A)$

<proof>

lemma *inconsistent_imp_incompatible*:

assumes $p \Vdash \varphi$ env $q \Vdash \text{Neg}(\varphi)$ env $p \in \mathbb{P}$ $q \in \mathbb{P}$

$\text{arity}(\varphi) \leq \text{length}(\text{env})$ $\varphi \in \text{formula}$ env $\in \text{list}(M)$

shows $p \perp q$

<proof>

notation *check* ($\langle _ \rangle$ [101] 100)

end — *forcing_data1*

locale *G_generic2* = *G_generic1* + *forcing_data2*

locale *G_generic2_AC* = *G_generic1_AC* + *G_generic2*

locale *G_generic3* = *G_generic2* + *forcing_data3*

locale *G_generic3_AC* = *G_generic2_AC* + *G_generic3*

locale *G_generic3_AC_CH* = *G_generic3_AC* + *M_ZFC2_ground_CH_trans*

sublocale *G_generic3_AC* \subseteq ext:*M_ZFC2_trans* $M[G]$

<proof>

lemma (in *forcing_data1*) *forces_neq_apply_imp_incompatible*:

assumes

$p \Vdash \cdot 0'1$ is 2. $[f, a, b^v]$

$q \Vdash \cdot 0'1$ is 2. $[f, a, b^w]$

$b \neq b'$

— More general version: taking general names b^v and b^w , satisfying $p \Vdash \cdot \neg \cdot 0 = 1 \cdot [b^v, b^w]$ and $q \Vdash \cdot \neg \cdot 0 = 1 \cdot [b^v, b^w]$.

and

types: $f \in M$ $a \in M$ $b \in M$ $b' \in M$ $p \in \mathbb{P}$ $q \in \mathbb{P}$

shows

$p \perp q$
 <proof>
 include *G_generic1_lemmas*
 <proof>

context *M_ctm2_AC*
begin

— Simplifying simp rules (because of the occurrence of *setclass*)

lemmas *sharp_simps* = *Card_rel_Union Card_rel_cardinal_rel Collect_abs*
Cons_abs Cons_in_M_iff Diff_closed Equal_abs Equal_in_M_iff Finite_abs
Forall_abs Forall_in_M_iff Inl_abs Inl_in_M_iff Inr_abs Inr_in_M_iff
Int_closed Inter_abs Inter_closed M_nat Member_abs Member_in_M_iff
Memrel_closed Nand_abs Nand_in_M_iff Nil_abs Nil_in_M Ord_cardinal_rel
Pow_rel_closed Un_closed Union_abs Union_closed and_abs and_closed
apply_abs apply_closed bij_rel_closed bijection_abs bool_of_o_abs
bool_of_o_closed cadd_rel_0 cadd_rel_closed cardinal_rel_0_iff_0
cardinal_rel_closed cardinal_rel_idem cartprod_abs cartprod_closed
cmult_rel_0 cmult_rel_1 cmult_rel_closed comp_closed composition_abs
cons_abs cons_closed converse_abs converse_closed csquare_lam_closed
csquare_rel_closed depth_closed domain_abs domain_closed eclose_abs
eclose_closed empty_abs field_abs field_closed finite_funspace_closed
finite_ordinal_abs fst_closed function_abs function_space_rel_closed
hd_abs image_abs image_closed inj_rel_closed injection_abs inter_abs
irreflexive_abs is_eclose_n_abs is_funspace_abs
iterates_closed length_closed lepoll_rel_refl
limit_ordinal_abs linear_rel_abs
mem_bij_abs mem_eclose_abs mem_inj_abs membership_abs
minimum_closed nat_case_abs nat_case_closed nonempty_not_abs
not_closed number1_abs number2_abs number3_abs omega_abs
or_abs or_closed order_isomorphism_abs ordermap_closed
ordertype_closed ordinal_abs pair_abs pair_in_M_iff powerset_abs
pred_closed pred_set_abs quaselist_abs quasinat_abs radd_closed
rall_abs range_abs range_closed relation_abs restrict_closed
restriction_abs rex_abs rmult_closed rtrancl_abs rtrancl_closed
rvimage_closed separation_closed setdiff_abs singleton_abs
singleton_in_M_iff snd_closed strong_replacement_closed subset_abs
succ_in_M_iff successor_abs successor_ordinal_abs sum_abs sum_closed
surj_rel_closed surjection_abs tl_abs trancl_abs trancl_closed
transitive_rel_abs transitive_set_abs typed_function_abs union_abs
upair_abs upair_in_M_iff vimage_abs vimage_closed well_ord_abs
nth_closed Aleph_rel_closed csucc_rel_closed
Card_rel_Aleph_rel

declare *sharp_simps*[*simp del, simplified setclass_iff, simp*]

lemmas *sharp_intros* = *nat_into_M Aleph_rel_closed Card_rel_Aleph_rel*

declare *sharp_intros*[*rule del, simplified setclass_iff, intro*]

```

end — M_ctm2_AC

context G_generic3_AC begin

context
  includes G_generic1_lemmas
begin

lemmas mg_sharp_simps = ext.Card_rel_Union ext.Card_rel_cardinal_rel
  ext.Collect_abs ext.Cons_abs ext.Cons_in_M_iff ext.Diff_closed
  ext.Equal_abs ext.Equal_in_M_iff ext.Finite_abs ext.Forall_abs
  ext.Forall_in_M_iff ext.Inl_abs ext.Inl_in_M_iff ext.Inr_abs
  ext.Inr_in_M_iff ext.Int_closed ext.Inter_abs ext.Inter_closed
  ext.M_nat ext.Member_abs ext.Member_in_M_iff ext.Memrel_closed
  ext.Nand_abs ext.Nand_in_M_iff ext.Nil_abs ext.Nil_in_M
  ext.Ord_cardinal_rel ext.Pow_rel_closed ext.Un_closed
  ext.Union_abs ext.Union_closed ext.and_abs ext.and_closed
  ext.apply_abs ext.apply_closed ext.bij_rel_closed
  ext.bijection_abs ext.bool_of_o_abs ext.bool_of_o_closed
  ext.cadd_rel_0 ext.cadd_rel_closed ext.cardinal_rel_0_iff_0
  ext.cardinal_rel_closed ext.cardinal_rel_idem ext.cartprod_abs
  ext.cartprod_closed ext.cmult_rel_0 ext.cmult_rel_1
  ext.cmult_rel_closed ext.comp_closed ext.composition_abs
  ext.cons_abs ext.cons_closed ext.converse_abs ext.converse_closed
  ext.csquare_lam_closed ext.csquare_rel_closed ext.depth_closed
  ext.domain_abs ext.domain_closed ext.eclose_abs ext.eclose_closed
  ext.empty_abs ext.field_abs ext.field_closed
  ext.finite_funspace_closed ext.finite_ordinal_abs
  ext.fst_closed ext.function_abs ext.function_space_rel_closed
  ext.hd_abs ext.image_abs ext.image_closed ext.inj_rel_closed
  ext.injection_abs ext.inter_abs ext.irreflexive_abs
  ext.is_eclose_n_abs ext.is_funspace_abs
  ext.iterates_closed ext.length_closed
  ext.lepoll_rel_refl ext.limit_ordinal_abs ext.linear_rel_abs
  ext.mem_bij_abs ext.mem_eclose_abs
  ext.mem_inj_abs ext.membership_abs
  ext.nat_case_abs ext.nat_case_closed
  ext.nonempty ext.not_abs ext.not_closed
  ext.number1_abs ext.number2_abs ext.number3_abs ext.omega_abs
  ext.or_abs ext.or_closed ext.order_isomorphism_abs
  ext.ordermap_closed ext.ordertype_closed ext.ordinal_abs
  ext.pair_abs ext.pair_in_M_iff ext.powerset_abs ext.pred_closed
  ext.pred_set_abs ext.quasilist_abs ext.quasinat_abs
  ext.radd_closed ext.rall_abs ext.range_abs ext.range_closed
  ext.relation_abs ext.restrict_closed ext.restriction_abs
  ext.rex_abs ext.rmult_closed ext.rtrancl_abs ext.rtrancl_closed
  ext.rvimage_closed ext.separation_closed ext.setdiff_abs
  ext.singleton_abs ext.singleton_in_M_iff ext.snd_closed

```

ext.strong_replacement_closed ext.subset_abs ext.succ_in_M_iff
ext.successor_abs ext.successor_ordinal_abs ext.sum_abs
ext.sum_closed ext.surj_rel_closed ext.surjection_abs ext.tl_abs
ext.trancl_abs ext.trancl_closed ext.transitive_rel_abs
ext.transitive_set_abs ext.typed_function_abs ext.union_abs
ext.upair_abs ext.upair_in_M_iff ext.vimage_abs ext.vimage_closed
ext.well_ord_abs ext.nth_closed ext.Aleph_rel_closed
ext.csucc_rel_closed ext.Card_rel_Aleph_rel

— The following was motivated by the fact that *ext.apply_closed* did not simplify appropriately.

declare *mg_sharp_simps*[*simp del, simplified setclass_iff, simp*]

lemmas *mg_sharp_intros* = *ext.nat_into_M ext.Aleph_rel_closed*
ext.Card_rel_Aleph_rel

declare *mg_sharp_intros*[*rule del, simplified setclass_iff, intro*]

— Kunen IV.2.31

lemma *forces_below_filter*:

assumes $M[G]$, $\text{map}(\text{val}(G), \text{env}) \models \varphi$ $p \in G$
 $\text{arity}(\varphi) \leq \text{length}(\text{env})$ $\varphi \in \text{formula}$ $\text{env} \in \text{list}(M)$
shows $\exists q \in G. q \preceq p \wedge q \Vdash \varphi$ env

<proof>

28.1 Preservation by ccc forcing notions

lemma *ccc_fun_closed_lemma_aux*:

assumes $f \cdot \text{dot} \in M$ $p \in M$ $a \in M$ $b \in M$

shows $\{q \in \mathbb{P} . q \preceq p \wedge (M, [q, \mathbb{P}, \text{leq}, \mathbf{1}, f \cdot \text{dot}, a^v, b^v] \models \text{forces}(\cdot 0'1 \text{ is } 2 \cdot))\} \in M$

<proof>

lemma *ccc_fun_closed_lemma_aux2*:

assumes $B \in M$ $f \cdot \text{dot} \in M$ $p \in M$ $a \in M$

shows $(\#\#M)(\lambda b \in B. \{q \in \mathbb{P} . q \preceq p \wedge (M, [q, \mathbb{P}, \text{leq}, \mathbf{1}, f \cdot \text{dot}, a^v, b^v] \models \text{forces}(\cdot 0'1 \text{ is } 2 \cdot))\})$

<proof>

lemma *ccc_fun_closed_lemma*:

assumes $A \in M$ $B \in M$ $f \cdot \text{dot} \in M$ $p \in M$

shows $(\lambda a \in A. \{b \in B. \exists q \in \mathbb{P}. q \preceq p \wedge (q \Vdash \cdot 0'1 \text{ is } 2 \cdot [f \cdot \text{dot}, a^v, b^v])\}) \in M$

<proof>

lemma *ccc_fun_approximation_lemma*:

notes *le_trans*[*trans*]

assumes $\text{ccc}^M(\mathbb{P}, \text{leq})$ $A \in M$ $B \in M$ $f \in M[G]$ $f : A \rightarrow B$

shows $\exists F \in M. F : A \rightarrow \text{Pow}^M(B) \wedge (\forall a \in A. f'a \in F'a \wedge |F'a|^M \leq \omega)$

<proof>

end — *G_generic1_lemmas* bundle

end — *G_generic3_AC*

end

29 Model of the negation of the Continuum Hypothesis

theory *Not_CH*

imports

Cardinal_Preservation

begin

We are taking advantage that the poset of finite functions is absolute, and thus we work with the unrelativized *Fn*. But it would have been more appropriate to do the following using the relative *Fn_rel*. As it turns out, the present theory was developed prior to having *Fn* relativized!

We also note that $Fn(\omega, \kappa \times \omega, \mathcal{Q})$ is separative, i.e. each $X \in Fn(\omega, \kappa \times \omega, \mathcal{Q})$ has two incompatible extensions; therefore we may recover part of our previous theorem *extensions_of_ctms_ZF*. But that result also included the possibility of not having *AC* in the ground model, which would not be sensible in a context where the cardinality of the continuum is under discussion. It is also the case that *extensions_of_ctms_ZF* was historically our first formalized result (with a different proof) that showed the forcing machinery had all of its elements in place.

abbreviation

Add_subs :: $i \Rightarrow i$ **where**

$Add_subs(\kappa) \equiv Fn(\omega, \kappa \times \omega, \mathcal{Q})$

abbreviation

Add_le :: $i \Rightarrow i$ **where**

$Add_le(\kappa) \equiv Fnle(\omega, \kappa \times \omega, \mathcal{Q})$

lemma (**in** *M_aleph*) *Aleph_rel2_closed*[*intro,simp*]: $M(\aleph_2^M)$
<proof>

locale *M_master* = *M_cohen* + *M_library* +

assumes

UN_lepoll_assumptions:

$M(A) \Longrightarrow M(b) \Longrightarrow M(f) \Longrightarrow M(A') \Longrightarrow separation(M, \lambda y. \exists x \in A'. y = \langle x, \mu \ i. x \in if_range_F_else_F((\cdot)(A), b, f, i))$)

29.1 Non-absolute concepts between extensions

sublocale *M_master* \subseteq *M_Pi_replacement*

<proof>

locale $M_master_sub = M_master + N:M_aleph\ N$ **for** $N +$
assumes
 $M_imp_N: M(x) \implies N(x)$ **and**
 $Ord_iff: Ord(x) \implies M(x) \longleftrightarrow N(x)$

sublocale $M_master_sub \subseteq M_N_Perm$
<proof>

context M_master_sub
begin

lemma $cardinal_rel_le_cardinal_rel: M(X) \implies |X|^N \leq |X|^M$
<proof>

lemma $Aleph_rel_sub_closed: Ord(\alpha) \implies M(\alpha) \implies N(\aleph_\alpha^M)$
<proof>

lemma $Card_rel_imp_Card_rel: Card^N(\kappa) \implies M(\kappa) \implies Card^M(\kappa)$
<proof>

lemma $csucc_rel_le_csucc_rel:$
 assumes $Ord(\kappa) M(\kappa)$
 shows $(\kappa^+)^M \leq (\kappa^+)^N$
<proof>

lemma $Aleph_rel_le_Aleph_rel: Ord(\alpha) \implies M(\alpha) \implies \aleph_\alpha^M \leq \aleph_\alpha^N$
<proof>

end — M_master_sub

lemmas (**in** M_ZF2_trans) $sep_instances =$
 $separation_ifrangeF_body\ separation_ifrangeF_body2\ separation_ifrangeF_body3$
 $separation_ifrangeF_body4\ separation_ifrangeF_body5\ separation_ifrangeF_body6$
 $separation_ifrangeF_body7\ separation_cardinal_rel_lesspoll_rel$
 $separation_is_dcwit_body\ separation_cdltgamma\ separation_cdeggamma$

lemmas (**in** M_ZF2_trans) $repl_instances = lam_replacement_inj_rel$

sublocale $M_ZFC2_ground_notCH_trans \subseteq M_master\ \#\#M$
<proof>

sublocale $M_ZFC2_trans \subseteq M_Pi_replacement\ \#\#M$
<proof>

29.2 Cohen forcing is ccc

context M_ctm2_AC

```

begin

lemma ccc_Add_subs_Aleph_2: cccM(Add_subs( $\aleph_2^M$ ), Add_le( $\aleph_2^M$ ))
⟨proof⟩

end — M_ctm2_AC

sublocale G_generic3_AC ⊆ M_master_sub ##M ##(M[G])
⟨proof⟩

lemma (in M_trans) mem_F_bound4:
  fixes F A
  defines F ≡ (·)
  shows x ∈ F(A, c) ⇒ c ∈ (range(f) ∪ domain(A))
  ⟨proof⟩

lemma (in M_trans) mem_F_bound5:
  fixes F A
  defines F ≡ λ x. A 'x
  shows x ∈ F(A, c) ⇒ c ∈ (range(f) ∪ domain(A))
  ⟨proof⟩

sublocale M_ctm2_AC ⊆ M_replacement_lepoll ##M (·)
⟨proof⟩

context G_generic3_AC begin

context
  includes G_generic1_lemmas
begin

lemma G_in_MG: G ∈ M[G]
⟨proof⟩

lemma ccc_preserves_Aleph_succ:
  assumes cccM( $\mathbb{P}$ , leq) Ord(z) z ∈ M
  shows CardM[G]( $\aleph_{succ(z)}^M$ )
  ⟨proof⟩

end — bundle G_generic1_lemmas

end — G_generic3_AC

context M_ctm1
begin

abbreviation
  Add :: i where
  Add ≡ Fn( $\omega$ ,  $\aleph_2^M \times \omega$ , 2)

```

```

end — M_ctm1

locale add_generic3 = G_generic3_AC Fn( $\omega$ ,  $\aleph_2^{\#\#M} \times \omega$ , 2) Fnle( $\omega$ ,  $\aleph_2^{\#\#M} \times \omega$ , 2) 0

sublocale add_generic3  $\subseteq$  cohen_data  $\omega$   $\aleph_2^M \times \omega$  2 <proof>

context add_generic3
begin

notation Leq (infixl  $\langle \preceq \rangle$  50)
notation Incompatible (infixl  $\langle \perp \rangle$  50)

lemma Add_subs_preserves_Aleph_succ: Ord(z)  $\implies z \in M \implies \text{Card}^{M[G]}(\aleph_{\text{succ}(z)}^M)$ 
<proof>

lemma Aleph_rel_nats_MG_eq_Aleph_rel_nats_M:
includes G_generic1_lemmas
assumes  $z \in \omega$ 
shows  $\aleph_z^{M[G]} = \aleph_z^M$ 
<proof>

abbreviation
f_G :: i (f_G) where
f_G  $\equiv \bigcup G$ 

abbreviation
dom_dense :: i  $\Rightarrow$  i where
dom_dense(x)  $\equiv \{p \in \text{Add} . x \in \text{domain}(p)\}$ 

declare (in M_ctm2_AC) Fnat_closed[simplified setclass_iff, simp, intro]
declare (in M_ctm2_AC) Fnle_nat_closed[simp del, rule del,
simplified setclass_iff, simp, intro]
declare (in M_ctm2_AC) cexp_rel_closed[simplified setclass_iff, simp, intro]
declare (in G_generic3_AC) ext.cexp_rel_closed[simplified setclass_iff, simp,
intro]

lemma dom_dense_closed[intro, simp]:  $x \in \aleph_2^M \times \omega \implies \text{dom\_dense}(x) \in M$ 
<proof>

lemma domain_f_G: assumes  $x \in \aleph_2^M$   $y \in \omega$ 
shows  $\langle x, y \rangle \in \text{domain}(f_G)$ 
<proof>

lemma f_G_funtype:
includes G_generic1_lemmas
shows  $f_G : \aleph_2^M \times \omega \rightarrow 2$ 
<proof>

```

lemma *inj_dense_closed*[*intro,simp*]:
 $w \in \aleph_2^M \implies x \in \aleph_2^M \implies \text{inj_dense}(\aleph_2^M, 2, w, x) \in M$
 ⟨*proof*⟩

lemma *Aleph_rel2_new_reals*:
assumes $w \in \aleph_2^M$ $x \in \aleph_2^M$ $w \neq x$
shows $(\lambda n \in \omega. f_G \text{ ` } \langle w, n \rangle) \neq (\lambda n \in \omega. f_G \text{ ` } \langle x, n \rangle)$
 ⟨*proof*⟩

definition
 $h_G :: i \text{ ` } \langle h_G \rangle$ **where**
 $h_G \equiv \lambda \alpha \in \aleph_2^M. \lambda n \in \omega. f_G \text{ ` } \langle \alpha, n \rangle$

lemma *h_G_in_MG*[*simp*]:
includes *G_generic1_lemmas*
shows $h_G \in M[G]$
 ⟨*proof*⟩

lemma *h_G_inj_Aleph_rel2_reals*: $h_G \in \text{inj}^{M[G]}(\aleph_2^M, \omega \rightarrow^{M[G]} 2)$
 ⟨*proof*⟩

lemma *Aleph2_extension_le_continuum_rel*:
includes *G_generic1_lemmas*
shows $\aleph_2^{M[G]} \leq 2^{\uparrow \aleph_0^{M[G]}, M[G]}$
 ⟨*proof*⟩

lemma *Aleph_rel_lt_continuum_rel*: $\aleph_1^{M[G]} < 2^{\uparrow \aleph_0^{M[G]}, M[G]}$
 ⟨*proof*⟩

corollary *not_CH*: $\aleph_1^{M[G]} \neq 2^{\uparrow \aleph_0^{M[G]}, M[G]}$
 ⟨*proof*⟩

end — *add_generic3*

29.3 Models of fragments of $ZFC + \neg CH$

definition
 $ContHyp :: o$ **where**
 $ContHyp \equiv \aleph_1 = 2^{\uparrow \aleph_0}$

⟨*ML*⟩
notation *ContHyp_rel* ($\langle CH \neg \rangle$)
 ⟨*ML*⟩

context *M_ZF_library*
begin

⟨*ML*⟩

$\langle proof \rangle$
end — *M_ZF_library*
 $\langle ML \rangle$
notation *is_ContHyp_fm* ($\langle \cdot CH \cdot \rangle$)
theorem *ctm_of_not_CH*:
assumes
 $M \approx \omega \text{ Transset}(M) \ M \models ZC \cup \{ \cdot Replacement(p) \cdot \ . \ p \in overhead_notCH \}$
 $\Phi \subseteq formula \ M \models \{ \cdot Replacement(ground_repl_fm(\varphi)) \cdot \ . \ \varphi \in \Phi \}$
shows
 $\exists N.$
 $M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZC \cup \{ \cdot \neg \cdot CH \cdot \} \cup \{ \cdot Replacement(\varphi) \cdot \ . \ \varphi \in \Phi \} \wedge$
 $(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$
 $\langle proof \rangle$
lemma *ZF_replacement_overhead_sub_ZFC*: $\{ \cdot Replacement(p) \cdot \ . \ p \in overhead \} \subseteq ZFC$
 $\langle proof \rangle$
lemma *ZF_replacement_overhead_notCH_sub_ZFC*: $\{ \cdot Replacement(p) \cdot \ . \ p \in overhead_notCH \} \subseteq ZFC$
 $\langle proof \rangle$
lemma *ZF_replacement_overhead_CH_sub_ZFC*: $\{ \cdot Replacement(p) \cdot \ . \ p \in overhead_CH \} \subseteq ZFC$
 $\langle proof \rangle$
corollary *ctm_ZFC_imp_ctm_not_CH*:
assumes
 $M \approx \omega \text{ Transset}(M) \ M \models ZFC$
shows
 $\exists N.$
 $M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZFC \cup \{ \cdot \neg \cdot CH \cdot \} \wedge$
 $(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$
 $\langle proof \rangle$
end

30 Preservation results for κ -closed forcing notions

theory *Kappa_Closed_Notions*
imports
Not_CH
begin

definition

$lerel :: i \Rightarrow i$ **where**
 $lerel(\alpha) \equiv Memrel(\alpha) \cup id(\alpha)$

lemma $lerelI[intro!]$: $x \leq y \Rightarrow y \in \alpha \Rightarrow Ord(\alpha) \Rightarrow \langle x, y \rangle \in lerel(\alpha)$
 $\langle proof \rangle$

lemma $lerelD[dest]$: $\langle x, y \rangle \in lerel(\alpha) \Rightarrow Ord(\alpha) \Rightarrow x \leq y$
 $\langle proof \rangle$

definition

$mono_seqspace :: [i, i, i] \Rightarrow i (\langle _ \rangle \langle _ \rangle \rightarrow '(_, _)' [61] 60)$ **where**
 $\alpha \langle _ \rangle (P, leq) \equiv mono_map(\alpha, Memrel(\alpha), P, leq)$

$\langle ML \rangle$

context $M_ZF_library$

begin

$\langle ML \rangle$

$\langle proof \rangle$

end — $M_ZF_library$

abbreviation

$mono_seqspace_r (\langle _ \rangle \langle _ \rangle \rightarrow '(_, _)' [61] 60)$ **where**
 $\alpha \langle _ \rangle^M (P, leq) \equiv mono_seqspace_rel(M, \alpha, P, leq)$

abbreviation

$mono_seqspace_r_set (\langle _ \rangle \langle _ \rangle \rightarrow '(_, _)' [61] 60)$ **where**
 $\alpha \langle _ \rangle^M (P, leq) \equiv mono_seqspace_rel(\#\#M, \alpha, P, leq)$

lemma $mono_seqspaceI[intro!]$:

includes $mono_map_rules$

assumes $f: A \rightarrow P \wedge x y. x \in A \Rightarrow y \in A \Rightarrow x < y \Rightarrow \langle f'x, f'y \rangle \in leq Ord(A)$

shows $f: A \langle _ \rangle (P, leq)$

$\langle proof \rangle$

lemma (**in** $M_ZF_library$) $mono_seqspace_rel_char$:

assumes $M(A) M(P) M(leq)$

shows $A \langle _ \rangle^M (P, leq) = \{f \in A \langle _ \rangle (P, leq). M(f)\}$

$\langle proof \rangle$

lemma (**in** $M_ZF_library$) $mono_seqspace_relI[intro!]$:

assumes $f: A \rightarrow^M P \wedge x y. x \in A \Rightarrow y \in A \Rightarrow x < y \Rightarrow \langle f'x, f'y \rangle \in leq$

$Ord(A) M(A) M(P) M(leq)$

shows $f: A \langle _ \rangle^M (P, leq)$

$\langle proof \rangle$

lemma *mono_seqspace_is_fun*[*dest*]:
includes *mono_map_rules*
shows $j: A \lt \rightarrow (P, leq) \implies j: A \rightarrow P$
 $\langle proof \rangle$

lemma *mono_map_lt_le_is_mono*[*dest*]:
includes *mono_map_rules*
assumes $j: A \lt \rightarrow (P, leq) \ a \in A \ c \in A \ a \leq c \ Ord(A) \ refl(P, leq)$
shows $\langle j'a, j'c \rangle \in leq$
 $\langle proof \rangle$

lemma (**in** *M_ZF_library*) *mem_mono_seqspace_abs*[*absolut*]:
assumes $M(f) \ M(A) \ M(P) \ M(leq)$
shows $f: A \lt \rightarrow^M (P, leq) \longleftrightarrow f: A \lt \rightarrow (P, leq)$
 $\langle proof \rangle$

definition
 $mono_map_lt_le :: [i, i] \Rightarrow i \text{ (infixr } \lt \rightarrow \leq \text{) } 60$ **where**
 $\alpha \lt \rightarrow \leq \beta \equiv \alpha \lt \rightarrow (\beta, lerel(\beta))$

lemma *mono_map_lt_leI*[*intro!*]:
includes *mono_map_rules*
assumes $f: A \rightarrow B \ \bigwedge x \ y. \ x \in A \implies y \in A \implies x < y \implies f'x \leq f'y \ Ord(A) \ Ord(B)$
shows $f: A \lt \rightarrow \leq B$
 $\langle proof \rangle$

definition
 $kappa_closed :: [i, i, i] \Rightarrow o \ (\lt_closed'(_, _))$ **where**
 $\kappa_closed(P, leq) \equiv \forall \delta. \ \delta < \kappa \longrightarrow (\forall f \in \delta \ \lt \rightarrow (P, converse(leq))). \ \exists q \in P. \ \forall \alpha \in \delta. \ \langle q, f'\alpha \rangle \in leq$

$\langle ML \rangle$

abbreviation
 $kappa_closed_r \ (\lt_closed'(_, _))$ [61] 60 **where**
 $\kappa_closed^M(P, leq) \equiv kappa_closed_rel(M, \kappa, P, leq)$

abbreviation
 $kappa_closed_r_set \ (\lt_closed'(_, _))$ [61] 60 **where**
 $\kappa_closed^M(P, leq) \equiv kappa_closed_rel(\#\#\#M, \kappa, P, leq)$

lemma (**in** *forcing_data3*) *forcing_a_value*:
assumes $p \Vdash \cdot 0: 1 \rightarrow 2. \ [f_dot, A^v, B^v] \ a \in A$
 $q \preceq p \ q \in \mathbb{P} \ p \in \mathbb{P} \ f_dot \in M \ A \in M \ B \in M$
shows $\exists d \in \mathbb{P}. \ \exists b \in B. \ d \preceq q \wedge d \Vdash \cdot 0' 1 \text{ is } 2. \ [f_dot, a^v, b^v]$

$\langle proof \rangle$
include *G_generic1_lemmas*
 $\langle proof \rangle$

locale $M_master_CH = M_master + M_library_DC$

sublocale $M_ZFC2_ground_CH_trans \subseteq M_master_CH \#\#M$
 $\langle proof \rangle$

context $G_generic3_AC_CH$ **begin**

context
includes $G_generic1_lemmas$
begin

lemma $separation_check_snd_aux$:
assumes $f_dot \in M \ \tau \in M \ \chi \in formula \ arity(\chi) \leq 7$
shows $separation(\#\#M, \lambda r. M, [fst(r), \mathbb{P}, leq, \mathbf{1}, f_dot, \tau, snd(r)^v] \models \chi)$
 $\langle proof \rangle$

lemma $separation_check_fst_snd_aux$:
assumes $f_dot \in M \ r \in M \ \chi \in formula \ arity(\chi) \leq 7$
shows $separation(\#\#M, \lambda p. M, [r, \mathbb{P}, leq, \mathbf{1}, f_dot, fst(p)^v, snd(p)^v] \models \chi)$
 $\langle proof \rangle$

lemma $separation_leq_and_forces_apply_aux$:
assumes $f_dot \in M \ B \in M$
shows $\forall n \in M. separation(\#\#M, \lambda x. snd(x) \preceq fst(x) \wedge$
 $(\exists b \in B. M, [snd(x), \mathbb{P}, leq, \mathbf{1}, f_dot, (\bigcup (n))^v, b^v] \models forces(\cdot 0'1 \text{ is } 2.)))$
 $\langle proof \rangle$

lemma $separation_leq_and_forces_apply_aux'$:
assumes $f_dot \in M \ p \in M \ B \in M$
shows $separation$
 $(\#\#M, \lambda p. snd(snd(p)) \preceq fst(snd(p)) \wedge$
 $(\exists b \in B. M, [snd(snd(p)), \mathbb{P}, leq, \mathbf{1}, f_dot, (\bigcup fst(p))^v, b^v] \models forces(\cdot 0'1 \text{ is } 2.)))$
 $\langle proof \rangle$

lemma $separation_closed_leq_and_forces_eq_check_aux$:
assumes $A \in M \ r \in G \ \tau \in M$
shows $(\#\#M)(\{q \in \mathbb{P}. \exists h \in A. q \preceq r \wedge q \Vdash \cdot 0 = 1. [\tau, h^v]\})$
 $\langle proof \rangle$

lemma $separation_closed_forces_apply_aux$:
assumes $B \in M \ f_dot \in M \ r \in M$
shows $(\#\#M)(\{(n, b) \in \omega \times B. r \Vdash \cdot 0'1 \text{ is } 2. [f_dot, n^v, b^v]\})$
 $\langle proof \rangle$

lemma $kunen_IV_6_9_function_space_rel_eq$:
assumes $\bigwedge p \ \tau. p \Vdash \cdot 0:1 \rightarrow 2. [\tau, A^v, B^v] \implies p \in \mathbb{P} \implies \tau \in M \implies$
 $\exists q \in \mathbb{P}. \exists h \in A \rightarrow^M B. q \preceq p \wedge q \Vdash \cdot 0 = 1. [\tau, h^v] \ A \in M \ B \in M$
shows
 $A \rightarrow^M B = A \rightarrow^{M[G]} B$
 $\langle proof \rangle$

30.1 $(\omega + 1)$ -Closed notions preserve countable sequences

lemma *succ_omega_closed_imp_no_new_nat_sequences*:
 assumes *succ(ω)-closed^M(\mathbb{P}, leq)* *f : $\omega \rightarrow B$ $f \in M[G]$ $B \in M$*
 shows *f $\in M$*
<proof>

declare *mono_seqspace_rel_closed*[*rule del*]
 — Mysteriously breaks the end of the next proof

lemma *succ_omega_closed_imp_no_new_reals*:
 assumes *succ(ω)-closed^M(\mathbb{P}, leq)*
 shows *$\omega \rightarrow^M 2 = \omega \rightarrow^{M[G]} 2$*
<proof>

lemma *succ_omega_closed_imp_Aleph_1_preserved*:
 assumes *succ(ω)-closed^M(\mathbb{P}, leq)*
 shows *$\aleph_1^M = \aleph_1^{M[G]}$*
<proof>

end — bundle *G_generic1_lemmas*

end — *G_generic3_AC*

end

31 Forcing extension satisfying the Continuum Hypothesis

theory *CH*
 imports
 Kappa_Closed_Notions
 Cohen_Posets_Relative
begin

context *M_ctm2_AC*
begin

declare *F_n_rel_closed*[*simp del, rule del, simplified setclass_iff, simp, intro*]
declare *F_{nle}_rel_closed*[*simp del, rule del, simplified setclass_iff, simp, intro*]

abbreviation
 Coll :: i where
 Coll $\equiv F_n^M(\aleph_1^M, \aleph_1^M, \omega \rightarrow^M 2)$

abbreviation
 Colleq :: i where
 Colleq $\equiv F_{nle}^M(\aleph_1^M, \aleph_1^M, \omega \rightarrow^M 2)$

lemma *Coll_in_M*[*intro,simp*]: $Coll \in M$ *<proof>*

lemma *Colleq_refl* : $refl(Coll, Colleq)$
<proof>

31.1 Collapse forcing is sufficiently closed

lemma *succ_omega_closed_Coll*: $succ(\omega)$ - $closed^M(Coll, Colleq)$
<proof>

end — *M_ctm2_AC*

locale *collapse_CH* = *G_generic3_AC_CH* $Fn^M(\aleph_1^{##M}, \aleph_1^M, \omega \rightarrow^M 2)$ $Fnle^M(\aleph_1^{##M}, \aleph_1^M, \omega \rightarrow^M 2)$ 0

sublocale *collapse_CH* \subseteq *forcing_notion* *Coll* *Colleq* 0
<proof>

context *collapse_CH*
begin

notation *Leq* (**infixl** $\langle \preceq \rangle$ 50)

notation *Incompatible* (**infixl** $\langle \perp \rangle$ 50)

abbreviation

$f_G :: i \langle f_G \rangle$ **where**
 $f_G \equiv \bigcup G$

lemma *f_G_in_MG*[*simp*]:
shows $f_G \in M[G]$
<proof>

abbreviation

$dom_dense :: i \Rightarrow i$ **where**
 $dom_dense(x) \equiv \{ p \in Coll . x \in domain(p) \}$

lemma *dom_dense_closed*[*intro,simp*]: $x \in M \implies dom_dense(x) \in M$
<proof>

lemma *domain_f_G*: **assumes** $x \in \aleph_1^M$
shows $x \in domain(f_G)$
<proof>

lemma *Un_filter_is_function*:
assumes *filter*(*G*)
shows *function*($\bigcup G$)
<proof>

lemma *f_G_funtype*:
shows $f_G : \aleph_1^M \rightarrow \omega \rightarrow^{M[G]} \mathcal{P}$
 $\langle \text{proof} \rangle$

abbreviation

surj_dense :: $i \Rightarrow i$ **where**
surj_dense(x) $\equiv \{ p \in \text{Coll} . x \in \text{range}(p) \}$

lemma *surj_dense_closed*[*intro,simp*]:
 $x \in \omega \rightarrow^M \mathcal{P} \implies \text{surj_dense}(x) \in M$
 $\langle \text{proof} \rangle$

lemma *reals_sub_image_f_G*:
assumes $x \in \omega \rightarrow^M \mathcal{P}$
shows $\exists \alpha \in \aleph_1^M . f_G \upharpoonright \alpha = x$
 $\langle \text{proof} \rangle$

lemma *f_G_surj_Aleph_rel1_reals*: $f_G \in \text{surj}^{M[G]}(\aleph_1^M, \omega \rightarrow^{M[G]} \mathcal{P})$
 $\langle \text{proof} \rangle$

lemma *continuum_rel_le_Aleph1_extension*:
includes *G_generic1_lemmas*
shows $\mathcal{P}^{\uparrow \aleph_0^{M[G]}, M[G]} \leq \aleph_1^{M[G]}$
 $\langle \text{proof} \rangle$

theorem *CH*: $\aleph_1^{M[G]} = \mathcal{P}^{\uparrow \aleph_0^{M[G]}, M[G]}$
 $\langle \text{proof} \rangle$

end — *collapse_CH*

31.2 Models of fragments of ZFC + CH

theorem *ctm_of_CH*:

assumes

$M \approx \omega \text{ Transset}(M)$

$M \models \text{ZC} \cup \{ \cdot \text{Replacement}(p) . p \in \text{overhead_CH} \}$

$\Phi \subseteq \text{formula } M \models \{ \cdot \text{Replacement}(\text{ground_repl_fm}(\varphi)) . \varphi \in \Phi \}$

shows

$\exists N .$

$M \subseteq N \wedge N \approx \omega \wedge \text{Transset}(N) \wedge N \models \text{ZC} \cup \{ \cdot \text{CH} \} \cup \{ \cdot \text{Replacement}(\varphi) .$

$\varphi \in \Phi \}$ \wedge

$(\forall \alpha . \text{Ord}(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$

$\langle \text{proof} \rangle$

corollary *ctm_ZFC_imp_ctm_CH*:

assumes

$M \approx \omega \text{ Transset}(M) \ M \models \text{ZFC}$

shows

$\exists N .$

$$M \subseteq N \wedge N \approx \omega \wedge \text{Transset}(N) \wedge N \models \text{ZFC} \cup \{\cdot\text{CH}\cdot\} \wedge$$

$$(\forall \alpha. \text{Ord}(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$$
 <proof>

end

32 From M to \mathcal{V}

theory *Absolute_Versions*

imports

CH

ZF.Cardinal_AC

begin

hide_const (open) *Order.pred*

32.1 Locales of a class M hold in \mathcal{V}

interpretation $V: M_trivial \mathcal{V}$

<proof>

lemmas *bad_simps = V.nonempty V.Forall_in_M_iff V.Inl_in_M_iff V.Inr_in_M_iff*
V.succ_in_M_iff V.singleton_in_M_iff V.Equal_in_M_iff V.Member_in_M_iff
V.Nand_in_M_iff
V.Cons_in_M_iff V.pair_in_M_iff V.upair_in_M_iff

lemmas *bad_M_trivial_simps[simp del] = V.Forall_in_M_iff V.Equal_in_M_iff*
V.nonempty

lemmas *bad_M_trivial_rules[rule del] = V.pair_in_MI V.singleton_in_MI*
V.pair_in_MD V.nat_into_M
V.depth_closed V.length_closed V.nat_case_closed V.separation_closed
V.Un_closed V.strong_replacement_closed V.nonempty

interpretation $V: M_basic \mathcal{V}$

<proof>

interpretation $V: M_eclose \mathcal{V}$

<proof>

lemmas *bad_M_basic_rules[simp del, rule del] =*
V.cartprod_closed V.finite_funspace_closed V.converse_closed
V.list_case'_closed V.pred_closed

interpretation $V: M_cardinal_arith \mathcal{V}$

<proof>

lemmas *bad_M_cardinals_rules[simp del, rule del] =*
V.iterates_closed V.M_nat V.trancl_closed V.rvimage_closed

interpretation $V:M_cardinal_arith_jump \mathcal{V}$
<proof>

lemma $choice_ax_Universe: choice_ax(\mathcal{V})$
<proof>

interpretation $V:M_master \mathcal{V}$
<proof>

named_theorems V_simps

— To work systematically, ASCII versions of ”_absolute” theorems as those below are preferable.

lemma $eqpoll_rel_absolute[V_simps]: x \approx^{\mathcal{V}} y \longleftrightarrow x \approx y$
<proof>

lemma $cardinal_rel_absolute[V_simps]: |x|^{\mathcal{V}} = |x|$
<proof>

lemma $Card_rel_absolute[V_simps]: Card^{\mathcal{V}}(a) \longleftrightarrow Card(a)$
<proof>

lemma $csucc_rel_absolute[V_simps]: (a^+)^{\mathcal{V}} = a^+$
<proof>

lemma $function_space_rel_absolute[V_simps]: x \rightarrow^{\mathcal{V}} y = x \rightarrow y$
<proof>

lemma $cexp_rel_absolute[V_simps]: x^{\uparrow y, \mathcal{V}} = x^{\uparrow y}$
<proof>

lemma $HAleph_rel_absolute[V_simps]: HAleph_rel(\mathcal{V}, a, b) = HAleph(a, b)$
<proof>

lemma $Aleph_rel_absolute[V_simps]: Ord(x) \implies \aleph_x^{\mathcal{V}} = \aleph_x$
<proof>

Example of absolute lemmas obtained from the relative versions. Note the *only* declarations

lemma $Ord_cardinal_idem': Ord(A) \implies ||A|| = |A|$
<proof>

lemma $Aleph_succ': Ord(\alpha) \implies \aleph_{succ(\alpha)} = \aleph_{\alpha}^+$
<proof>

These two results are new, first obtained in relative form (not ported).

lemma $csucc_cardinal:$
assumes $Ord(\kappa)$ **shows** $|\kappa|^+ = \kappa^+$

<proof>

lemma *csucc_le_mono*:
 assumes $\kappa \leq \nu$ **shows** $\kappa^+ \leq \nu^+$
 <proof>

Example of transferring results from a transitive model to \mathcal{V}

lemma (in *M_Perm*) *eqpoll_rel_transfer_absolute*:
 assumes $M(A)$ $M(B)$ $A \approx^M B$
 shows $A \approx B$
 <proof>

The “relationalized” *CH* with respect to \mathcal{V} corresponds to the real *CH*.

lemma *is_ContHyp_iff_CH*: $is_ContHyp(\mathcal{V}) \longleftrightarrow ContHyp$
 <proof>

end

33 Main definitions of the development

theory *Definitions_Main*
 imports
 Absolute_Versions
begin

This theory gathers the main definitions of the *Transitive_Models* session and the present one.

It might be considered as the bare minimum reading requisite to trust that our development indeed formalizes the theory of forcing. This should be mathematically clear since this is the only known method for obtaining proper extensions of ctms while preserving the ordinals.

The main theorem of this session and all of its relevant definitions appear in Section 33.4. The reader trusting all the libraries on which our development is based, might jump directly to Section 33.3, which treats relative cardinal arithmetic as implemented in *Transitive_Models*. But in case one wants to dive deeper, the following sections treat some basic concepts of the ZF logic (Section 33.1) and in the ZF-Constructible library (Section 33.2) on which our definitions are built.

declare *[[show_question_marks=false]]*

33.1 ZF

For the basic logic ZF we restrict ourselves to just a few concepts.

thm *bij_def[unfolded inj_def surj_def]*

$$\begin{aligned} \text{bij}(A, B) &\equiv \\ &\{f \in A \rightarrow B . \forall w \in A. \forall x \in A. f \text{ ` } w = f \text{ ` } x \longrightarrow w = x\} \cap \\ &\{f \in A \rightarrow B . \forall y \in B. \exists x \in A. f \text{ ` } x = y\} \end{aligned}$$

thm *eqpoll_def*

$$A \approx B \equiv \exists f. f \in \text{bij}(A, B)$$

thm *Transset_def*

$$\text{Transset}(i) \equiv \forall x \in i. x \subseteq i$$

thm *Ord_def*

$$\text{Ord}(i) \equiv \text{Transset}(i) \wedge (\forall x \in i. \text{Transset}(x))$$

thm *lt_def le_iff*

$$\begin{aligned} i < j &\equiv i \in j \wedge \text{Ord}(j) \\ i \leq j &\longleftrightarrow i < j \vee i = j \wedge \text{Ord}(j) \end{aligned}$$

With the concepts of empty set and successor in place,

lemma *empty_def'*: $\forall x. x \notin 0$ *<proof>*

lemma *succ_def'*: $\text{succ}(i) = i \cup \{i\}$ *<proof>*

we can define the set of natural numbers ω . In the sources, it is defined as a fixpoint, but here we just write its characterization as the first limit ordinal.

thm *Limit_nat[unfolded Limit_def] nat_le_Limit[unfolded Limit_def]*

$$\begin{aligned} \text{Ord}(\omega) \wedge 0 < \omega \wedge (\forall y. y < \omega \longrightarrow \text{succ}(y) < \omega) \\ \text{Ord}(i) \wedge 0 < i \wedge (\forall y. y < i \longrightarrow \text{succ}(y) < i) \implies \omega \leq i \end{aligned}$$

Then, addition and predecessor on ω are inductively characterized as follows:

thm *add_0_right add_succ_right pred_0 pred_succ_eq*

$$\begin{aligned} m +_{\omega} \text{succ}(n) &= \text{succ}(m +_{\omega} n) \\ m \in \omega \implies m +_{\omega} 0 &= m \\ \text{pred}(0) &= 0 \\ \text{pred}(\text{succ}(y)) &= y \end{aligned}$$

Lists on a set A can be characterized by being recursively generated from the empty list $[]$ and the operation $Cons$ that adds a new element to the left end; the induction theorem for them shows that the characterization is “complete”.

thm *Nil Cons list.induct*

$$\begin{aligned} & [] \in list(A) \\ & \llbracket a \in A; l \in list(A) \rrbracket \implies Cons(a, l) \in list(A) \\ & \llbracket x \in list(A); P([]); \bigwedge a l. \llbracket a \in A; l \in list(A); P(l) \rrbracket \implies P(Cons(a, l)) \rrbracket \\ & \implies P(x) \end{aligned}$$

Length, concatenation, and n th element of lists are recursively characterized as follows.

thm *length.simps app.simps nth_0 nth_Cons*

$$\begin{aligned} length([]) &= 0 \\ length(Cons(a, l)) &= succ(length(l)) \\ [] @ ys &= ys \\ Cons(a, l) @ ys &= Cons(a, l @ ys) \\ nth(0, Cons(a, l)) &= a \\ n \in \omega \implies nth(succ(n), Cons(a, l)) &= nth(n, l) \end{aligned}$$

We have the usual Haskell-like notation for iterated applications of $Cons$:

lemma *Cons_app*: $[a,b,c] = Cons(a, Cons(b, Cons(c, [])))$ *<proof>*

Relative quantifiers restrict the range of the bound variable to a class M of type $i \Rightarrow o$; that is, a truth-valued function with set arguments.

lemma $\forall x[M]. P(x) \equiv \forall x. M(x) \longrightarrow P(x)$
 $\exists x[M]. P(x) \equiv \exists x. M(x) \wedge P(x)$
<proof>

Finally, a set can be viewed (“cast”) as a class using the following function of type $i \Rightarrow i \Rightarrow o$.

thm *setclass_iff*

$$(\#\#A)(x) \longleftrightarrow x \in A$$

33.2 Relative concepts

A list of relative concepts (mostly from the ZF-Constructible library) follows next.

thm *big_union_def*

$$\mathit{big_union}(M, A, z) \equiv \forall x[M]. x \in z \longleftrightarrow (\exists y[M]. y \in A \wedge x \in y)$$

thm *upair_def*

$$\mathit{upair}(M, a, b, z) \equiv a \in z \wedge b \in z \wedge (\forall x[M]. x \in z \longrightarrow x = a \vee x = b)$$

thm *pair_def*

$$\begin{aligned} \mathit{pair}(M, a, b, z) &\equiv \\ \exists x[M]. \mathit{upair}(M, a, a, x) \wedge (\exists y[M]. \mathit{upair}(M, a, b, y) \wedge \mathit{upair}(M, x, y, z)) \end{aligned}$$

thm *successor_def[unfolded is_cons_def union_def]*

$$\begin{aligned} \mathit{successor}(M, a, z) &\equiv \\ \exists x[M]. \mathit{upair}(M, a, a, x) \wedge (\forall xa[M]. xa \in z \longleftrightarrow xa \in x \vee xa \in a) \end{aligned}$$

thm *empty_def*

$$\mathit{empty}(M, z) \equiv \forall x[M]. x \notin z$$

thm *transitive_set_def[unfolded subset_def]*

$$\mathit{transitive_set}(M, a) \equiv \forall x[M]. x \in a \longrightarrow (\forall xa[M]. xa \in x \longrightarrow xa \in a)$$

thm *ordinal_def*

$$\begin{aligned} \mathit{ordinal}(M, a) &\equiv \\ \mathit{transitive_set}(M, a) \wedge (\forall x[M]. x \in a \longrightarrow \mathit{transitive_set}(M, x)) \end{aligned}$$

thm *image_def*

$$\begin{aligned} \mathit{image}(M, r, A, z) &\equiv \\ \forall y[M]. y \in z \longleftrightarrow (\exists w[M]. w \in r \wedge (\exists x[M]. x \in A \wedge \mathit{pair}(M, x, y, w))) \end{aligned}$$

thm *fun_apply_def*

$is_apply(M, f, x, y) \equiv$
 $\exists xs[M].$
 $\quad \exists fxs[M]. \text{upair}(M, x, x, xs) \wedge \text{image}(M, f, xs, fxs) \wedge \text{big_union}(M, fxs, y)$

thm *is_function_def*

$is_function(M, r) \equiv$
 $\forall x[M].$
 $\quad \forall y[M].$
 $\quad \quad \forall y'[M].$
 $\quad \quad \quad \forall p[M].$
 $\quad \quad \quad \quad \forall p'[M].$
 $\quad \quad \quad \quad \quad \text{pair}(M, x, y, p) \longrightarrow$
 $\quad \quad \quad \quad \quad \text{pair}(M, x, y', p') \longrightarrow p \in r \longrightarrow p' \in r \longrightarrow y = y'$

thm *is_relation_def*

$is_relation(M, r) \equiv \forall z[M]. z \in r \longrightarrow (\exists x[M]. \exists y[M]. \text{pair}(M, x, y, z))$

thm *is_domain_def*

$is_domain(M, r, z) \equiv$
 $\forall x[M]. x \in z \longleftrightarrow (\exists w[M]. w \in r \wedge (\exists y[M]. \text{pair}(M, x, y, w)))$

thm *typed_function_def*

$typed_function(M, A, B, r) \equiv$
 $is_function(M, r) \wedge$
 $is_relation(M, r) \wedge$
 $is_domain(M, r, A) \wedge$
 $(\forall u[M]. u \in r \longrightarrow (\forall x[M]. \forall y[M]. \text{pair}(M, x, y, u) \longrightarrow y \in B))$

thm *is_function_space_def*[*unfolded is_funspace_def*]
function_space_rel_def *surjection_def*

$is_function_space(M, A, B, fs) \equiv$
 $M(fs) \wedge (\forall f[M]. f \in fs \longleftrightarrow typed_function(M, A, B, f))$
 $A \rightarrow^M B \equiv THE\ d.\ is_function_space(M, A, B, d)$
 $surjection(M, A, B, f) \equiv$
 $typed_function(M, A, B, f) \wedge$
 $(\forall y[M]. y \in B \longrightarrow (\exists x[M]. x \in A \wedge is_apply(M, f, x, y)))$

Relative version of the *ZFC* axioms

thm *extensionality_def*

$$\text{extensionality}(M) \equiv \forall x[M]. \forall y[M]. (\forall z[M]. z \in x \longleftrightarrow z \in y) \longrightarrow x = y$$

thm *foundation_ax_def*

$$\begin{aligned} \text{foundation_ax}(M) &\equiv \\ \forall x[M]. (\exists y[M]. y \in x) &\longrightarrow (\exists y[M]. y \in x \wedge \neg (\exists z[M]. z \in x \wedge z \in y)) \end{aligned}$$

thm *upair_ax_def*

$$\text{upair_ax}(M) \equiv \forall x[M]. \forall y[M]. \exists z[M]. \text{upair}(M, x, y, z)$$

thm *Union_ax_def*

$$\text{Union_ax}(M) \equiv \forall x[M]. \exists z[M]. \text{big_union}(M, x, z)$$

thm *power_ax_def*[*unfolded powerset_def subset_def*]

$$\text{power_ax}(M) \equiv \forall x[M]. \exists z[M]. \forall xa[M]. xa \in z \longleftrightarrow (\forall xb[M]. xb \in xa \longrightarrow xb \in x)$$

thm *infinity_ax_def*

$$\begin{aligned} \text{infinity_ax}(M) &\equiv \\ \exists I[M]. & \\ (\exists z[M]. \text{empty}(M, z) \wedge z \in I) \wedge & \\ (\forall y[M]. y \in I \longrightarrow (\exists sy[M]. \text{successor}(M, y, sy) \wedge sy \in I)) & \end{aligned}$$

thm *choice_ax_def*

$$\text{choice_ax}(M) \equiv \forall x[M]. \exists a[M]. \exists f[M]. \text{ordinal}(M, a) \wedge \text{surjection}(M, a, x, f)$$

thm *separation_def*

$$\text{separation}(M, P) \equiv \forall z[M]. \exists y[M]. \forall x[M]. x \in y \longleftrightarrow x \in z \wedge P(x)$$

thm *univalent_def*

$univalent(M, A, P) \equiv$
 $\forall x[M]. x \in A \longrightarrow (\forall y[M]. \forall z[M]. P(x, y) \wedge P(x, z) \longrightarrow y = z)$

thm *strong_replacement_def*

$strong_replacement(M, P) \equiv$
 $\forall A[M].$
 $univalent(M, A, P) \longrightarrow (\exists Y[M]. \forall b[M]. b \in Y \longleftrightarrow (\exists x[M]. x \in A \wedge P(x, b)))$

Internalized formulas

“Codes” for formulas (as sets) are constructed from natural numbers using *Member*, *Equal*, *Nand*, and *Forall*.

thm *Member Equal Nand Forall formula.induct*

$\llbracket x \in \omega; y \in \omega \rrbracket \Longrightarrow \cdot x \in y \cdot \in formula$
 $\llbracket x \in \omega; y \in \omega \rrbracket \Longrightarrow \cdot x = y \cdot \in formula$
 $\llbracket p \in formula; q \in formula \rrbracket \Longrightarrow \cdot \neg(p \wedge q) \cdot \in formula$
 $p \in formula \Longrightarrow (\cdot \forall p \cdot) \in formula$
 $\llbracket x \in formula; \wedge x y. \llbracket x \in \omega; y \in \omega \rrbracket \Longrightarrow P(\cdot x \in y \cdot);$
 $\wedge x y. \llbracket x \in \omega; y \in \omega \rrbracket \Longrightarrow P(\cdot x = y \cdot);$
 $\wedge p q. \llbracket p \in formula; P(p); q \in formula; P(q) \rrbracket \Longrightarrow P(\cdot \neg(p \wedge q) \cdot);$
 $\wedge p. \llbracket p \in formula; P(p) \rrbracket \Longrightarrow P((\cdot \forall p \cdot))$
 $\Longrightarrow P(x)$

Definitions for the other connectives and the internal existential quantifier are also provided. For instance, negation:

thm *Neg_def*

$\cdot \neg p \cdot \equiv \cdot \neg(p \wedge p) \cdot$

thm *arity.simps*

$arity(\cdot x \in y \cdot) = succ(x) \cup succ(y)$
 $arity(\cdot x = y \cdot) = succ(x) \cup succ(y)$
 $arity(\cdot \neg(p \wedge q) \cdot) = arity(p) \cup arity(q)$
 $arity((\cdot \forall p \cdot)) = pred(arity(p))$

We have the satisfaction relation between \in -models and first order formulas (given a “environment” list representing the assignment of free variables),

thm *mem_iff_sats equal_iff_sats sats_Nand_iff sats_Forall_iff*

$$\begin{aligned}
& \llbracket nth(i, env) = x; nth(j, env) = y; env \in list(A) \rrbracket \\
& \implies x \in y \longleftrightarrow A, env \models \cdot i \in j \cdot \\
& \llbracket nth(i, env) = x; nth(j, env) = y; env \in list(A) \rrbracket \\
& \implies x = y \longleftrightarrow A, env \models \cdot i = j \cdot \\
& env \in list(A) \implies (A, env \models \cdot \neg(p \wedge q) \cdot) \longleftrightarrow \neg((A, env \models p) \wedge (A, env \models q)) \\
& env \in list(A) \implies (A, env \models (\cdot \forall p \cdot)) \longleftrightarrow (\forall x \in A. A, Cons(x, env) \models p)
\end{aligned}$$

as well as the satisfaction of an arbitrary set of sentences.

thm *satT_def*

$$A \models \Phi \equiv \forall \varphi \in \Phi. A, [] \models \varphi$$

The internalized (viz. as elements of the set *formula*) version of the axioms follow next.

thm *ZF_union_iff_sats ZF_power_iff_sats ZF_pairing_iff_sats*
ZF_foundation_iff_sats ZF_extensionality_iff_sats
ZF_infinity_iff_sats sats_ZF_separation_fm_iff
sats_ZF_replacement_fm_iff ZF_choice_iff_sats

$$\begin{aligned}
& Union_ax(\#\#A) \longleftrightarrow A, [] \models \cdot Union\ Ax \cdot \\
& power_ax(\#\#A) \longleftrightarrow A, [] \models \cdot Powerset\ Ax \cdot \\
& upair_ax(\#\#A) \longleftrightarrow A, [] \models \cdot Pairing \cdot \\
& foundation_ax(\#\#A) \longleftrightarrow A, [] \models \cdot Foundation \cdot \\
& extensionality(\#\#A) \longleftrightarrow A, [] \models \cdot Extensionality \cdot \\
& infinity_ax(\#\#A) \longleftrightarrow A, [] \models \cdot Infinity \cdot \\
& \varphi \in formula \implies \\
& (M, [] \models \cdot Separation(\varphi) \cdot) \longleftrightarrow \\
& (\forall env \in list(M). \\
& \quad arity(\varphi) \leq 1 +_{\omega} length(env) \longrightarrow separation(\#\#M, \lambda x. M, [x] @ env \models \varphi)) \\
& \varphi \in formula \implies \\
& (M, [] \models \cdot Replacement(\varphi) \cdot) \longleftrightarrow (\forall env. replacement_assm(M, env, \varphi)) \\
& choice_ax(\#\#A) \longleftrightarrow A, [] \models \cdot AC \cdot
\end{aligned}$$

Above, we use the following:

thm *replacement_assm_def*

$$\begin{aligned}
& replacement_assm(M, env, \varphi) \equiv \\
& \varphi \in formula \longrightarrow \\
& env \in list(M) \longrightarrow \\
& arity(\varphi) \leq 2 +_{\omega} length(env) \longrightarrow \\
& strong_replacement(\#\#M, \lambda x y. M, [x, y] @ env \models \varphi)
\end{aligned}$$

Finally, the axiom sets are defined as follows.

thm *ZF_fin_def ZF_schemes_def Zermelo_fms_def ZC_def ZF_def ZFC_def*

$ZF_fin \equiv$
 $\{\cdot Extensionality \cdot, \cdot Foundation \cdot, \cdot Pairing \cdot, \cdot Union Ax \cdot, \cdot Infinity \cdot,$
 $\cdot Powerset Ax \cdot\}$
 $ZF_schemes \equiv$
 $\{\cdot Separation(p) \cdot \ . \ p \in formula\} \cup \{\cdot Replacement(p) \cdot \ . \ p \in formula\}$
 $\cdot Z \cdot \equiv ZF_fin \cup \{\cdot Separation(p) \cdot \ . \ p \in formula\}$
 $ZC \equiv \cdot Z \cdot \cup \{\cdot AC \cdot\}$
 $ZF \equiv ZF_schemes \cup ZF_fin$
 $ZFC \equiv ZF \cup \{\cdot AC \cdot\}$

33.3 Relativization of infinitary arithmetic

In order to state the defining property of the relative equipotence relation, we work under the assumptions of the locale $M_cardinals$. They comprise a finite set of instances of Separation and Replacement to prove closure properties of the transitive class M .

lemma (in $M_cardinals$) *eqpoll_def'*:
assumes $M(A) \ M(B)$ **shows** $A \approx^M B \longleftrightarrow (\exists f[M]. f \in bij(A,B))$
<proof>

Below, μ denotes the minimum operator on the ordinals.

lemma *cardinalities_defs*:
fixes $M::i \Rightarrow o$
shows
 $|A|^M \equiv \mu i. M(i) \wedge i \approx^M A$
 $Card^M(\alpha) \equiv \alpha = |\alpha|^M$
 $\kappa^{\uparrow \nu, M} \equiv |\nu \rightarrow^M \kappa|^M$
 $(\kappa^+)^M \equiv \mu x. M(x) \wedge Card^M(x) \wedge \kappa < x$
<proof>

context M_aleph
begin

Analogous to the previous Lemma *eqpoll_def'*, we are now under the assumptions of the locale M_aleph . The axiom instances included are sufficient to state and prove the defining properties of the relativized *Aleph* function (in particular, the required ability to perform transfinite recursions).

thm *Aleph_rel_zero Aleph_rel_succ Aleph_rel_limit*

$\aleph_0^M = \omega$
 $\llbracket Ord(\alpha); M(\alpha) \rrbracket \implies \aleph_{succ(\alpha)}^M = (\aleph_\alpha^{M+})^M$
 $\llbracket Limit(\alpha); M(\alpha) \rrbracket \implies \aleph_\alpha^M = (\bigcup_{j \in \alpha} \aleph_j^M)$

end — M_aleph

lemma *ContHyp_rel_def'*:
fixes $N::i\Rightarrow o$
shows
 $CH^N \equiv \aleph_1^N = 2^{\uparrow\aleph_0^{N,N}}$
<proof>

Under appropriate hypotheses (this time, from the locale *M_ZF_library*), CH^M is equivalent to its fully relational version *is_ContHyp*. As a sanity check, we see that if the transitive class is indeed \mathcal{V} , we recover the original *CH*.

thm *M_ZF_library.is_ContHyp_iff_is_ContHyp_iff_CH[unfolded ContHyp_def]*

$$\begin{aligned} M_ZF_library(M) &\Longrightarrow is_ContHyp(M) \longleftrightarrow CH^M \\ is_ContHyp(\mathcal{V}) &\longleftrightarrow \aleph_1 = 2^{\uparrow\aleph_0} \end{aligned}$$

In turn, the fully relational version evaluated on a nonempty transitive A is equivalent to the satisfaction of the first-order formula $\cdot CH \cdot$.

thm *is_ContHyp_iff_sats*

$$\llbracket env \in list(A); 0 \in A \rrbracket \Longrightarrow is_ContHyp(\#\#A) \longleftrightarrow A, env \models \cdot CH \cdot$$

33.4 Forcing

Our first milestone was to obtain a proper extension using forcing. Its original proof didn't required the previous developments involving the relativization of material on cardinal arithmetic. Now it is derived from a stronger result, namely *extensions_of_ctms* below.

thm *extensions_of_ctms_ZF*

$$\begin{aligned} \llbracket M \approx \omega; Transset(M); M \models ZF \rrbracket \\ \Longrightarrow \exists N. M \subseteq N \wedge \\ N \approx \omega \wedge \\ Transset(N) \wedge \\ N \models ZF \wedge \\ M \neq N \wedge (\forall \alpha. Ord(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N) \wedge ((M, [] \models \cdot AC \cdot) \longrightarrow \\ N \models ZFC) \end{aligned}$$

We can finally state our main results, namely, the existence of models for $ZFC + CH$ and $ZFC + \neg CH$ under the assumption of a ctm of *ZFC*.

thm *ctm_ZFC_imp_ctm_not_CH*

$$\begin{aligned} & \llbracket M \approx \omega; \text{Transset}(M); M \models \text{ZFC} \rrbracket \\ \implies & \exists N. M \subseteq N \wedge \\ & N \approx \omega \wedge \\ & \text{Transset}(N) \wedge N \models \text{ZFC} \cup \{\neg \text{CH}\} \wedge (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \\ & \alpha \in N) \end{aligned}$$

thm *ctm_ZFC_imp_ctm_CH*

$$\begin{aligned} & \llbracket M \approx \omega; \text{Transset}(M); M \models \text{ZFC} \rrbracket \\ \implies & \exists N. M \subseteq N \wedge \\ & N \approx \omega \wedge \\ & \text{Transset}(N) \wedge N \models \text{ZFC} \cup \{\text{CH}\} \wedge (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \\ & \in N) \end{aligned}$$

These results can be strengthened by enumerating six finite sets of replacement instances which are sufficient to develop forcing and for the construction of the aforementioned models: *instances1_fms* through *instances3_fms*, *instances_ground_fms*, and *instances_ground_notCH_fms*, which are then collected into the 31-element set *overhead_notCH*. For example, we have:

thm *instances1_fms_def*

$$\begin{aligned} \text{instances1_fms} & \equiv \\ & \{\text{eclose_closed_fm}, \text{eclose_abs_fm}, \text{wfrec_rank_fm}, \text{transrec_VFrom_fm}\} \end{aligned}$$

thm *overhead_def overhead_notCH_def*

$$\begin{aligned} \text{overhead} & \equiv \text{instances1_fms} \cup \text{instances_ground_fms} \\ \text{overhead_notCH} & \equiv \\ & \text{overhead} \cup \text{instances2_fms} \cup \text{instances3_fms} \cup \text{instances_ground_notCH_fms} \\ \text{overhead_CH} & \equiv \text{overhead_notCH} \cup \{\text{dc_abs_fm}\} \end{aligned}$$

One further instance is needed to force *CH*, with a total count of 32 instances:

thm *overhead_CH_def*

$$\text{overhead_CH} \equiv \text{overhead_notCH} \cup \{\text{dc_abs_fm}\}$$

thm *extensions_of_ctms*

$$\begin{aligned}
& \llbracket M \approx \omega; \text{Transset}(M); M \models \cdot Z \cdot \cup \{\cdot \text{Replacement}(p) \cdot \mid p \in \text{overhead}\}; \\
& \Phi \subseteq \text{formula}; M \models \{\cdot \text{Replacement}(\text{ground_repl_fm}(\varphi)) \cdot \mid \varphi \in \Phi\} \rrbracket \\
\implies & \exists N. M \subseteq N \wedge \\
& N \approx \omega \wedge \\
& \text{Transset}(N) \wedge \\
& M \neq N \wedge \\
& (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N) \wedge \\
& ((M, [] \models \cdot AC \cdot) \longrightarrow N, [] \models \cdot AC \cdot) \wedge \\
& N \models \cdot Z \cdot \cup \{\cdot \text{Replacement}(\varphi) \cdot \mid \varphi \in \Phi\}
\end{aligned}$$

thm *ctm_of_not_CH*

$$\begin{aligned}
& \llbracket M \approx \omega; \text{Transset}(M); M \models ZC \cup \{\cdot \text{Replacement}(p) \cdot \mid p \in \text{overhead_notCH}\}; \\
& \Phi \subseteq \text{formula}; M \models \{\cdot \text{Replacement}(\text{ground_repl_fm}(\varphi)) \cdot \mid \varphi \in \Phi\} \rrbracket \\
\implies & \exists N. M \subseteq N \wedge \\
& N \approx \omega \wedge \\
& \text{Transset}(N) \wedge \\
& N \models ZC \cup \{\cdot \neg \cdot CH \cdot\} \cup \{\cdot \text{Replacement}(\varphi) \cdot \mid \varphi \in \Phi\} \wedge \\
& (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N)
\end{aligned}$$

thm *ctm_of_CH*

$$\begin{aligned}
& \llbracket M \approx \omega; \text{Transset}(M); M \models ZC \cup \{\cdot \text{Replacement}(p) \cdot \mid p \in \text{overhead_CH}\}; \\
& \Phi \subseteq \text{formula}; M \models \{\cdot \text{Replacement}(\text{ground_repl_fm}(\varphi)) \cdot \mid \varphi \in \Phi\} \rrbracket \\
\implies & \exists N. M \subseteq N \wedge \\
& N \approx \omega \wedge \\
& \text{Transset}(N) \wedge \\
& N \models ZC \cup \{\cdot CH \cdot\} \cup \{\cdot \text{Replacement}(\varphi) \cdot \mid \varphi \in \Phi\} \wedge \\
& (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N)
\end{aligned}$$

In the above three statements, the function *ground_repl_fm* takes an element φ of *formula* and returns the replacement instance in the ground model that produces the φ -replacement instance in the generic extension. The next result is stated in the context *G_generic1*, which assumes the existence of a generic filter.

context *G_generic1*
begin

thm *sats_ground_repl_fm_imp_sats_ZF_replacement_fm*

$$\begin{aligned}
& \llbracket \varphi \in \text{formula}; M, [] \models \cdot \text{Replacement}(\text{ground_repl_fm}(\varphi)) \cdot \rrbracket \\
\implies & M[G], [] \models \cdot \text{Replacement}(\varphi) \cdot
\end{aligned}$$

end — *G_generic1*

end

34 Some demonstrations

```

theory Demonstrations
  imports
    Definitions_Main
begin

```

The following theory is only intended to explore some details of the formalization and to show the appearance of relevant internalized formulas. It is **not** intended as the entry point of the session. For that purpose, consult *Independence_CH.Definitions_Main*

The snippet (by M. Pagano) commented out below outputs a directed graph picturing the locale structure.

```

locale Demo = M_trivial + M_AC +
  fixes t1 t2
  assumes
    ts_in_nat[simp]: t1 ∈ ω t2 ∈ ω
  and
    power_infty: power_ax(M) M(ω)
begin

```

The next fake lemma is intended to explore the instances of the axiom schemes that are needed to build our forcing models. They are categorized as plain replacements (using *strong_replacement*), “lambda-replacements” using a higher order function, replacements to perform transfinite and general well-founded recursion (using *transrec_replacement* and *wfrec_replacement* respectively) and for the construction of fixpoints (using *iterates_replacement*). Lastly, separations instances.

```

lemma
  assumes
    sorried_replacements:
      ∧ P. strong_replacement(M,P)
      ∧ F. lam_replacement(M,F)
      ∧ Q S. iterates_replacement(M,Q,S)
      ∧ Q S. wfrec_replacement(M,Q,S)
      ∧ Q S. transrec_replacement(M,Q,S)
  and
    sorried_separations: ∧ Q. separation(M,Q)
  shows
    M_master(M)
    ⟨proof⟩
no_notation mem (infixl ⟨∈⟩ 50)
no_notation conj (infixr ⟨∧⟩ 35)
no_notation disj (infixr ⟨∨⟩ 30)
no_notation iff (infixr ⟨⟷⟩ 25)
no_notation imp (infixr ⟨⟶⟩ 25)
no_notation not (⟷ ¬) [40] 40)

```

no_notation *All* ($\langle'(\forall _)\rangle$)
no_notation *Ex* ($\langle'(\exists _)\rangle$)

no_notation *Member* ($\langle _ \in / _ \rangle$)
no_notation *Equal* ($\langle _ = / _ \rangle$)
no_notation *Nand* ($\langle _ \neg'(_ \wedge / _ \wedge) \rangle$)
no_notation *And* ($\langle _ \wedge / _ \rangle$)
no_notation *Or* ($\langle _ \vee / _ \rangle$)
no_notation *Iff* ($\langle _ \leftrightarrow / _ \rangle$)
no_notation *Implies* ($\langle _ \rightarrow / _ \rangle$)
no_notation *Neg* ($\langle _ \neg _ \rangle$)
no_notation *Forall* ($\langle'(\forall (_))\rangle$)
no_notation *Exists* ($\langle'(\exists (_))\rangle$)

notation *Member* (**infixl** $\langle \in \rangle$ 50)
notation *Equal* (**infixl** $\langle \equiv \rangle$ 50)
notation *Nand* ($\langle _ \neg'(_ \wedge / _ \wedge) \rangle$)
notation *And* (**infixr** $\langle \wedge \rangle$ 35)
notation *Or* (**infixr** $\langle \vee \rangle$ 30)
notation *Iff* (**infixr** $\langle \longleftrightarrow \rangle$ 25)
notation *Implies* (**infixr** $\langle \longrightarrow \rangle$ 25)
notation *Neg* ($\langle _ \neg _ \rangle$ [40] 40)
notation *Forall* ($\langle'(\forall _)\rangle$)
notation *Exists* ($\langle'(\exists _)\rangle$)

lemma *forces*($t_1 \in t_2$) = ($0 \in 1 \wedge \text{forces_mem_fm}(1, 2, 0, t_1 + \omega 4, t_2 + \omega 4)$)
<proof>

definition *forces_0_mem_1* **where** *forces_0_mem_1* $\equiv \text{forces_mem_fm}(1, 2, 0, t_1 + \omega 4, t_2 + \omega 4)$

thm *forces_0_mem_1_def* [
unfolded frc_at_fm_def ftype_fm_def
name1_fm_def name2_fm_def snd_snd_fm_def hcomp_fm_def
ecloseN_fm_def eclose_n1_fm_def eclose_n2_fm_def
is_eclose_fm_def mem_eclose_fm_def eclose_n_fm_def
is_If_fm_def least_fm_def Replace_fm_def Collect_fm_def
fm_definitions,simplified]

named_theorems *incr_bv_new_simps*

schematic_goal *incr_bv_Neg*:
 $\text{mem}(n, \omega) \implies \text{mem}(\varphi, \text{formula}) \implies \text{incr_bv}(\text{Neg}(\varphi))'n = ?x$
<proof>

schematic_goal *incr_bv_Exists* [*incr_bv_new_simps*]:

$mem(n,\omega) \implies mem(\varphi,formula) \implies incr_bv(Exists(\varphi)) 'n = ?x$
<proof>

end — *Demo*

end

References

- [1] E. GUNTHER, M. PAGANO, P. SÁNCHEZ TERRAF, First steps towards a formalization of forcing, in: Proceedings of the 13th Workshop on Logical and Semantic Frameworks with Applications, LSFA 2018, Fortaleza, Brazil, September 26-28, 2018, pp. 119–136 (2018).
- [2] E. GUNTHER, M. PAGANO, P. SÁNCHEZ TERRAF, Mechanization of Separation in Generic Extensions, *arXiv e-prints* **1901.03313** (2019).
- [3] E. GUNTHER, M. PAGANO, P. SÁNCHEZ TERRAF, Formalization of Forcing in Isabelle/ZF, arXiv e-prints, in: N. Peltier, V. Sofronie-Stokkermans (Eds.), Automated Reasoning. 10th International Joint Conference, IJCAR 2020, Paris, France, July 1–4, 2020, Proceedings, Part II, Lecture Notes in Artificial Intelligence **12167**, Springer International Publishing: 221–235 (2020).
- [4] L.C. PAULSON, K. GRABCZEWSKI, Mechanizing set theory, *J. Autom. Reasoning* **17**: 291–323 (1996).