

The Independence of the Continuum Hypothesis in Isabelle/ZF

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

May 20, 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)	10
3	Cohen forcing notions	13
3.1	Combinatorial results on Cohen posets	14
3.2	The well-founded relation <i>ed</i>	25
4	Well-founded relation on names	30
5	Concepts involved in instances of Replacement	45
5.1	Formulas used to prove some generic instances.	50
5.2	The relation <i>frecrel</i>	51
5.3	Definition of Forces	52
5.3.1	Definition of <i>forces</i> for equality and membership	52
5.3.2	The well-founded relation <i>forcerec</i>	53

*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>frc_at</i> , forcing for atomic formulas	53
5.5	Forcing for general formulas	56
5.5.1	The primitive recursion	59
5.6	The arity of <i>forces</i>	59
6	The ZFC axioms, internalized	67
6.1	The Axiom of Separation, internalized	68
6.2	The Axiom of Replacement, internalized	72
7	Interface between set models and Constructibility	78
7.1	Interface with <i>M_trivial</i>	79
7.2	Interface with <i>M_basic</i>	80
7.3	Interface with <i>M_trancl</i>	86
7.4	Interface with <i>M_eclose</i>	88
7.5	Interface for proving Collects and Replace in M.	94
7.6	More Instances of Separation	110
8	More Instances of Replacement	114
9	Further instances of axiom-schemes	142
10	Transitive set models of ZF	156
10.1	A forcing locale and generic filters	157
11	The definition of <i>forces</i>	160
11.1	The relation <i>frecrel</i>	160
11.2	Recursive expression of <i>frc_at</i>	171
11.3	Absoluteness of <i>frc_at</i>	172
11.4	Forcing for atomic formulas in context	175
12	Names and generic extensions	177
12.1	Values and check-names	178
13	The Forcing Theorems	191
13.1	The forcing relation in context	191
13.2	Kunen 2013, Lemma IV.2.37(a)	192
13.3	Kunen 2013, Lemma IV.2.37(a)	192
13.4	Kunen 2013, Lemma IV.2.37(b)	192
13.5	Kunen 2013, Lemma IV.2.38	194
13.6	The relation of forcing and atomic formulas	194
13.7	The relation of forcing and connectives	195
13.8	Kunen 2013, Lemma IV.2.29	196
13.9	Auxiliary results for Lemma IV.2.40(a)	197
13.10	Induction on names	200
13.11	Lemma IV.2.40(a), in full	202

13.12	Lemma IV.2.40(b)	202
13.13	The Strengthening Lemma	208
13.14	The Density Lemma	210
13.15	The Truth Lemma	212
13.16	The “Definition of forcing”	221
14	Ordinals in generic extensions	222
15	Auxiliary renamings for Separation	223
16	The Axiom of Separation in $M[G]$	234
17	The Axiom of Pairing in $M[G]$	241
18	The Axiom of Unions in $M[G]$	242
19	The Powerset Axiom in $M[G]$	244
20	The Axiom of Extensionality in $M[G]$	249
21	The Axiom of Foundation in $M[G]$	250
22	The Axiom of Replacement in $M[G]$	251
23	The Axiom of Infinity in $M[G]$	257
24	The Axiom of Choice in $M[G]$	258
24.1	$M[G]$ is a transitive model of ZF	261
25	Separative notions and proper extensions	264
26	A poset of successions	266
26.1	Cohen extension is proper	270
27	The existence of generic extensions	271
27.1	The generic extension is countable	271
27.2	Extensions of ctm's of fragments of ZFC	272
28	Preservation of cardinals in generic extensions	274
28.1	Preservation by ccc forcing notions	279
29	Model of the negation of the Continuum Hypothesis	284
29.1	Non-absolute concepts between extensions	285
29.2	Cohen forcing is ccc	287
29.3	Models of fragments of $ZFC + \neg CH$	294

30 Preservation results for κ-closed forcing notions	297
30.1 $(\omega + 1)$ -Closed notions preserve countable sequences	305
31 Forcing extension satisfying the Continuum Hypothesis	312
31.1 Collapse forcing is sufficiently closed	313
31.2 Models of fragments of $ZFC + CH$	319
32 From M to \mathcal{V}	321
32.1 Locales of a class M hold in \mathcal{V}	321
33 Main definitions of the development	324
33.1 ZF	325
33.2 Relative concepts	327
33.3 Relativization of infinitary arithmetic	332
33.4 Forcing	333
34 Some demonstrations	336

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 $compat_inI$:
 $[[d \in A ; \langle d, p \rangle \in r ; \langle d, q \rangle \in r]] \Longrightarrow compat_in(A, r, p, q)$
by (*auto simp add: compat_in_def*)

lemma $refl_compat$:
 $[[refl(A, r) ; \langle p, q \rangle \in r \mid p = q \mid \langle q, p \rangle \in r ; p \in A ; q \in A]] \Longrightarrow compat_in(A, r, p, q)$
by (*auto simp add: refl_def compat_inI*)

lemma $chain_compat$:
 $refl(A, r) \Longrightarrow linear(A, r) \Longrightarrow (\forall p \in A. \forall q \in A. compat_in(A, r, p, q))$
by (*simp add: refl_compat linear_def*)

lemma $subset_fun_image$: $f: N \rightarrow P \Longrightarrow f''N \subseteq P$
by (*auto simp add: image_fun apply_funtype*)

lemma $refl_monot_domain$: $refl(B, r) \Longrightarrow A \subseteq B \Longrightarrow refl(A, r)$
unfolding $refl_def$ **by** *blast*

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** \preceq 50)
where $x \preceq y \equiv \langle x, y \rangle \in leq$

lemma $refl_leq$:
 $r \in \mathbb{P} \Longrightarrow r \preceq r$
using leq_preord **unfolding** $preorder_on_def$ $refl_def$ **by** *simp*

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})

by (*insert leq_preord*, *auto simp add: preorder_on_def refl_def dense_def*)

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*: $a \preceq b \Longrightarrow b \preceq c \Longrightarrow a \in \mathbb{P} \Longrightarrow b \in \mathbb{P} \Longrightarrow c \in \mathbb{P} \Longrightarrow a \preceq c$
using *leq_preord trans_onD unfolding preorder_on_def by blast*

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

using *leq_preord trans_onD subsetD unfolding preorder_on_def by blast*

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

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

lemma *compatI[intro!]*: $d \in \mathbb{P} \Longrightarrow d \preceq p \Longrightarrow d \preceq q \Longrightarrow \text{compat}(p, q)$
unfolding *compat_def compat_in_def by blast*

lemma *Incompatible_imp_not_eq*: $\llbracket p \perp q; p \in \mathbb{P}; q \in \mathbb{P} \rrbracket \Longrightarrow p \neq q$
using *refl_leq by blast*

lemma *denseD [dest]*: $\text{dense}(D) \Longrightarrow p \in \mathbb{P} \Longrightarrow \exists d \in D. d \preceq p$
unfolding *dense_def by blast*

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

lemma *dense_belowD [dest]*:
assumes *dense_below*(D, p) $q \in \mathbb{P} \ q \preceq p$
shows $\exists d \in D. d \in \mathbb{P} \wedge d \preceq q$
using *assms unfolding dense_below_def by simp*

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 $\text{dense_below}(D, p)$
using *assms unfolding dense_below_def by simp*

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

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

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

lemma *dense_below_under*:

assumes $\text{dense_below}(D, p) \ p \in \mathbb{P} \ q \in \mathbb{P} \ q \preceq p$
shows $\text{dense_below}(D, q)$
using *assms leq_transD by blast*

lemma *ideal_dense_below*:

assumes $\bigwedge q. q \in \mathbb{P} \implies q \preceq p \implies q \in D$
shows $\text{dense_below}(D, p)$
using *assms refl_leq by blast*

lemma *dense_below_dense_below*:

assumes $\text{dense_below}(\{q \in \mathbb{P}. \text{dense_below}(D, q)\}, p) \ p \in \mathbb{P}$
shows $\text{dense_below}(D, p)$
using *assms leq_transD refl_leq by blast*

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* : $\text{filter}(G) \implies x \in G \implies x \in \mathbb{P}$
by (*auto simp add : subsetD filter_def*)

lemma *filter_subset_notion*[*dest*]: $\text{filter}(G) \implies G \subseteq \mathbb{P}$
by (*auto dest: filterD*)

lemma *filter_leqD* : $\text{filter}(G) \implies x \in G \implies y \in \mathbb{P} \implies x \preceq y \implies y \in G$
by (*simp add: filter_def increasing_def*)

lemma *filter_imp_compat*: $\text{filter}(G) \implies p \in G \implies q \in G \implies \text{compat}(p, q)$
unfolding *filter_def compat_in_def compat_def by blast*

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

assumes $filter(G)$ **and** $p \in G$ **and** $q \in G$
shows $\exists r \in G. r \preceq p \wedge r \preceq q$
using $assms$
unfolding $compat_in_def$ $filter_def$ **by** $blast$

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 $upclosureI$ [$intro$] : $p \in \mathbb{P} \Longrightarrow a \in A \Longrightarrow a \preceq p \Longrightarrow p \in upclosure(A)$
by ($simp$ $add:upclosure_def$, $auto$)

lemma $upclosureE$ [$elim$] :
 $p \in upclosure(A) \Longrightarrow (\bigwedge x a. x \in \mathbb{P} \Longrightarrow a \in A \Longrightarrow a \preceq x \Longrightarrow R) \Longrightarrow R$
by ($auto$ $simp$ $add:upclosure_def$)

lemma $upclosureD$ [$dest$] :
 $p \in upclosure(A) \Longrightarrow \exists a \in A. (a \preceq p) \wedge p \in \mathbb{P}$
by ($simp$ $add:upclosure_def$)

lemma $upclosure_increasing$:
assumes $A \subseteq \mathbb{P}$
shows $increasing(upclosure(A))$
unfolding $increasing_def$ $upclosure_def$
using leq_transD' [$OF' \langle A \subseteq \mathbb{P} \rangle$] **by** $auto$

lemma $upclosure_in_P$: $A \subseteq \mathbb{P} \Longrightarrow upclosure(A) \subseteq \mathbb{P}$
using $subsetI$ $upclosure_def$ **by** $simp$

lemma $A_sub_upclosure$: $A \subseteq \mathbb{P} \Longrightarrow A \subseteq upclosure(A)$
using $subsetI$ leq_preord
unfolding $upclosure_def$ $preorder_on_def$ $refl_def$ **by** $auto$

lemma $elem_upclosure$: $A \subseteq \mathbb{P} \Longrightarrow x \in A \Longrightarrow x \in upclosure(A)$
by ($blast$ $dest:A_sub_upclosure$)

lemma $closure_compat_filter$:
assumes $A \subseteq \mathbb{P}$ ($\forall p \in A. \forall q \in A. compat_in(A, leq, p, q)$)
shows $filter(upclosure(A))$
unfolding $filter_def$
proof($auto$)
show $increasing(upclosure(A))$
using $assms$ $upclosure_increasing$ **by** $simp$
next
let $?UA = upclosure(A)$
show $compat_in(upclosure(A), leq, p, q)$ **if** $p \in ?UA$ $q \in ?UA$ **for** p q
proof -

from that
obtain $a\ b$ **where** $1:a\in A\ b\in A\ a\preceq p\ b\preceq q\ p\in\mathbb{P}\ q\in\mathbb{P}$
using $upclosureD[OF\ \langle p\in?UA\rangle]\ upclosureD[OF\ \langle q\in?UA\rangle]$ **by auto**
with $assms(2)$
obtain d **where** $d\in A\ d\preceq a\ d\preceq b$
unfolding $compat_in_def$ **by auto**
with 1
have $d\preceq p\ d\preceq q\ d\in?UA$
using $A_sub_upclosure[THEN\ subsetD]\ \langle A\subseteq\mathbb{P}\rangle$
 $leq_transD'[of\ A\ d\ a]\ leq_transD'[of\ A\ d\ b]$ **by auto**
then
show $?thesis$ **unfolding** $compat_in_def$ **by auto**
qed
qed

lemma $aux_RS1: f\in N\ \rightarrow\ \mathbb{P}\ \Longrightarrow\ n\in N\ \Longrightarrow\ f^n\in upclosure(f\ \text{``}N)$
using $elem_upclosure[OF\ subset_fun_image]\ image_fun$
by $(simp, blast)$

lemma $decr_succ_decr:$
assumes $f\in nat\ \rightarrow\ \mathbb{P}\ preorder_on(\mathbb{P}, leq)$
 $\forall n\in nat. \langle f^n\ succ(n), f^n\rangle\in leq$
 $m\in nat$
shows $n\in nat\ \Longrightarrow\ n\leq m\ \Longrightarrow\ \langle f^n\ m, f^n\rangle\in leq$
using $\langle m\in_ \rangle$
proof($induct\ m$)
case 0
then show $?case$ **using** $assms\ refl_leq$ **by simp**
next
case $(succ\ x)$
then
have $1:f^n\ succ(x)\preceq f^n\ x\ f^n\ n\in\mathbb{P}\ f^n\ x\in\mathbb{P}\ f^n\ succ(x)\in\mathbb{P}$
using $assms$ **by** $simp_all$
consider $(lt)\ n<succ(x)\ | (eq)\ n=succ(x)$
using $succ\ le_succ_iff$ **by auto**
then
show $?case$
proof($cases$)
case lt
with 1 **show** $?thesis$ **using** $leI\ succ\ leq_transD$ **by auto**
next
case eq
with 1 **show** $?thesis$ **using** $refl_leq$ **by simp**
qed
qed

lemma $decr_seq_linear:$
assumes $refl(\mathbb{P}, leq)\ f\in nat\ \rightarrow\ \mathbb{P}$
 $\forall n\in nat. \langle f^n\ succ(n), f^n\rangle\in leq$

```

    trans[ $\mathbb{P}$ ](leq)
  shows linear(f “ nat, leq)
proof -
  have preorder_on( $\mathbb{P}$ , leq)
    unfolding preorder_on_def using assms by simp
  {
    fix n m
    assume n ∈ nat m ∈ nat
    then
    have f‘m ≤ f‘n ∨ f‘n ≤ f‘m
    proof(cases m ≤ n)
      case True
      with ⟨n ∈ _⟩ ⟨m ∈ _⟩
      show ?thesis
        using decr_succ_decr[of f n m] assms leI ⟨preorder_on( $\mathbb{P}$ , leq)⟩ by simp
    next
      case False
      with ⟨n ∈ _⟩ ⟨m ∈ _⟩
      show ?thesis
        using decr_succ_decr[of f m n] assms leI not_le_iff_lt ⟨preorder_on( $\mathbb{P}$ , leq)⟩
    by simp
  }
  qed
}
then
show ?thesis
  unfolding linear_def using ball_image_simp assms by auto
qed

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)  $\equiv$  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 ∈  $\mathbb{P}$  f : nat →  $\mathbb{P}$  f ‘ 0 = p
     $\bigwedge n. n \in \text{nat} \implies f ‘ \text{succ}(n) \leq f ‘ n \wedge f ‘ \text{succ}(n) \in \mathcal{D} ‘ n$ 
  shows

```

```

     $\exists G. p \in G \wedge D\_generic(G)$ 
proof -
  note assms
  moreover from this
  have  $f''nat \subseteq \mathbb{P}$ 
    by (simp add:subset_fun_image)
  moreover from calculation
  have  $refl(f''nat, leq) \wedge trans[\mathbb{P}](leq)$ 
    using leq_preord unfolding preorder_on_def by (blast intro:refl_monot_domain)
  moreover from calculation
  have  $\forall n \in nat. f' succ(n) \preceq f' n$  by (simp)
  moreover from calculation
  have linear( $f''nat, leq$ )
    using leq_preord and decr_seq_linear unfolding preorder_on_def by (blast)
  moreover from calculation
  have ( $\forall p \in f''nat. \forall q \in f''nat. compat\_in(f''nat, leq, p, q)$ )
    using chain_compat by (auto)
  ultimately
  have filter(upclosure( $f''nat$ )) (is filter( $?G$ ))
    using closure_compat_filter by simp
  moreover
  have  $\forall n \in nat. \mathcal{D}' n \cap ?G \neq 0$ 
proof
  fix n
  assume  $n \in nat$ 
  with assms
  have  $f' succ(n) \in ?G \wedge f' succ(n) \in \mathcal{D}' n$ 
    using aux_RS1 by simp
  then
  show  $\mathcal{D}' n \cap ?G \neq 0$  by blast
qed
  moreover from assms
  have  $p \in ?G$ 
    using aux_RS1 by auto
  ultimately
  show thesis unfolding D_generic_def by auto
qed

```

end — *countable_generic*

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

```

consts RS_seq ::  $[i, i, i, i, i] \Rightarrow i$ 
primrec
  RS_seq( $0, P, leq, p, enum, \mathcal{D}$ ) = p
  RS_seq(succ(n), P, leq, p, enum,  $\mathcal{D}$ ) =
    enum'( $\mu m. \langle enum'm, RS\_seq(n, P, leq, p, enum, \mathcal{D}) \rangle \in leq \wedge enum'm \in \mathcal{D}' n$ )

```

context *countable_generic*

begin

lemma countable_RS_sequence_aux:

fixes p $enum$
defines $f(n) \equiv RS_seq(n, \mathbb{P}, leq, p, enum, \mathcal{D})$
and $Q(q, k, m) \equiv enum\ 'm \preceq q \wedge enum\ 'm \in \mathcal{D}\ 'k$
assumes $n \in nat$ $p \in \mathbb{P}$ $\mathbb{P} \subseteq range(enum)$ $enum: nat \rightarrow M$
 $\wedge x k. x \in \mathbb{P} \implies k \in nat \implies \exists q \in \mathbb{P}. q \preceq x \wedge q \in \mathcal{D}\ 'k$
shows
 $f(succ(n)) \in \mathbb{P} \wedge f(succ(n)) \preceq f(n) \wedge f(succ(n)) \in \mathcal{D}\ 'n$
using $\langle n \in nat \rangle$
proof (induct)
case 0
from $assms$
obtain q where $q \in \mathbb{P}$ $q \preceq p$ $q \in \mathcal{D}\ '0$ by blast
moreover from this and $\langle \mathbb{P} \subseteq range(enum) \rangle$
obtain m where $m \in nat$ $enum\ 'm = q$
using $Pi_rangeD[OF \langle enum: nat \rightarrow M \rangle]$ by blast
moreover
have $\mathcal{D}\ '0 \subseteq \mathbb{P}$
using $apply_funtype[OF countable_subs_of_P]$ by simp
moreover note $\langle p \in \mathbb{P} \rangle$
ultimately
show ?case
using $LeastI[of Q(p, 0) m]$ unfolding $Q_def\ f_def$ by auto
next
case (succ n)
with $assms$
obtain q where $q \in \mathbb{P}$ $q \preceq f(succ(n))$ $q \in \mathcal{D}\ 'succ(n)$ by blast
moreover from this and $\langle \mathbb{P} \subseteq range(enum) \rangle$
obtain m where $m \in nat$ $enum\ 'm \preceq f(succ(n))$ $enum\ 'm \in \mathcal{D}\ 'succ(n)$
using $Pi_rangeD[OF \langle enum: nat \rightarrow M \rangle]$ by blast
moreover note $succ$
moreover from $calculation$
have $\mathcal{D}\ 'succ(n) \subseteq \mathbb{P}$
using $apply_funtype[OF countable_subs_of_P]$ by auto
ultimately
show ?case
using $LeastI[of Q(f(succ(n)), succ(n)) m]$ unfolding $Q_def\ f_def$ by auto
qed

lemma countable_RS_sequence:

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

```

from assms
show  $f'0 = p$  by simp
{
  fix  $x k$ 
  assume  $x \in \mathbb{P} \ k \in \text{nat}$ 
  then
  have  $\exists q \in \mathbb{P}. q \preceq x \wedge q \in \mathcal{D} \text{ ` } k$ 
    using seq_of_densens_apply_funtype[OF countable_subs_of_P]
    unfolding dense_def by blast
}
with assms
show  $f'\text{succ}(n) \preceq f'n \wedge f'\text{succ}(n) \in \mathcal{D} \text{ ` } n \ f'\text{succ}(n) \in \mathbb{P}$ 
  unfolding f_def using countable_RS_sequence_aux by simp_all
qed

```

```

lemma RS_seq_type:
  assumes  $n \in \text{nat} \ p \in \mathbb{P} \ \mathbb{P} \subseteq \text{range}(enum) \ enum: \text{nat} \rightarrow M$ 
  shows  $RS\_seq(n, \mathbb{P}, leq, p, enum, \mathcal{D}) \in \mathbb{P}$ 
  using assms countable_RS_sequence(1,3)
  by (induct; simp)

```

```

lemma RS_seq_funtype:
  assumes  $p \in \mathbb{P} \ \mathbb{P} \subseteq \text{range}(enum) \ enum: \text{nat} \rightarrow M$ 
  shows  $(\lambda n \in \text{nat}. RS\_seq(n, \mathbb{P}, leq, p, enum, \mathcal{D})): \text{nat} \rightarrow \mathbb{P}$ 
  using assms lam_type RS_seq_type by auto

```

```

lemmas countable_rasiowa_sikorski =
  RS_sequence_imp_rasiowa_sikorski[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_lesspoll_kappa = zero_lesspoll[OF 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 $\exists d \in \text{inj_dense}(I, J, w, x). \langle d, p \rangle \in \text{Fnle}(\omega, I \times \omega, J)$

proof -

obtain n **where** $\langle w, n \rangle \notin \text{domain}(p)$ $\langle x, n \rangle \notin \text{domain}(p)$ $n \in \omega$

proof -

{
 assume $\langle w, n \rangle \in \text{domain}(p) \vee \langle x, n \rangle \in \text{domain}(p)$ **if** $n \in \omega$ **for** n
 then
 have $\omega \subseteq \text{range}(\text{domain}(p))$ **by** *blast*
 then
 have $\neg \text{Finite}(p)$
 using *Finite_range Finite_domain subset_Finite nat_not_Finite*
 by *auto*
 with $\langle p \in _ \rangle$
 have *False*
 using *Fn_nat_eq_FiniteFun FiniteFun.dom_subset[of I \times ω J]*
 Fin_into_Finite **by** *auto*
}

with *that*— the shape of the goal puts assumptions in this variable
show *?thesis* **by** *auto*

qed

moreover

note $\langle p \in _ \rangle$ *assms*

moreover from *calculation*

have $\text{cons}(\langle \langle x, n \rangle, 0 \rangle, p) \in \text{Fn}(\omega, I \times \omega, J)$

using *FiniteFun.consI[of $\langle x, n \rangle$ $I \times \omega$ 0 J p]*
Fn_nat_eq_FiniteFun **by** *auto*

ultimately

have $\text{cons}(\langle \langle w, n \rangle, 1 \rangle, \text{cons}(\langle \langle x, n \rangle, 0 \rangle, p)) \in \text{Fn}(\omega, I \times \omega, J)$

using *FiniteFun.consI[of $\langle w, n \rangle$ $I \times \omega$ 1 J $\text{cons}(\langle \langle x, n \rangle, 0 \rangle, p)$]*
Fn_nat_eq_FiniteFun **by** *auto*

with $\langle n \in \omega \rangle$

show *?thesis*

using $\langle p \in _ \rangle$ **by** (*intro bexI*) *auto*

qed

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

3.1 Combinatorial results on Cohen posets

sublocale *cohen_data* \subseteq *forcing_notion* *Fn*(κ, I, J) *Fnle*(κ, I, J) 0

```

proof
  show  $0 \in Fn(\kappa, I, J)$ 
    using zero_lt_kappa zero_in_Fn by simp
  then
  show  $\forall p \in Fn(\kappa, I, J). \langle p, 0 \rangle \in Fnle(\kappa, I, J)$ 
    unfolding preorder_on_def refl_def trans_on_def
    by auto
next
  show  $preorder\_on(Fn(\kappa, I, J), Fnle(\kappa, I, J))$ 
    unfolding preorder_on_def refl_def trans_on_def
    by blast
qed

context cohen_data
begin

lemma compat_imp_Un_is_function:
  assumes  $G \subseteq Fn(\kappa, I, J) \wedge p \sqsubseteq q. p \in G \implies q \in G \implies compat(p, q)$ 
  shows  $function(\bigcup G)$ 
  unfolding function_def
proof (intro allI ballI impI)
  fix  $x \ y \ y'$ 
  assume  $\langle x, y \rangle \in \bigcup G \ \langle x, y' \rangle \in \bigcup G$ 
  then
  obtain  $p \ q$  where  $\langle x, y \rangle \in p \ \langle x, y' \rangle \in q \ p \in G \ q \in G$ 
    by auto
  moreover from this and assms
  obtain  $r$  where  $r \in Fn(\kappa, I, J) \ r \supseteq p \ r \supseteq q$ 
    using compatD[of  $p \ q$ ] by force
  moreover from this
  have  $function(r)$ 
    using Fn_is_function by simp
  ultimately
  show  $y = y'$ 
    unfolding function_def by fastforce
qed

lemma Un_filter_is_function:  $filter(G) \implies function(\bigcup G)$ 
  using compat_imp_Un_is_function filter_imp_compat[of  $G$ ]
  filter_subset_notion
  by simp

end — cohen_data

locale M_cohen = M_delta +
  assumes
    countable_lepoll_assms2:

```

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

and

countable_lepoll_assms3:

$M(A) \implies M(f) \implies M(b) \implies M(D) \implies M(r') \implies M(A') \implies$

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

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

$M(\text{Fnle}^M(\aleph_1^M, \aleph_1^M, \omega \rightarrow^M 2))$

by *simp*

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*

relativize relational *compat_in* *is_compat_in* **external**

synthesize *compat_in* **from_definition** *is_compat_in* **assuming** *nonempty*
context

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

begin

arity_theorem **for** *compat_in_fm*

end

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

assumes

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

shows

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

using *assms* **unfolding** *is_compat_in_def* *compat_in_def* **by** *simp*

definition

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

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

relativize relational *antichain* *antichain_rel*

definition

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

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

abbreviation

antichain_rel_abbr :: $[i \Rightarrow o, i, i, i] \Rightarrow o$ ($\langle \text{antichain}'(_, _, _) \rangle$) **where**

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

abbreviation

$antichain_r_set :: [i,i,i,i] \Rightarrow o (\langle antichain-'(_,_,')\rangle)$ **where**
 $antichain^M(P,leq,A) \equiv antichain_rel(\#\#M,P,leq,A)$

context $M_trivial$ **begin****lemma** $antichain_abs$ [*absolut*]:

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

unfolding $antichain_rel_def$ $antichain_def$ **by** (*simp add:absolut*)

end — $M_trivial$ **relativize relational** ccc ccc_rel **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**

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

using $is_cardinal_iff$ **unfolding** ccc_rel_def **by** (*simp add:absolut*)**end** — $M_cardinals$ **context** $M_FiniteFun$ **begin****lemma** $Fnle_nat_closed$ [*intro,simp*]:**assumes** $M(I)$ $M(J)$ **shows** $M(Fnle(\omega,I,J))$ **unfolding** $Fnle_def$ $Fnlerel_def$ $Rrel_def$ **using** $supset_separation$ $FiniteFun_closed$ $Fn_nat_eq_FiniteFun$ *assms* **by** *simp***lemma** Fn_nat_closed :**assumes** $M(A)$ $M(B)$ **shows** $M(Fn(\omega,A,B))$ **using** *assms* $Fn_nat_eq_FiniteFun$ **by** *simp***end** — $M_FiniteFun$

context *M_add_reals*

begin

lemma *lam_replacement_drSR_Y*: $M(A) \implies M(D) \implies M(r') \implies \text{lam_replacement}(M, \text{drSR_Y}(r', D, A))$

using *lam_replacement_drSR_Y*

by *simp*

lemma (**in** *M_trans*) *mem_F_bound3*:

fixes *F A*

defines $F \equiv dC_F$

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

using *apply_0 unfolding F_def*

by (*cases M(c), auto simp:F_def drSR_Y_def dC_F_def*)

lemma *ccc_rel_Fn_nat*:

assumes $M(I)$

shows $\text{ccc}^M(\text{Fn}(\text{nat}, I, 2), \text{Fnle}(\text{nat}, I, 2))$

proof -

have *repFun_dom_closed*: $M(\{\text{domain}(p) . p \in A\})$ **if** $M(A)$ **for** *A*

using *RepFun_closed domain_replacement_simp transM[OF_⟨M(A)⟩]* **that**

by *auto*

from *assms*

have $M(\text{Fn}(\text{nat}, I, 2))$ **using** *Fn_nat_eq_FiniteFun* **by** *simp*

{

fix *A*

assume $\neg |A|^M \leq \text{nat } M(A) \ A \subseteq \text{Fn}(\text{nat}, I, 2)$

moreover from *this*

have *countable_rel*($M, \{p \in A. \text{domain}(p) = d\}$) **if** $M(d)$ **for** *d*

proof (*cases d <math>\prec^M \text{nat} \wedge d \subseteq I>)*

case *True*

with $\langle A \subseteq \text{Fn}(\text{nat}, I, 2) \rangle \langle M(A) \rangle$

have $\{p \in A . \text{domain}(p) = d\} \subseteq d \rightarrow^M 2$

using *domain_of_fun function_space_rel_char[of_2]*

by (*auto, subgoal_tac M(domain(x)), simp_all add:transM[of_A], force*)

moreover from *True* $\langle M(d) \rangle$

have *Finite*($d \rightarrow^M 2$)

using *Finite_Pi[THEN [2] subset_Finite, of_d λ_. 2]*

lesspoll_rel_nat_is_Finite_rel function_space_rel_char[of_2]

by *auto*

moreover from $\langle M(d) \rangle$

have $M(d \rightarrow^M 2)$

by *simp*

moreover from $\langle M(A) \rangle$

have $M(\{p \in A . \text{domain}(p) = d\})$

using *separation_closed domain_eq_separation[OF ⟨M(d)⟩]* **by** *simp*

ultimately

show *?thesis* **using** *subset_Finite[of_d →^M 2]*

```

    by (auto intro!:Finite_imp_countable_rel)
next
case False
with  $\langle A \subseteq Fn(nat, I, 2) \rangle \langle M(A) \rangle$ 
have  $domain(p) \neq d$  if  $p \in A$  for  $p$ 
proof -
  note False that  $\langle M(A) \rangle$ 
  moreover from this
  obtain  $d'$  where  $d' \subseteq I$   $p \in d' \rightarrow 2$   $d' \prec \omega$ 
  using  $FnD[OF subsetD[OF \langle A \subseteq \_ \rangle \langle p \in A \rangle]]$ 
  by auto
  moreover from this
  have  $p \approx d'$   $domain(p) = d'$ 
  using  $function_eqpoll Pi\_iff$ 
  by auto
  ultimately
  show ?thesis
  using  $lesspoll\_nat\_imp\_lesspoll\_rel$   $transM[of p]$ 
  by auto
qed
then
show ?thesis
  using  $empty\_lepoll\_relI$  by auto
qed
have  $2:M(x) \implies x \in dC\_F(X, i) \implies M(i)$  for  $x \in X$   $i$ 
  unfolding  $dC\_F\_def$ 
  by auto
moreover
have  $uncountable\_rel(M, \{domain(p) . p \in A\})$ 
proof
  interpret  $M\_replacement\_lepoll$   $M$   $dC\_F$ 
  using  $lam\_replacement\_dC\_F$   $domain\_eq\_separation$   $lam\_replacement\_inj\_rel$ 
 $lam\_replacement\_minimum$ 
  unfolding  $dC\_F\_def$ 
  proof( $unfold\_locales, simp\_all$ )
  fix  $X$   $b$   $f$ 
  assume  $M(X)$   $M(b)$   $M(f)$ 
  with  $2[of X]$ 
  show  $lam\_replacement(M, \lambda x. \mu i. x \in if\_range\_F\_else\_F(\lambda d. \{p \in X .$ 
 $domain(p) = d\}, b, f, i))$ 
  using  $lam\_replacement\_dC\_F$   $domain\_eq\_separation$ 
 $mem\_F\_bound3$   $countable\_lepoll\_assms2$   $repFun\_dom\_closed$ 
  by ( $rule\_tac$   $lam\_Least\_assumption\_general[where U=\lambda_. \{domain(x).$ 
 $x \in X\}], auto)$ 
  qed (auto)
  have  $\exists a \in A. x = domain(a) \implies M(dC\_F(A, x))$  for  $x$ 
  using  $\langle M(A) \rangle$   $transM[OF \_ \langle M(A) \rangle]$  by (auto)
  moreover
  have  $w \in A \wedge domain(w) = x \implies M(x)$  for  $w \in X$ 

```

```

using transM[OF _ ⟨M(A)⟩] by auto
ultimately
interpret M_cardinal_UN_lepoll_dC_F(A) {domain(p). p∈A}
using lam_replacement_dC_F lam_replacement_inj_rel ⟨M(A)⟩
  lepoll_assumptions domain_eq_separation lam_replacement_minimum
  countable_lepoll_assms2 repFun_dom_closed
  lepoll_assumptions1[OF ⟨M(A)⟩ repFun_dom_closed[OF ⟨M(A)⟩]]
apply(unfold_locales)
by(simp_all del:if_range_F_else_F_def,
  rule_tac lam_Least_assumption_general[where U=λ_. {domain(x).
x∈A}])
  (auto simp del:if_range_F_else_F_def simp add:dC_F_def)
from ⟨A ⊆ Fn(nat, I, 2)⟩
have x:(⋃ d∈{domain(p) . p ∈ A}. {p∈A. domain(p) = d}) = A
  by auto
moreover
assume countable_rel(M, {domain(p) . p ∈ A})
moreover
note ⟨∧d. M(d) ⇒ countable_rel(M, {p∈A. domain(p) = d})⟩
moreover from ⟨M(A)⟩
have p∈A ⇒ M(domain(p)) for p
  by (auto dest: transM)
ultimately
have countable_rel(M, A)
  using countable_rel_imp_countable_rel_UN
  unfolding dC_F_def
  by auto
with ⟨¬ |A|^M ≤ nat⟩ ⟨M(A)⟩
show False
  using countable_rel_iff_cardinal_rel_le_nat by simp
qed
moreover from ⟨A ⊆ Fn(nat, I, 2)⟩ ⟨M(A)⟩
have p ∈ A ⇒ Finite(domain(p)) for p
  using lesspoll_rel_nat_is_Finite_rel[of domain(p)]
  lesspoll_nat_imp_lesspoll_rel[of domain(p)]
  domain_of_fun[of p _ λ_. 2] by (auto dest:transM)
moreover
note repFun_dom_closed[OF ⟨M(A)⟩]
ultimately
obtain D where delta_system(D) D ⊆ {domain(p) . p ∈ A} D ≈M ⋈IM M(D)
  using delta_system_uncountable_rel[of {domain(p) . p ∈ A}] by auto
then
have delta:∀ d1∈D. ∀ d2∈D. d1 ≠ d2 → d1 ∩ d2 = ⋂ D
  using delta_system_root_eq_Inter
  by simp
moreover from ⟨D ≈M ⋈IM M(D)⟩, ⟨M(D)⟩
have uncountable_rel(M, D)
  using uncountable_rel_iff_subset_eqpoll_rel_Aleph_rel1 by auto
moreover from this and ⟨D ⊆ {domain(p) . p ∈ A}⟩

```

```

obtain  $p1$  where  $p1 \in A$   $\text{domain}(p1) \in D$ 
  using uncountable_rel_not_empty[of  $D$ ] by blast
moreover from this and  $\langle p1 \in A \implies \text{Finite}(\text{domain}(p1)) \rangle$ 
have  $\text{Finite}(\text{domain}(p1))$ 
  using Finite_domain by simp
moreover
define  $r$  where  $r \equiv \bigcap D$ 
moreover from  $\langle M(D) \rangle$ 
have  $M(r)$   $M(r \times 2)$ 
  unfolding r_def by simp_all
ultimately
have  $\text{Finite}(r)$  using subset_Finite[of  $r$   $\text{domain}(p1)$ ]
  by auto
have countable_rel( $M, \{\text{restrict}(p, r) \mid p \in A\}$ )
proof -
  note  $\langle M(\text{Fn}(\text{nat}, I, 2)) \rangle$   $\langle M(r) \rangle$ 
  moreover from this
  have  $f \in \text{Fn}(\text{nat}, I, 2) \implies M(\text{restrict}(f, r))$  for  $f$ 
    by (blast dest: transM)
  ultimately
  have  $f \in \text{Fn}(\text{nat}, I, 2) \implies \text{restrict}(f, r) \in \text{Pow\_rel}(M, r \times 2)$  for  $f$ 
    using restrict_subset_Sigma[of  $f$   $\lambda \_ . 2$   $r$ ] Pow_rel_char
    by (auto del: FnD dest!: FnD simp: Pi_def) (auto dest: transM)
  with  $\langle A \subseteq \text{Fn}(\text{nat}, I, 2) \rangle$ 
  have  $\{\text{restrict}(f, r) \mid f \in A\} \subseteq \text{Pow\_rel}(M, r \times 2)$ 
    by fast
  moreover from  $\langle M(A) \rangle$   $\langle M(r) \rangle$ 
  have  $M(\{\text{restrict}(f, r) \mid f \in A\})$ 
    using RepFun_closed restrict_strong_replacement transM[OF  $\_ \langle M(A) \rangle$ ]
by auto
  moreover
  note  $\langle \text{Finite}(r) \rangle$   $\langle M(r) \rangle$ 
  ultimately
  show ?thesis
    using Finite_Sigma[THEN Finite_Pow_rel, of  $r$   $\lambda \_ . 2$ ]
    by (intro Finite_imp_countable_rel) (auto intro: subset_Finite)
qed
moreover from  $\langle M(A) \rangle$   $\langle M(D) \rangle$ 
have  $M(\{p \in A. \text{domain}(p) \in D\})$ 
  using domain_mem_separation by simp
have uncountable_rel( $M, \{p \in A. \text{domain}(p) \in D\}$ ) (is uncountable_rel( $M, ?X$ ))
proof
  from  $\langle D \subseteq \{\text{domain}(p) \mid p \in A\} \rangle$ 
  have  $(\lambda p \in ?X. \text{domain}(p)) \in \text{surj}(?X, D)$ 
    using lam_type unfolding surj_def by auto
  moreover from  $\langle M(A) \rangle$   $\langle M(?X) \rangle$ 
  have  $M(\lambda p \in ?X. \text{domain}(p))$ 
    using lam_closed[OF domain_replacement  $\langle M(?X) \rangle$ ] transM[OF  $\_ \langle M(?X) \rangle$ ]
by simp

```

```

moreover
note  $\langle M(?X) \rangle \langle M(D) \rangle$ 
moreover from calculation
have surjection:  $(\lambda p \in ?X. \text{domain}(p)) \in \text{surj}^M(?X, D)$ 
  using surj_rel_char by simp
moreover
assume countable_rel( $M, ?X$ )
moreover
note  $\langle \text{uncountable\_rel}(M, D) \rangle$ 
ultimately
show False
  using surj_rel_countable_rel[OF _ surjection] by auto
qed
moreover
have  $D = (\bigcup f \in \text{Pow\_rel}(M, r \times \mathbb{2}). \{y . p \in A, \text{restrict}(p, r) = f \wedge y = \text{domain}(p)\})$ 
 $\wedge \text{domain}(p) \in D$ )
proof -
  {
    fix  $z$ 
    assume  $z \in D$ 
    with  $\langle M(D) \rangle$ 
    have  $\langle M(z) \rangle$  by (auto dest:transM)
    from  $\langle z \in D \rangle \langle D \subseteq \_ \rangle \langle M(A) \rangle$ 
    obtain  $p$  where  $\text{domain}(p) = z \wedge p \in A \wedge M(p)$ 
    by (auto dest:transM)
    moreover from  $\langle A \subseteq \text{Fn}(\text{nat}, I, \mathbb{2}) \rangle \langle M(z) \rangle$  and this
    have  $p \in z \rightarrow^M \mathbb{2}$ 
    using domain_of_fun_function_space_rel_char by (auto del:FxD
dest!:FxD)
    moreover from this  $\langle M(z) \rangle$ 
    have  $p \in z \rightarrow \mathbb{2}$ 
    using domain_of_fun_function_space_rel_char by (auto)
    moreover from this  $\langle M(r) \rangle$ 
    have  $\text{restrict}(p, r) \subseteq r \times \mathbb{2}$ 
    using function_restrictI[of p r] fun_is_function[of p z lambda. 2]
    restrict_subset_Sigma[of p z lambda. 2 r] function_space_rel_char
    by (auto simp:Pi_def)
    moreover from  $\langle M(p) \rangle \langle M(r) \rangle$ 
    have  $M(\text{restrict}(p, r))$  by simp
    moreover
    note  $\langle M(r) \rangle$ 
    ultimately
    have  $\exists p \in A. \text{restrict}(p, r) \in \text{Pow\_rel}(M, r \times \mathbb{2}) \wedge \text{domain}(p) = z$ 
    using Pow_rel_char by auto
  }
then
show ?thesis
  by (intro equalityI) (force)+
qed

```

```

from  $\langle M(D) \rangle \langle M(r) \rangle$ 
have  $M(\{y . p \in A, \text{restrict}(p, r) = f \wedge y = \text{domain}(p) \wedge \text{domain}(p) \in D\})$  (is
 $M(?Y(A, f))$ )
  if  $M(f) M(A)$  for  $f A$ 
  using  $\text{drSR\_Y\_closed}[\text{unfolded drSR\_Y\_def}]$  that
  by simp
then
obtain  $f$  where  $\text{uncountable\_rel}(M, ?Y(A, f)) M(f)$ 
proof -
  have  $1: M(i)$ 
  if  $M(B) M(x)$ 
     $x \in \{y . x \in B, \text{restrict}(x, r) = i \wedge y = \text{domain}(x) \wedge \text{domain}(x) \in D\}$ 
  for  $B x i$ 
  using that  $\langle M(r) \rangle$ 
  by (auto dest:transM)
  note  $\langle M(r) \rangle$ 
  moreover from this
  have  $M(\text{Pow}^M(r \times 2))$  by simp
  moreover
  note  $\langle M(A) \rangle \langle \bigwedge f A. M(f) \implies M(A) \implies M(?Y(A, f)) \rangle \langle M(D) \rangle$ 
  moreover from calculation
  interpret  $M\_replacement\_lepoll M \text{drSR\_Y}(r, D)$ 
  using  $\text{countable\_lepoll\_assms3 lam\_replacement\_inj\_rel lam\_replacement\_drSR\_Y}$ 
 $\text{drSR\_Y\_closed lam\_Least\_assumption\_drSR\_Y lam\_replacement\_minimum}$ 
  by (unfold locales, simp_all add: drSR\_Y\_def, rule_tac 1, simp_all)
  from calculation
  have  $x \in \text{Pow}^M(r \times 2) \implies M(\text{drSR\_Y}(r, D, A, x))$  for  $x$ 
  unfolding  $\text{drSR\_Y\_def}$  by (auto dest:transM)
  ultimately
  interpret  $M\_cardinal\_UN\_lepoll \_ ?Y(A) \text{Pow\_rel}(M, r \times 2)$ 
  using  $\text{countable\_lepoll\_assms3 lam\_replacement\_drSR\_Y}$ 
 $\text{lepoll\_assumptions}[\text{where } S = \text{Pow\_rel}(M, r \times 2), \text{unfolded lepoll\_assumptions\_defs}]$ 
 $\text{lam\_Least\_assumption\_drSR\_Y}[\text{unfolded drSR\_Y\_def}] \text{lam\_replacement\_minimum}$ 
  unfolding  $\text{drSR\_Y\_def}$ 
  apply unfold locales
  apply (simp_all add: lam\_replacement\_inj\_rel del: if\_range\_F\_else\_F\_def, rule_tac
 $1, \text{simp\_all})$ 
  by ((fastforce dest:transM[OF _  $\langle M(A) \rangle$ ]) +)
  {
  from  $\langle \text{Finite}(r) \rangle \langle M(r) \rangle$ 
  have  $\text{countable\_rel}(M, \text{Pow\_rel}(M, r \times 2))$ 
  using  $\text{Finite\_Sigma}[\text{THEN Finite\_Pow\_rel}] \text{Finite\_imp\_countable\_rel}$ 
  by simp
  moreover
  assume  $M(f) \implies \text{countable\_rel}(M, ?Y(A, f))$  for  $f$ 
  moreover
  note  $\langle D = (\bigcup f \in \text{Pow\_rel}(M, r \times 2) . ?Y(A, f)) \rangle \langle M(r) \rangle$ 
  moreover
  note  $\langle \text{uncountable\_rel}(M, D) \rangle$ 

```

```

    ultimately
    have False
      using countable_rel_imp_countable_rel_UN by (auto dest: transM)
  }
  with that
  show ?thesis
    by auto
qed
moreover from this ⟨M(A)⟩ and ⟨M(f) ⟹ M(A) ⟹ M(?Y(A,f))⟩
have M(?Y(A,f))
  by blast
ultimately
obtain j where j ∈ inj_rel(M,nat, ?Y(A,f)) M(j)
  using uncountable_rel_iff_nat_lt_cardinal_rel[THEN iffD1, THEN leI,
    THEN cardinal_rel_le_imp_lepoll_rel, THEN lepoll_relD]
  by auto
with ⟨M(?Y(A,f))⟩
have j'0 ≠ j'1 j'0 ∈ ?Y(A,f) j'1 ∈ ?Y(A,f)
  using inj_is_fun[THEN apply_type, of j nat ?Y(A,f)]
  inj_rel_char
  unfolding inj_def by auto
then
obtain p q where domain(p) ≠ domain(q) p ∈ A q ∈ A
  domain(p) ∈ D domain(q) ∈ D
  restrict(p,r) = restrict(q,r) by auto
moreover from this and delta
have domain(p) ∩ domain(q) = r
  unfolding r_def by simp
moreover
note ⟨A ⊆ Fn(nat, I, 2)⟩ Fn_nat_abs[OF ⟨M(I)⟩ nat_into_M[of 2],simplified]
moreover from calculation
have p ∈ FnM(nat, I, 2) q ∈ FnM(nat, I, 2)
  by auto
moreover from calculation
have p ∪ q ∈ Fn(nat, I, 2)
  using restrict_eq_imp_compat_rel InfCard_rel_nat
  by simp
ultimately
have ∃ p ∈ A. ∃ q ∈ A. p ≠ q ∧ compat_in(Fn(nat, I, 2), Fnle(nat, I, 2), p, q)
  unfolding compat_in_def
  by (rule_tac bexI[of _ p], rule_tac bexI[of _ q]) blast
}
moreover from assms
have M(Fnle(ω,I,2))
  by simp
moreover note ⟨M(Fn(ω,I,2))⟩
ultimately
show ?thesis using def_ccc_rel by (auto simp:absolut antichain_def) fastforce
qed

```

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

using *subsetD[OF mem_eclose_subset]*

by *simp*

lemma *ecloseE* :

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

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

using *assms*

proof (*induct rule:eclose_induct_down*)

case (1 *y*)

then

show *?case*

using *arg_into_eclose* **by** *auto*

next

case (2 *y z*)

from $\langle y \in A \vee (\exists B \in A. y \in \text{eclose}(B)) \rangle$

consider (*inA*) $y \in A \mid$ (*exB*) $(\exists B \in A. y \in \text{eclose}(B))$

by *auto*

then show *?case*

proof (*cases*)

case *inA*

then

show *?thesis* **using** 2 *arg_into_eclose* **by** *auto*

next

case *exB*

then obtain *B* **where** $y \in \text{eclose}(B)$ $B \in A$

by *auto*

then

show *?thesis* **using** 2 *ecloseD[of y B z]* **by** *auto*

qed

qed

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

by (*blast dest: ecloseE*)

lemma *in_eclose_sing* :

assumes $x \in \text{eclose}(\{a\})$ $a \in \text{eclose}(z)$
shows $x \in \text{eclose}(\{z\})$
proof -
from $\langle x \in \text{eclose}(\{a\}) \rangle$
consider $x=a \mid x \in \text{eclose}(a)$
using *eclose_singE* **by** *auto*
then
show *?thesis*
using *eclose_sing mem_eclose_trans assms*
by (*cases, auto*)
qed

lemma *in_dom_in_eclose* :
assumes $x \in \text{domain}(z)$
shows $x \in \text{eclose}(z)$
proof -
from *assms*
obtain y **where** $\langle x,y \rangle \in z$
unfolding *domain_def* **by** *auto*
then
show *?thesis*
unfolding *Pair_def*
using *ecloseD[of {x,x}] ecloseD[of {{x,x},{x,y}}] arg_into_eclose*
by *auto*
qed

termed 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!]*: $t \in \text{domain}(x) \implies \text{ed}(t,x)$
unfolding *ed_def* .

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

lemma *rank_ed*:
assumes $\text{ed}(y,x)$
shows $\text{succ}(\text{rank}(y)) \leq \text{rank}(x)$
proof
from *assms*
obtain p **where** $\langle y,p \rangle \in x$ **by** *auto*
moreover
obtain s **where** $y \in s$ $s \in \langle y,p \rangle$ **unfolding** *Pair_def* **by** *auto*
ultimately
have $\text{rank}(y) < \text{rank}(s)$ $\text{rank}(s) < \text{rank}(\langle y,p \rangle)$ $\text{rank}(\langle y,p \rangle) < \text{rank}(x)$

```

    using rank_lt by blast+
  then
  show rank(y) < rank(x)
    using lt_trans by blast
qed

```

```

lemma edrel_dest [dest]: x ∈ edrel(A) ⇒ ∃ a ∈ A. ∃ b ∈ A. x = ⟨a,b⟩
  by(auto simp add:ed_def edrel_def Rrel_def)

```

```

lemma edrelD : x ∈ edrel(A) ⇒ ∃ a ∈ A. ∃ b ∈ A. x = ⟨a,b⟩ ∧ a ∈ domain(b)
  by(auto simp add:ed_def edrel_def Rrel_def)

```

```

lemma edrelI [intro!]: x ∈ A ⇒ y ∈ A ⇒ x ∈ domain(y) ⇒ ⟨x,y⟩ ∈ edrel(A)
  by (simp add:ed_def edrel_def Rrel_def)

```

```

lemma edrel_trans: Transset(A) ⇒ y ∈ A ⇒ x ∈ domain(y) ⇒ ⟨x,y⟩ ∈ edrel(A)
  by (rule edrelI, auto simp add:Transset_def domain_def Pair_def)

```

```

lemma domain_trans: Transset(A) ⇒ y ∈ A ⇒ x ∈ domain(y) ⇒ x ∈ A
  by (auto simp add: Transset_def domain_def Pair_def)

```

```

lemma relation_edrel : relation(edrel(A))
  by(auto simp add: relation_def)

```

```

lemma field_edrel : field(edrel(A)) ⊆ A
  by blast

```

```

lemma edrel_sub_memrel: edrel(A) ⊆ trancl(Memrel(eclose(A)))

```

```

proof

```

```

  let

```

```

    ?r=trancl(Memrel(eclose(A)))

```

```

  fix z

```

```

  assume z ∈ edrel(A)

```

```

  then

```

```

    obtain x y where x ∈ A y ∈ A z = ⟨x,y⟩ x ∈ domain(y)

```

```

      using edrelD

```

```

      by blast

```

```

  moreover from this

```

```

    obtain u v where x ∈ u u ∈ v v ∈ y

```

```

      unfolding domain_def Pair_def by auto

```

```

  moreover from calculation

```

```

    have x ∈ eclose(A) y ∈ eclose(A) y ⊆ eclose(A)

```

```

      using arg_into_eclose Transset_eclose[unfolded Transset_def]

```

```

      by simp_all

```

```

  moreover from calculation

```

```

    have v ∈ eclose(A)

```

```

      by auto

```

```

  moreover from calculation

```

```

    have u ∈ eclose(A)

```

```

    using Transset_eclose[unfolded Transset_def]
    by auto
  moreover from calculation
  have  $\langle x, u \rangle \in ?r \ \langle u, v \rangle \in ?r \ \langle v, y \rangle \in ?r$ 
    by (auto simp add: r_into_trancl)
  moreover from this
  have  $\langle x, y \rangle \in ?r$ 
    using trancl_trans[OF trancl_trans[of v y]]
    by simp
  ultimately
  show  $z \in ?r$ 
    by simp
qed

lemma wf_edrel : wf(edrel(A))
  using wf_subset[of trancl(Memrel(eclose(A)))] edrel_sub_memrel
  wf_trancl wf_Memrel
  by auto

lemma ed_induction:
  assumes  $\bigwedge x. [\bigwedge y. \text{ed}(y, x) \implies Q(y)] \implies Q(x)$ 
  shows  $Q(a)$ 
proof(induct rule: wf_induct2[OF wf_edrel[of eclose({a})], of a eclose({a})])
  case 1
  then show ?case using arg_into_eclose by simp
next
  case 2
  then show ?case using field_edrel .
next
  case (3 x)
  then
  show ?case
    using asms[of x] edrelI domain_trans[OF Transset_eclose 3(1)] by blast
qed

lemma dom_under_edrel_eclose:  $\text{edrel}(\text{eclose}(\{x\})) -'' \{x\} = \text{domain}(x)$ 
proof(intro equalityI)
  show  $\text{edrel}(\text{eclose}(\{x\})) -'' \{x\} \subseteq \text{domain}(x)$ 
    unfolding edrel_def Rrel_def ed_def
    by auto
next
  show  $\text{domain}(x) \subseteq \text{edrel}(\text{eclose}(\{x\})) -'' \{x\}$ 
    unfolding edrel_def Rrel_def
    using in_dom_in_eclose eclose_sing arg_into_eclose
    by blast
qed

lemma ed_eclose :  $\langle y, z \rangle \in \text{edrel}(A) \implies y \in \text{eclose}(z)$ 
  by (drule edrelD, auto simp add: domain_def in_dom_in_eclose)

```

lemma *tr_edrel_eclose* : $\langle y, z \rangle \in \text{edrel}(\text{eclose}(\{x\}))^+ \implies y \in \text{eclose}(z)$
by (*rule trancl_induct*, (*simp add: ed_eclose mem_eclose_trans*)+)

lemma *restrict_edrel_eq* :
assumes $z \in \text{domain}(x)$
shows $\text{edrel}(\text{eclose}(\{x\})) \cap \text{eclose}(\{z\}) \times \text{eclose}(\{z\}) = \text{edrel}(\text{eclose}(\{z\}))$
proof (*intro equalityI subsetI*)
let $?ec = \lambda y . \text{edrel}(\text{eclose}(\{y\}))$
let $?ez = \text{eclose}(\{z\})$
let $?rr = ?ec(x) \cap ?ez \times ?ez$
fix y
assume $y \in ?rr$
then
obtain $a\ b$ **where** $\langle a, b \rangle \in ?rr$ $a \in ?ez$ $b \in ?ez$ $\langle a, b \rangle \in ?ec(x)$ $y = \langle a, b \rangle$
by *blast*
moreover from this
have $a \in \text{domain}(b)$
using *edrelD* **by** *blast*
ultimately
show $y \in \text{edrel}(\text{eclose}(\{z\}))$
by *blast*
next
let $?ec = \lambda y . \text{edrel}(\text{eclose}(\{y\}))$
let $?ez = \text{eclose}(\{z\})$
let $?rr = ?ec(x) \cap ?ez \times ?ez$
fix y
assume $y \in \text{edrel}(\text{eclose}(\{z\}))$
then
obtain $a\ b$ **where** $a \in ?ez$ $b \in ?ez$ $y = \langle a, b \rangle$ $a \in \text{domain}(b)$
using *edrelD* **by** *blast*
moreover
from this assms
have $z \in \text{eclose}(x)$
using *in_dom_in_eclose* **by** *simp*
moreover
from assms calculation
have $a \in \text{eclose}(\{x\})$ $b \in \text{eclose}(\{x\})$
using *in_eclose_sing* **by** *simp_all*
moreover from calculation
have $\langle a, b \rangle \in \text{edrel}(\text{eclose}(\{x\}))$
by *blast*
ultimately
show $y \in ?rr$
by *simp*
qed

lemma *tr_edrel_subset* :
assumes $z \in \text{domain}(x)$

```

shows tr_down(edrel(eclose({x})),z) ⊆ eclose({z})
proof(intro subsetI)
  let ?r=λ x . edrel(eclose({x}))
  fix y
  assume y ∈ tr_down(?r(x),z)
  then
  have ⟨y,z⟩ ∈ ?r(x)+
    using tr_downD by simp
  with assms
  show y ∈ eclose({z})
    using tr_edrel_eclose eclose_sing by simp
qed

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⇒i where
  ftype ≡ fst

```

```

definition
  name1 :: i⇒i where
  name1(x) ≡ fst(snd(x))

```

```

definition
  name2 :: i⇒i where
  name2(x) ≡ fst(snd(snd(x)))

```

```

definition
  cond_of :: i⇒i where
  cond_of(x) ≡ snd(snd(snd((x))))

```

```

lemma components_simp:
  ftype(⟨f,n1,n2,c⟩) = f
  name1(⟨f,n1,n2,c⟩) = n1
  name2(⟨f,n1,n2,c⟩) = n2
  cond_of(⟨f,n1,n2,c⟩) = c
unfolding ftype_def name1_def name2_def cond_of_def
by simp_all

```

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

definition

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

lemma $components_in_eclose :$

$n1 \in ecloseN(\langle f, n1, n2, c \rangle)$
 $n2 \in ecloseN(\langle f, n1, n2, c \rangle)$
unfolding $ecloseN_def$ $eclose_n_def$
using $components_simp$ arg_into_eclose **by** $auto$

lemmas $names_simp = components_simp(2)$ $components_simp(3)$

lemma $ecloseNI1 :$

assumes $x \in eclose(n1) \vee x \in eclose(n2)$
shows $x \in ecloseN(\langle f, n1, n2, c \rangle)$
unfolding $ecloseN_def$ $eclose_n_def$
using $assms$ $eclose_sing$ $names_simp$
by $auto$

lemmas $ecloseNI = ecloseNI1$

lemma $ecloseN_mono :$

assumes $u \in ecloseN(x)$ $name1(x) \in ecloseN(y)$ $name2(x) \in ecloseN(y)$
shows $u \in ecloseN(y)$

proof -

from $\langle u \in _ \rangle$
consider $(a) u \in eclose(\{name1(x)\}) \mid (b) u \in eclose(\{name2(x)\})$
unfolding $ecloseN_def$ $eclose_n_def$ **by** $auto$
then
show $?thesis$
proof $cases$
case a
with $\langle name1(x) \in _ \rangle$
show $?thesis$
unfolding $ecloseN_def$ $eclose_n_def$
using $eclose_singE[OF a]$ $mem_eclose_trans[of u name1(x)]$ **by** $auto$
next
case b
with $\langle name2(x) \in _ \rangle$
show $?thesis$
unfolding $ecloseN_def$ $eclose_n_def$
using $eclose_singE[OF b]$ $mem_eclose_trans[of u name2(x)]$ **by** $auto$

qed

qed

definition

$is_ftype :: (i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$ **where**
 $is_ftype \equiv is_fst$

definition

$ftype_fm :: [i, i] \Rightarrow i$ **where**
 $ftype_fm \equiv fst_fm$

lemma $is_ftype_iff_sats$ [iff_sats]:

assumes

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

shows

$is_ftype(\#\#A, x, y) \longleftrightarrow sats(A, ftype_fm(a, b), env)$

unfolding $ftype_fm_def$ is_ftype_def

using $assms$ $sats_fst_fm$

by $simp$

definition

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

definition

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

lemma $sats_name1_fm$ [$simp$]:

$\llbracket x \in nat; y \in nat; env \in list(A) \rrbracket \Longrightarrow$

$(A, env \models name1_fm(x, y)) \longleftrightarrow is_name1(\#\#A, nth(x, env), nth(y, env))$

unfolding $name1_fm_def$ is_name1_def

using $sats_fst_fm$ $sats_snd_fm$ $sats_hcomp_fm$ [$of \ A \ is_fst(\#\#A) \ _ \ fst_fm$
 $is_snd(\#\#A)$]

by $simp$

lemma $is_name1_iff_sats$ [iff_sats]:

assumes

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

shows

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

using $assms$ $sats_name1_fm$

by $simp$

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**
 $snd_snd_fm(x, t) \equiv hcomp_fm(snd_fm, snd_fm, x, t)$

lemma $sats_snd2_fm$ [$simp$]:

$\llbracket x \in \text{nat}; y \in \text{nat}; \text{env} \in \text{list}(A) \rrbracket \implies$
 $(A, \text{env} \models \text{snd_snd_fm}(x,y)) \longleftrightarrow \text{is_snd_snd}(\#\#A, \text{nth}(x,\text{env}), \text{nth}(y,\text{env}))$
unfolding snd_snd_fm_def is_snd_snd_def
using sats_snd_fm sats_hcomp_fm [of A $\text{is_snd}(\#\#A)$ $_ \text{snd_fm}$ $\text{is_snd}(\#\#A)$]
by simp

definition

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

definition

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

lemma sats_name2_fm :

$\llbracket x \in \text{nat}; y \in \text{nat}; \text{env} \in \text{list}(A) \rrbracket \implies$
 $(A, \text{env} \models \text{name2_fm}(x,y)) \longleftrightarrow \text{is_name2}(\#\#A, \text{nth}(x,\text{env}), \text{nth}(y,\text{env}))$
unfolding name2_fm_def is_name2_def
using sats_fst_fm sats_snd2_fm sats_hcomp_fm [of A $\text{is_fst}(\#\#A)$ $_ \text{fst_fm}$ $\text{is_snd_snd}(\#\#A)$]
by simp

lemma is_name2_iff_sats [iff_sats]:

assumes
 $\text{nth}(a,\text{env}) = x$ $\text{nth}(b,\text{env}) = y$ $a \in \text{nat}$ $b \in \text{nat}$ $\text{env} \in \text{list}(A)$
shows
 $\text{is_name2}(\#\#A, x, y) \longleftrightarrow A, \text{env} \models \text{name2_fm}(a, b)$
using assms sats_name2_fm
by simp

definition

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

definition

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

lemma sats_cond_of_fm :

$\llbracket x \in \text{nat}; y \in \text{nat}; \text{env} \in \text{list}(A) \rrbracket \implies$
 $(A, \text{env} \models \text{cond_of_fm}(x,y)) \longleftrightarrow \text{is_cond_of}(\#\#A, \text{nth}(x,\text{env}), \text{nth}(y,\text{env}))$
unfolding cond_of_fm_def is_cond_of_def
using sats_snd_fm sats_snd2_fm sats_hcomp_fm [of A $\text{is_snd}(\#\#A)$ $_ \text{snd_fm}$ $\text{is_snd_snd}(\#\#A)$]
by simp

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

assumes
 $\text{nth}(a,\text{env}) = x$ $\text{nth}(b,\text{env}) = y$ $a \in \text{nat}$ $b \in \text{nat}$ $\text{env} \in \text{list}(A)$

shows
 $is_cond_of(\#\#A,x,y) \leftrightarrow A, env \models cond_of_fm(a,b)$
using *assms sats_cond_of_fm*
by *simp*

lemma *components_type[TC]* :
assumes $a \in nat$ $b \in nat$
shows
 $f_type_fm(a,b) \in formula$
 $name1_fm(a,b) \in formula$
 $name2_fm(a,b) \in formula$
 $cond_of_fm(a,b) \in formula$
using *assms*
unfolding *f_type_fm_def fst_fm_def snd_fm_def snd_snd_fm_def name1_fm_def name2_fm_def cond_of_fm_def hcomp_fm_def*
by *simp_all*

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] :
 $\llbracket en \in nat ; t \in nat \rrbracket \implies ecloseN_fm(en,t) \in formula$
unfolding *ecloseN_fm_def* *eclose_n1_fm_def* *eclose_n2_fm_def* **by** *simp*

lemma *sats_ecloseN_fm* [*simp*]:
 $\llbracket en \in nat; t \in nat ; env \in list(A) \rrbracket$
 $\implies (A, env \models ecloseN_fm(en,t)) \longleftrightarrow is_ecloseN(\#\#A,nth(t,env),nth(en,env))$
unfolding *ecloseN_fm_def* *is_ecloseN_def* *eclose_n1_fm_def* *eclose_n2_fm_def*
is_eclose_n_def
using *nth_0* *nth_ConsI* *sats_name1_fm* *sats_name2_fm* *singleton_iff_sats*[*symmetric*]
by *auto*

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

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

lemma *frecR_ftypeD* :
assumes *frecR*(x,y)
shows $(ftype(x) = 0 \wedge ftype(y) = 1) \vee (ftype(x) = 1 \wedge ftype(y) = 0)$
using *assms* **unfolding** *frecR_def* **by** *auto*

lemma *frecRI1*: $s \in domain(n1) \vee s \in domain(n2) \implies frecR(\langle 1, s, n1, q \rangle, \langle 0, n1,$
 $n2, q \rangle)$
unfolding *frecR_def* **by** (*simp add:components_simp*)

lemma *frecRI1'*: $s \in domain(n1) \cup domain(n2) \implies frecR(\langle 1, s, n1, q \rangle, \langle 0, n1,$
 $n2, q \rangle)$
unfolding *frecR_def* **by** (*simp add:components_simp*)

lemma *frecRI2*: $s \in domain(n1) \vee s \in domain(n2) \implies frecR(\langle 1, s, n2, q \rangle, \langle 0,$
 $n1, n2, q \rangle)$
unfolding *frecR_def* **by** (*simp add:components_simp*)

lemma *frecRI2'*: $s \in domain(n1) \cup domain(n2) \implies frecR(\langle 1, s, n2, q \rangle, \langle 0, n1,$
 $n2, q \rangle)$
unfolding *frecR_def* **by** (*simp add:components_simp*)

lemma *frecRI3*: $\langle s, r \rangle \in n2 \implies \text{frecR}(\langle 0, n1, s, q \rangle, \langle 1, n1, n2, q \rangle)$
unfolding *frecR_def* **by** (*auto simp add:components_simp*)

lemma *frecRI3'*: $s \in \text{domain}(n2) \implies \text{frecR}(\langle 0, n1, s, q \rangle, \langle 1, n1, n2, q \rangle)$
unfolding *frecR_def* **by** (*auto simp add:components_simp*)

lemma *frecR_D1* :
 $\text{frecR}(x,y) \implies \text{ftype}(y) = 0 \implies \text{ftype}(x) = 1 \wedge$
 $(\text{name1}(x) \in \text{domain}(\text{name1}(y)) \cup \text{domain}(\text{name2}(y)) \wedge (\text{name2}(x) = \text{name1}(y)$
 $\vee \text{name2}(x) = \text{name2}(y)))$
unfolding *frecR_def*
by *auto*

lemma *frecR_D2* :
 $\text{frecR}(x,y) \implies \text{ftype}(y) = 1 \implies \text{ftype}(x) = 0 \wedge$
 $\text{ftype}(x) = 0 \wedge \text{ftype}(y) = 1 \wedge \text{name1}(x) = \text{name1}(y) \wedge \text{name2}(x) \in$
 $\text{domain}(\text{name2}(y))$
unfolding *frecR_def*
by *auto*

lemma *frecR_DI* :
assumes $\text{frecR}(\langle a,b,c,d \rangle, \langle \text{ftype}(y), \text{name1}(y), \text{name2}(y), \text{cond_of}(y) \rangle)$
shows $\text{frecR}(\langle a,b,c,d \rangle, y)$
using *assms*
unfolding *frecR_def*
by (*force simp add:components_simp*)

reldb_add *ftype is_ftype*
reldb_add *name1 is_name1*
reldb_add *name2 is_name2*

relativize *frecR is_frecR*

schematic_goal *sats_frecR_fm_auto*:
assumes
 $i \in \text{nat } j \in \text{nat } \text{env} \in \text{list}(A)$
shows
 $\text{is_frecR}(\#\#A, \text{nth}(i, \text{env}), \text{nth}(j, \text{env})) \longleftrightarrow A, \text{env} \models \text{?fr_fm}(i, j)$
unfolding *is_frecR_def*
by (*insert assms ; (rule sep_rules' cartprod_iff_sats components_iff_sats*
 $| \text{simp del:sats_cartprod_fm})+$)

synthesize *frecR from_schematic sats_frecR_fm_auto*

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

lemma *eq_ftypep_not_frecR*:
assumes $\text{ftype}(x) = \text{ftype}(y)$
shows $\neg \text{frecR}(x, y)$
using *assms frecR_ftypeD* **by** *force*

definition

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

lemma $rank_names_types$ [TC]:

shows $Ord(rank_names(x))$
unfolding $rank_names_def$ max_def **using** Ord_rank Ord_Un **by** $auto$

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**
 $type_form(x) \equiv if\ ftype(x) = 0\ then\ 1\ else\ mtype_form(x)$

lemma $type_form_tc$ [TC]:

shows $type_form(x) \in \mathbb{3}$
unfolding $type_form_def$ $mtype_form_def$ **by** $auto$

lemma $frecR_le_rnk_names$:

assumes $frecR(x,y)$
shows $rank_names(x) \leq rank_names(y)$

proof -

obtain $a\ b\ c\ d$ **where**

$H: a = name1(x)\ b = name2(x)$
 $c = name1(y)\ d = name2(y)$
 $(a \in domain(c) \cup domain(d) \wedge (b=c \vee b = d)) \vee (a = c \wedge b \in domain(d))$

using $assms$

unfolding $frecR_def$

by $force$

then

consider

$(m)\ a \in domain(c) \wedge (b = c \vee b = d)$
 $| (n)\ a \in domain(d) \wedge (b = c \vee b = d)$
 $| (o)\ b \in domain(d) \wedge a = c$

by $auto$

then

show $?thesis$

proof($cases$)

case m

then

have $rank(a) < rank(c)$

using $eclose_rank_lt\ in_dom_in_eclose$

by $simp$

with $\langle rank(a) < rank(c) \rangle\ H\ m$

show $?thesis$

unfolding $rank_names_def$

```

    using Ord_rank max_cong max_cong2 leI
    by auto
next
case n
then
have rank(a) < rank(d)
  using eclose_rank_lt in_dom_in_eclose
  by simp
with ⟨rank(a) < rank(d)⟩ H n
show ?thesis
  unfolding rank_names_def
  using Ord_rank max_cong2 max_cong max_commutes[of rank(c) rank(d)]
leI
  by auto
next
case o
then
have rank(b) < rank(d) (is ?b < ?d) rank(a) = rank(c) (is ?a = _)
  using eclose_rank_lt in_dom_in_eclose
  by simp_all
with H
show ?thesis
  unfolding rank_names_def
  using Ord_rank max_commutes max_cong2[OF leI[OF ⟨?b < ?d⟩], of ?a]
  by simp
qed
qed

```

definition

```

Γ :: i ⇒ i where
Γ(x) = 3 ** rank_names(x) ++ type_form(x)

```

```

lemma Γ_type [TC]:
shows Ord(Γ(x))
  unfolding Γ_def by simp

```

```

lemma Γ_mono :

```

```

  assumes frecR(x,y)
  shows Γ(x) < Γ(y)

```

```

proof -

```

```

  have F: type_form(x) < 3 type_form(y) < 3
    using ltI
    by simp_all

```

```

from assms

```

```

have A: rank_names(x) ≤ rank_names(y) (is ?x ≤ ?y)
  using frecR_le_rnk_names
  by simp

```

```

then

```

```

have Ord(?y)

```

```

    unfolding rank_names_def
    using Ord_rank_max_def
    by simp
note leE[OF ‹?x ≤ ?y›]
then
show ?thesis
proof(cases)
  case 1
  then
  show ?thesis
  unfolding Γ_def
  using oadd_lt_mono2 ‹?x < ?y› F
  by auto
next
case 2
consider (a) ftype(x) = 0 ∧ ftype(y) = 1 | (b) ftype(x) = 1 ∧ ftype(y) = 0
  using frecR_ftypeD[OF ‹frecR(x,y)›]
  by auto
then show ?thesis
proof(cases)
  case b
  moreover from this
  have type_form(y) = 1
    using type_form_def by simp
  moreover from calculation
  have name2(x) = name1(y) ∨ name2(x) = name2(y) (is ?τ = ?σ' ∨ ?τ =
?τ')
    name1(x) ∈ domain(name1(y)) ∪ domain(name2(y)) (is ?σ ∈ domain(?σ')
∪ domain(?τ'))
    using assms unfolding type_form_def frecR_def by auto
  moreover from calculation
  have E: rank(?τ) = rank(?σ') ∨ rank(?τ) = rank(?τ') by auto
  from calculation
  consider (c) rank(?σ) < rank(?σ') | (d) rank(?σ) < rank(?τ')
    using eclose_rank_lt_in_dom_in_eclose by force
  then
  have rank(?σ) < rank(?τ)
  proof (cases)
    case c
    with ‹rank_names(x) = rank_names(y) ›
    show ?thesis
    unfolding rank_names_def mtype_form_def type_form_def
    using max_D2[OF E c] E assms Ord_rank
    by simp
  next
  case d
  with ‹rank_names(x) = rank_names(y) ›
  show ?thesis
  unfolding rank_names_def mtype_form_def type_form_def

```

```

    using max_D2[OF _ d] max_commutes E assms Ord_rank disj_commute
      by simp
  qed
  with b
  have type_form(x) = 0 unfolding type_form_def mtype_form_def by simp
  with ⟨rank_names(x) = rank_names(y)⟩ ⟨type_form(y) = 1⟩ ⟨type_form(x)
= 0⟩
  show ?thesis
    unfolding Γ_def by auto
  next
  case a
  then
  have name1(x) = name1(y) (is ?σ = ?σ')
    name2(x) ∈ domain(name2(y)) (is ?τ ∈ domain(?τ'))
    type_form(x) = 1
    using assms
    unfolding type_form_def frecR_def
    by auto
  then
  have rank(?σ) = rank(?σ') rank(?τ) < rank(?τ')
    using eclose_rank_lt_in_dom_in_eclose
    by simp_all
  with ⟨rank_names(x) = rank_names(y)⟩
  have rank(?τ') ≤ rank(?σ')
    using Ord_rank max_D1
    unfolding rank_names_def
    by simp
  with a
  have type_form(y) = 2
    unfolding type_form_def mtype_form_def
    using not_lt_iff_le assms
    by simp
  with ⟨rank_names(x) = rank_names(y)⟩ ⟨type_form(y) = 2⟩ ⟨type_form(x)
= 1⟩
  show ?thesis
    unfolding Γ_def by auto
  qed
  qed
  qed

```

definition

```

frecrel :: i ⇒ i where
frecrel(A) ≡ Rrel(frecR,A)

```

lemma frecrelI :

```

assumes x ∈ A y ∈ A frecR(x,y)
shows ⟨x,y⟩ ∈ frecrel(A)
using assms unfolding frecrel_def Rrel_def by auto

```

```

lemma frecrelD :
  assumes  $\langle x,y \rangle \in \text{frecrel}(A1 \times A2 \times A3 \times A4)$ 
  shows
     $\text{ftype}(x) \in A1$   $\text{ftype}(x) \in A1$ 
     $\text{name1}(x) \in A2$   $\text{name1}(y) \in A2$ 
     $\text{name2}(x) \in A3$   $\text{name2}(x) \in A3$ 
     $\text{cond\_of}(x) \in A4$   $\text{cond\_of}(y) \in A4$ 
     $\text{frecrel}(x,y)$ 
  using assms
  unfolding frecrel_def Rrel_def ftype_def by (auto simp add:components_simp)

lemma wf_frecrel :
  shows  $\text{wf}(\text{frecrel}(A))$ 
proof -
  have  $\text{frecrel}(A) \subseteq \text{measure}(A,\Gamma)$ 
    unfolding frecrel_def Rrel_def measure_def
    using  $\Gamma\_mono$ 
    by force
  then
  show ?thesis
    using wf_subset wf_measure by auto
qed

lemma core_induction_aux:
  fixes  $A1 A2 :: i$ 
  assumes
     $\text{Transset}(A1)$ 
     $\bigwedge \tau \vartheta p. p \in A2 \implies [\bigwedge q \sigma. [q \in A2 ; \sigma \in \text{domain}(\vartheta)] \implies Q(0,\tau,\sigma,q)] \implies$ 
 $Q(1,\tau,\vartheta,p)$ 
     $\bigwedge \tau \vartheta p. p \in A2 \implies [\bigwedge q \sigma. [q \in A2 ; \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta)] \implies Q(1,\sigma,\tau,q)$ 
 $\wedge Q(1,\sigma,\vartheta,q)] \implies Q(0,\tau,\vartheta,p)$ 
  shows  $a \in 2 \times A1 \times A1 \times A2 \implies Q(\text{ftype}(a),\text{name1}(a),\text{name2}(a),\text{cond\_of}(a))$ 
proof (induct a rule:wf_induct[OF wf_frecrel[of 2 × A1 × A1 × A2]])
  case ( $1\ x$ )
  let  $? \tau = \text{name1}(x)$ 
  let  $? \vartheta = \text{name2}(x)$ 
  let  $?D = 2 \times A1 \times A1 \times A2$ 
  assume  $x \in ?D$ 
  then
  have  $\text{cond\_of}(x) \in A2$ 
    by (auto simp add:components_simp)
  from  $\langle x \in ?D \rangle$ 
  consider (eq)  $\text{ftype}(x)=0 \mid$  (mem)  $\text{ftype}(x)=1$ 
    by (auto simp add:components_simp)
  then
  show ?case
proof cases
  case eq
  then

```

```

    have  $Q(1, \sigma, ?\tau, q) \wedge Q(1, \sigma, ?\vartheta, q)$  if  $\sigma \in \text{domain}(?\tau) \cup \text{domain}(?\vartheta)$  and
     $q \in A2$  for  $q \sigma$ 
  proof -
    from 1
    have  $?\tau \in A1$   $?\vartheta \in A1$   $?\tau \in \text{eclose}(A1)$   $?\vartheta \in \text{eclose}(A1)$ 
      using arg_into_eclose
      by (auto simp add:components_simp)
    moreover from  $\langle \text{Transset}(A1) \rangle$  that(1)
    have  $\sigma \in \text{eclose}(?\tau) \cup \text{eclose}(?\vartheta)$ 
      using in_dom_in_eclose
      by auto
    then
    have  $\sigma \in A1$ 
      using mem_eclose_subset[OF  $\langle ?\tau \in A1 \rangle$  mem_eclose_subset[OF  $\langle ?\vartheta \in A1 \rangle$ ]
      Transset_eclose_eq_arg[OF  $\langle \text{Transset}(A1) \rangle$ ]
      by auto
    with  $\langle q \in A2 \rangle$   $\langle ?\vartheta \in A1 \rangle$   $\langle \text{cond\_of}(x) \in A2 \rangle$   $\langle ?\tau \in A1 \rangle$ 
    have  $\text{frecR}(\langle 1, \sigma, ?\tau, q \rangle, x)$  (is  $\text{frecR}(?T, \_)$ )
       $\text{frecR}(\langle 1, \sigma, ?\vartheta, q \rangle, x)$  (is  $\text{frecR}(?U, \_)$ )
      using frecRI1'[OF that(1)] frecR_DI  $\langle \text{ftype}(x) = 0 \rangle$ 
      frecRI2'[OF that(1)]
      by (auto simp add:components_simp)
    with  $\langle x \in ?D \rangle$   $\langle \sigma \in A1 \rangle$   $\langle q \in A2 \rangle$ 
    have  $\langle ?T, x \rangle \in \text{frecrel}(?D)$   $\langle ?U, x \rangle \in \text{frecrel}(?D)$ 
      using frecrelI[of ?T ?D x] frecrelI[of ?U ?D x]
      by (auto simp add:components_simp)
    with  $\langle q \in A2 \rangle$   $\langle \sigma \in A1 \rangle$   $\langle ?\tau \in A1 \rangle$   $\langle ?\vartheta \in A1 \rangle$ 
    have  $Q(1, \sigma, ?\tau, q)$ 
      using 1
      by (force simp add:components_simp)
    moreover from  $\langle q \in A2 \rangle$   $\langle \sigma \in A1 \rangle$   $\langle ?\tau \in A1 \rangle$   $\langle ?\vartheta \in A1 \rangle$   $\langle \langle ?U, x \rangle \in \text{frecrel}(?D) \rangle$ 
    have  $Q(1, \sigma, ?\vartheta, q)$ 
      using 1 by (force simp add:components_simp)
    ultimately
    show ?thesis
      by simp
  qed
  with assms(3)  $\langle \text{ftype}(x) = 0 \rangle$   $\langle \text{cond\_of}(x) \in A2 \rangle$ 
  show ?thesis
    by auto
next
case mem
have  $Q(0, ?\tau, \sigma, q)$  if  $\sigma \in \text{domain}(?\vartheta)$  and  $q \in A2$  for  $q \sigma$ 
proof -
  from 1 assms
  have  $?\tau \in A1$   $?\vartheta \in A1$   $\text{cond\_of}(x) \in A2$   $?\tau \in \text{eclose}(A1)$   $?\vartheta \in \text{eclose}(A1)$ 
    using arg_into_eclose
    by (auto simp add:components_simp)
  with  $\langle \text{Transset}(A1) \rangle$  that(1)

```

```

have  $\sigma \in \text{eclose}(\vartheta)$ 
  using in_dom_in_eclose
  by auto
then
  have  $\sigma \in A1$ 
  using mem_eclose_subset[OF  $\langle \vartheta \in A1 \rangle$ ] Transset_eclose_eq_arg[OF  $\langle \text{Transset}(A1) \rangle$ ]
  by auto
  with  $\langle q \in A2 \rangle \langle \vartheta \in A1 \rangle \langle \text{cond\_of}(x) \in A2 \rangle \langle \tau \in A1 \rangle \langle \text{ftype}(x) = 1 \rangle$ 
  have frecR( $\langle 0, \tau, \sigma, q \rangle, x$ ) (is frecR( $?T, \_$ ))
    using frecR13'[OF that(1)] frecR_DI
    by (auto simp add:components_simp)
  with  $\langle x \in ?D \rangle \langle \sigma \in A1 \rangle \langle q \in A2 \rangle \langle \tau \in A1 \rangle$ 
  have  $\langle ?T, x \rangle \in \text{frecrel}(?D)$   $?T \in ?D$ 
    using frecrelI[of ?T ?D x]
    by (auto simp add:components_simp)
  with  $\langle q \in A2 \rangle \langle \sigma \in A1 \rangle \langle \tau \in A1 \rangle \langle \vartheta \in A1 \rangle 1$ 
  show ?thesis
    by (force simp add:components_simp)
qed
with assms(2)  $\langle \text{ftype}(x) = 1 \rangle \langle \text{cond\_of}(x) \in A2 \rangle$ 
show ?thesis
  by auto
qed
qed

```

```

lemma def_frecrel : frecrel(A) = {z ∈ A × A. ∃ x y. z =  $\langle x, y \rangle \wedge \text{frecR}(x, y)$ }
  unfolding frecrel_def Rrel_def ..

```

```

lemma frecrel_fst_snd:
  frecrel(A) = {z ∈ A × A .
    ftype(fst(z)) = 1 ∧
    ftype(snd(z)) = 0 ∧ name1(fst(z)) ∈ domain(name1(snd(z))) ∪ domain(name2(snd(z))) ∧
    (name2(fst(z)) = name1(snd(z)) ∨ name2(fst(z)) = name2(snd(z)))
    ∨ (ftype(fst(z)) = 0 ∧
    ftype(snd(z)) = 1 ∧ name1(fst(z)) = name1(snd(z)) ∧ name2(fst(z)) ∈
    domain(name2(snd(z))))}
  unfolding def_frecrel frecR_def
  by (intro equalityI subsetI CollectI; elim CollectE; auto)

```

```

end
theory FrecR_Arities
  imports
    FrecR
  begin

```

```

context
  notes FOL_arities[simp]

```

begin

arity_theorem intermediate for fst_fm

lemma *arity_fst_fm* [arity] :
[[$x \in \text{nat}$; $t \in \text{nat}$]] \implies $\text{arity}(\text{fst_fm}(x,t)) = \text{succ}(x) \cup \text{succ}(t)$
using *arity_fst_fm'*
by *auto*

arity_theorem intermediate for snd_fm

lemma *arity_snd_fm* [arity] :
[[$x \in \text{nat}$; $t \in \text{nat}$]] \implies $\text{arity}(\text{snd_fm}(x,t)) = \text{succ}(x) \cup \text{succ}(t)$
using *arity_snd_fm'*
by *auto*

lemma *arity_snd_snd_fm* [arity] :
[[$x \in \text{nat}$; $t \in \text{nat}$]] \implies $\text{arity}(\text{snd_snd_fm}(x,t)) = \text{succ}(x) \cup \text{succ}(t)$
unfolding *snd_snd_fm_def hcomp_fm_def*
using *arity_snd_fm arity_empty_fm union_abs2 pred_Un_distrib*
by *auto*

lemma *arity_fstype_fm* [arity] :
[[$x \in \text{nat}$; $t \in \text{nat}$]] \implies $\text{arity}(\text{fstype_fm}(x,t)) = \text{succ}(x) \cup \text{succ}(t)$
unfolding *fstype_fm_def*
using *arity_fst_fm*
by *auto*

lemma *arity_name1_fm* [arity] :
[[$x \in \text{nat}$; $t \in \text{nat}$]] \implies $\text{arity}(\text{name1_fm}(x,t)) = \text{succ}(x) \cup \text{succ}(t)$
unfolding *name1_fm_def hcomp_fm_def*
using *arity_fst_fm arity_snd_fm union_abs2 pred_Un_distrib*
by *auto*

lemma *arity_name2_fm* [arity] :
[[$x \in \text{nat}$; $t \in \text{nat}$]] \implies $\text{arity}(\text{name2_fm}(x,t)) = \text{succ}(x) \cup \text{succ}(t)$
unfolding *name2_fm_def hcomp_fm_def*
using *arity_fst_fm arity_snd_snd_fm union_abs2 pred_Un_distrib*
by *auto*

lemma *arity_cond_of_fm* [arity] :
[[$x \in \text{nat}$; $t \in \text{nat}$]] \implies $\text{arity}(\text{cond_of_fm}(x,t)) = \text{succ}(x) \cup \text{succ}(t)$
unfolding *cond_of_fm_def hcomp_fm_def*
using *arity_snd_fm arity_snd_snd_fm union_abs2 pred_Un_distrib*
by *auto*

lemma *arity_eclose_n1_fm* [arity] :
[[$x \in \text{nat}$; $t \in \text{nat}$]] \implies $\text{arity}(\text{eclose_n1_fm}(x,t)) = \text{succ}(x) \cup \text{succ}(t)$
unfolding *eclose_n1_fm_def*
using *arity_is_eclose_fm arity_singleton_fm arity_name1_fm union_abs2 pred_Un_distrib*
by *auto*

```

lemma arity_eclose_n2_fm [arity] :
   $\llbracket x \in \text{nat} ; t \in \text{nat} \rrbracket \implies \text{arity}(\text{eclose\_n2\_fm}(x,t)) = \text{succ}(x) \cup \text{succ}(t)$ 
  unfolding eclose_n2_fm_def
  using arity_is_eclose_fm arity_singleton_fm arity_name2_fm union_abs2 pred_Un_distrib
  by auto

lemma arity_ecloseN_fm [arity] :
   $\llbracket x \in \text{nat} ; t \in \text{nat} \rrbracket \implies \text{arity}(\text{ecloseN\_fm}(x,t)) = \text{succ}(x) \cup \text{succ}(t)$ 
  unfolding ecloseN_fm_def
  using arity_eclose_n1_fm arity_eclose_n2_fm arity_union_fm union_abs2
  pred_Un_distrib
  by auto

lemma arity_frecl_fm [arity]:
   $\llbracket a \in \text{nat}; b \in \text{nat} \rrbracket \implies \text{arity}(\text{frecl\_fm}(a,b)) = \text{succ}(a) \cup \text{succ}(b)$ 
  unfolding frecl_fm_def
  using arity_ftype_fm arity_name1_fm arity_name2_fm arity_domain_fm
  arity_empty_fm arity_union_fm pred_Un_distrib arity_succ_fm
  by auto

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 ( $\langle \aleph \rangle$  [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 satisfactions 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]
```

```
synthesize setdiff from_definition setdiff assuming nonempty
arity_theorem for setdiff_fm
```

```
synthesize is_converse from_definition assuming nonempty
arity_theorem for is_converse_fm
```

```
relationalize first_rel is_first external
synthesize first_fm from_definition is_first assuming nonempty
```

```
relationalize minimum_rel is_minimum external
```

```
definition is_minimum' where
```

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

```
synthesize minimum from_definition is_minimum' assuming nonempty
arity_theorem for minimum_fm
```

```
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)
```

```
using sats_lambda_fm[OF assms, of A r] by simp
```

— same as *sats_is_wfrec_fm*, but changing length assumptions to *l* being in the

model

lemma *sats_is_wfrec_fm'*:

assumes *MH_iff_sats*:

!!*a0 a1 a2 a3 a4*.

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

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

shows

$[[x \in \text{nat}; y \in \text{nat}; z \in \text{nat}; env \in \text{list}(A); 0 \in A]]$

$\implies \text{sats}(A, \text{is_wfrec_fm}(p, x, y, z), env) \longleftrightarrow$

$\text{is_wfrec}(\#\#A, MH, \text{nth}(x, env), \text{nth}(y, env), \text{nth}(z, env))$

using *MH_iff_sats* [*THEN iff_sym*] *nth_closed* *sats_is_recfun_fm*

by (*simp add: is_wfrec_fm_def is_wfrec_def*) *blast*

lemma *is_wfrec_iff_sats'*[*iff_sats*]:

assumes *MH_iff_sats*:

!!*a0 a1 a2 a3 a4*.

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

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

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

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

shows

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

using *assms(2-4)* *sats_is_wfrec_fm'*[*OF assms(1,5-9)*] **by** *simp*

lemma *is_wfrec_on_iff_sats*[*iff_sats*]:

assumes *MH_iff_sats*:

!!*a0 a1 a2 a3 a4*.

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

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

shows

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

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

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

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

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

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

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

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

using *assms* *sats_is_wfrec_fm'*[*OF assms*] **unfolding** *is_wfrec_on_def* **by** *simp*

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**

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

relativize *rcheck* *is_rcheck*

synthesize *is_rcheck* from_definition
arity_theorem for *is_rcheck_fm*

— The function used for the replacement.

definition

$PHcheck :: [i \Rightarrow o, i, i, i, i] \Rightarrow o$ **where**
 $PHcheck(M, o, f, y, p) \equiv M(p) \wedge (\exists fy[M]. fun_apply(M, f, y, fy) \wedge pair(M, fy, o, p))$

synthesize *PHcheck* from_definition assuming *nonempty*
arity_theorem for *PHcheck_fm*

— The recursive call for check. We could use the meta-program relationalize for this; but it makes some proofs more involved.

definition

$is_Hcheck :: [i \Rightarrow o, i, i, i, i] \Rightarrow o$ **where**
 $is_Hcheck(M, o, z, f, hc) \equiv is_Replace(M, z, PHcheck(M, o, f), hc)$

synthesize *is_Hcheck* from_definition assuming *nonempty*

lemma *arity_is_Hcheck_fm*:

assumes $m \in nat$ $n \in nat$ $p \in nat$ $o \in nat$
shows $arity(is_Hcheck_fm(m, n, p, o)) = succ(o) \cup succ(n) \cup succ(p) \cup succ(m)$
unfolding *is_Hcheck_fm_def*
using *assms arity_Replace_fm[rule_format, OF PHcheck_fm_type _ _ _ arity_PHcheck_fm]*
pred_Un_distrib Un_assoc Un_nat_type
by *simp*

— The relational version of check is hand-made because our automatic tool does not handle *wfrec*.

definition

$is_check :: [i \Rightarrow o, i, i, i] \Rightarrow o$ **where**
 $is_check(M, o, x, z) \equiv \exists rch[M]. is_rcheck(M, x, rch) \wedge is_wfrec(M, is_Hcheck(M, o), rch, x, z)$

— Finally, we internalize the formula.

definition

$check_fm :: [i, i, i] \Rightarrow i$ **where**
 $check_fm(o, x, z) \equiv Exists(And(is_rcheck_fm(1+\omega x, 0), is_wfrec_fm(is_Hcheck_fm(6+\omega o, 2, 1, 0), 0, 1+\omega x, 1+\omega z)))$

lemma *check_fm_type[TC]*: $x \in nat \Longrightarrow o \in nat \Longrightarrow z \in nat \Longrightarrow check_fm(x, o, z) \in formula$

by (*simp add:check_fm_def*)

lemma *sats_check_fm* :

assumes
 $o \in nat$ $x \in nat$ $z \in nat$ $env \in list(M)$ $0 \in M$
shows

$(M, env \models check_fm(o,x,z)) \longleftrightarrow is_check(\#\#M, nth(o,env), nth(x,env), nth(z,env))$
proof -
have *sats_is_Hcheck_fm*:
 $\bigwedge a0\ a1\ a2\ a3\ a4\ a6. \llbracket a0 \in M; a1 \in M; a2 \in M; a3 \in M; a4 \in M; a6 \in M \rrbracket \implies$
 $is_Hcheck(\#\#M, a6, a2, a1, a0) \longleftrightarrow$
 $(M, [a0, a1, a2, a3, a4, r, a6]@env \models is_Hcheck_fm(6, 2, 1, 0))$ **if** $r \in M$ **for** r
using *that assms*
by *simp*
then
have $(M, [r]@env \models is_wfrec_fm(is_Hcheck_fm(6+\omega, 0, 2, 1, 0), 0, 1+\omega x, 1+\omega z))$
 $\longleftrightarrow is_wfrec(\#\#M, is_Hcheck(\#\#M, nth(o,env)), r, nth(x,env), nth(z,env))$
if $r \in M$ **for** r
using *that assms is_wfrec_iff_sats'[symmetric]*
by *simp*
then
show *?thesis*
unfolding *is_check_def check_fm_def*
using *assms is_rcheck_iff_sats[symmetric]*
by *simp*
qed

lemma *iff_sats_check_fm[iff_sats]* :
assumes
 $nth(o, env) = oa\ nth(x, env) = xa\ nth(z, env) = za\ o \in nat\ x \in nat\ z \in nat$
 $env \in list(A)\ 0 \in A$
shows $is_check(\#\#A, oa, xa, za) \longleftrightarrow A, env \models check_fm(o,x,z)$
using *assms sats_check_fm[symmetric]*
by *auto*

lemma *arity_check_fm[arity]*:
assumes $m \in nat\ n \in nat\ o \in nat$
shows $arity(check_fm(m,n,o)) = succ(o) \cup succ(n) \cup succ(m)$
unfolding *check_fm_def*
using *assms arity_is_wfrec_fm[rule_format, OF _____ arity_is_Hcheck_fm]*
 $pred\ Un_distrib\ Un_assoc\ arity_tran_closure_fm$
by *(auto simp add:arity)*

notation $check_fm (\langle _ \rangle^v _ is _ \rangle)$

— The pair of elements belongs to some set. The intended set is the preorder.

definition

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

synthesize *is_leq from_definition* **assuming** *nonempty*

arity_theorem **for** *is_leq_fm*

abbreviation

$fm_leq :: [i, i, i] \Rightarrow i (\langle _ \rangle^{\leq} _ \rangle)$ **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 $f_type : \{\langle 0, 1 \rangle, \langle 1, 0 \rangle\} \in 2 \rightarrow 3$

using Pi_iff **unfolding** $function_def$ **by** $auto$

— $thmInternalize.sum_type$ clashes with $thmRenaming.sum_type$.

hide_fact $Internalize.sum_type$

lemma $ren_type :$

assumes $l \in nat$

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

using sum_type [of $2\ 3\ l\ l\ \{\langle 0, 1 \rangle, \langle 1, 0 \rangle\}\ id(l)\ f_type$ $assms\ id_type$

unfolding ϱ_repl_def **by** $auto$

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

$\in formula$

using ren_tc [of $\varphi\ 2 +_{\omega} len\ 3 +_{\omega} len\ \varrho_repl(len)$] ren_type

unfolding $Lambda_in_M_fm_def$

by $simp$

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

using Pi_iff **unfolding** $function_def$ **by** $auto$

lemma $ren_type' :$

assumes $l \in nat$

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

using sum_type [of $3\ 4\ l\ l\ \{\langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 2, 3 \rangle\}\ id(l)\ f_type'$ $assms\ id_type$

unfolding $\varrho_pair_repl_def$ **by** $auto$

lemma *LambdaPair_in_M_fm_type*[TC]: $\varphi \in \text{formula} \implies \text{len} \in \text{nat} \implies \text{LambdaPair_in_M_fm}(\varphi, \text{len}) \in \text{formula}$
using *ren_tc*[OF *ren_type'*, of φ len] *Lambda_in_M_fm_type*
unfolding *LambdaPair_in_M_fm_def*
by *simp*

5.2 The relation *frecrel*

definition

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

synthesize *frecrel* **from_definition**

arity_theorem for *frecrelP_fm*

definition

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

synthesize *frecrel* **from_definition** *is_frecrel*

arity_theorem for *frecrel_fm*

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**
 $x = \langle f, n1, n2, p \rangle$ $f \in 2$ $n1 \in \text{ecloseN}(z)$ $n2 \in \text{ecloseN}(z)$ $p \in P$
using *assms* **unfolding** *names_below_def* **by** *auto*

synthesize *number2* **from_definition**

lemma *number2_iff* :

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

unfolding *number2_def* *number1_def* **by** *auto*

arity_theorem for *number2_fm*

reldb_add *ecloseN* *is_ecloseN*

relativize *names_below* *is_names_below*

synthesize *is_names_below* **from_definition**

arity_theorem for *is_names_below_fm*

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

$pair(M, z, t1t2p, t)$

synthesize is_tuple **from** **definition**

arity **theorem** **for** is_tuple_fm

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**

$eq_case(\tau, \vartheta, p, P, leq, f) \equiv \forall \sigma. \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \longrightarrow$

$(\forall q. q \in P \wedge \langle q, p \rangle \in leq \longrightarrow (f' \langle 1, \sigma, \tau, q \rangle = 1 \longleftrightarrow f' \langle 1, \sigma, \vartheta, q \rangle = 1))$

relativize eq_case **is** eq_case

synthesize eq_case **from** **definition** is_eq_case

$p \Vdash \tau \in \theta$ 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**

$mem_case(\tau, \vartheta, p, P, leq, f) \equiv \forall v \in P. \langle v, p \rangle \in leq \longrightarrow$

$(\exists q. \exists \sigma. \exists r. r \in P \wedge q \in P \wedge \langle q, v \rangle \in leq \wedge \langle \sigma, r \rangle \in \vartheta \wedge \langle q, r \rangle \in leq \wedge f' \langle 0, \tau, \sigma, q \rangle = 1)$

relativize mem_case **is** mem_case

synthesize mem_case **from** **definition** is_mem_case

arity **theorem** **intermediate** **for** eq_case_fm

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

$arity(eq_case_fm(n1, n2, p, P, leq, f)) =$

$\text{succ}(n1) \cup \text{succ}(n2) \cup \text{succ}(p) \cup \text{succ}(P) \cup \text{succ}(leq) \cup \text{succ}(f)$

using $assms$ $arity_eq_case_fm'$

by $auto$

arity **theorem** **intermediate** **for** mem_case_fm

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

$arity(mem_case_fm(n1, n2, p, P, leq, f)) =$

$\text{succ}(n1) \cup \text{succ}(n2) \cup \text{succ}(p) \cup \text{succ}(P) \cup \text{succ}(leq) \cup \text{succ}(f)$

using $assms$ $arity_mem_case_fm'$

by $auto$

definition

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

relativize $Hfrc$ **is** $Hfrc$

synthesize $Hfrc$ **from_definition** is_Hfrc

definition

$is_Hfrc_at :: [i \Rightarrow o, i, i, i, i] \Rightarrow o$ **where**
 $is_Hfrc_at(M, P, leq, fnnc, f, b) \equiv$
 $(empty(M, b) \wedge \neg is_Hfrc(M, P, leq, fnnc, f))$
 $\vee (number1(M, b) \wedge is_Hfrc(M, P, leq, fnnc, f))$

synthesize $Hfrc_at$ **from_definition** is_Hfrc_at

arity_theorem **intermediate** **for** $Hfrc_fm$

lemma $arity_Hfrc_fm[arity]$:

assumes

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

shows

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

using $assms$ $arity_Hfrc_fm'$

by $auto$

arity_theorem **for** $Hfrc_at_fm$

5.3.2 The well-founded relation *forcerel*

definition

$forcerel :: i \Rightarrow i \Rightarrow i$ **where**
 $forcerel(P, x) \equiv frecrel(names_below(P, x)) \hat{+}$

definition

$is_forcerel :: [i \Rightarrow o, i, i, i] \Rightarrow o$ **where**
 $is_forcerel(M, P, x, z) \equiv \exists r[M]. \exists nb[M]. tran_closure(M, r, z) \wedge$
 $(is_names_below(M, P, x, nb) \wedge is_frecrel(M, nb, r))$

synthesize $is_forcerel$ **from_definition**

arity_theorem **for** $is_forcerel_fm$

5.4 *frc_at*, forcing for atomic formulas

definition

$frc_at :: [i, i, i] \Rightarrow i$ **where**
 $frc_at(P, leq, fnnc) \equiv wfrec(frecrel(names_below(P, fnnc)), fnnc,$
 $\lambda x f. bool_of_o(Hfrc(P, leq, x, f)))$

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

definition

$is_frc_at :: [i \Rightarrow o, i, i, i, i] \Rightarrow o$ **where**
 $is_frc_at(M, P, leq, x, z) \equiv \exists r[M]. is_forcere_l(M, P, x, r) \wedge$
 $is_wfrec(M, is_Hfrc_at(M, P, leq), r, x, z)$

definition

$frc_at_fm :: [i, i, i, i] \Rightarrow i$ **where**
 $frc_at_fm(p, l, x, z) \equiv Exists(And(is_forcere_l_fm(succ(p), succ(x), 0),$
 $is_wfrec_fm(Hfrc_at_fm(6 + \omega p, 6 + \omega l, 2, 1, 0), 0, succ(x), succ(z))))$

lemma $frc_at_fm_type$ $[TC]$:

$\llbracket p \in nat; l \in nat; x \in nat; z \in nat \rrbracket \Longrightarrow frc_at_fm(p, l, x, z) \in formula$
unfolding $frc_at_fm_def$ **by** $simp$

lemma $arity_frc_at_fm$ $[arity]$:

assumes $p \in nat$ $l \in nat$ $x \in nat$ $z \in nat$
shows $arity(frc_at_fm(p, l, x, z)) = succ(p) \cup succ(l) \cup succ(x) \cup succ(z)$

proof -

let $?\varphi = Hfrc_at_fm(6 + \omega p, 6 + \omega l, 2, 1, 0)$

note $assms$

moreover from $this$

have $arity(?\varphi) = (7 + \omega p) \cup (7 + \omega l)$ $?\varphi \in formula$

using $arity_Hfrc_at_fm$ ord_simp_union

by $auto$

moreover from $calculation$

have $arity(is_wfrec_fm(?\varphi, 0, succ(x), succ(z))) = 2 + \omega p \cup (2 + \omega l) \cup (2 + \omega x)$

$\cup (2 + \omega z)$

using $arity_is_wfrec_fm$ $[OF \langle ?\varphi \in _ \rangle _ _ _ _ \langle arity(?\varphi) = _ \rangle]$ $pred_Un_distrib$

$pred_succ_eq$

$union_abs1$

by $auto$

moreover from $assms$

have $arity(is_forcere_l_fm(succ(p), succ(x), 0)) = 2 + \omega p \cup (2 + \omega x)$

using $arity_is_forcere_l_fm$ ord_simp_union

by $auto$

ultimately

show $?thesis$

unfolding $frc_at_fm_def$

using $arity_is_forcere_l_fm$ $pred_Un_distrib$

by $(auto simp:FOL_arities)$

qed

lemma $sats_frc_at_fm$:

assumes

$p \in nat$ $l \in nat$ $i \in nat$ $j \in nat$ $env \in list(A)$ $i < length(env)$ $j < length(env)$

shows

$(A, env \models frc_at_fm(p, l, i, j)) \longleftrightarrow$

$is_frc_at(\#\#A, nth(p, env), nth(l, env), nth(i, env), nth(j, env))$

proof -

```

{
  fix r pp ll
  assume r ∈ A
  have is_Hfrc_at(##A, nth(p, env), nth(l, env), a2, a1, a0) ↔
    (A, [a0, a1, a2, a3, a4, r]@env ⊨ Hfrc_at_fm(6+ωp, 6+ωl, 2, 1, 0))
  if a0 ∈ A a1 ∈ A a2 ∈ A a3 ∈ A a4 ∈ A for a0 a1 a2 a3 a4
  using that assms ⟨r ∈ A⟩
  Hfrc_at_iff_sats[of 6+ωp 6+ωl 2 1 0 [a0, a1, a2, a3, a4, r]@env A] by simp
  with ⟨r ∈ A⟩
  have (A, [r]@env ⊨ is_wfrec_fm(Hfrc_at_fm(6+ωp, 6+ωl, 2, 1, 0), 0, i+ω1,
j+ω1)) ↔
    is_wfrec(##A, is_Hfrc_at(##A, nth(p, env), nth(l, env)), r, nth(i, env),
nth(j, env))
  using assms sats_is_wfrec_fm
  by simp
}
moreover
have (A, Cons(r, env) ⊨ is_forcerel_fm(succ(p), succ(i), 0)) ↔
  is_forcerel(##A, nth(p, env), nth(i, env), r) if r ∈ A for r
  using assms sats_is_forcerel_fm that
  by simp
ultimately
show ?thesis
  unfolding is_frc_at_def frc_at_fm_def
  using assms
  by simp
qed

```

lemma *frc_at_fm_iff_sats*:

```

assumes nth(i, env) = w nth(j, env) = x nth(k, env) = y nth(l, env) = z
  i ∈ nat j ∈ nat k ∈ nat l ∈ nat env ∈ list(A) k < length(env) l < length(env)
shows is_frc_at(##A, w, x, y, z) ↔ (A, env ⊨ frc_at_fm(i, j, k, l))
using assms sats_frc_at_fm
by simp

```

declare *frc_at_fm_iff_sats* [*iff_sats*]

definition

```

forces_eq' :: [i, i, i, i] ⇒ o where
forces_eq'(P, l, p, t1, t2) ≡ frc_at(P, l, ⟨0, t1, t2, p⟩) = 1

```

definition

```

forces_mem' :: [i, i, i, i] ⇒ o where
forces_mem'(P, l, p, t1, t2) ≡ frc_at(P, l, ⟨1, t1, t2, p⟩) = 1

```

definition

```

forces_neq' :: [i, i, i, i] ⇒ o where
forces_neq'(P, l, p, t1, t2) ≡ ¬ (∃ q ∈ P. ⟨q, p⟩ ∈ l ∧ forces_eq'(P, l, q, t1, t2))

```

definition

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

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

definition

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

definition

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

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

synthesize forces_eq from_definition is_forces_eq'

synthesize forces_mem from_definition is_forces_mem'

synthesize forces_neq from_definition is_forces_neq' assuming nonempty

synthesize forces_nmem from_definition is_forces_nmem' assuming nonempty

context

notes $Un_assoc[simp]$ $Un_trasposition_aux2[simp]$

begin

arity_theorem for $forces_eq_fm$

arity_theorem for $forces_mem_fm$

arity_theorem for $forces_neq_fm$

arity_theorem for $forces_nmem_fm$

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 $ren_forces_nand_type[TC]$:

$\varphi \in formula \Longrightarrow ren_forces_nand(\varphi) \in formula$

unfolding $ren_forces_nand_def$

```

by simp

lemma arity_ren_forces_nand :
  assumes  $\varphi \in \text{formula}$ 
  shows  $\text{arity}(\text{ren\_forces\_nand}(\varphi)) \leq \text{succ}(\text{arity}(\varphi))$ 
proof -
  consider (lt)  $1 < \text{arity}(\varphi) \mid (\text{ge}) \neg 1 < \text{arity}(\varphi)$ 
  by auto
  then
  show ?thesis
  proof cases
    case lt
    with  $\langle \varphi \in \_ \rangle$ 
    have  $2 < \text{succ}(\text{arity}(\varphi)) \ 2 < \text{arity}(\varphi) +_{\omega} 2$ 
    using succ_ltI by auto
    with  $\langle \varphi \in \_ \rangle$ 
    have  $\text{arity}(\text{iterates}(\lambda p. \text{incr\_bv}(p) '1, 2, \varphi)) = 2 +_{\omega} \text{arity}(\varphi)$ 
    using arity_incr_bv_lemma lt
    by auto
    with  $\langle \varphi \in \_ \rangle$ 
    show ?thesis
    unfolding ren_forces_nand_def
    using lt pred_Un_distrib union_abs1 Un_assoc[symmetric] Un_le_compat
    by (simp add: FOL_arity)
  next
  case ge
  with  $\langle \varphi \in \_ \rangle$ 
  have  $\text{arity}(\varphi) \leq 1 \ \text{pred}(\text{arity}(\varphi)) \leq 1$ 
  using not_lt_iff_le le_trans[OF le_pred]
  by simp_all
  with  $\langle \varphi \in \_ \rangle$ 
  have  $\text{arity}(\text{iterates}(\lambda p. \text{incr\_bv}(p) '1, 2, \varphi)) = (\text{arity}(\varphi))$ 
  using arity_incr_bv_lemma ge
  by simp
  with  $\langle \text{arity}(\varphi) \leq 1 \rangle \ \langle \varphi \in \_ \rangle \ \langle \text{pred}(\_) \leq 1 \rangle$ 
  show ?thesis
  unfolding ren_forces_nand_def
  using pred_Un_distrib union_abs1 Un_assoc[symmetric] union_abs2
  by (simp add: FOL_arity)
qed
qed

lemma sats_ren_forces_nand:
   $[q, P, \text{leq}, o, p] @ \text{env} \in \text{list}(M) \implies \varphi \in \text{formula} \implies$ 
   $(M, [q, p, P, \text{leq}, o] @ \text{env} \models \text{ren\_forces\_nand}(\varphi)) \iff (M, [q, P, \text{leq}, o] @ \text{env} \models$ 
 $\varphi)$ 
  unfolding ren_forces_nand_def
  using sats_incr_bv_iff [of _ _ M _ [q]]
  by simp

```

definition

```

ren_forces_forall :: i⇒i where
ren_forces_forall(φ) ≡
  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(λp. incr_bv(p)'5 , 5, φ))))))))))

```

lemma *arity_ren_forces_all* :

```

assumes φ∈formula
shows arity(ren_forces_forall(φ)) = 5 ∪ arity(φ)

```

proof -

```

consider (lt) 5<arity(φ) | (ge) ¬ 5 < arity(φ)
  by auto
then
show ?thesis
proof cases
  case lt
    with ⟨φ∈_⟩
    have 5 < succ(arity(φ)) 5<arity(φ)+ω2 5<arity(φ)+ω3 5<arity(φ)+ω4
      using succ_ltI by auto
    with ⟨φ∈_⟩
    have arity(iterates(λp. incr_bv(p)'5,5,φ)) = 5+ωarity(φ)
      using arity_incr_bv_lemma lt
      by simp
    with ⟨φ∈_⟩
    show ?thesis
      unfolding ren_forces_forall_def
      using pred_Un_distrib union_abs1 Un_assoc[symmetric] union_abs2
      by (simp add:FOL_arity)
  next
    case ge
      with ⟨φ∈_⟩
      have arity(φ) ≤ 5 pred5(arity(φ)) ≤ 5
        using not_lt_iff_le le_trans[OF le_pred]
        by simp_all
      with ⟨φ∈_⟩
      have arity(iterates(λp. incr_bv(p)'5,5,φ)) = arity(φ)
        using arity_incr_bv_lemma ge
        by simp
      with ⟨arity(φ) ≤ 5⟩ ⟨φ∈_⟩ ⟨pred5(_) ≤ 5⟩
      show ?thesis
        unfolding ren_forces_forall_def
        using pred_Un_distrib union_abs1 Un_assoc[symmetric] union_abs2
        by (simp add:FOL_arity)

```

qed

qed

lemma *ren_forces_forall_type*[TC] :
 $\varphi \in \text{formula} \implies \text{ren_forces_forall}(\varphi) \in \text{formula}$
unfolding *ren_forces_forall_def* **by** *simp*

lemma *sats_ren_forces_forall* :
 $[x,P,leq,o,p] @ \text{env} \in \text{list}(M) \implies \varphi \in \text{formula} \implies$
 $(M, [x,p,P,leq,o] @ \text{env} \models \text{ren_forces_forall}(\varphi)) \longleftrightarrow (M, [p,P,leq,o,x] @ \text{env} \models \varphi)$
unfolding *ren_forces_forall_def*
using *sats_incr_bv_iff* [*of* $_ _ M _ [p,P,leq,o,x]$]
by *simp*

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]: $\varphi \in \text{formula} \implies \text{forces}'(\varphi) \in \text{formula}$
by (*induct* φ *set:formula; simp*)

lemma *forces_type*[TC] : $\varphi \in \text{formula} \implies \text{forces}(\varphi) \in \text{formula}$
unfolding *forces_def* **by** *simp*

5.6 The arity of forces

lemma *arity_forces_at*:

assumes $x \in \text{nat } y \in \text{nat}$

shows $\text{arity}(\text{forces}(\text{Member}(x,y))) = (\text{succ}(x) \cup \text{succ}(y)) +_\omega 4$

$\text{arity}(\text{forces}(\text{Equal}(x,y))) = (\text{succ}(x) \cup \text{succ}(y)) +_\omega 4$

unfolding *forces_def*

using *assms* *arity_forces_mem_fm* *arity_forces_eq_fm* *succ_Un_distrib* *ord_simp_union*

by (*auto simp:FOL_aritys,(rule_tac le_anti_sym,simp_all,(rule_tac not_le_anti_sym,simp_all))+*)

lemma *arity_forces'*:

assumes $\varphi \in \text{formula}$

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

using *assms*

proof (*induct set:formula*)

case (*Member* x y)

```

then
show ?case
using arity_forces_mem_fm succ_Un_distrib ord_simp_union leI not_le_iff_lt
by simp
next
case (Equal x y)
then
show ?case
using arity_forces_eq_fm succ_Un_distrib ord_simp_union leI not_le_iff_lt
by simp
next
case (Nand  $\varphi$   $\psi$ )
let ? $\varphi'$  = ren_forces_nand(forces'( $\varphi$ ))
let ? $\psi'$  = ren_forces_nand(forces'( $\psi$ ))
have arity(is_leq_fm(3, 0, 1)) = 4
using arity_is_leq_fm succ_Un_distrib ord_simp_union
by simp
have  $3 \leq (4 +_{\omega} \text{arity}(\varphi)) \cup (4 +_{\omega} \text{arity}(\psi))$  (is  $\_ \leq$  ?rhs)
using ord_simp_union by simp
from  $\langle \varphi \in \_ \rangle$  Nand
have pred(arity(? $\varphi'$ ))  $\leq$  ?rhs pred(arity(? $\psi'$ ))  $\leq$  ?rhs
proof -
from  $\langle \varphi \in \_ \rangle$   $\langle \psi \in \_ \rangle$ 
have A: pred(arity(? $\varphi'$ ))  $\leq$  arity(forces'( $\varphi$ ))
pred(arity(? $\psi'$ ))  $\leq$  arity(forces'( $\psi$ ))
using pred_mono[OF  $\_$  arity_ren_forces_nand] pred_succ_eq
by simp_all
from Nand
have  $3 \cup \text{arity}(\text{forces}'(\varphi)) \leq \text{arity}(\varphi) +_{\omega} 4$ 
 $3 \cup \text{arity}(\text{forces}'(\psi)) \leq \text{arity}(\psi) +_{\omega} 4$ 
using Un_le by simp_all
with Nand
show pred(arity(? $\varphi'$ ))  $\leq$  ?rhs
pred(arity(? $\psi'$ ))  $\leq$  ?rhs
using le_trans[OF A(1)] le_trans[OF A(2)] le_Un_iff
by simp_all
qed
with Nand  $\langle \_ = 4 \rangle$ 
show ?case
using pred_Un_distrib Un_assoc[symmetric] succ_Un_distrib union_abs1
Un_leI3[OF  $\langle 3 \leq ?rhs \rangle$ ]
by (simp add: FOL_aritys)
next
case (Forall  $\varphi$ )
let ? $\varphi'$  = ren_forces_forall(forces'( $\varphi$ ))
show ?case
proof (cases arity( $\varphi$ ) = 0)
case True
with Forall

```

```

show ?thesis
proof -
  from Forall True
  have arity(forces'( $\varphi$ ))  $\leq 5$ 
    using le_trans[of _ 4 5] by auto
  with  $\langle \varphi \in \_ \rangle$ 
  have arity( $? \varphi'$ )  $\leq 5$ 
    using arity_ren_forces_all[OF forces'_type[OF  $\langle \varphi \in \_ \rangle$ ]] union_abs2
    by auto
  with Forall True
  show ?thesis
    using pred_mono[OF _  $\langle \text{arity}(\varphi') \leq 5 \rangle$ ]
    by simp
qed
next
case False
with Forall
show ?thesis
proof -
  from Forall False
  have arity( $? \varphi'$ ) =  $5 \cup \text{arity}(\text{forces}'(\varphi))$ 
    arity(forces'( $\varphi$ ))  $\leq 5 +_\omega \text{arity}(\varphi)$ 
     $4 \leq 3 +_\omega \text{arity}(\varphi)$ 
    using Ord_0_lt arity_ren_forces_all
    le_trans[OF _ add_le_mono[of 4 5, OF _ le_refl]]
    by auto
  with  $\langle \varphi \in \_ \rangle$ 
  have  $5 \cup \text{arity}(\text{forces}'(\varphi)) \leq 5 +_\omega \text{arity}(\varphi)$ 
    using ord_simp_union by auto
  with  $\langle \varphi \in \_ \rangle$   $\langle \text{arity}(\varphi') = 5 \cup \_ \rangle$ 
  show ?thesis
    using pred_Un_distrib succ_pred_eq[OF _  $\langle \text{arity}(\varphi) \neq 0 \rangle$ ]
    pred_mono[OF _ Forall(2)] Un_le[OF  $\langle 4 \leq 3 +_\omega \text{arity}(\varphi) \rangle$ ]
    by simp
qed
qed
qed

lemma arity_forces :
  assumes  $\varphi \in \text{formula}$ 
  shows  $\text{arity}(\text{forces}(\varphi)) \leq 4 +_\omega \text{arity}(\varphi)$ 
  unfolding forces_def
  using assms arity_forces' le_trans ord_simp_union FOL_aritys by auto

lemma arity_forces_le :
  assumes  $\varphi \in \text{formula}$   $n \in \text{nat}$   $\text{arity}(\varphi) \leq n$ 
  shows  $\text{arity}(\text{forces}(\varphi)) \leq 4 +_\omega n$ 
  using assms le_trans[OF _ add_le_mono[OF le_refl[of 5]  $\langle \text{arity}(\varphi) \leq \_ \rangle$ ]] ar-
  ity_forces

```

by auto

definition rename_split_fm where

rename_split_fm(φ) \equiv ($\cdot \exists (\cdot \exists (\cdot \exists (\cdot \exists (\cdot \exists (\cdot \exists \dots \text{snd}(9) \text{ is } 0 \cdot \wedge \dots \text{fst}(9) \text{ is } 4 \cdot \wedge \dots 1=11 \cdot$
 \wedge
 $\dots 2=12 \cdot \wedge \dots 3=13 \cdot \wedge \dots 5=7 \cdot \wedge$
 $(\lambda p. \text{incr_bv}(p) \wedge 8(\text{forces}(\varphi)) \dots \dots \dots) \dots \dots \dots)$)

lemma rename_split_fm_type[TC]: $\varphi \in \text{formula} \implies \text{rename_split_fm}(\varphi) \in \text{formula}$
unfolding rename_split_fm_def by simp

schematic_goal arity_rename_split_fm: $\varphi \in \text{formula} \implies \text{arity}(\text{rename_split_fm}(\varphi))$
 $= ?m$

using arity_forces[of φ] forces_type **unfolding** rename_split_fm_def
by (simp add:arity Un_assoc[symmetric] union_abs1)

lemma arity_rename_split_fm_le:

assumes $\varphi \in \text{formula}$

shows $\text{arity}(\text{rename_split_fm}(\varphi)) \leq 8 \cup (6 +_{\omega} \text{arity}(\varphi))$

proof -

from assms

have arity_forces_6: $\neg 1 < \text{arity}(\varphi) \implies 6 \leq n \implies \text{arity}(\text{forces}(\varphi)) \leq n$ **for** n

using le_trans lt_trans[of $_ 5 n$] not_lt_iff_le[of $1 \text{arity}(\varphi)$]

by (auto intro!:le_trans[OF arity_forces])

have pred1_arity_forces: $\neg 1 < \text{arity}(\varphi) \implies \text{pred}^n(\text{arity}(\text{forces}(\varphi))) \leq 8$ **if**
 $n \in \text{nat}$ **for** n

using that pred_le[of 7] le_succ[THEN $[2]$ le_trans] arity_forces_6

by (induct rule:nat_induct) auto

have arity_forces_le_succ6: $\text{pred}^n(\text{arity}(\text{forces}(\varphi))) \leq \text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{arity}(\varphi))))))$
if $n \in \text{nat}$ **for** n

using that assms arity_forces[of φ , THEN le_trans,

OF $_ \text{le_succ}$, THEN le_trans, OF $_ _ \text{le_succ}$] le_trans[OF pred_le[OF

$_ \text{le_succ}$]

by (induct rule:nat_induct) auto

note trivial_arity_6 = arity_forces_6

arity_forces_le_succ6[of 1 , simplified] arity_forces_le_succ6[of 2 , simplified]

arity_forces_le_succ6[of 3 , simplified] arity_forces_le_succ6[of 4 , simplified]

arity_forces_le_succ6[of 5 , simplified] arity_forces_le_succ6[of 6 , simplified]

pred1_arity_forces[of 1 , simplified] pred1_arity_forces[of 2 , simplified]

pred1_arity_forces[of 3 , simplified] pred1_arity_forces[of 4 , simplified]

pred1_arity_forces[of 5 , simplified] pred1_arity_forces[of 6 , simplified]

show ?thesis

using assms arity_forces[of φ] arity_forces[of φ , THEN le_trans, OF $_ \text{le_succ}$]

arity_forces[of φ , THEN le_trans, OF $_ \text{le_succ}$, THEN le_trans, OF $_ _ \text{le_succ}$]

unfolding rename_split_fm_def

by (simp add:arity Un_assoc[symmetric] union_abs1 arity_forces[of φ] forces_type)

((subst arity_incr_bv_lemma; auto simp: arity_ord_simp_union_forces_type

trivial_arity)+)

qed

definition *body_ground_repl_fm* where

$body_ground_repl_fm(\varphi) \equiv (\exists (\cdot \exists \cdot is_Vset_fm(2, 0) \wedge \cdot \cdot 1 \in 0 \cdot \wedge rename_split_fm(\varphi) \dots) \cdot)$

lemma *body_ground_repl_fm_type*[TC]: $\varphi \in formula \implies body_ground_repl_fm(\varphi) \in formula$
unfolding *body_ground_repl_fm_def* by *simp*

lemma *arity_body_ground_repl_fm_le*:

notes *le_trans*[*trans*]

assumes $\varphi \in formula$

shows $arity(body_ground_repl_fm(\varphi)) \leq 6 \cup (arity(\varphi) +_{\omega} 4)$

proof -

from $\langle \varphi \in formula \rangle$

have *ineq*: $n \cup pred(pred(arity(rename_split_fm(\varphi))))$

$\leq m \cup pred(pred(8 \cup (arity(\varphi) +_{\omega} 6)))$ **if** $n \leq m$ $n \in nat$ $m \in nat$ **for** n m

using that *arity_rename_split_fm_le*[of φ , THEN [2] *pred_mono*, THEN [2]

pred_mono,

THEN [2] *Un_mono*[THEN *subset_imp_le*, OF *le_imp_subset*]] *le_imp_subset*

by *auto*

moreover

have *eq1*: $pred(pred(pred(4 \cup 2 \cup pred(pred(pred(pred(pred(pred(pred(pred(9 \cup 1 \cup 3 \cup 2))))))))))) = 1$

by (*auto simp:pred_Un_distrib*)

ultimately

have $pred(pred(pred(4 \cup 2 \cup pred(pred(pred(pred(pred(pred(pred(pred(9 \cup 1 \cup 3 \cup 2))))))))))) \cup pred(pred(arity(rename_split_fm(\varphi))) \leq 1 \cup pred(pred(8 \cup (arity(\varphi) +_{\omega} 6)))$

by *auto*

also from $\langle \varphi \in formula \rangle$

have $1 \cup pred(pred(8 \cup (arity(\varphi) +_{\omega} 6))) \leq 6 \cup (4 +_{\omega} arity(\varphi))$

by (*auto simp:pred_Un_distrib Un_assoc[symmetric] ord_simp_union*)

finally

show *?thesis*

using $\langle \varphi \in formula \rangle$ **unfolding** *body_ground_repl_fm_def*

by (*simp add:arity pred_Un_distrib, subst arity_transrec_fm[of is_HVfrom_fm(8,2,1,0) 3 1]*)

(*simp add:arity pred_Un_distrib, simp_all,*

auto simp add:eq1 arity_is_HVfrom_fm[of 8 2 1 0])

qed

definition *ground_repl_fm* where

$ground_repl_fm(\varphi) \equiv least_fm(body_ground_repl_fm(\varphi), 1)$

lemma *ground_repl_fm_type*[TC]:

$\varphi \in formula \implies ground_repl_fm(\varphi) \in formula$

```

unfolding ground_repl_fm_def by simp

lemma arity_ground_repl_fm:
  assumes  $\varphi \in \text{formula}$ 
  shows  $\text{arity}(\text{ground\_repl\_fm}(\varphi)) \leq 5 \cup (3 +_{\omega} \text{arity}(\varphi))$ 
proof -
  from assms
  have  $\text{pred}(\text{arity}(\text{body\_ground\_repl\_fm}(\varphi))) \leq 5 \cup (3 +_{\omega} \text{arity}(\varphi))$ 
    using arity_body_ground_repl_fm_le pred_mono succ_Un_distrib
    by (rule_tac pred_le) auto
  with assms
  have  $2 \cup \text{pred}(\text{arity}(\text{body\_ground\_repl\_fm}(\varphi))) \leq 5 \cup (3 +_{\omega} \text{arity}(\varphi))$ 
    using Un_le le_Un_iff by auto
  then
  show ?thesis
    using assms arity_forces arity_body_ground_repl_fm_le
    unfolding least_fm_def ground_repl_fm_def
    apply (auto simp add:arity Un_assoc[symmetric])
    apply (simp add: pred_Un Un_assoc, simp add: Un_assoc[symmetric] union_abs1
pred_Un)
    by (simp only: Un_commute, subst Un_commute, simp add:ord_simp_union,force)
qed

synthesize is_ordermap from_definition assuming nonempty

synthesize is_ordertype from_definition assuming nonempty

synthesize is_order_body from_definition assuming nonempty
arity_theorem for is_order_body_fm

definition omap_wfrec_body where
  omap_wfrec_body( $A, r$ )  $\equiv (\cdot \exists \cdot \text{image\_fm}(2, 0, 1) \wedge \text{pred\_set\_fm}(9 +_{\omega} A, 3, 9 +_{\omega} r,$ 
   $0) \dots)$ 

lemma type_omap_wfrec_body_fm :  $A \in \text{nat} \implies r \in \text{nat} \implies \text{omap\_wfrec\_body}(A, r) \in \text{formula}$ 
unfolding omap_wfrec_body_def by simp

lemma arity_omap_wfrec_aux :  $A \in \text{nat} \implies r \in \text{nat} \implies \text{arity}(\text{omap\_wfrec\_body}(A, r))$ 
 $= (9 +_{\omega} A) \cup (9 +_{\omega} r)$ 
unfolding omap_wfrec_body_def
using arity_image_fm arity_pred_set_fm pred_Un_distrib union_abs2[of 3]
union_abs1
by (simp add:FOL_aritys, auto simp add:Un_assoc[symmetric] union_abs1)

lemma arity_omap_wfrec :  $A \in \text{nat} \implies r \in \text{nat} \implies$ 
 $\text{arity}(\text{is\_wfrec\_fm}(\text{omap\_wfrec\_body}(A, r), r +_{\omega} 3, 1, 0)) = (4 +_{\omega} A) \cup (4 +_{\omega} r)$ 
using Arities.arity_is_wfrec_fm[OF _ _ _ _ _ arity_omap_wfrec_aux, of A r
 $3 +_{\omega} r 1 0]$ 
pred_Un_distrib union_abs1 union_abs2 type_omap_wfrec_body_fm

```

by auto

lemma *arity_isordermap*: $A \in \text{nat} \implies r \in \text{nat} \implies d \in \text{nat} \implies$
 $\text{arity}(\text{is_ordermap_fm}(A, r, d)) = \text{succ}(d) \cup (\text{succ}(A) \cup \text{succ}(r))$
unfolding *is_ordermap_fm_def*
using *arity_lambda_fm* [where $i = (4 + \omega A) \cup (4 + \omega r)$, *OF_____arity_omap_wfrec*,
unfolded_omap_wfrec_body_def] *pred_Un_distrib union_abs1*
by auto

lemma *arity_is_ordertype*: $A \in \text{nat} \implies r \in \text{nat} \implies d \in \text{nat} \implies$
 $\text{arity}(\text{is_ordertype_fm}(A, r, d)) = \text{succ}(d) \cup (\text{succ}(A) \cup \text{succ}(r))$
unfolding *is_ordertype_fm_def*
using *arity_isordermap arity_image_fm pred_Un_distrib FOL_arities*
by auto

lemma *arity_is_order_body*: $\text{arity}(\text{is_order_body_fm}(0, 1)) = 2$
using *arity_is_order_body_fm arity_is_ordertype ord_simp_union*
by (*simp add:FOL_arities*)

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

relationalize *H_order_pred* *is_H_order_pred*

synthesize *is_H_order_pred* **from_definition** *assuming nonempty*

definition *order_pred_wfrec_body* **where**
 $order_pred_wfrec_body(M, A, r, z, x) \equiv \exists y[M].$
 $pair(M, x, y, z) \wedge$
 $(\exists f[M].$
 $(\forall z[M].$
 $z \in f \longleftrightarrow$
 $(\exists xa[M].$
 $\exists y[M].$
 $\exists xaa[M].$
 $\exists sx[M].$
 $\exists r_sx[M].$
 $\exists f_r_sx[M].$
 $pair(M, xa, y, z) \wedge$
 $pair(M, xa, x, xaa) \wedge$
 $upair(M, xa, xa, sx) \wedge$
 $pre_image(M, r, sx, r_sx) \wedge$
 $restriction(M, f, r_sx, f_r_sx) \wedge$
 $xaa \in r \wedge (\exists a[M]. image(M, f_r_sx, a, y) \wedge$
 $pred_set(M, A, xa, r, a))) \wedge$
 $(\exists a[M]. image(M, f, a, y) \wedge pred_set(M, A, x, r, a)))$

synthesize *order_pred_wfrec_body* **from_definition**

arity_theorem for *order_pred_wfrec_body_fm*

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

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

relativize functional *funspace_succ_rep_intf* *funspace_succ_rep_intf_rel*

— The definition obtained next uses *is_cons* instead of *upair* as in Paulson's `~/src/ZF/Constructible/Relative.thy`.

relationalize *funspace_succ_rep_intf_rel* *is_funspace_succ_rep_intf*

synthesize *is_funspace_succ_rep_intf* **from definition**

arity_theorem for *is_funspace_succ_rep_intf_fm*

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

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

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

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

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

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

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

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

definition *replacement_assm* **where**

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

definition *ground_replacement_assm* **where**

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

end

6 The ZFC axioms, internalized

theory *Internal_ZFC_Axioms*

imports

Fm_Definitions

begin

schematic_goal *ZF_union_auto*:

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

unfolding *Union_ax_def*

by ((*rule sep_rules* | *simp*)+)

synthesize *ZF_union* **from_schematic** *ZF_union_auto*

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

schematic_goal *ZF_power_auto*:

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

unfolding *power_ax_def powerset_def subset_def*

by ((*rule sep_rules* | *simp*)+)

synthesize *ZF_power* **from_schematic** *ZF_power_auto*

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

schematic_goal *ZF_pairing_auto*:

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

unfolding *upair_ax_def*

by ((*rule sep_rules* | *simp*)+)

synthesize *ZF_pairing* **from_schematic** *ZF_pairing_auto*

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

schematic_goal *ZF_foundation_auto*:

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

unfolding *foundation_ax_def*

by ((*rule sep_rules* | *simp*)+)

synthesize *ZF_foundation* **from_schematic** *ZF_foundation_auto*

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

schematic_goal *ZF_extensionality_auto*:
 $extensionality(\#\#A) \longleftrightarrow (A, [] \models ?ztext)$
unfolding *extensionality_def*
by ((*rule sep_rules* | *simp*)⁺)

synthesize *ZF_extensionality_from_schematic* *ZF_extensionality_auto*
notation *ZF_extensionality_fm* ($\langle \cdot Extensionality \cdot \rangle$)

schematic_goal *ZF_infinity_auto*:
 $infinity_ax(\#\#A) \longleftrightarrow (A, [] \models (? \varphi(i,j,h)))$
unfolding *infinity_ax_def*
by ((*rule sep_rules* | *simp*)⁺)

synthesize *ZF_infinity_from_schematic* *ZF_infinity_auto*
notation *ZF_infinity_fm* ($\langle \cdot Infinity \cdot \rangle$)

schematic_goal *ZF_choice_auto*:
 $choice_ax(\#\#A) \longleftrightarrow (A, [] \models (? \varphi(i,j,h)))$
unfolding *choice_ax_def*
by ((*rule sep_rules* | *simp*)⁺)

synthesize *ZF_choice_from_schematic* *ZF_choice_auto*
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*]:
 $\llbracket n \in nat; p \in formula \rrbracket \implies Forall^{\wedge n}(p) \in formula$
by (*induct set:nat, auto*)

lemma *last_init_eq* :
assumes $l \in list(A)$ $length(l) = succ(n)$
shows $\exists a \in A. \exists l' \in list(A). l = l'@[a]$
proof-
from $\langle l \in _ \rangle \langle length(_) = _ \rangle$
have $rev(l) \in list(A)$ $length(rev(l)) = succ(n)$
by *simp_all*

```

then
obtain  $a\ l'$  where  $a \in A\ l' \in \text{list}(A)\ \text{rev}(l) = \text{Cons}(a, l')$ 
  by  $(\text{cases}; \text{simp})$ 
then
have  $l = \text{rev}(l') @ [a]\ \text{rev}(l') \in \text{list}(A)$ 
  using  $\text{rev\_rev\_ident}[OF\ \langle l \in \_ \rangle]$  by  $\text{auto}$ 
with  $\langle a \in \_ \rangle$ 
show  $?thesis$  by  $\text{blast}$ 
qed

```

```

lemma  $\text{take\_drop\_eq}$  :
  assumes  $l \in \text{list}(M)$ 
  shows  $\bigwedge n . n < \text{succ}(\text{length}(l)) \implies l = \text{take}(n, l) @ \text{drop}(n, l)$ 
  using  $\langle l \in \text{list}(M) \rangle$ 
proof  $\text{induct}$ 
  case  $\text{Nil}$ 
  then show  $?case$  by  $\text{auto}$ 
next
  case  $(\text{Cons}\ a\ l)$ 
  then show  $?case$ 
proof -
  {
    fix  $i$ 
    assume  $i < \text{succ}(\text{succ}(\text{length}(l)))$ 
    with  $\langle l \in \text{list}(M) \rangle$ 
    consider  $(lt)\ i = 0 \mid (eq)\ \exists k \in \text{nat} . i = \text{succ}(k) \wedge k < \text{succ}(\text{length}(l))$ 
    using  $\langle l \in \text{list}(M) \rangle\ \text{le\_natI}\ \text{nat\_imp\_quasinat}$ 
    by  $(\text{cases}\ \text{rule}:\text{nat\_cases}[of\ i]; \text{auto})$ 
    then
    have  $\text{take}(i, \text{Cons}(a, l)) @ \text{drop}(i, \text{Cons}(a, l)) = \text{Cons}(a, l)$ 
    using  $\text{Cons}$ 
    by  $(\text{cases}; \text{auto})$ 
  }
  then show  $?thesis$  using  $\text{Cons}$  by  $\text{auto}$ 
qed
qed

```

```

lemma  $\text{list\_split}$  :
  assumes  $n \leq \text{succ}(\text{length}(\text{rest}))\ \text{rest} \in \text{list}(M)$ 
  shows  $\exists re \in \text{list}(M) . \exists st \in \text{list}(M) . \text{rest} = re @ st \wedge \text{length}(re) = \text{pred}(n)$ 
proof -
  from  $\text{assms}$ 
  have  $\text{pred}(n) \leq \text{length}(\text{rest})$ 
  using  $\text{pred\_mono}[OF\ \_ \langle n \leq \_ \rangle]\ \text{pred\_succ\_eq}$  by  $\text{auto}$ 
  with  $\langle \text{rest} \in \_ \rangle$ 
  have  $\text{pred}(n) \in \text{nat}\ \text{rest} = \text{take}(\text{pred}(n), \text{rest}) @ \text{drop}(\text{pred}(n), \text{rest})$  (is  $\_ = ?re @$ 
   $?st)$ 
  using  $\text{take\_drop\_eq}[OF\ \langle \text{rest} \in \_ \rangle]\ \text{le\_natI}$  by  $\text{auto}$ 
  then

```

```

have length(?re) = pred(n) ?re∈list(M) ?st∈list(M)
  using length_take[rule_format,OF _ ⟨pred(n)∈_⟩] ⟨pred(n) ≤ _⟩ ⟨rest∈_⟩
  unfolding min_def
  by auto
then
show ?thesis
  using rev_bexI[of _ _ λ re. ∃ st∈list(M). rest = re @ st ∧ length(re) = pred(n)]
    ⟨length(?re) = _⟩ ⟨rest = _⟩
  by auto
qed

```

lemma sats_nForall:

```

assumes
  φ ∈ formula
shows
  n∈nat ⇒ ms ∈ list(M) ⇒
    (M, ms ⊨ (Forall^n(φ))) ↔
    (∀ rest ∈ list(M). length(rest) = n → M, rest @ ms ⊨ φ)
proof (induct n arbitrary:ms set:nat)
  case 0
  with assms
  show ?case by simp
next
  case (succ n)
  have (∀ rest∈list(M). length(rest) = succ(n) → P(rest,n)) ↔
    (∀ t∈M. ∀ res∈list(M). length(res) = n → P(res @ [t],n))
  if n∈nat for n P
  using that last_init_eq by force
  from this[of _ λrest _. (M, rest @ ms ⊨ φ)] ⟨n∈nat⟩
  have (∀ rest∈list(M). length(rest) = succ(n) → M, rest @ ms ⊨ φ) ↔
    (∀ t∈M. ∀ res∈list(M). length(res) = n → M, (res @ [t]) @ ms ⊨ φ)
  by simp
  with assms succ(1,3) succ(2)[of Cons(_,ms)]
  show ?case
  using arity_sats_iff[of φ _ M Cons(_, ms @ _)] app_assoc
  by (simp)
qed

```

definition

```

sep_body_fm :: i ⇒ i where
sep_body_fm(p) ≡ (·∀ (·∃ (·∀ ·0 ∈ 1 · ↔ ·0 ∈ 2 · ∧ incr_bv1^2 (p) ...)·)·)

```

lemma sep_body_fm_type [TC]: p ∈ formula ⇒ sep_body_fm(p) ∈ formula
by (simp add: sep_body_fm_def)

lemma sats_sep_body_fm:

```

assumes
  φ ∈ formula ms∈list(M) rest∈list(M)
shows

```

$(M, \text{rest} @ \text{ms} \models \text{sep_body_fm}(\varphi)) \longleftrightarrow$
 $\text{separation}(\#\#M, \lambda x. M, [x] @ \text{rest} @ \text{ms} \models \varphi)$
using *assms* *formula_add_params1*[of $_2 _ _ [_ , _]$]
unfolding *sep_body_fm_def* *separation_def* **by** *simp*

definition

$ZF_separation_fm :: i \Rightarrow i (\cdot \text{Separation}'(_') \cdot)$ **where**
 $ZF_separation_fm(p) \equiv \text{Forall}^\wedge(\text{pred}(\text{arity}(p)))(\text{sep_body_fm}(p))$

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

by (*simp* *add*: *ZF_separation_fm_def*)

lemma *sats_ZF_separation_fm_iff*:

assumes

$\varphi \in \text{formula}$

shows

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

\longleftrightarrow

$(\forall \text{env} \in \text{list}(M). \text{arity}(\varphi) \leq 1 +_\omega \text{length}(\text{env}) \longrightarrow$
 $\text{separation}(\#\#M, \lambda x. M, [x] @ \text{env} \models \varphi))$

proof (*intro iffI ballI impI*)

let $?n = \text{pred}(\text{arity}(\varphi))$

fix *env*

assume $M, [] \models ZF_separation_fm(\varphi)$

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

moreover from *this*

have $\text{arity}(\varphi) \leq \text{succ}(\text{length}(\text{env}))$ **by** *simp*

then

obtain *some* **where** $\text{some} \in \text{list}(M) \text{ rest} \in \text{list}(M)$

$\text{env} = \text{some} @ \text{rest} \text{ length}(\text{some}) = \text{pred}(\text{arity}(\varphi))$

using *list_split[OF* $\langle \text{arity}(\varphi) \leq \text{succ}(_) \rangle \langle \text{env} \in _ \rangle$ **by** *force*

moreover from $\langle \varphi \in _ \rangle$

have $\text{arity}(\varphi) \leq \text{succ}(\text{pred}(\text{arity}(\varphi)))$

using *succpred_leI* **by** *simp*

moreover

note *assms*

moreover

assume $M, [] \models ZF_separation_fm(\varphi)$

moreover from *calculation*

have $M, \text{some} \models \text{sep_body_fm}(\varphi)$

using *sats_nForall*[of *sep_body_fm*(φ) *?n*]

unfolding *ZF_separation_fm_def* **by** *simp*

ultimately

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

unfolding *ZF_separation_fm_def*

using *sats_sep_body_fm*[of φ $[]$ M *some*]

arity_sats_iff[of φ *rest* M $[]$ @ *some*]

separation_cong[of $\#\#M$ $\lambda x. M$, *Cons*(*x*, *some* @ *rest*) $\models \varphi _$]

```

  by simp
next — almost equal to the previous implication
let ?n=pred(arity(φ))
assume asm:∀ env∈list(M). arity(φ) ≤ 1 +ω length(env) →
  separation(##M, λx. M, [x] @ env ⊨ φ)
{
  fix some
  assume some∈list(M) length(some) = pred(arity(φ))
  moreover
  note ⟨φ∈_⟩
  moreover from calculation
  have arity(φ) ≤ 1 +ω length(some)
    using le_trans[OF succpred_leI] succpred_leI by simp
  moreover from calculation and asm
  have separation(##M, λx. M, [x] @ some ⊨ φ) by blast
  ultimately
  have M, some ⊨ sep_body_fm(φ)
  using sats_sep_body_fm[of φ [] M some]
    arity_sats_iff[of φ _ M [_,_] @ some]
    strong_replacement_cong[of ##M λx y. M, Cons(x, Cons(y, some @ _)) ⊨
φ _ ]
  by simp
}
with ⟨φ∈_⟩
show M, [] ⊨ ZF_separation_fm(φ)
  using sats_nForall[of sep_body_fm(φ) ?n]
  unfolding ZF_separation_fm_def
  by simp
qed

```

6.2 The Axiom of Replacement, internalized

schematic_goal *sats_univalent_fm_auto*:
assumes

$$\begin{aligned}
Q_iff_sats: & \bigwedge x y z. x \in A \implies y \in A \implies z \in A \implies \\
& Q(x, z) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, env)))) \models Q1_fm) \\
& \bigwedge x y z. x \in A \implies y \in A \implies z \in A \implies \\
& Q(x, y) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, env)))) \models Q2_fm)
\end{aligned}$$

and

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

shows

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

unfolding *univalent_def*

by (*insert asms; (rule sep_rules Q_iff_sats | simp)+*)

synthesize_notc *univalent_from_schematic* *sats_univalent_fm_auto*

lemma *univalent_fm_type* [TC]: $q1 \in formula \implies q2 \in formula \implies i \in nat \implies$

univalent_fm(*q2*,*q1*,*i*) ∈ *formula*
by (*simp add:univalent_fm_def*)

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

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

unfolding *univalent_fm_def* **using** *asms sats_univalent_fm_auto[OF Q_iff_sats]*
by *simp*

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*] :

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

unfolding *swap_vars_def* **by** *simp*

lemma *sats_swap_vars* :

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

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

unfolding *swap_vars_def*

using *sats_incr_bv_iff [of _ _ M _ [y,x]]* **by** *simp*

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 $\text{univalent_Q1}(\varphi) \in \text{formula} \ \text{univalent_Q2}(\varphi) \in \text{formula}$

unfolding *univalent_Q1_def univalent_Q2_def* **using** *asms* **by** *simp_all*

lemma *sats_univalent_fm_asm*:

assumes

$x \in A \ y \in A \ z \in A \ \text{env} \in \text{list}(A) \ \varphi \in \text{formula}$

shows

$(A, ([x,z] @ env) \models \varphi) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, env)))) \models (\text{univalent_Q1}(\varphi))$
 $(A, ([x,y] @ env) \models \varphi) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, env)))) \models (\text{univalent_Q2}(\varphi))$

unfolding *univalent_Q1_def univalent_Q2_def*

using

sats_incr_bv_iff[of _ _ A _ []] — simplifies iterates of $\lambda x. \text{incr_bv}(x) \text{ ' } 0$

sats_incr_bv1_iff[of _ *Cons*(*x*,*env*) A *z* *y*]

sats_swap_vars *assms*

by *simp_all*

definition

rep_body_fm :: $i \Rightarrow i$ **where**

rep_body_fm(*p*) \equiv *Forall*(*Implies*(

univalent_fm(*univalent_Q1*(*incr_bv*(*p*) '2), *univalent_Q2*(*incr_bv*(*p*) '2), 0),

Exists(*Forall*(

Iff(*Member*(0,1), *Exists*(*And*(*Member*(0,3), *incr_bv*(*incr_bv*(*p*) '2) '2))))))

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

by (*simp add: rep_body_fm_def*)

lemmas *ZF_replacement_simps* = *formula_add_params1*[of φ 2 _ *M* [_ , _]]

sats_incr_bv_iff[of _ _ *M* _ []] — simplifies iterates of $\lambda x. \text{incr_bv}(x) \text{ ' } 0$

sats_incr_bv1_iff[of _ _ *M* _ [_ , _]] — simplifies $\lambda x. \text{incr_bv}(x) \text{ ' } 2$

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

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

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

using *assms ZF_replacement_simps*

unfolding *rep_body_fm_def strong_replacement_def univalent_def*

unfolding *univalent_fm_def univalent_Q1_def univalent_Q2_def*

by *simp*

definition

ZF_replacement_fm :: $i \Rightarrow i$ (\cdot *Replacement'* ($_$) \cdot) **where**

ZF_replacement_fm(*p*) \equiv *Forall* \wedge (*pred*(*pred*(*arity*(*p*))))(*rep_body_fm*(*p*))

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

by (*simp add: ZF_replacement_fm_def*)

lemma *sats_ZF_replacement_fm_iff*:

assumes

$\varphi \in \text{formula}$

shows

$(M, [] \models \cdot \text{Replacement}(\varphi) \cdot) \longleftrightarrow (\forall \text{env. } \text{replacement_assm}(M, \text{env}, \varphi))$

unfolding *replacement_assm_def*

```

proof (intro iffI allI impI)
  let ?n=pred(pred(arity(φ)))
  fix env
  assume M, [] ⊨ ZF_replacement_fm(φ) arity(φ) ≤ 2 +ω length(env) env∈list(M)
  moreover from this
  have arity(φ) ≤ succ(succ(length(env))) by (simp)
  moreover from calculation
  have pred(arity(φ)) ≤ succ(length(env))
    using pred_mono[OF _ ‹arity(φ)≤succ(_›] pred_succ_eq by simp
  moreover from calculation
  obtain some rest where some∈list(M) rest∈list(M)
    env = some @ rest length(some) = pred(pred(arity(φ)))
    using list_split[OF ‹pred(_) ≤ _› ‹env∈_›] by auto
  moreover
  note ‹φ∈_›
  moreover from this
  have arity(φ) ≤ succ(succ(pred(pred(arity(φ)))))
    using le_trans[OF succpred_leI] succpred_leI by simp
  moreover from calculation
  have M, some ⊨ rep_body_fm(φ)
    using sats_nForall[of rep_body_fm(φ) ?n]
    unfolding ZF_replacement_fm_def
    by simp
  ultimately
  show strong_replacement(##M, λx y. M, [x, y] @ env ⊨ φ)
    using sats_rep_body_fm[of φ [] M some]
    arity_sats_iff[of φ rest M [_,_] @ some]
    strong_replacement_cong[of ##M λx y. M, Cons(x, Cons(y, some @ rest))
  ⊨ φ _]
    by simp
next — almost equal to the previous implication
  let ?n=pred(pred(arity(φ)))
  assume asm:∀ env. φ ∈ formula →
    env ∈ list(M) → arity(φ) ≤ 2 +ω length(env) →
    strong_replacement(##M, λx y. M, [x, y] @ env ⊨ φ)
  {
    fix some
    assume some∈list(M) length(some) = pred(pred(arity(φ)))
    moreover
    note ‹φ∈_›
    moreover from calculation
    have arity(φ) ≤ 2 +ω length(some)
      using le_trans[OF succpred_leI] succpred_leI by simp
    moreover from calculation and asm
    have strong_replacement(##M, λx y. M, [x, y] @ some ⊨ φ) by blast
    ultimately
    have M, some ⊨ rep_body_fm(φ)
      using sats_rep_body_fm[of φ [] M some]
      arity_sats_iff[of φ _ M [_,_] @ some]
  }

```

```

    strong_replacement_cong[of ##M λx y. M, Cons(x, Cons(y, some @ _))] ⊨
  φ _ ]
  by simp
}
with ⟨φ∈_⟩
show M, [] ⊨ ZF_replacement_fm(φ)
  using sats_nForall[of rep_body_fm(φ) ?n]
  unfolding ZF_replacement_fm_def
  by simp
qed

```

definition

```

ZF_schemes :: i where
ZF_schemes ≡ {·Separation(p)· . p ∈ formula } ∪ {·Replacement(p)· . p ∈ formula }

```

lemma *Un_subset_formula* [TC]: $A ⊆ formula ∧ B ⊆ formula ⇒ A ∪ B ⊆ formula$
by *auto*

lemma *ZF_schemes_subset_formula* [TC]: $ZF_schemes ⊆ formula$
unfolding *ZF_schemes_def* **by** *auto*

lemma *ZF_fin_subset_formula* [TC]: $ZF_fin ⊆ formula$
unfolding *ZF_fin_def* **by** *simp*

definition

```

ZF :: i where
ZF ≡ ZF_schemes ∪ ZF_fin

```

lemma *ZF_subset_formula* [TC]: $ZF ⊆ formula$
unfolding *ZF_def* **by** *auto*

definition

```

ZFC :: i where
ZFC ≡ ZF ∪ {·AC·}

```

definition

```

ZF_minus_P :: i where
ZF_minus_P ≡ ZF - {·Powerset Ax· }

```

definition

```

Zermelo_fms :: i (⟨·Z·⟩) where
Zermelo_fms ≡ ZF_fin ∪ {·Separation(p)· . p ∈ formula }

```

definition

```

ZC :: i where
ZC ≡ Zermelo_fms ∪ {·AC·}

```

lemma *ZFC_subset_formula*: $ZFC ⊆ formula$

by (*simp add:ZFC_def Un_subset_formula*)

Satisfaction of a set of sentences

definition

satT :: [*i*,*i*] ⇒ *o* (*⊢* [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 $A \models \Phi$

using *assms unfolding satT_def by simp*

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

unfolding *satT_def by simp*

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

by *blast*

lemma *satT_Un_iff*: $M \models \Phi \cup \Psi \iff M \models \Phi \wedge M \models \Psi$ **by** *auto*

lemma *sats_ZFC_iff_sats_ZF_AC*:

$(N \models ZFC) \iff (N \models ZF) \wedge (N, [] \models \cdot AC \cdot)$

unfolding *ZFC_def ZF_def by auto*

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

unfolding *ZF_def ZF_schemes_def Zermelo_fms_def ZF_fin_def by auto*

lemma *satT_ZFC_imp_satT_ZC*: $M \models ZFC \implies M \models ZC$

unfolding *ZFC_def ZF_def ZF_schemes_def ZC_def Zermelo_fms_def by auto*

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

unfolding *ZF_def ZF_schemes_def Zermelo_fms_def ZF_fin_def by auto*

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

unfolding *ZFC_def ZF_def ZF_schemes_def ZC_def Zermelo_fms_def by auto*

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

unfolding *ZF_def ZF_schemes_def by auto*

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

unfolding *ZFC_def ZF_def ZF_schemes_def by auto*

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

```

formula} ⊆ ZFC
  unfolding ZFC_def ZF_def ZF_schemes_def by auto

lemma ZF_replacement_ground_repl_fm_type: {·Replacement(ground_repl_fm(φ)).
  · φ ∈ formula} ⊆ formula
  by auto

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: φ ∈ formula ⇒ env ∈ list(M) ⇒
                  arity(φ) ≤ 1 +ω length(env) ⇒
                  separation(##M,λx. (M, [x] @ env ⊨ φ))

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
using Lambda_in_M_fm_type
unfolding replacement_instances1_defs instances1_fms_def by simp

```

```

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 -

```

```

  obtain  $z$  where empty( $\#\#M, z$ )  $z \in M$ 
  using empty_intf[OF infinity_ax]
  by auto
  moreover from this
  have  $z=0$ 
  using transitivity empty_def
  by auto
  ultimately
  show ?thesis

```

by *simp*
qed

lemma *separation_in_ctm* :

assumes

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

$\text{arity}(\varphi) \leq 1 + \omega \text{ length}(\text{env})$ **and**

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

shows

$\text{separation}(\#\#M, Q)$

using *assms separation_ax satsQ transitivity*

separation_cong[of $\#\#M \lambda y. (M, [y]@env \models \varphi) Q$]

by *simp*

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* $\#\#M$

using *transitivity zero_in_M exI*[of $\lambda x. x \in M$]

by *unfold_locales simp_all*

sublocale *M_Z_trans* \subseteq *M_trivial* $\#\#M$

using *upair_ax Union_ax* **by** *unfold_locales*

7.2 Interface with *M_basic*

definition *Intersection* **where**

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

synthesize *Intersection* **from_definition** *Intersection* **assuming** *nonempty*

arity_theorem **for** *Intersection_fm*

definition *CartProd* **where**

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

synthesize *CartProd* **from_definition** *CartProd* **assuming** *nonempty*

arity_theorem **for** *CartProd_fm*

definition *ImageSep* **where**

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

synthesize *ImageSep* **from_definition** **assuming** *nonempty*

arity_theorem **for** *ImageSep_fm*

definition *Converse* where

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

synthesize *Converse* from_definition *Converse* assuming *nonempty*

arity_theorem for *Converse_fm*

definition *Restrict* where

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

synthesize *Restrict* from_definition *Restrict* assuming *nonempty*

arity_theorem for *Restrict_fm*

definition *Comp* where

$$\begin{aligned} \text{Comp}(N,R,S,xz) \equiv & \exists x[N]. \exists y[N]. \exists z[N]. \exists xy[N]. \exists yz[N]. \\ & \text{pair}(N,x,z,xz) \wedge \text{pair}(N,x,y,xy) \wedge \text{pair}(N,y,z,yz) \wedge xy \in S \wedge yz \in R \end{aligned}$$

synthesize *Comp* from_definition *Comp* assuming *nonempty*

arity_theorem for *Comp_fm*

definition *Pred* where

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

synthesize *Pred* from_definition *Pred* assuming *nonempty*

arity_theorem for *Pred_fm*

definition *is_Memrel* where

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

synthesize *is_Memrel* from_definition *is_Memrel* assuming *nonempty*

arity_theorem for *is_Memrel_fm*

definition *RecFun* where

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

synthesize *RecFun* from_definition *RecFun* assuming *nonempty*

arity_theorem for *RecFun_fm*

arity_theorem for *rtran_closure_mem_fm*

synthesize *wellfounded_trancl* from_definition assuming *nonempty*

arity_theorem for *wellfounded_trancl_fm*

context *M_Z_trans*

begin

lemma *inter_sep_intf* :

assumes $A \in M$
shows $\text{separation}(\#\#M, \lambda x . \forall y \in M . y \in A \longrightarrow x \in y)$
using $\text{assms separation_in_ctm}[of \text{Intersection_fm}(1,0) [A] \text{Intersection}(\#\#M,A)]$
 $\text{Intersection_iff_sats}[of 1 [_,A] A 0 _ M] \text{arity_Intersection_fm} \text{Intersection_fm_type}$
 $\text{ord_simp_union zero_in_M}$
unfolding Intersection_def
by simp

lemma diff_sep_intf :
assumes $B \in M$
shows $\text{separation}(\#\#M, \lambda x . x \notin B)$
using $\text{assms separation_in_ctm}[of \text{Neg}(\text{Member}(0,1)) [B] \lambda x . x \notin B] \text{ord_simp_union}$
by simp

lemma cartprod_sep_intf :
assumes $A \in M$ **and** $B \in M$
shows $\text{separation}(\#\#M, \lambda z . \exists x \in M . x \in A \wedge (\exists y \in M . y \in B \wedge \text{pair}(\#\#M, x, y, z)))$
using $\text{assms separation_in_ctm}[of \text{CartProd_fm}(1,2,0) [A,B] \text{CartProd}(\#\#M,A,B)]$
 $\text{CartProd_iff_sats}[of 1 [_,A,B] A 2 B 0 _ M] \text{arity_CartProd_fm} \text{CartProd_fm_type}$
 $\text{ord_simp_union zero_in_M}$
unfolding CartProd_def
by simp

lemma image_sep_intf :
assumes $A \in M$ **and** $B \in M$
shows $\text{separation}(\#\#M, \lambda y . \exists p \in M . p \in B \wedge (\exists x \in M . x \in A \wedge \text{pair}(\#\#M, x, y, p)))$
using $\text{assms separation_in_ctm}[of \text{ImageSep_fm}(1,2,0) [A,B] \text{ImageSep}(\#\#M,A,B)]$
 $\text{ImageSep_iff_sats}[of 1 [_,A,B] _ 2 _ 0 _ M] \text{arity_ImageSep_fm} \text{ImageSep_fm_type}$
 $\text{ord_simp_union zero_in_M}$
unfolding ImageSep_def
by simp

lemma converse_sep_intf :
assumes $R \in M$
shows $\text{separation}(\#\#M, \lambda z . \exists p \in M . p \in R \wedge (\exists x \in M . \exists y \in M . \text{pair}(\#\#M, x, y, p) \wedge \text{pair}(\#\#M, y, x, z)))$
using $\text{assms separation_in_ctm}[of \text{Converse_fm}(1,0) [R] \text{Converse}(\#\#M,R)]$
 $\text{Converse_iff_sats}[of 1 [_,R] _ 0 _ M] \text{arity_Converse_fm} \text{Converse_fm_type}$
 $\text{ord_simp_union zero_in_M}$
unfolding Converse_def
by simp

lemma restrict_sep_intf :
assumes $A \in M$
shows $\text{separation}(\#\#M, \lambda z . \exists x \in M . x \in A \wedge (\exists y \in M . \text{pair}(\#\#M, x, y, z)))$
using $\text{assms separation_in_ctm}[of \text{Restrict_fm}(1,0) [A] \text{Restrict}(\#\#M,A)]$

$Restrict_iff_sats[of\ 1\ [_,A]\ 0\ M]\ arity_Restrict_fm\ Restrict_fm_type$
 $ord_simp_union\ zero_in_M$
unfolding $Restrict_def$
by $simp$

lemma $comp_sep_intf$:
assumes $R \in M$ and $S \in M$
shows $separation(\#\#M, \lambda xz. \exists x \in M. \exists y \in M. \exists z \in M. \exists xy \in M. \exists yz \in M.$
 $pair(\#\#M, x, z, xz) \wedge pair(\#\#M, x, y, xy) \wedge pair(\#\#M, y, z, yz) \wedge xy \in S \wedge$
 $yz \in R)$
using $assms\ separation_in_ctm[of\ Comp_fm(1,2,0)\ [R,S]\ Comp(\#\#M,R,S)]$
 $Comp_iff_sats[of\ 1\ [_,R,S]\ 2\ 0\ M]\ arity_Comp_fm\ Comp_fm_type$
 $ord_simp_union\ zero_in_M$
unfolding $Comp_def$
by $simp$

lemma $pred_sep_intf$:
assumes $R \in M$ and $X \in M$
shows $separation(\#\#M, \lambda y. \exists p \in M. p \in R \wedge pair(\#\#M, y, X, p))$
using $assms\ separation_in_ctm[of\ Pred_fm(1,2,0)\ [R,X]\ Pred(\#\#M,R,X)]$
 $Pred_iff_sats[of\ 1\ [_,R,X]\ 2\ 0\ M]\ arity_Pred_fm\ Pred_fm_type$
 $ord_simp_union\ zero_in_M$
unfolding $Pred_def$
by $simp$

lemma $memrel_sep_intf$:
 $separation(\#\#M, \lambda z. \exists x \in M. \exists y \in M. pair(\#\#M, x, y, z) \wedge x \in y)$
using $separation_in_ctm[of\ is_Memrel_fm(0)\ []\ is_Memrel(\#\#M)]$
 $is_Memrel_iff_sats[of\ 0\ [_] \ M]\ arity_is_Memrel_fm\ is_Memrel_fm_type$
 $ord_simp_union\ zero_in_M$
unfolding is_Memrel_def
by $simp$

lemma $is_recfun_sep_intf$:
assumes $r \in M$ $f \in M$ $g \in M$ $a \in M$ $b \in M$
shows $separation(\#\#M, \lambda x. \exists xa \in M. \exists xb \in M.$
 $pair(\#\#M, x, a, xa) \wedge xa \in r \wedge pair(\#\#M, x, b, xb) \wedge xb \in r \wedge$
 $(\exists fx \in M. \exists gx \in M. fun_apply(\#\#M, f, x, fx) \wedge fun_apply(\#\#M, g, x, gx)$
 \wedge
 $fx \neq gx))$
using $assms\ separation_in_ctm[of\ RecFun_fm(1,2,3,4,5,0)\ [r,f,g,a,b]\ RecFun(\#\#M,r,f,g,a,b)]$
 $RecFun_iff_sats[of\ 1\ [_,r,f,g,a,b]\ 2\ 3\ 4\ 5\ 0\ M]\ arity_RecFun_fm$
 $RecFun_fm_type$
 $ord_simp_union\ zero_in_M$
unfolding $RecFun_def$
by $simp$

lemmas $M_basic_sep_instances =$
 $inter_sep_intf\ diff_sep_intf\ cartprod_sep_intf$

```

image_sep_intf converse_sep_intf restrict_sep_intf
pred_sep_intf memrel_sep_intf comp_sep_intf is_recfun_sep_intf

end — M_Z_trans

sublocale M_Z_trans ⊆ M_basic_no_repl ##M
  using power_ax M_basic_sep_instances
  by unfold_locales simp_all

lemma Replace_eq_Collect:
  assumes  $\bigwedge x y y'. x \in A \implies P(x,y) \implies P(x,y') \implies y=y'$  {y . x ∈ A, P(x, y)}
  ⊆ B
  shows {y . x ∈ A, P(x, y)} = {y ∈ B .  $\exists x \in A. P(x,y)$ }
  using assms by blast

context M_Z_trans
begin

lemma Pow_inter_M_closed: assumes A ∈ M shows Pow(A) ∩ M ∈ M
proof -
  have {a ∈ Pow(A) . a ∈ M} = Pow(A) ∩ M by auto
  then
  show ?thesis
    using power_ax powerset_abs assms unfolding power_ax_def
    by auto
qed

lemma Pow'_inter_M_closed: assumes A ∈ M shows {a ∈ Pow(A) . a ∈ M}
∈ M
  using power_ax powerset_abs assms unfolding power_ax_def by auto

end — M_Z_trans

context M_basic_no_repl
begin

lemma Replace_funspace_succ_rep_intf_sub:
  assumes
    M(A) M(n)
  shows
    {z . p ∈ A, funspace_succ_rep_intf_rel(M,p,z,n)}
    ⊆ PowM(PowM( $\bigcup \text{domain}(A) \cup (\{n\} \times \text{range}(A)) \cup (\bigcup (\{n\} \times \text{range}(A)))$ )))
  unfolding funspace_succ_rep_intf_rel_def using assms mem_Pow_rel_abs
  by clarsimp (auto simp: cartprod_def)

lemma funspace_succ_rep_intf_uniq:
  assumes
    funspace_succ_rep_intf_rel(M,p,z,n) funspace_succ_rep_intf_rel(M,p,z',n)
  shows

```

```

    z = z'
  using assms unfolding funspace_succ_rep_intf_rel_def by auto

lemma Replace_funspace_succ_rep_intf_eq:
  assumes
    M(A) M(n)
  shows
    {z . p ∈ A, funspace_succ_rep_intf_rel(M,p,z,n)} =
    {z ∈ PowM(PowM( $\bigcup$  domain(A)  $\cup$  ({n}  $\times$  range(A))  $\cup$  ( $\bigcup$  ({n}  $\times$  range(A))))
    .
     $\exists p \in A. \text{funspace\_succ\_rep\_intf\_rel}(M,p,z,n)$ 
  using assms Replace_eq_Collect[OF funspace_succ_rep_intf_uniq, of A,
    OF  $\_ \_$  Replace_funspace_succ_rep_intf_sub[of A n], of  $\lambda x y z. x \lambda x y z. n$ ]
  by (intro equalityI)
    (auto dest:transM simp:funspace_succ_rep_intf_rel_def)

end — M_basic_no_repl

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

relationalize fsri is_fsri
synthesize is_fsri from_definition assuming nonempty
arity_theorem for is_fsri_fm

context M_Z_trans
begin

lemma separation_fsri:
  ( $\#\#M$ )(A)  $\implies$  ( $\#\#M$ )(B)  $\implies$  separation( $\#\#M$ , is_fsri( $\#\#M$ ,A,B))
  using separation_in_ctm[where env=[A,B] and  $\varphi = \text{is\_fsri\_fm}(1,2,0)$ ]
    zero_in_M is_fsri_iff_sats[symmetric] arity_is_fsri_fm is_fsri_fm_type
  by (simp_all add: ord_simp_union)

lemma separation_funspace_succ_rep_intf_rel:
  ( $\#\#M$ )(A)  $\implies$  ( $\#\#M$ )(B)  $\implies$  separation( $\#\#M$ ,  $\lambda z. \exists p \in A. \text{funspace\_succ\_rep\_intf\_rel}(\#\#M,p,z,B)$ )
  using separation_fsri zero_in_M
  by (rule_tac separation_cong[THEN iffD1, of  $\_ \text{is\_fsri}(\#\#M,A,B)$ ])
    (auto simp flip:setclass_iff dest:transM
      simp:is_fsri_def funspace_succ_rep_intf_rel_def, force)

lemma Replace_funspace_succ_rep_intf_in_M:
  assumes
    A ∈ M n ∈ M
  shows
    {z . p ∈ A, funspace_succ_rep_intf_rel( $\#\#M$ ,p,z,n)} ∈ M
  proof -
    have ( $\#\#M$ )( $\{z \in \text{Pow}^M(\text{Pow}^M(\bigcup \text{domain}(A) \cup (\{n\} \times \text{range}(A)) \cup (\bigcup \{n\}))$ 

```

```

× range(A))))).
  ∃ p ∈ A. funspace_succ_rep_intf_rel(##M,p,z,n))
using assms separation_funspace_succ_rep_intf_rel
by (intro separation_closed) (auto simp flip:setclass_iff)
with assms
show ?thesis
using Replace_funspace_succ_rep_intf_eq by auto
qed

lemma funspace_succ_rep_intf:
assumes  $n \in M$ 
shows
  strong_replacement(##M,
     $\lambda p z. \exists f \in M. \exists b \in M. \exists nb \in M. \exists cnbf \in M.$ 
     $\text{pair}(##M,f,b,p) \wedge \text{pair}(##M,n,b,nb) \wedge \text{is\_cons}(##M,nb,f,cnbf) \wedge$ 
     $\text{upair}(##M,cnbf,cnbf,z)$ )
using assms pair_in_M_iff[simplified] cons_closed[simplified]
unfolding strong_replacement_def univalent_def
apply (clarsimp, rename_tac A)
apply (rule_tac x={z . p ∈ A, funspace_succ_rep_intf_rel(##M,p,z,n)} in
beXI)
apply (auto simp:funspace_succ_rep_intf_rel_def
  Replace_funspace_succ_rep_intf_in_M[unfolded funspace_succ_rep_intf_rel_def,
simplified])
done

end — M_Z_trans

sublocale M_Z_trans  $\subseteq$  M_basic ##M
using power_ax M_basic_sep_instances funspace_succ_rep_intf
by unfold_locales auto

```

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))
using assms separation_in_ctm[of rtran_closure_mem_fm(1,2,0) [A,r] rtran_closure_mem(##M,A,r)]
arity_rtran_closure_mem_fm ord_simp_union zero_in_M
by simp

```

```

lemma wftrancl_separation_intf:
assumes  $r \in M$  and  $Z \in M$ 
shows separation (##M, wellfounded_trancl(##M,Z,r))
using assms separation_in_ctm[of wellfounded_trancl_fm(1,2,0) [Z,r] well-
founded_trancl(##M,Z,r)]

```

arity_wellfounded_trancl_fm ord_simp_union zero_in_M
by simp

To prove $\omega \in M$ we get an infinite set I from *infinity_ax* closed under 0 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, $\lambda x. x \in \text{nat}$)*
proof -
 have $(\forall v \in M. \text{separation}(##M, \lambda x. (M, [x, v] \models \text{finite_ordinal_fm}(0))))$
 using *separation_ax arity_finite_ordinal_fm*
 by simp
 then
 have $(\forall v \in M. \text{separation}(##M, \text{finite_ordinal}(##M)))$
 unfolding *separation_def*
 by simp
 then
 have *separation(##M, finite_ordinal(##M))*
 using *separation_in_ctm zero_in_M*
 by auto
 then
 show *?thesis*
 unfolding *separation_def*
 by simp
qed

lemma nat_subset_I: $\exists I \in M. \text{nat} \subseteq I$
proof -
 have $\text{nat} \subseteq I$
 if $I \in M$ **and** $0 \in I$ **and** $\bigwedge x. x \in I \implies \text{succ}(x) \in I$ **for** I
 using *that*
 by $(\text{rule_tac } \text{subsetI}, \text{induct_tac } x, \text{simp_all})$
 moreover
 obtain I **where**
 $I \in M$ $0 \in I$ $\bigwedge x. x \in I \implies \text{succ}(x) \in I$
 using *infinity_ax transitivity*
 unfolding *infinity_ax_def*
 by auto
 ultimately
 show *?thesis*
 by auto
qed

lemma nat_in_M: $\text{nat} \in M$
proof -
 have $\{x \in B \mid x \in A\} = A$ **if** $A \subseteq B$ **for** $A B$
 using *that* **by auto**
 moreover
 obtain I **where**
 $I \in M$ $\text{nat} \subseteq I$

```

    using nat_subset_I by auto
  moreover from this
  have {x∈I . x∈nat} ∈ M
    using finite_sep_intf separation_closed[of λx . x∈nat]
    by simp
  ultimately
  show ?thesis
    by simp
qed

end — M_ZF1_trans

sublocale M_ZF1_trans ⊆ M_trancl ##M
  using rtrancl_separation_intf wtrancl_separation_intf nat_in_M
  wellfounded_trancl_def
  by unfold_locales auto

```

7.4 Interface with *M_eclose*

```

lemma repl_sats:
  assumes
    sat:  $\bigwedge x z. x \in M \implies z \in M \implies (M, \text{Cons}(x, \text{Cons}(z, \text{env})) \models \varphi) \longleftrightarrow P(x, z)$ 
  shows
    strong_replacement(##M,  $\lambda x z. (M, \text{Cons}(x, \text{Cons}(z, \text{env})) \models \varphi) \longleftrightarrow$ 
      strong_replacement(##M, P)
    by (rule strong_replacement_cong, simp add: sat)

```

```

arity_theorem for list_functor_fm
arity_theorem for formula_functor_fm
arity_theorem for Inl_fm
arity_theorem for Inr_fm
arity_theorem for Nil_fm
arity_theorem for Cons_fm
arity_theorem for quasilist_fm
arity_theorem for tl_fm
arity_theorem for big_union_fm

```

```

context M_ZF1_trans
begin

```

This lemma obtains *iterates_replacement* for predicates without parameters.

```

lemma iterates_repl_intf :
  assumes
    v∈M and
    isfm: is_F_fm ∈ formula and
    arty: arity(is_F_fm) = 2 and
    satsf:  $\bigwedge a b \text{ env}'. \llbracket a \in M ; b \in M ; \text{env}' \in \text{list}(M) \rrbracket$ 
       $\implies \text{is}_F(a, b) \longleftrightarrow (M, [b, a]@\text{env}' \models \text{is}_F\_fm)$ 
  and is_F_fm_replacement:

```

```

 $\wedge env. (\exists \cdot \langle 1, 0 \rangle \text{ is } 2 \cdot \wedge \text{is\_wfrec\_fm}(\text{iterates\_MH\_fm}(\text{is\_F\_fm}, 9, 2, 1, 0), 3, 1, 0)$ 
 $\cdot) \in \text{formula} \implies env \in \text{list}(M) \implies$ 
 $\text{arity}((\exists \cdot \langle 1, 0 \rangle \text{ is } 2 \cdot \wedge \text{is\_wfrec\_fm}(\text{iterates\_MH\_fm}(\text{is\_F\_fm}, 9, 2, 1, 0), 3, 1, 0)$ 
 $\cdot)) \leq 2 +_{\omega} \text{length}(env) \implies$ 
 $\text{strong\_replacement}(\#\#M, \lambda x y.$ 
 $M, [x, y] @ env \models (\exists \cdot \langle 1, 0 \rangle \text{ is } 2 \cdot \wedge \text{is\_wfrec\_fm}(\text{iterates\_MH\_fm}(\text{is\_F\_fm}, 9, 2, 1, 0), 3, 1, 0)$ 
 $\cdot))$ 
shows
 $\text{iterates\_replacement}(\#\#M, \text{is\_F}, v)$ 
proof -
let  $?f = (\exists \cdot \langle 1, 0 \rangle \text{ is } 2 \cdot \wedge \text{is\_wfrec\_fm}(\text{iterates\_MH\_fm}(\text{is\_F\_fm}, 9, 2, 1, 0), 3, 1, 0)$ 
 $\cdot)$ 
have  $\text{arity}(?f) = 4$   $?f \in \text{formula}$ 
using  $\text{arity\_iterates\_MH\_fm}$  [where  $\text{isF} = \text{is\_F\_fm}$  and  $i = 2$ ]
 $\text{arity\_wfrec\_replacement\_fm}$  [where  $i = 10$ ]  $\text{isfm}$   $\text{arty}$   $\text{ord\_simp\_union}$ 
by  $\text{simp\_all}$ 
{
fix  $n$ 
assume  $n \in \text{nat}$ 
then
have  $\text{Memrel}(\text{succ}(n)) \in M$ 
using  $\text{nat\_into\_M}$   $\text{Memrel\_closed}$ 
by  $\text{simp}$ 
moreover
{
fix  $a0 a1 a2 a3 a4 y x z$ 
assume  $[a0, a1, a2, a3, a4, y, x, z] \in \text{list}(M)$ 
moreover
note  $\langle v \in M \rangle \langle \text{Memrel}(\text{succ}(n)) \in M \rangle$ 
moreover from calculation
have  $(M, [b, a, c, d, a0, a1, a2, a3, a4, y, x, z, \text{Memrel}(\text{succ}(n)), v] \models \text{is\_F\_fm}) \longleftrightarrow$ 
 $\text{is\_F}(a, b)$ 
if  $a \in M$   $b \in M$   $c \in M$   $d \in M$  for  $a$   $b$   $c$   $d$ 
using  $\text{that satsf}$  [of  $a$   $b$   $[c, d, a0, a1, a2, a3, a4, y, x, z, \text{Memrel}(\text{succ}(n)), v]$ ]
by  $\text{simp}$ 
moreover from calculation
have  $(M, [a0, a1, a2, a3, a4, y, x, z, \text{Memrel}(\text{succ}(n)), v] \models \text{iterates\_MH\_fm}(\text{is\_F\_fm}, 9, 2, 1, 0))$ 
 $\longleftrightarrow$ 
 $\text{iterates\_MH}(\#\#M, \text{is\_F}, v, a2, a1, a0)$ 
using  $\text{sats\_iterates\_MH\_fm}$  [of  $M$   $\text{is\_F}$   $\text{is\_F\_fm}$ ]
by  $\text{simp}$ 
}
moreover from calculation
have  $(M, [y, x, z, \text{Memrel}(\text{succ}(n)), v] \models \text{is\_wfrec\_fm}(\text{iterates\_MH\_fm}(\text{is\_F\_fm}, 9, 2, 1, 0), 3, 1, 0))$ 
 $\longleftrightarrow$ 
 $\text{is\_wfrec}(\#\#M, \text{iterates\_MH}(\#\#M, \text{is\_F}, v), \text{Memrel}(\text{succ}(n)), x, y)$ 
if  $y \in M$   $x \in M$   $z \in M$  for  $y$   $x$   $z$ 
using  $\text{that sats\_is\_wfrec\_fm}$   $\langle v \in M \rangle$  by  $\text{simp}$ 
moreover from calculation

```

```

have (M, [x,z,Memrel(succ(n)),v] ⊨ ?f) ↔
  (∃ y∈M. pair(##M,x,y,z) ∧
   is_wfrec(##M, iterates_MH(##M,is_F,v) , Memrel(succ(n)), x, y))
if x∈M z∈M for x z
using that ⟨v∈M⟩
by (simp del:pair_abs)
moreover
note ⟨arity(?f) = 4⟩ ⟨?f∈formula⟩
moreover from calculation ⟨v∈_⟩
have strong_replacement(##M,λx z. (M, [x,z,Memrel(succ(n)),v] ⊨ ?f))
using is_F_fm_replacement
by simp
ultimately
have strong_replacement(##M,λx z.
  ∃ y∈M. pair(##M,x,y,z) ∧ is_wfrec(##M, iterates_MH(##M,is_F,v)
  ,
  Memrel(succ(n)), x, y))
using repl_sats[of M ?f [Memrel(succ(n)),v]]
by (simp del:pair_abs)
}
then
show ?thesis
  unfolding iterates_replacement_def wfrec_replacement_def
  by simp
qed

lemma eclose_repl1_intf:
assumes A∈M
shows iterates_replacement(##M, big_union(##M), A)
using assms arity_big_union_fm
  iterates_repl_intf[where is_F_fm=big_union_fm(1,0)]
  replacement_ax1(1)[unfolded replacement_assm_def]
  ord_simp_union
by simp

lemma eclose_repl2_intf:
assumes A∈M
shows strong_replacement(##M,λn y. n∈nat ∧ is_iterates(##M, big_union(##M),
A, n, y))
proof -
let ?f = And(Member(0,3),is_iterates_fm(big_union_fm(1,0),2,0,1))
note nat_in_M ⟨A∈M⟩
moreover from this
have big_union(##M,a,b) ↔
  (M, [b,a,c,d,e,f,g,h,i,j,k,n,y,A,nat] ⊨ big_union_fm(1,0))
if a∈M b∈M c∈M d∈M e∈M f∈Mg∈Mh∈Mi∈Mj∈M k∈M n∈M y∈M
for a b c d e f g h i j k n y
using that by simp

```

```

moreover from calculation
have (M, [n,y,A,nat] ⊨ is_iterates_fm(big_union_fm(1,0),2,0,1)) ↔
  is_iterates(##M, big_union(##M), A, n, y)
  if n∈M y∈M for n y
  using that sats_is_iterates_fm[of M big_union(##M)]
  by simp
moreover from calculation
have (M, [n,y,A,nat] ⊨ ?f) ↔
  n∈nat ∧ is_iterates(##M, big_union(##M), A, n, y)
  if n∈M y∈M for n y
  using that
  by simp
moreover
have arity(?f) = 4
  using arity_is_iterates_fm[where p=big_union_fm(1,0) and i=2]
  arity_big_union_fm arity_And ord_simp_union
  by simp
ultimately
show ?thesis
  using repl_sats[of M ?f [A,nat]] replacement_ax1(2)[unfolded replacement_assm_def]
  by simp
qed

end — M_ZF1_trans

sublocale M_ZF1_trans ⊆ M_eclose ##M
  using eclose_repl1_intf eclose_repl2_intf
  by unfold_locales auto

Interface with M_eclose.
schematic_goal sats_is_Vset_fm_auto:
  assumes
    i∈nat v∈nat env∈list(A) 0∈A
    i < length(env) v < length(env)
  shows
    is_Vset(##A,nth(i, env),nth(v, env)) ↔ (A, env ⊨ ?ivs_fm(i,v))
  unfolding is_Vset_def is_Vfrom_def
  by (insert assms; (rule sep_rules is_HVfrom_iff_sats is_transrec_iff_sats |
simp)+)

synthesize is_Vset from_schematic sats_is_Vset_fm_auto
arity_theorem for is_Vset_fm

declare is_Hrank_fm_def[fm_definitions add]

context M_ZF1_trans
begin

lemma wfrec_rank :

```

```

assumes  $X \in M$ 
shows  $wfrec\_replacement(\#\#M, is\_Hrank(\#\#M), rrank(X))$ 
proof -
let  $?f = \text{Exists}(\text{And}(\text{pair\_fm}(1,0,2), is\_wfrec\_fm(is\_Hrank\_fm(2,1,0), 3,1,0)))$ 
note  $assms\_zero\_in\_M$ 
moreover from this
have
   $is\_Hrank(\#\#M, a2, a1, a0) \longleftrightarrow$ 
     $(M, [a0, a1, a2, a3, a4, y, x, z, rrank(X)] \models is\_Hrank\_fm(2,1,0))$ 
  if  $a4 \in M \ a3 \in M \ a2 \in M \ a1 \in M \ a0 \in M \ y \in M \ x \in M \ z \in M$  for  $a4 \ a3 \ a2 \ a1 \ a0 \ y \ x \ z$ 
  using  $that \ rrank\_in\_M \ is\_Hrank\_iff\_sats$ 
  by  $simp$ 
moreover from calculation
have  $(M, [y, x, z, rrank(X)] \models is\_wfrec\_fm(is\_Hrank\_fm(2,1,0), 3,1,0)) \longleftrightarrow$ 
   $is\_wfrec(\#\#M, is\_Hrank(\#\#M), rrank(X), x, y)$ 
  if  $y \in M \ x \in M \ z \in M$  for  $y \ x \ z$ 
  using  $that \ rrank\_in\_M \ sats \ is\_wfrec\_fm$ 
  by  $simp$ 
moreover from calculation
have  $(M, [x, z, rrank(X)] \models ?f) \longleftrightarrow$ 
   $(\exists y \in M. \text{pair}(\#\#M, x, y, z) \wedge is\_wfrec(\#\#M, is\_Hrank(\#\#M),$ 
 $rrank(X), x, y))$ 
  if  $x \in M \ z \in M$  for  $x \ z$ 
  using  $that \ rrank\_in\_M$ 
  by  $(simp \ del: \text{pair\_abs})$ 
moreover
have  $arity(?f) = 3$ 
using  $arity\_wfrec\_replacement\_fm$  where  $p = is\_Hrank\_fm(2,1,0)$  and  $i = 3, simplified$ 
   $arity\_is\_Hrank\_fm[of \ 2 \ 1 \ 0, simplified] \ ord\_simp\_union$ 
  by  $simp$ 
moreover from calculation
have  $strong\_replacement(\#\#M, \lambda x z. (M, [x, z, rrank(X)] \models ?f))$ 
  using  $replacement\_ax1(3)[unfolded \ replacement\_assm\_def] \ rrank\_in\_M$ 
  by  $simp$ 
ultimately
show  $?thesis$ 
  using  $repl\_sats[of \ M \ ?f \ [rrank(X)]]$ 
  unfolding  $wfrec\_replacement\_def$ 
  by  $(simp \ del: \text{pair\_abs})$ 
qed

```

lemma $trans_repl_HVFrom :$

assumes $A \in M \ i \in M$

shows $transrec_replacement(\#\#M, is_HVfrom(\#\#M, A), i)$

proof -

let $?f = \text{Exists}(\text{And}(\text{pair_fm}(1,0,2), is_wfrec_fm(is_HVfrom_fm(8,2,1,0), 4,1,0)))$

note $facts = assms_zero_in_M$

moreover

have $\exists sa \in M. \exists esa \in M. \exists mesa \in M.$

```

    upair(##M,a,a,sa) ∧ is_eclose(##M,sa,esa) ∧ membership(##M,esa,mesa)
  if a∈M for a
  using that upair_ax eclose_closed Memrel_closed
  unfolding upair_ax_def
  by (simp del:upair_abs)
moreover
{
  fix mesa
  assume mesa∈M
  moreover
  note facts
  moreover from calculation
  have is_HVfrom(##M,A,a2, a1, a0) ↔
    (M, [a0,a1,a2,a3,a4,y,x,z,A,mesa] ⊨ is_HVfrom_fm(8,2,1,0))
  if a4∈M a3∈M a2∈M a1∈M a0∈M y∈M x∈M z∈M for a4 a3 a2 a1 a0 y x z
  using that sats_is_HVfrom_fm
  by simp
  moreover from calculation
  have (M, [y,x,z,A,mesa] ⊨ is_wfrec_fm(is_HVfrom_fm(8,2,1,0),4,1,0)) ↔
    is_wfrec(##M, is_HVfrom(##M,A),mesa, x, y)
  if y∈M x∈M z∈M for y x z
  using that sats_is_wfrec_fm
  by simp
  moreover from calculation
  have (M, [x,z,A,mesa] ⊨ ?f) ↔
    (∃ y∈M. pair(##M,x,y,z) ∧ is_wfrec(##M, is_HVfrom(##M,A) ,
mesa, x, y))
  if x∈M z∈M for x z
  using that
  by (simp del:pair_abs)
  moreover
  have arity(?f) = 4
  using arity_wfrec_replacement_fm[where p=is_HVfrom_fm(8,2,1,0) and
i=9]
  arity_is_HVfrom_fm ord_simp_union
  by simp
  moreover from calculation
  have strong_replacement(##M,λx z. (M, [x,z,A,mesa] ⊨ ?f))
  using replacement_ax1(4)[unfolded replacement_assm_def]
  by simp
  ultimately
  have wfrec_replacement(##M,is_HVfrom(##M,A),mesa)
  using repl_sats[of M ?f [A,mesa]]
  unfolding wfrec_replacement_def
  by (simp del:pair_abs)
}
ultimately
show ?thesis
  unfolding transrec_replacement_def

```

by *simp*
qed

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

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

shows

$\{y \in A . Q(y)\} \in M$

proof -

have *separation*($\#\#M, \lambda x. (M, [x]@env \models \varphi)$)

using *assms separation_ax* by *simp*

then

show *?thesis*

using $\langle A \in M \rangle$ *satsQ* *transitivity separation_closed*

separation_cong[of $\#\#M \ \lambda y. (M, [y]@env \models \varphi) \ Q$]

by *simp*

qed

— This version has a weaker assumption.

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

$\text{sats}Q: \bigwedge x. x \in A \implies (M, [x]@env \models \varphi) \longleftrightarrow Q(x)$

shows

$\{y \in A . Q(y)\} \in M$

proof -

let $? \varphi' = \text{And}(\varphi, \text{Member}(0, \text{length}(\text{env}) +_{\omega} 1))$

note *assms*

moreover

have $\text{arity}(\varphi') \leq 1 +_{\omega} \text{length}(\text{env}@[A])$

using *assms Un_le le_trans*[of $\text{arity}(\varphi) \ 1 +_{\omega} \text{length}(\text{env}) \ 2 +_{\omega} \text{length}(\text{env})$]

by (*force simp:FOL_aritys*)

moreover from *calculation*

have $? \varphi' \in \text{formula } \text{nth}(\text{length}(\text{env}), \text{env} @ [A]) = A$

using *nth_append*

by *auto*

moreover from *calculation*

have $\bigwedge x . x \in M \implies (M, [x]@env@[A] \models ? \varphi') \longleftrightarrow Q(x) \wedge x \in A$

using *arity_sats_iff*[of $_ [A] _ []@env$]

by auto
ultimately
show *?thesis*
using *Collect_in_M[of ? φ' env@[A] $\lambda x . Q(x) \wedge x \in A$, OF $_ _ _ \langle A \in M \rangle$]*
by auto
qed

end — *M_ZF1_trans*

context *M_Z_trans*
begin

lemma *strong_replacement_in_ctm:*

assumes

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

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

env $\in \text{list}(M)$

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

using *assms*

strong_replacement_cong[of $\#\#M \lambda x y. M, [x, y]@env \models \varphi \lambda x y. y = f(x)$]

unfolding *replacement_assm_def*

by auto

lemma *strong_replacement_rel_in_ctm :*

assumes

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

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

env $\in \text{list}(M)$

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

using *assms*

strong_replacement_cong[of $\#\#M \lambda x y. M, [x, y]@env \models \varphi f$]

unfolding *replacement_assm_def*

by auto

lemma *Replace_in_M :*

assumes

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

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

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

)

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

proof -
let $?\varphi' = \text{And}(\varphi, \text{Member}(0, \text{length}(\text{env}) +_{\omega} 2))$
note *assms*
moreover from this
have $\text{arity}(\varphi') \leq 2 +_{\omega} \text{length}(\text{env}@[A])$
using $\text{Un_le_le_trans}[\text{of } \text{arity}(\varphi) \ 2 +_{\omega} (\text{length}(\text{env})) \ 3 +_{\omega} \text{length}(\text{env})]$
by (*force simp:FOL_aritys*)
moreover from calculation
have $? \varphi' \in \text{formula } \text{nth}(\text{length}(\text{env}), \text{env} @ [A]) = A$
using *nth_append by auto*
moreover from calculation
have $\bigwedge x y. x \in M \implies y \in M \implies (M, [x, y] @ \text{env} @ [A] \models ?\varphi') \longleftrightarrow y = f(x) \wedge x \in A$
using *arity_sats_iff[of _ [A] _ [_, _] @ env]*
by auto
moreover from calculation
have *strong_replacement*($\#\#M, \lambda x y. M, [x, y] @ \text{env} @ [A] \models ?\varphi'$)
using *phi'_replacement assms(1-6) unfolding replacement_assm_def by simp*
ultimately
have $\exists \text{strong_replacement}(\#\#M, \lambda x y. y = f(x) \wedge x \in A)$
using
strong_replacement_cong[of $\#\#M \ \lambda x y. M, [x, y] @ \text{env} @ [A] \models ?\varphi' \ \lambda x y. y = f(x) \wedge x \in A$]
by simp
then
have $\{y . x \in A, y = f(x)\} \in M$
using $\langle A \in M \rangle$ *strong_replacement_closed*[OF \exists , of A] *fclosed by simp*
moreover
have $\{f(x). x \in A\} = \{y . x \in A, y = f(x)\}$
by auto
ultimately
show *?thesis by simp*
qed

lemma *Replace_relativized_in_M* :

assumes

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

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

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

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

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

$A \in M$ *env* $\in \text{list}(M)$ **and**

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

)

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

using *assms Replace_in_M*[of φ] **by auto**

lemma *ren_action* :

assumes

env $\in \text{list}(M)$ $x \in M$ $y \in M$ $z \in M$

```

shows  $\forall i . i < 2 + \omega \text{length}(env) \longrightarrow$ 
       $\text{nth}(i, [x, z] @ env) = \text{nth}(\rho\_repl(\text{length}(env))) 'i, [z, x, y] @ env$ 
proof -
  let  $?f = \{\langle 0, 1 \rangle, \langle 1, 0 \rangle\}$ 
  have  $1: (\bigwedge j. j < \text{length}(env) \implies \text{nth}(j, env) = \text{nth}(\text{id}(\text{length}(env))) 'j, env)$ 
    using assms ltD by simp
  have  $2: \text{nth}(j, [x, z]) = \text{nth} (?f 'j, [z, x, y])$  if  $j < 2$  for  $j$ 
  proof -
    consider  $j=0 \mid j=1$  using ltD[OF <j<2>] by auto
    then show ?thesis
    proof(cases)
      case  $1$ 
      then show ?thesis using apply_equality f_type by simp
    next
      case  $2$ 
      then show ?thesis using apply_equality f_type by simp
    qed
  qed
  show ?thesis
    using sum_action[OF _____ f_type id_type _____ 2 1, simplified]
assms
    unfolding  $\rho\_repl\_def$  by simp
  qed

lemma Lambda_in_M :
  assumes
    f_fm:  $\varphi \in \text{formula}$  and
    f_ar:  $\text{arity}(\varphi) \leq 2 + \omega \text{length}(env)$  and
    fsats:  $\bigwedge x y. x \in A \implies y \in M \implies (M, [x, y] @ env \models \varphi) \longleftrightarrow \text{is}_f(x, y)$  and
    fabs:  $\bigwedge x y. x \in A \implies y \in M \implies \text{is}_f(x, y) \longleftrightarrow y = f(x)$  and
    fclosed:  $\bigwedge x. x \in A \implies f(x) \in M$  and
     $A \in M$   $env \in \text{list}(M)$  and
    phi'_replacement2:  $\text{replacement\_assm}(M, env @ [A], \text{Lambda\_in\_M\_fm}(\varphi, \text{length}(env)))$ 
  shows  $(\lambda x \in A . f(x)) \in M$ 
  unfolding lam_def
  proof -
    let  $?ren = \rho\_repl(\text{length}(env))$ 
    let  $?j = 2 + \omega \text{length}(env)$ 
    let  $?k = 3 + \omega \text{length}(env)$ 
    let  $?psi = ren(\varphi) ' ?j ' ?k ' ?ren$ 
    let  $?phi' = \text{Exists}(\text{And}(\text{pair\_fm}(1, 0, 2), ?psi))$ 
    let  $?p = \lambda x y. \exists z \in M. \text{pair}(\#\#M, x, z, y) \wedge \text{is}_f(x, z)$ 
    have  $?phi' \in \text{formula}$   $?psi \in \text{formula}$ 
    using  $\langle env \in \_ \rangle \text{length\_type } f\_fm \text{ ren\_type } ren\_tc[\text{of } \varphi \ 2 + \omega \text{length}(env) \ 3 + \omega \text{length}(env)$ 
       $?ren]$ 
    by simp_all
    moreover from this
    have  $\text{arity} (?psi) \leq 3 + \omega (\text{length}(env))$   $\text{arity} (?psi) \in \text{nat}$ 
    using assms arity_ren[OF f_fm _ _ ren_type, of length(env)] by simp_all

```

```

then
have arity(?φ') ≤ 2+ω(length(env))
  using Un_le pred_Un_distrib assms pred_le
  by (simp add:arity)
moreover from this calculation
have x∈A ⇒ y∈M ⇒ (M,[x,y]@env ⊨ ?φ') ↔ ?p(x,y) for x y
  using ⟨env∈_⟩ length_type[OF ⟨env∈_⟩] assms transitivity[OF _ ⟨A∈M⟩]
  sats_iff_sats_ren[OF f_fm _ _ _ ren_type f_ar ren_action[rule_format,of
_ x y],of _ M ]
  by auto
moreover
have x∈A ⇒ y∈M ⇒ ?p(x,y) ↔ y = ⟨x,f(x)⟩ for x y
  using assms transitivity[OF _ ⟨A∈_⟩] fclosed
  by simp
moreover
have ∧ x . x∈A ⇒ ⟨x,f(x)⟩ ∈ M
  using transitivity[OF _ ⟨A∈M⟩] pair_in_M_iff fclosed by simp
ultimately
show {⟨x,f(x)⟩ . x∈A } ∈ M
  using Replace_in_M[of ?φ' env A] phi'_replacement2 ⟨A∈M⟩ ⟨env∈_⟩
  by simp
qed

```

lemma ren_action' :

```

assumes
  env∈list(M) x∈M y∈M z∈M u∈M
shows ∀ i . i < 3+ω(length(env)) →
  nth(i,[x,z,u]@env) = nth(ρ_pair_repl(length(env)) 'i,[x,z,y,u]@env)

```

proof -

```

let ?f={⟨0, 0⟩, ⟨1, 1⟩, ⟨2,3⟩}
have 1:(∧j. j < length(env) ⇒ nth(j, env) = nth(id(length(env)) 'j, env))
  using assms ltD by simp
have 2:nth(j, [x,z,u]) = nth(?f 'j, [x,z,y,u]) if j<3 for j

```

proof -

```

consider j=0 | j=1 | j=2 using ltD[OF ⟨j<3⟩] by auto
then show ?thesis

```

proof(cases)

case 1

```

then show ?thesis using apply_equality f_type' by simp

```

next

case 2

```

then show ?thesis using apply_equality f_type' by simp

```

next

case 3

```

then show ?thesis using apply_equality f_type' by simp

```

qed

qed

show ?thesis

```

using sum_action[OF _ _ _ f_type' id_type _ _ _ _ 2 1,simplified]

```

assms

unfolding $\rho_pair_repl_def$ **by** *simp*
qed

lemma *LambdaPair_in_M* :

assumes

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

f_ar: $arity(\varphi) \leq 3 + \omega \ length(env)$ **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 $(\lambda x \in A. f(fst(x), snd(x))) \in M$

proof -

let $?ren = \rho_pair_repl(length(env))$

let $?j = 3 + \omega \ length(env)$

let $?k = 4 + \omega \ length(env)$

let $?psi = ren(\varphi) ' ?j ' ?k ' ?ren$

let $?phi' = Exists(Exists(And(fst_fm(2, 0), (And(snd_fm(2, 1), ?psi)))))$

let $?p = \lambda x y. is_f(fst(x), snd(x), y)$

have $?phi' \in formula \ ?psi \in formula$

using $\langle env \in _ \rangle \ length_type \ f_fm \ ren_type' \ ren_tc[of \ \varphi \ ?j \ ?k \ ?ren]$

by *simp_all*

moreover from *this*

have $arity(?psi) \leq 4 + \omega \ (length(env)) \ arity(?psi) \in nat$

using *assms* $arity_ren[OF \ f_fm \ _ \ _ \ ren_type', of \ length(env)]$ **by** *simp_all*

moreover from *calculation*

have $1: arity(?phi') \leq 2 + \omega \ (length(env)) \ ?phi' \in formula$

using *Un_le_pred_Un_distrib* *assms* *pred_le*

by (*simp_all* *add:arity*)

moreover from *this* *calculation*

have $2: x \in A \implies y \in M \implies (M, [x, y] @ env \models ?phi') \longleftrightarrow ?p(x, y)$ **for** $x \ y$

using

sats_iff_sats_ren[OF \ f_fm \ _ \ _ \ ren_type' \ f_ar

ren_action'[rule_format, of \ fst(x) \ x \ snd(x) \ y], simplified]

$\langle env \in _ \rangle \ length_type[OF \ \langle env \in _ \rangle]$ *transitivity[OF \ _ \ \langle A \in M \rangle]*

fst_snd_closed_pair_in_M_iff fsats[of \ fst(x) \ snd(x) \ y, symmetric]

fst_abs_snd_abs

by *auto*

moreover from *assms*

have $3: x \in A \implies y \in M \implies ?p(x, y) \longleftrightarrow y = f(fst(x), snd(x))$ **for** $x \ y$

using *fclosed* *fst_snd_closed_pair_in_M_iff* *fabs* *transitivity*

by *auto*

moreover

have $4: \bigwedge x. x \in A \implies \langle x, f(fst(x), snd(x)) \rangle \in M \wedge x. x \in A \implies f(fst(x), snd(x))$

$\in M$

using *transitivity[OF \ _ \ \langle A \in M \rangle]* *pair_in_M_iff* *fclosed* *fst_snd_closed*

```

    by simp_all
  ultimately
  show ?thesis
    using Lambda_in_M[unfolded Lambda_in_M_fm_def, of ?φ', OF _ _ _ _]
  ---
    phi'_replacement3[unfolded LambdaPair_in_M_fm_def]
  ⟨env∈_⟩ ⟨A∈_⟩ by simp

qed

lemma (in M_ZF1_trans) lam_replacement2_in_ctm :
  assumes
    f_fm: φ ∈ formula and
    f_ar: arity(φ) ≤ 3 + ω length(env) and
    fsats: ∧ x z r. x ∈ M ⇒ z ∈ M ⇒ r ∈ M ⇒ (M, [x, z, r]@env ⊨ φ) ↔ is_f(x, z, r)
  and
    fabs: ∧ x z r. x ∈ M ⇒ z ∈ M ⇒ r ∈ M ⇒ is_f(x, z, r) ↔ r = f(x, z) and
    fclosed: ∧ x z. x ∈ M ⇒ z ∈ M ⇒ f(x, z) ∈ M and
    env ∈ list(M) and
    phi'_replacement3: ∧ A. A ∈ M ⇒ replacement_assm(M, env@[A], LambdaPair_in_M_fm(φ, length(env)))
  shows lam_replacement(##M, λx. f(fst(x), snd(x)))
  using
    LambdaPair_in_M fabs
    f_ar ord_simp_union transitivity assms fst_snd_closed
  by (rule_tac lam_replacement_iff_lam_closed[THEN iffD2], simp_all)

simple_rename ren_U src [z1, x_P, x_leq, x_o, x_t, z2_c]
  tgt [z2_c, z1, z, x_P, x_leq, x_o, x_t]

simple_rename ren_V src [fz, x_P, x_leq, x_o, x_f, x_t, gz]
  tgt [gz, fz, z, x_P, x_leq, x_o, x_f, x_t]

simple_rename ren_V3 src [fz, x_P, x_leq, x_o, x_f, gz, hz]
  tgt [hz, gz, fz, z, x_P, x_leq, x_o, x_f]

lemma separation_sat_after_function_1:
  assumes [a, b, c, d] ∈ list(M) and χ ∈ formula and arity(χ) ≤ 6
  and
    f_fm: f_fm ∈ formula and
    f_ar: arity(f_fm) ≤ 6 and
    fsats: ∧ fx x. fx ∈ M ⇒ x ∈ M ⇒ (M, [fx, x]@[a, b, c, d] ⊨ f_fm) ↔ fx = f(x)
  and
    fclosed: ∧ x. x ∈ M ⇒ f(x) ∈ M and
    g_fm: g_fm ∈ formula and
    g_ar: arity(g_fm) ≤ 7 and
    gsats: ∧ gx fx x. gx ∈ M ⇒ fx ∈ M ⇒ x ∈ M ⇒ (M, [gx, fx, x]@[a, b, c, d] ⊨
  g_fm) ↔ gx = g(x) and
    gclosed: ∧ x. x ∈ M ⇒ g(x) ∈ M
  shows separation(##M, λr. M, [f(r), a, b, c, d, g(r)] ⊨ χ)

```

proof -

note $types = assms(1-4)$

let $?\psi = ren(\chi) '6 '7 'ren_U_fn$

let $?\psi' = Exists(And(f_fm, Exists(And(g_fm, ?\psi))))$

let $?g = \lambda z. [f(z), a, b, c, d, g(z)]$

let $?env = [a, b, c, d]$

let $?eta = \lambda z. [g(z), f(z), z] @ ?env$

note $types$

moreover from this

have $arity(\chi) \leq 7 ?\psi \in formula$

using $ord_simp_union\ ren_tc\ ren_U_thm(2)[folded\ ren_U_fn_def]\ le_trans[of\ arity(\chi)\ 6]$

by $simp_all$

moreover from calculation

have $arity(?\psi) \leq 7 ?\psi' \in formula$

using $arity_ren\ ren_U_thm(2)[folded\ ren_U_fn_def]\ f_fm\ g_fm$

by $simp_all$

moreover from calculation $f_ar\ g_ar\ f_fm\ g_fm$

have $arity(?\psi') \leq 5$

using $ord_simp_union\ pred_le\ arity_type$

by $(simp\ add:arity)$

moreover from calculation $fclosed\ gclosed$

have $0: (M, [f(z), a, b, c, d, g(z)] \models \chi) \longleftrightarrow (M, ?\eta(z) \models ?\psi)$ **if** $(\#\#M)(z)$ **for** z

using $sats_iff_sats_ren[of\ \chi\ 6\ 7\ _ _ ?\eta(z)]$

$ren_U_thm(1)[where\ A=M, folded\ ren_U_fn_def]\ ren_U_thm(2)[folded\ ren_U_fn_def]$ **that**

by $simp$

moreover from calculation

have $1: (M, ?\eta(z) \models ?\psi) \longleftrightarrow M, [z] @ ?env \models ?\psi'$ **if** $(\#\#M)(z)$ **for** z

using $that\ fsats[OF\ fclosed[of\ z], of\ z]\ gsats[of\ g(z)\ f(z)\ z]\ fclosed\ gclosed\ f_fm\ g_fm$

proof $(rule_tac\ iffI, simp, rule_tac\ rev_bexI[where\ x=f(z)], simp, (auto)[1])$

assume $M, [z] @ [a, b, c, d] \models (\cdot \exists \cdot f_fm \wedge (\cdot \exists \cdot g_fm \wedge ren(\chi) '6 '7 'ren_U_fn \cdot))$

then

have $\exists xa \in M. (M, [xa, z, a, b, c, d] \models f_fm) \wedge$
 $(\exists x \in M. (M, [x, xa, z, a, b, c, d] \models g_fm) \wedge$
 $(M, [x, xa, z, a, b, c, d] \models ren(\chi) '6 '7 'ren_U_fn))$

using $that\ calculation\ by\ auto$

then

obtain $xa\ x\ where\ x \in M\ xa \in M\ M, [xa, z, a, b, c, d] \models f_fm$
 $(M, [x, xa, z, a, b, c, d] \models g_fm)$
 $(M, [x, xa, z, a, b, c, d] \models ren(\chi) '6 '7 'ren_U_fn)$

using $that\ calculation\ by\ auto$

moreover from this

have $xa=f(z)\ x=g(z)$ **using** $fsats[of\ xa]\ gsats[of\ x\ xa]$ **that** **by** $simp_all$

ultimately

show $M, [g(z), f(z), z] @ [a, b, c, d] \models ren(\chi) '6 '7 'ren_U_fn$

by $auto$

```

qed
moreover from calculation
have separation(##M, λz. (M,[z]@?env |= ?ψ'))
  using separation_ax
  by simp_all
ultimately
show ?thesis
  by(rule_tac separation_cong[THEN iffD2,OF iff_trans[OF 0 1]],clarify,force)
qed

lemma separation_sat_after_function3:
  assumes [a, b, c, d]∈list(M) and χ∈formula and arity(χ) ≤ 7
  and
  f_fm: f_fm ∈ formula and
  f_ar: arity(f_fm) ≤ 6 and
  fsats: ∧ fx x. fx∈M ⇒ x∈M ⇒ (M,[fx,x]@[a, b, c, d] |= f_fm) ⇔ fx=f(x)
and
  fclosed: ∧x . x∈M ⇒ f(x) ∈ M and
  g_fm: g_fm ∈ formula and
  g_ar: arity(g_fm) ≤ 7 and
  gsats: ∧ gx fx x. gx∈M ⇒ fx∈M ⇒ x∈M ⇒ (M,[gx,fx,x]@[a, b, c, d] |=
g_fm) ⇔ gx=g(x) and
  gclosed: ∧x . x∈M ⇒ g(x) ∈ M and
  h_fm: h_fm ∈ formula and
  h_ar: arity(h_fm) ≤ 8 and
  hsats: ∧ hx gx fx x. hx∈M ⇒ gx∈M ⇒ fx∈M ⇒ x∈M ⇒ (M,[hx,gx,fx,x]@[a,
b, c, d] |= h_fm) ⇔ hx=h(x) and
  hclosed: ∧x . x∈M ⇒ h(x) ∈ M
  shows separation(##M, λr. M, [f(r), a, b, c, d, g(r), h(r)] |= χ)
proof -
  note types = assms(1-3)
  let ?φ=χ
  let ?ψ=ren(?φ)‘7‘8‘ren_V3_fn
  let ?ψ'=Exists(And(f_fm,Exists(And(g_fm,Exists(And(h_fm,?ψ)))))
  let ?ρ=λz.[f(z), a, b, c, d,g(z), h(z)]
  let ?env=[a, b, c, d]
  let ?η=λz.[h(z),g(z),f(z),z]@?env
  note types
  moreover from this
  have ?φ∈formula by simp
  moreover from calculation
  have arity(?φ) ≤ 9 ?ψ∈formula
  using ord_simp_union ren_tc ren_V3_thm(2)[folded ren_V3_fn_def] le_trans[of
arity(χ) 7]
  by simp_all
  moreover from calculation
  have arity(?ψ) ≤ 8 ?ψ'∈formula
  using arity_ren ren_V3_thm(2)[folded ren_V3_fn_def] f_fm g_fm h_fm
  by (simp_all)

```

moreover from *this* $f_ar\ g_ar\ f_fm\ g_fm\ h_fm\ h_ar\ \langle ?\psi' \in _ \rangle$
have $arity(?\psi') \leq 5$
using *ord_simp_union arity_type nat_into_Ord*
by (*simp add:arity,(rule_tac pred_le,simp,rule_tac Un_le,simp)+,simp_all*
add: \langle ?\psi \in _ \rangle)
moreover from *calculation fclosed gclosed hclosed*
have $0:(M, ?\rho(z) \models ?\varphi) \longleftrightarrow (M, ?\eta(z) \models ?\psi)$ **if** $(\#\#M)(z)$ **for** z
using *sats_iff_sats_ren[of ?\varphi 7 8 ?\rho(z) M ?\eta(z)]*
ren_V3_thm(1)[where A=M,folded ren_V3_fn_def,simplified] ren_V3_thm(2)[folded
ren_V3_fn_def] that
by *simp*
moreover from *calculation*
have $1:(M, ?\eta(z) \models ?\psi) \longleftrightarrow M, [z] @ ?env \models ?\psi'$ **if** $(\#\#M)(z)$ **for** z
using *that fsats[OF fclosed[of z],of z] gsats[of g(z) f(z) z]*
hsats[of h(z) g(z) f(z) z]
fclosed gclosed hclosed f_fm g_fm h_fm
apply(*rule_tac iffI,simp,rule_tac rev_bexI[where x=f(z)],simp*)
apply(*rule_tac conjI,simp,rule_tac rev_bexI[where x=g(z)],simp*)
apply(*rule_tac conjI,simp,rule_tac rev_bexI[where x=h(z)],simp,rule_tac*
conjI,simp,simp)
proof -
assume $M, [z] @ [a, b, c, d] \models (\cdot \exists \cdot f_fm \wedge (\cdot \exists \cdot g_fm \wedge (\cdot \exists \cdot h_fm \wedge ren(\chi) \text{ ' 7$
 $\text{ ' 8 ' ren_V3_fn} \cdot \cdot \cdot)) \cdot \cdot \cdot)$
with *calculation that*
have $\exists x \in M. (M, [x, z, a, b, c, d] \models f_fm) \wedge$
 $(\exists xa \in M. (M, [xa, x, z, a, b, c, d] \models g_fm) \wedge (\exists xb \in M. (M, [xb, xa, x, z, a,$
 $a, b, c, d] \models h_fm) \wedge (M, [xb, xa, x, z, a, b, c, d] \models ren(\chi) \text{ ' 7 ' 8 ' ren_V3_fn})))$
by *auto*
with *calculation*
obtain x **where** $x \in M$ $(M, [x, z, a, b, c, d] \models f_fm)$
 $(\exists xa \in M. (M, [xa, x, z, a, b, c, d] \models g_fm) \wedge (\exists xb \in M. (M, [xb, xa, x, z, a,$
 $b, c, d] \models h_fm) \wedge (M, [xb, xa, x, z, a, b, c, d] \models ren(\chi) \text{ ' 7 ' 8 ' ren_V3_fn})))$
by *force*
moreover from *this*
have $x=f(z)$ **using** *fsats[of x] that by simp*
moreover from *calculation*
obtain xa **where** $xa \in M$ $(M, [xa, x, z, a, b, c, d] \models g_fm)$
 $(\exists xb \in M. (M, [xb, xa, x, z, a, b, c, d] \models h_fm) \wedge (M, [xb, xa, x, z, a, b, c,$
 $d] \models ren(\chi) \text{ ' 7 ' 8 ' ren_V3_fn})))$
by *auto*
moreover from *calculation*
have $xa=g(z)$ **using** *gsats[of xa x] that by simp*
moreover from *calculation*
obtain xb **where** $xb \in M$ $(M, [xb, xa, x, z, a, b, c, d] \models h_fm)$
 $(M, [xb, xa, x, z, a, b, c, d] \models ren(\chi) \text{ ' 7 ' 8 ' ren_V3_fn})))$
by *auto*
moreover from *calculation*
have $xb=h(z)$ **using** *hsats[of xb xa x] that by simp*
ultimately

```

  show  $M, [h(z), g(z), f(z), z] @ [a, b, c, d] \models \text{ren}(\chi) \text{ ' } 7 \text{ ' } 8 \text{ ' } \text{ren\_V3\_fn}$ 
  by auto
qed
moreover from calculation  $\langle ?\psi' \in \_ \rangle$ 
have separation( $\#\#M, \lambda z. (M, [z] @ ?env \models ?\psi')$ )
  using separation_ax
  by simp
ultimately
show ?thesis
  by(rule_tac separation_cong[THEN iffD2, OF iff_trans[OF 0 1]], clarify, force)
qed

```

lemma separation_sat_after_function:

```

  assumes  $[a, b, c, d, \tau] \in \text{list}(M)$  and  $\chi \in \text{formula}$  and  $\text{arity}(\chi) \leq 7$ 
  and
  f_fm:  $f\_fm \in \text{formula}$  and
  f_ar:  $\text{arity}(f\_fm) \leq 7$  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 separation( $\#\#M, \lambda r. M, [f(r), a, b, c, d, \tau, g(r)] \models \chi$ )

```

proof -

```

  note types = assms(1-3)
  let ? $\varphi = \chi$ 
  let ? $\psi = \text{ren}(\varphi) \text{ ' } 7 \text{ ' } 8 \text{ ' } \text{ren\_V\_fn}$ 
  let ? $\psi' = \text{Exists}(And(f\_fm, \text{Exists}(And(g\_fm, ?\psi))))$ 
  let ? $\rho = \lambda z. [f(z), a, b, c, d, \tau, g(z)]$ 
  let ?env =  $[a, b, c, d, \tau]$ 
  let ? $\eta = \lambda z. [g(z), f(z), z] @ ?env$ 
  note types
  moreover from this
  have ? $\varphi \in \text{formula}$  by simp
  moreover from calculation
  have  $\text{arity}(\varphi) \leq 8$  ? $\varphi \in \text{formula}$ 
  using ord_simp_union ren_tc ren_V_thm(2)[folded ren_V_fn_def] le_trans[of
 $\text{arity}(\chi) 7]$ 
  by simp_all
  moreover from calculation
  have  $\text{arity}(\psi) \leq 8$  ? $\psi \in \text{formula}$ 
  using arity_ren ren_V_thm(2)[folded ren_V_fn_def] f_fm g_fm
  by (simp_all)
  moreover from calculation f_ar g_ar f_fm g_fm
  have  $\text{arity}(\psi') \leq 6$ 
  using ord_simp_union pred_le arity_type

```

```

    by (simp add:arity)
  moreover from calculation fclosed gclosed
  have 0:(M, ?ρ(z) ⊨ ?φ) ↔ (M, ?η(z) ⊨ ?ψ) if (##M)(z) for z
    using sats_iff_sats_ren[of ?φ 7 8 ?ρ(z) _ ?η(z)]
      ren_V_thm(1)[where A=M, folded ren_V_fn_def] ren_V_thm(2)[folded
ren_V_fn_def] that
    by simp
  moreover from calculation
  have 1:(M, ?η(z) ⊨ ?ψ) ↔ M, [z]@?env ⊨ ?ψ' if (##M)(z) for z
    using that fsats[OF fclosed[of z], of z] gsats[of g(z) f(z) z]
      fclosed gclosed f_fm g_fm
    apply(rule_tac iffI, simp, rule_tac rev_bexI[where x=f(z)], simp)
    apply(auto)[1]
  proof -
    assume M, [z] @ [a, b, c, d, τ] ⊨ (·∃·f_fm ∧ (·∃·g_fm ∧ ren(χ) ' 7 ' 8 '
ren_V_fn··))
    then have ∃ xa∈M. (M, [xa, z, a, b, c, d, τ] ⊨ f_fm) ∧
      (∃ x∈M. (M, [x, xa, z, a, b, c, d, τ] ⊨ g_fm) ∧ (M, [x, xa, z, a, b, c, d, τ]
⊨ ren(χ) ' 7 ' 8 ' ren_V_fn))
      using that calculation by auto
    then
      obtain xa where xa∈M M, [xa, z, a, b, c, d, τ] ⊨ f_fm
        (∃ x∈M. (M, [x, xa, z, a, b, c, d, τ] ⊨ g_fm) ∧ (M, [x, xa, z, a, b, c, d, τ] ⊨
ren(χ) ' 7 ' 8 ' ren_V_fn))
        by auto
      moreover from this
      have xa=f(z) using fsats[of xa] that by simp
      moreover from calculation
      obtain x where x∈M M, [x, xa, z, a, b, c, d, τ] ⊨ g_fm M, [x, xa, z, a, b, c,
d, τ] ⊨ ren(χ) ' 7 ' 8 ' ren_V_fn
        by auto
      moreover from calculation
      have x=g(z) using gsats[of x xa] that by simp
      ultimately
      show M, [g(z), f(z), z] @ [a, b, c, d, τ] ⊨ ren(χ) ' 7 ' 8 ' ren_V_fn
        by auto
    qed
  moreover from calculation
  have separation(##M, λz. (M, [z]@?env ⊨ ?ψ'))
    using separation_ax
    by simp_all
  ultimately
  show ?thesis
    by(rule_tac separation_cong[THEN iffD2, OF iff_trans[OF 0 1]], clarify, force)
  qed
end

```

definition *separation_assm_fm* :: $[i, i, i] \Rightarrow i$
 where

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

lemma *separation_assm_fm_type*[TC]:

$A \in \omega \implies y \in \omega \implies f_fm \in formula \implies separation_assm_fm(A, y, f_fm) \in formula$

unfolding *separation_assm_fm_def*

by *simp*

lemma *arity_separation_assm_fm* : $A \in \omega \implies x \in \omega \implies f_fm \in formula \implies arity(separation_assm_fm(A, x, f_fm)) = succ(A) \cup succ(x) \cup pred(pred(arity(f_fm)))$

using *pred_Un_distrib*

unfolding *separation_assm_fm_def*

by (*auto simp add:arity*)

definition *separation_assm_bin_fm* **where**

$separation_assm_bin_fm(A,y,f_fm) \equiv$

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

lemma *separation_assm_bin_fm_type*[TC]:

$A \in \omega \implies y \in \omega \implies f_fm \in formula \implies separation_assm_bin_fm(A, y, f_fm) \in formula$

unfolding *separation_assm_bin_fm_def*

by *simp*

lemma *arity_separation_assm_bin_fm* : $A \in \omega \implies x \in \omega \implies f_fm \in formula \implies$

$arity(separation_assm_bin_fm(A, x, f_fm)) = succ(A) \cup succ(x) \cup (pred^4(arity(f_fm)))$

using *pred_Un_distrib*

unfolding *separation_assm_bin_fm_def*

by (*auto simp add:arity*)

context *M_Z_trans*

begin

lemma *separation_assm_sats* :

assumes

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

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

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

$is_f(x, y)$ **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 $separation(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, f(x) \rangle)$

proof -

let $?\varphi' = separation_assm_fm(1, 0, \varphi)$

let $?p = \lambda y. \exists x \in M. x \in A \wedge y = \langle x, f(x) \rangle$

```

from f_fm
have  $?\varphi' \in \text{formula}$ 
  by simp
moreover from this f_ar f_fm
have  $\text{arity}(\varphi') = 2$ 
  using arity_separation_assm_fm[of 1 0  $\varphi$ ] ord_simp_union
  by simp
moreover from  $\langle A \in M \rangle$  calculation
have  $\text{separation}(\#\#M, \lambda y. M, [y, A] \models ?\varphi')$ 
  using separation_ax by auto
moreover
have  $y \in M \implies (M, [y, A] \models ?\varphi') \longleftrightarrow ?p(y)$  for  $y$ 
  using assms transitivity[OF  $\_ \langle A \in M \rangle$ ]
  unfolding separation_assm_fm_def
  by auto
ultimately
show ?thesis
  by(rule_tac separation_cong[THEN iffD1], auto)
qed

```

lemma *separation_assm_bin_sats* :

```

assumes
  f_fm:  $\varphi \in \text{formula}$  and
  f_ar:  $\text{arity}(\varphi) = 3$  and
  fsats:  $\bigwedge \text{env } x z y. \text{env} \in \text{list}(M) \implies x \in M \implies z \in M \implies y \in M \implies (M, [x, z, y] @ \text{env} \models \varphi) \longleftrightarrow \text{is\_f}(x, z, y)$  and
  fabs:  $\bigwedge x z y. x \in M \implies z \in M \implies y \in M \implies \text{is\_f}(x, z, y) \longleftrightarrow y = f(x, z)$  and
  fclosed:  $\bigwedge x z. x \in M \implies z \in M \implies f(x, z) \in M$  and
   $A \in M$ 
shows  $\text{separation}(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, f(\text{fst}(x), \text{snd}(x)) \rangle)$ 
proof -
let  $?\varphi' = \text{separation\_assm\_bin\_fm}(1, 0, \varphi)$ 
let  $?p = \lambda y. \exists x \in M. x \in A \wedge y = \langle x, f(\text{fst}(x), \text{snd}(x)) \rangle$ 
from f_fm
have  $?\varphi' \in \text{formula}$ 
  by simp
moreover from this f_ar f_fm
have  $\text{arity}(\varphi') = 2$ 
  using arity_separation_assm_bin_fm[of 1 0  $\varphi$ ] ord_simp_union
  by simp
moreover from  $\langle A \in M \rangle$  calculation
have  $\text{separation}(\#\#M, \lambda y. M, [y, A] \models ?\varphi')$ 
  using separation_ax by auto
moreover
have  $y \in M \implies (M, [y, A] \models ?\varphi') \longleftrightarrow ?p(y)$  for  $y$ 
  using assms transitivity[OF  $\_ \langle A \in M \rangle$ ] pair_in_M_iff fst_abs snd_abs fst_closed
  snd_closed
  unfolding separation_assm_bin_fm_def
  by auto

```

ultimately
show *?thesis*
 by(*rule_tac separation_cong[THEN iffD1],auto*)
qed

lemma *separation_Union*: $A \in M \implies$
 $separation(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, Union(x) \rangle)$
using *separation_assm_sats[of big_union_fm(0,1)] arity_big_union_fm ord_simp_union*
Union_closed[simplified]
by *simp*

lemma *lam_replacement_Union*: $lam_replacement(\#\#M, Union)$
using *lam_replacement_Union' separation_Union transM* **by** *simp*

lemma *separation_fst*: $A \in M \implies$
 $separation(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, fst(x) \rangle)$
using *separation_assm_sats[of fst_fm(0,1)] arity_fst_fm ord_simp_union*
fst_closed fst_abs
by *simp*

lemma *lam_replacement_fst*: $lam_replacement(\#\#M, fst)$
using *lam_replacement_fst' separation_fst transM* **by** *simp*

lemma *separation_snd*: $A \in M \implies$
 $separation(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, snd(x) \rangle)$
using *separation_assm_sats[of snd_fm(0,1)] arity_snd_fm ord_simp_union*
snd_closed[simplified] snd_abs
by *simp*

lemma *lam_replacement_snd*: $lam_replacement(\#\#M, snd)$
using *lam_replacement_snd' separation_snd transM* **by** *simp*

Binary lambda-replacements

lemma *separation_Image*: $A \in M \implies$
 $separation(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, fst(x) \text{ “ } snd(x) \rangle)$
using *arity_image_fm ord_simp_union*
nonempty_image_closed image_abs
by (*rule_tac separation_assm_bin_sats[of image_fm(0,1,2)],auto*)

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

lemma *separation_middle_del*: $A \in M \implies$
 $separation(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, middle_del(fst(x), snd(x)) \rangle)$
using *arity_is_middle_del_fm ord_simp_union nonempty*
fst_abs snd_abs fst_closed snd_closed pair_in_M iff
by (*rule_tac separation_assm_bin_sats[of is_middle_del_fm(0,1,2)],*
auto simp:is_middle_del_def middle_del_def)

lemma *lam_replacement_middle_del*: $\text{lam_replacement}(\#\#M, \lambda r. \text{middle_del}(\text{fst}(r), \text{snd}(r)))$
using *lam_replacement_middle_del'* *separation_middle_del*
by *simp*

lemma *separation_prodRepl*: $A \in M \implies$
 $\text{separation}(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, \text{prodRepl}(\text{fst}(x), \text{snd}(x)) \rangle)$
using *arity_is_prodRepl_fm* *ord_simp_union* *nonempty*
fst_abs_snd_abs *fst_closed_snd_closed* *pair_in_M_iff*
by (*rule_tac* *separation_assm_bin_sats*[of *is_prodRepl_fm*(0,1,2)],
auto simp:is_prodRepl_def prodRepl_def)

lemma *lam_replacement_prodRepl*: $\text{lam_replacement}(\#\#M, \lambda r. \text{prodRepl}(\text{fst}(r), \text{snd}(r)))$
using *lam_replacement_prodRepl'* *separation_prodRepl*
by *simp*

end — *M_Z_trans*

context *M_trivial*
begin

lemma *first_closed*:
 $M(B) \implies M(r) \implies \text{first}(u, r, B) \implies M(u)$
using *transM[OF first_is_elem]* **by** *simp*

is_iff_rel **for** *first*
unfolding *is_first_def first_rel_def* **by** *auto*

is_iff_rel **for** *minimum*
unfolding *is_minimum_def minimum_rel_def*
using *is_first_iff The_abs* *nonempty*
by *force*

end — *M_trivial*

context *M_Z_trans*
begin

lemma (**in** *M_basic*) *is_minimum_equivalence* :
 $M(R) \implies M(X) \implies M(u) \implies \text{is_minimum}(M, R, X, u) \longleftrightarrow \text{is_minimum}'(M, R, X, u)$
unfolding *is_minimum_def is_minimum'_def is_The_def is_first_def* **by**
simp

lemma *separation_minimum*: $A \in M \implies$
 $\text{separation}(\#\#M, \lambda y. \exists x \in M. x \in A \wedge y = \langle x, \text{minimum}(\text{fst}(x), \text{snd}(x)) \rangle)$
using *arity_minimum_fm* *ord_simp_union* *is_minimum_iff* *minimum_abs*
is_minimum_equivalence *nonempty* *minimum_closed* *minimum_abs*
by (*rule_tac* *separation_assm_bin_sats*[of *minimum_fm*(0,1,2)], *auto*)

```

lemma lam_replacement_minimum: lam_replacement(##M, λx . minimum(fst(x),snd(x)))
  using lam_replacement_minimum' separation_minimum
  by simp

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] ⇒ o where
  radd_body(R,S) ≡ λz. (∃ x y. z = ⟨Inl(x), Inr(y)⟩) ∨
    (∃ x' x. z = ⟨Inl(x'), Inl(x)⟩ ∧ ⟨x', x⟩ ∈ R) ∨
    (∃ y' y. z = ⟨Inr(y'), Inr(y)⟩ ∧ ⟨y', y⟩ ∈ S)

```

```

relativize functional radd_body radd_body_rel
relationalize radd_body_rel is_radd_body

```

```

synthesize is_radd_body from_definition
arity_theorem for is_radd_body_fm

```

```

definition rmult_body :: [i,i,i] ⇒ o where
  rmult_body(b,d) ≡ λz. ∃ x' y' x y. z = ⟨⟨x', y'⟩, x, y⟩ ∧ (⟨x', x⟩ ∈ b ∨
    x' = x ∧ ⟨y', y⟩ ∈ d)

```

```

relativize functional rmult_body rmult_body_rel
relationalize rmult_body_rel is_rmult_body

```

```

synthesize is_rmult_body from_definition
arity_theorem for is_rmult_body_fm

```

```

lemma (in M_replacement) separation_well_ord_iso:
  (M)(f) ⇒ (M)(r) ⇒ (M)(A) ⇒ separation
  (M, λx. x ∈ A → (∃ y[M]. ∃ p[M]. is_apply(M, f, x, y) ∧ pair(M, y, x, p)
  ∧ p ∈ r))
  using separation_imp separation_in lam_replacement_identity lam_replacement_constant
    lam_replacement_apply[of f] lam_replacement_product
  by simp

```

```

definition is_obase_body :: [i⇒o,i,i,i] ⇒ o where
  is_obase_body(N,A,r,x) ≡ x ∈ A →

```

$$\neg (\exists y[N].$$

$$\quad \exists g[N].$$

$$\quad \text{ordinal}(N, y) \wedge$$

$$\quad (\exists my[N].$$

$$\quad \quad \exists pxr[N].$$

$$\quad \quad \text{membership}(N, y, my) \wedge$$

$$\quad \quad \text{pred_set}(N, A, x, r, pxr) \wedge$$

$$\quad \quad \text{order_isomorphism}(N, pxr, r, y, my, g)))$$

synthesize *is_obase_body* from_definition

arity_theorem for *is_obase_body_fm*

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

***is_obase_equals*(*N, A, r, a*) $\equiv \exists x[N].$**

$\exists g[N].$

$\exists mx[N].$

$\exists par[N].$

***ordinal*(*N, x*) \wedge**

***membership*(*N, x, mx*) \wedge**

***pred_set*(*N, A, a, r, par*) $\wedge \text{order_isomorphism}(N, par,$**

***r, x, mx, g*)**

synthesize *is_obase_equals* from_definition

arity_theorem for *is_obase_equals_fm*

synthesize *PiP_rel* from_definition assuming *nonempty*

arity_theorem for *PiP_rel_fm*

synthesize *injP_rel* from_definition assuming *nonempty*

arity_theorem for *injP_rel_fm*

synthesize *surjP_rel* from_definition assuming *nonempty*

arity_theorem for *surjP_rel_fm*

context *M_ZF1_trans*

begin

lemma *radd_body_abs*:

assumes $(\#\#M)(R) (\#\#M)(S) (\#\#M)(x)$

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

using *assms pair_in_M_iff Inl_in_M_iff Inr_in_M_iff*

unfolding *radd_body_def is_radd_body_def*

by (*auto*)

lemma *separation_radd_body*:

$(\#\#M)(R) \Longrightarrow (\#\#M)(S) \Longrightarrow \text{separation}$

$(\#\#M, \lambda z. (\exists x y. z = \langle \text{Inl}(x), \text{Inr}(y) \rangle)) \vee$

$(\exists x' x. z = \langle \text{Inl}(x'), \text{Inl}(x) \rangle \wedge \langle x', x \rangle \in R) \vee$

$(\exists y' y. z = \langle \text{Inr}(y'), \text{Inr}(y) \rangle \wedge \langle y', y \rangle \in S)$
using *separation_in_ctm* [**where** $\varphi = \text{is_radd_body_fm}(1,2,0)$ **and** $\text{env} = [R,S]$]
is_radd_body_defarity_is_radd_body_fm_ord_simp_union_is_radd_body_fm_type
radd_body_abs
unfolding *radd_body_def*
by *simp*

lemma *rmult_body_abs*:
assumes $(\#\#M)(b) (\#\#M)(d) (\#\#M)(x)$
shows $\text{is_rmult_body}(\#\#M, b, d, x) \longleftrightarrow \text{rmult_body}(b, d, x)$
using *assms_pair_in_M_iff_apply_closed*
unfolding *rmult_body_def_is_rmult_body_def*
by (*auto*)

lemma *separation_rmult_body*:
 $(\#\#M)(b) \implies (\#\#M)(d) \implies \text{separation}$
 $(\#\#M, \lambda z. \exists x' y' x y. z = \langle \langle x', y' \rangle, x, y \rangle \wedge (\langle x', x \rangle \in b \vee x' = x \wedge \langle y', y \rangle \in d))$
using *separation_in_ctm* [**where** $\varphi = \text{is_rmult_body_fm}(1,2,0)$ **and** $\text{env} = [b,d]$]
is_rmult_body_defarity_is_rmult_body_fm_ord_simp_union_is_rmult_body_fm_type
rmult_body_abs
unfolding *rmult_body_def*
by *simp*

lemma *separation_is_obase*:
 $(\#\#M)(f) \implies (\#\#M)(r) \implies (\#\#M)(A) \implies \text{separation}$
 $(\#\#M, \lambda x. x \in A \longrightarrow \neg (\exists y[\#\#M]. \exists g[\#\#M]. \text{ordinal}(\#\#M, y) \wedge (\exists my[\#\#M]. \exists pxr[\#\#M]. \text{membership}(\#\#M, y, my) \wedge \text{pred_set}(\#\#M, A, x, r, pxr) \wedge \text{order_isomorphism}(\#\#M, pxr, r, y, my, g))))$
using *separation_in_ctm* [**where** $\varphi = \text{is_obase_body_fm}(1,2,0)$ **and** $\text{env} = [A,r]$]
is_obase_body_defarity_is_obase_body_fm_ord_simp_union_is_obase_body_fm_type
by *simp*

lemma *separation_obase_equals*:
 $(\#\#M)(f) \implies (\#\#M)(r) \implies (\#\#M)(A) \implies \text{separation}$
 $(\#\#M, \lambda a. \exists x[\#\#M]. \exists g[\#\#M]. \exists mx[\#\#M]. \exists par[\#\#M]. \text{ordinal}(\#\#M, x) \wedge \text{membership}(\#\#M, x, mx) \wedge \text{pred_set}(\#\#M, A, a, r, par) \wedge \text{order_isomorphism}(\#\#M, par, r, x, mx, g))$

using *separation_in_ctm* [**where** $\varphi = \text{is_obase_equals_fm}(1,2,0)$ **and** $\text{env} = [A,r]$]
is_obase_equals_def *arity_is_obase_equals_fm* *ord_simp_union* *is_obase_equals_fm_type*
by *simp*

lemma *separation_PiP_rel*:

$(\#\#M)(A) \implies \text{separation}(\#\#M, \text{PiP_rel}(\#\#M,A))$
using *separation_in_ctm* [**where** $\text{env} = [A]$ **and** $\varphi = \text{PiP_rel_fm}(1,0)$]
nonempty_PiP_rel_iff_sats [*symmetric*] *arity_PiP_rel_fm* *PiP_rel_fm_type*
by (*simp_all* *add: ord_simp_union*)

lemma *separation_injP_rel*:

$(\#\#M)(A) \implies \text{separation}(\#\#M, \text{injP_rel}(\#\#M,A))$
using *separation_in_ctm* [**where** $\text{env} = [A]$ **and** $\varphi = \text{injP_rel_fm}(1,0)$]
nonempty_injP_rel_iff_sats [*symmetric*] *arity_injP_rel_fm* *injP_rel_fm_type*
by (*simp_all* *add: ord_simp_union*)

lemma *separation_surjP_rel*:

$(\#\#M)(A) \implies (\#\#M)(B) \implies \text{separation}(\#\#M, \text{surjP_rel}(\#\#M,A,B))$
using *separation_in_ctm* [**where** $\text{env} = [A,B]$ **and** $\varphi = \text{surjP_rel_fm}(1,2,0)$]
nonempty_surjP_rel_iff_sats [*symmetric*] *arity_surjP_rel_fm* *surjP_rel_fm_type*
by (*simp_all* *add: ord_simp_union*)

lemma *separation_is_function*:

$\text{separation}(\#\#M, \text{is_function}(\#\#M))$
using *separation_in_ctm* [**where** $\text{env} = []$ **and** $\varphi = \text{function_fm}(0)$] *arity_function_fm*
by *simp*

end — *M_ZF1_trans*

definition *fstsnd_in_sndsnd* :: $[i] \Rightarrow o$ **where**

$\text{fstsnd_in_sndsnd} \equiv \lambda x. \text{fst}(\text{snd}(x)) \in \text{snd}(\text{snd}(x))$

relativize *fstsnd_in_sndsnd* *is_fstsnd_in_sndsnd*

synthesize *is_fstsnd_in_sndsnd* **from_definition** **assuming** *nonempty*

arity_theorem **for** *is_fstsnd_in_sndsnd_fm*

definition *sndfst_eq_fstsnd* :: $[i] \Rightarrow o$ **where**

$\text{sndfst_eq_fstsnd} \equiv \lambda x. \text{snd}(\text{fst}(x)) = \text{fst}(\text{snd}(x))$

relativize *sndfst_eq_fstsnd* *is_sndfst_eq_fstsnd*

synthesize *is_sndfst_eq_fstsnd* **from_definition** **assuming** *nonempty*

arity_theorem **for** *is_sndfst_eq_fstsnd_fm*

context *M_ZF1_trans*

begin

lemma *fstsnd_in_sndsnd_abs*:

assumes $(\#\#M)(x)$

shows $\text{is_fstsnd_in_sndsnd}(\#\#M,x) \longleftrightarrow \text{fstsnd_in_sndsnd}(x)$

```

using assms pair_in_M_iff fst_abs snd_abs fst_snd_closed
unfolding fstsnd_in_sndsnd_def is_fstsnd_in_sndsnd_def
by auto

lemma separation_fstsnd_in_sndsnd:
  separation(##M, λx. fst(snd(x)) ∈ snd(snd(x)))
  using separation_in_ctm[where env=[] and φ=is_fstsnd_in_sndsnd_fm(0)]
and Q=fstsnd_in_sndsnd]
  nonempty_fstsnd_in_sndsnd_abs arity_is_fstsnd_in_sndsnd_fm
  unfolding fstsnd_in_sndsnd_def
  by simp

lemma sndfst_eq_fstsnd_abs:
  assumes (##M)(x)
  shows is_sndfst_eq_fstsnd(##M,x) ↔ sndfst_eq_fstsnd(x)
  using assms pair_in_M_iff fst_abs snd_abs fst_snd_closed
  unfolding sndfst_eq_fstsnd_def is_sndfst_eq_fstsnd_def
  by auto

lemma separation_sndfst_eq_fstsnd:
  separation(##M, λx. snd(fst(x)) = fst(snd(x)))
  using separation_in_ctm[where env=[] and φ=is_sndfst_eq_fstsnd_fm(0)]
and Q=sndfst_eq_fstsnd]
  nonempty_sndfst_eq_fstsnd_abs arity_is_sndfst_eq_fstsnd_fm
  unfolding sndfst_eq_fstsnd_def
  by simp

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 ∈ ω ⇒ a1 ∈ ω ⇒ a2 ∈ ω ⇒
  composition_fm(a0,a1,a2) ∈ formula
  unfolding composition_fm_def by simp

arity_theorem for composition_fm

definition is_omega_funspace :: [i⇒o,i,i,i]⇒o where
  is_omega_funspace(N,B,n,z) ≡ ∃ o[N]. omega(N,o) ∧ n∈o ∧ is_funspace(N, n,
B, z)

```

**synthesize ω _funspace from_definition is_omega_funspace assuming nonempty
arity_theorem for omega_funspace_fm**

definition HAleph_wfrec_repl_body where

$$\begin{aligned}
& \text{HAleph_wfrec_repl_body}(N, \text{mesa}, x, z) \equiv \exists y[N]. \\
& \quad \text{pair}(N, x, y, z) \wedge \\
& \quad (\exists g[N]. \\
& \quad \quad (\forall u[N]. \\
& \quad \quad \quad u \in g \longleftrightarrow \\
& \quad \quad \quad (\exists a[N]. \\
& \quad \quad \quad \quad \exists y[N]. \\
& \quad \quad \quad \quad \quad \exists ax[N]. \\
& \quad \quad \quad \quad \quad \quad \exists sx[N]. \\
& \quad \quad \quad \quad \quad \quad \quad \exists r_sx[N]. \\
& \quad \quad \quad \quad \quad \quad \quad \quad \exists f_r_sx[N]. \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{pair}(N, a, y, u) \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{pair}(N, a, x, ax) \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{upair}(N, a, a, sx) \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{pre_image}(N, \text{mesa}, sx, r_sx) \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{restriction}(N, g, r_sx, f_r_sx) \wedge ax \in \text{mesa} \wedge \text{is_HAleph}(N, a, f_r_sx, y))) \\
& \wedge \\
& \quad \text{is_HAleph}(N, x, g, y))
\end{aligned}$$

arity_theorem for ordinal_fm

arity_theorem for is_Limit_fm

arity_theorem for empty_fm

arity_theorem for fun_apply_fm

synthesize HAleph_wfrec_repl_body from_definition assuming nonempty

arity_theorem for HAleph_wfrec_repl_body_fm

definition dcwit_repl_body where

$$\begin{aligned}
& \text{dcwit_repl_body}(N, \text{mesa}, A, a, s, R) \equiv \lambda x z. \exists y[N]. \text{pair}(N, x, y, z) \wedge \\
& \quad \text{is_wfrec} \\
& \quad (N, \lambda n f. \text{is_nat_case} \\
& \quad \quad (N, a, \\
& \quad \quad \quad \lambda m \text{bmfm}. \\
& \quad \quad \quad \quad \exists \text{fm}[N]. \\
& \quad \quad \quad \quad \quad \exists \text{cp}[N]. \\
& \quad \quad \quad \quad \quad \quad \text{is_apply}(N, f, m, \text{fm}) \wedge \\
& \quad \quad \quad \quad \quad \quad \text{is_Collect}(N, A, \lambda x. \exists \text{fm}x[N]. (N(x) \\
& \wedge \text{fm}x \in R) \wedge \text{pair}(N, \text{fm}, x, \text{fm}x), \text{cp}) \wedge \\
& \quad \quad \quad \quad \quad \quad \quad \text{is_apply}(N, s, \text{cp}, \text{bmfm}), \\
& \quad \quad \quad \quad \quad \quad \quad \quad n), \\
& \quad \quad \quad \quad \quad \quad \quad \quad \text{mesa}, x, y)
\end{aligned}$$

manual_schematic for dcwit_repl_body assuming nonempty

unfolding dcwit_repl_body_def

by (rule iff_sats is_nat_case_iff_sats is_eclose_iff_sats sep_rules | simp)+

synthesize dcwit_repl_body from_schematic

definition dcwit_aux_fm where

$$\begin{aligned}
\text{dcwit_aux_fm}(A,s,R) \equiv & (\cdot \exists \cdot 4' 2 \text{ is } 0 \cdot \wedge \\
& (\cdot \exists \cdot \text{Collect_fm} \\
& \quad (\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(A)))))))))), \\
& \quad (\cdot \exists \cdot 0 \in \\
& \quad \quad \text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(R)))))))))) \\
& \cdot \wedge \\
& \quad \text{pair_fm}(3, 1, 0) \cdot \cdot), \\
& \quad 0) \wedge \\
& \quad \cdot \text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(s))))))))' 0 \text{ is} \\
& 2 \cdot \dots) \cdot
\end{aligned}$$

arity_theorem for dcwit_aux_fm

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

by (simp_all add: dcwit_aux_fm_def)

definition is_nat_case_dcwit_aux_fm where

$$\begin{aligned}
\text{is_nat_case_dcwit_aux_fm}(A,a,s,R) \equiv & \text{is_nat_case_fm} \\
& (\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(a)))))), \text{dcwit_aux_fm}(A,s,R), \\
& 2, 0)
\end{aligned}$$

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

by (simp_all add: is_nat_case_dcwit_aux_fm_def)

manual_arity for is_nat_case_dcwit_aux_fm

unfolding is_nat_case_dcwit_aux_fm_def

by (rule arity_dcwit_aux_fm[THEN [6] arity_is_nat_case_fm]) simp_all

manual_arity for dcwit_repl_body_fm

using arity_is_nat_case_dcwit_aux_fm[THEN [6] arity_is_wfrec_fm]

unfolding dcwit_repl_body_fm_def is_nat_case_dcwit_aux_fm_def dcwit_aux_fm_def

by (auto simp add: arity(1-33))

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

by (simp_all add: FOL_arity arity_dcwit_repl_body_fm ord_simp_union)

definition fst2_snd2

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

relativize functional fst2_snd2 fst2_snd2_rel

relationalize fst2_snd2_rel is_fst2_snd2

lemma (in *M_trivial*) *fst2_snd2_abs*:
assumes $M(x) M(res)$
shows $is_fst2_snd2(M, x, res) \longleftrightarrow res = fst2_snd2(x)$
unfolding *is_fst2_snd2_def* *fst2_snd2_def*
using *fst_rel_abs* *snd_rel_abs* *fst_abs* *snd_abs* *assms*
by *simp*

synthesize *is_fst2_snd2* **from_definition** **assuming** *nonempty*
arity_theorem **for** *is_fst2_snd2_fm*

definition *sndfst_fst2_snd2*
where $sndfst_fst2_snd2(x) \equiv \langle snd(fst(x)), fst(fst(x)), snd(snd(x)) \rangle$

relativize functional *sndfst_fst2_snd2* *sndfst_fst2_snd2_rel*
relationalize *sndfst_fst2_snd2_rel* *is_sndfst_fst2_snd2*
synthesize *is_sndfst_fst2_snd2* **from_definition** **assuming** *nonempty*
arity_theorem **for** *is_sndfst_fst2_snd2_fm*

definition *order_eq_map* **where**
 $order_eq_map(M, A, r, a, z) \equiv \exists x[M]. \exists g[M]. \exists mx[M]. \exists par[M].$
 $ordinal(M, x) \ \& \ pair(M, a, x, z) \ \& \ membership(M, x, mx) \ \&$
 $pred_set(M, A, a, r, par) \ \& \ order_isomorphism(M, par, r, x, mx, g)$

synthesize *order_eq_map* **from_definition** **assuming** *nonempty*
arity_theorem **for** *is_ord_iso_fm*
arity_theorem **for** *order_eq_map_fm*

synthesize *is_banach_functor* **from_definition** **assuming** *nonempty*
arity_theorem **for** *is_banach_functor_fm*

definition *banach_body_iterates* **where**
 $banach_body_iterates(M, X, Y, f, g, W, n, x, z) \equiv$
 $\exists y[M].$

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

$$(\exists fa[M].$$

$$(\forall z[M].$$

$$z \in fa \longleftrightarrow$$

$$(\exists xa[M].$$

$$\exists y[M].$$

$$\exists xaa[M].$$

$$\exists sx[M].$$

$$\exists r_sx[M].$$

$$\exists f_r_sx[M]. \exists sn[M]. \exists msn[M]. \text{successor}(M, n, sn)$$

\wedge

$$membership(M, sn, msn) \wedge$$

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

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

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

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

synthesize *is_quasimat* from_definition assuming nonempty
arity_theorem for *is_quasimat_fm*

synthesize *banach_body_iterates* from_definition assuming nonempty
arity_theorem for *banach_body_iterates_fm*

definition *banach_is_iterates_body* where

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

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

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

synthesize banach_is_iterates_body from_definition assuming nonempty
arity_theorem for banach_is_iterates_body_fm

definition trans_apply_image where

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

relativize functional trans_apply_image trans_apply_image_rel
relationalize trans_apply_image is_trans_apply_image

schematic_goal arity_is_recfun_fm[arity]:

$$p \in \text{formula} \implies a \in \omega \implies z \in \omega \implies r \in \omega \implies \text{arity}(\text{is_recfun_fm}(p, a, z, r)) = ?ar$$

unfolding is_recfun_fm_def

by (simp add:arity)

schematic_goal arity_is_wfrec_fm[arity]:

$$p \in \text{formula} \implies a \in \omega \implies z \in \omega \implies r \in \omega \implies \text{arity}(\text{is_wfrec_fm}(p, a, z, r)) = ?ar$$

unfolding is_wfrec_fm_def

by (simp add:arity)

schematic_goal arity_is_transrec_fm[arity]:

$$p \in \text{formula} \implies a \in \omega \implies z \in \omega \implies \text{arity}(\text{is_transrec_fm}(p, a, z)) = ?ar$$

unfolding is_transrec_fm_def

by (simp add:arity)

synthesize is_trans_apply_image from_definition assuming nonempty
arity_theorem for is_trans_apply_image_fm

definition transrec_apply_image_body where

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

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

synthesize transrec_apply_image_body from_definition assuming nonempty arity_theorem for transrec_apply_image_body_fm

definition is_trans_apply_image_body where

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

synthesize is_trans_apply_image_body from_definition assuming nonempty arity_theorem for is_trans_apply_image_body_fm

definition replacement_is_omega_funspace_fm where replacement_is_omega_funspace_fm
 $\equiv \omega_funspace_fm(2, 0, 1)$

definition wfrec_Aleph_fm where wfrec_Aleph_fm $\equiv HAleph_wfrec_repl_body_fm(2, 0, 1)$

definition replacement_is_fst2_snd2_fm where replacement_is_fst2_snd2_fm

```

≡ is_fst2_snd2_fm(0,1)
definition replacement_is_sndfst_fst2_snd2_fm where replacement_is_sndfst_fst2_snd2_fm
≡ is_sndfst_fst2_snd2_fm(0,1)
definition omap_replacement_fm where omap_replacement_fm ≡ order_eq_map_fm(2,3,0,1)
definition rec_constr_abs_fm where rec_constr_abs_fm ≡ transrec_apply_image_body_fm(3,2,0,1)
definition banach_replacement_iterates_fm where banach_replacement_iterates_fm
≡ banach_is_iterates_body_fm(6,5,4,3,2,0,1)
definition rec_constr_fm where rec_constr_fm ≡ is_trans_apply_image_body_fm(3,2,0,1)

definition dc_abs_fm where dc_abs_fm ≡ dcwit_repl_body_fm(6,5,4,3,2,0,1)
definition lam_replacement_check_fm where lam_replacement_check_fm ≡ 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 ≡
  { 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 ⊆ formula
using Lambda_in_M_fm_type
unfolding instances_ground_fms_def replacement_instances_ground_defs
by simp

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
≡
  { rec_constr_abs_fm,

```

```

    rec_constr_fm }

lemma instances_ground_notCH_fms_type[TC]: instances_ground_notCH_fms
  ⊆ formula
  unfolding instances_ground_notCH_fms_def rec_constr_abs_fm_def
    rec_constr_fm_def
  by simp

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∈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) ⇒ (##M)(A) ⇒ (##M)(a) ⇒ (##M)(s) ⇒ (##M)(R)
  ⇒
  strong_replacement(##M, dcwit_repl_body(##M,mesa,A,a,s,R))
using strong_replacement_rel_in_ctm[where φ=dcwit_repl_body_fm(6,5,4,3,2,0,1)
  and env=[R,s,a,A,mesa] and f=dcwit_repl_body(##M,mesa,A,a,s,R)]
  zero_in_M_arity_dcwit_repl_body dcwit_replacement
unfolding dc_abs_fm_def
by simp

lemma dcwit_repl:
  (##M)(sa) ⇒
  (##M)(esa) ⇒
  (##M)(mesa) ⇒ (##M)(A) ⇒ (##M)(a) ⇒ (##M)(s) ⇒
  (##M)(R) ⇒
  strong_replacement
  ((##M), λx z. ∃ y[(##M)]. pair((##M), x, y, z) ∧
  is_wfrec
  ((##M), λn f. is_nat_case

```

```

((##M), a,
 λm bmf.
   ∃ fm[(##M)].
   ∃ cp[(##M)].
   is_apply((##M), f, m, fm) ∧
   is_Collect((##M), A, λx. ∃ fmx[(##M)].
((##M)(x) ∧ fmx ∈ R) ∧ pair((##M), fm, x, fmx), cp) ∧
   is_apply((##M), s, cp, bmf)),
  n),
  mesa, x, y))
using replacement_dcwit_repl_body unfolding dcwit_repl_body_def by simp

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))
using assms separation_in_ctm[where env=[A,a,g,R] and φ=is_dcwit_body_fm(1,2,3,4,0),
  OF _ _ _ is_dcwit_body_iff_sats[symmetric],
  of λ_.A λ_.a λ_.g λ_.R λx. x]
  nonempty_arity_is_dcwit_body_fm is_dcwit_body_fm_type
by (simp add:ord_simp_union)

end — M_ZF1_trans

sublocale M_ZF1_trans ⊆ M_replacement ##M
using M_replacement_ZF_instances M_separation_ZF_instances
by unfold_locales simp

context M_ZF1_trans
begin

lemma separation_Pow_rel: A ∈ M ⇒
  separation(##M, λy. ∃ x ∈ M . x ∈ A ∧ y = ⟨x, Pow##M(x)⟩)
using separation_assm_sats[of is_Pow_fm(0,1)] arity_is_Pow_fm ord_simp_union
  Pow_rel_closed nonempty_Pow_rel_iff
by simp

lemma strong_replacement_Powapply_rel:
  f ∈ M ⇒ strong_replacement(##M, λx y. y = Powapply##M(f,x))

```

```

using Powapply_rel_replacement separation_Pow_rel transM
by simp

end — M_ZF1_trans

sublocale M_ZF1_trans  $\subseteq$  M_Vfrom ##M
using power_ax strong_replacement_Powapply_rel phrank_repl trans_repl_HVFrom
wfrec_rank
by unfold_locales auto

sublocale M_ZF1_trans  $\subseteq$  M_Perm ##M
using separation_PiP_rel separation_injP_rel separation_surjP_rel
lam_replacement_imp_strong_replacement[OF
lam_replacement_Sigfun[OF lam_replacement_constant]]
Pi_replacement1 unfolding Sigfun_def
by unfold_locales simp_all

sublocale M_ZF1_trans  $\subseteq$  M_pre_seqspace ##M
by unfold_locales

context M_ZF1_trans
begin

lemma separation_inj_rel:  $A \in M \implies$ 
  separation(##M,  $\lambda y. \exists x \in M. x \in A \wedge y = \langle x, \text{inj\_rel}(##M, \text{fst}(x), \text{snd}(x)) \rangle$ )
using arity_is_inj_fm ord_simp_union
nonempty_inj_rel_closed[simplified] inj_rel_iff[simplified]
by (rule_tac separation_assm_bin_sats[of is_inj_fm(0,1,2)])
(simp_all add:setclass_def)

lemma lam_replacement_inj_rel: lam_replacement(##M,  $\lambda x. \text{inj\_rel}(##M, \text{fst}(x), \text{snd}(x))$ )
using lam_replacement_inj_rel' separation_inj_rel
by simp

end — M_ZF1_trans

lemma (in M_basic) rel2_trans_apply:
   $M(f) \implies \text{relation2}(M, \text{is\_trans\_apply\_image}(M, f), \text{trans\_apply\_image}(f))$ 
unfolding is_trans_apply_image_def trans_apply_image_def relation2_def
by auto

lemma (in M_basic) apply_image_closed:
shows  $M(f) \implies \forall x[M]. \forall g[M]. M(\text{trans\_apply\_image}(f, x, g))$ 
unfolding trans_apply_image_def by simp

context M_ZF_ground_notCH_trans
begin

```

lemma *replacement_transrec_apply_image_body* :
 $(\#\#M)(f) \implies (\#\#M)(\text{mesa}) \implies \text{strong_replacement}(\#\#M, \text{transrec_apply_image_body}(\#\#M, f, \text{mesa}))$
using *strong_replacement_rel_in_ctm* [where $\varphi = \text{transrec_apply_image_body_fm}(3, 2, 0, 1)$
and *env* = [mesa, f]
zero_in_M arity_transrec_apply_image_body_fm ord_simp_union
ZF_ground_notCH_replacements(1)
by *simp*

lemma *transrec_replacement_apply_image*:
assumes $(\#\#M)(f) (\#\#M)(\alpha)$
shows $\text{transrec_replacement}(\#\#M, \text{is_trans_apply_image}(\#\#M, f), \alpha)$
using *replacement_transrec_apply_image_body* [unfolded *transrec_apply_image_body_def*]
assms
Memrel_closed_singleton_closed eclose_closed
unfolding *transrec_replacement_def wfrec_replacement_def is_wfrec_def M_is_recfun_def*
by *simp*

lemma *rec_trans_apply_image_abs*:
assumes $(\#\#M)(f) (\#\#M)(x) (\#\#M)(y) \text{Ord}(x)$
shows $\text{is_transrec}(\#\#M, \text{is_trans_apply_image}(\#\#M, f), x, y) \longleftrightarrow y = \text{transrec}(x, \text{trans_apply_image}(f))$
using *transrec_abs* [OF *transrec_replacement_apply_image_rel2_trans_apply*]
assms apply_image_closed
by *simp*

lemma *replacement_is_trans_apply_image*:
 $(\#\#M)(f) \implies (\#\#M)(\beta) \implies \text{strong_replacement}(\#\#M, \lambda x z .$
 $\exists y[\#\#M]. \text{pair}(\#\#M, x, y, z) \wedge x \in \beta \wedge (\text{is_transrec}(\#\#M, \text{is_trans_apply_image}(\#\#M,$
 $f), x, y)))$
unfolding *is_transrec_def is_wfrec_def M_is_recfun_def*
apply (*rule_tac strong_replacement_cong* [where $P = \lambda x z. M, [x, z, \beta, f] \models \text{is_trans_apply_image_body_fm}(3, 2, 0, 1), \text{THEN}$
iffD1])
apply (*rule_tac is_trans_apply_image_body_iff_sats* [symmetric, unfolded *is_trans_apply_image_body_def*,
env = [_, _, β, f]])
apply (*simp_all add: zero_in_M*)
apply (*rule_tac ZF_ground_notCH_replacements(2)* [unfolded *replacement_assm_def*,
rule_format, where *env* = [β, f], *simplified*])
apply (*simp_all add: arity_is_trans_apply_image_body_fm is_trans_apply_image_body_fm_type*
ord_simp_union)
done

lemma *trans_apply_abs*:
 $(\#\#M)(f) \implies (\#\#M)(\beta) \implies \text{Ord}(\beta) \implies (\#\#M)(x) \implies (\#\#M)(z) \implies$
 $(x \in \beta \wedge z = \langle x, \text{transrec}(x, \lambda a g. f \text{ ` } (g \text{ ` } a) \rangle) \longleftrightarrow$
 $(\exists y[\#\#M]. \text{pair}(\#\#M, x, y, z) \wedge x \in \beta \wedge (\text{is_transrec}(\#\#M, \text{is_trans_apply_image}(\#\#M,$
 $f), x, y)))$
using *rec_trans_apply_image_abs Ord_in_Ord*

$\text{transrec_closed}[OF \text{ transrec_replacement_apply_image _rel2_trans_apply, of } f, \text{simplified}]$
 $\text{apply_image_closed}$
unfolding $\text{trans_apply_image_def}$
by auto

lemma $\text{replacement_trans_apply_image}$:
 $(\#\#M)(f) \implies (\#\#M)(\beta) \implies \text{Ord}(\beta) \implies$
 $\text{strong_replacement}(\#\#M, \lambda x y. x \in \beta \wedge y = \langle x, \text{transrec}(x, \lambda a g. f \text{ ‘ } (g \text{ ‘ } a) \rangle))$
using $\text{strong_replacement_cong}[THEN \text{ iffD1, OF_replacement_is_trans_apply_image, simplified}]$
 $\text{trans_apply_abs _Ord_in_Ord}$
by simp

end — $M_ZF_ground_notCH_trans$

definition ifrFb_body **where**
 $\text{ifrFb_body}(M, b, f, x, i) \equiv x \in$
 $(\text{if } b = 0 \text{ then if } i \in \text{range}(f) \text{ then}$
 $\text{if } M(\text{converse}(f) \text{ ‘ } i) \text{ then converse}(f) \text{ ‘ } i \text{ else } 0 \text{ else } 0 \text{ else if } M(i) \text{ then } i \text{ else } 0)$

relativize functional ifrFb_body ifrFb_body_rel
relationalize ifrFb_body_rel is_ifrFb_body

synthesize is_ifrFb_body from_definition **assuming** nonempty
arity_theorem for is_ifrFb_body_fm

definition $\text{ifrangeF_body} :: [i \Rightarrow o, i, i, i, i] \Rightarrow o$ **where**
 $\text{ifrangeF_body}(M, A, b, f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. \text{ifrFb_body}(M, b, f, x, i) \rangle$

relativize functional ifrangeF_body ifrangeF_body_rel
relationalize ifrangeF_body_rel is_ifrangeF_body

synthesize is_ifrangeF_body from_definition **assuming** nonempty
arity_theorem for $\text{is_ifrangeF_body_fm}$

lemma (**in** M_Z_trans) $\text{separation_is_ifrangeF_body}$:
 $(\#\#M)(A) \implies (\#\#M)(r) \implies (\#\#M)(s) \implies \text{separation}(\#\#M, \text{is_ifrangeF_body}(\#\#M, A, r, s))$
using $\text{separation_in_ctm}[\text{where } \varphi = \text{is_ifrangeF_body_fm}(1, 2, 3, 0) \text{ and } \text{env} = [A, r, s]]$
 $\text{zero_in_M_arity_is_ifrangeF_body_fm _ord_simp_union _is_ifrangeF_body_fm_type}$
by simp

lemma (**in** M_basic) $\text{is_ifrFb_body_closed}$: $M(r) \implies M(s) \implies \text{is_ifrFb_body}(M, r, s, x, i) \implies M(i)$
unfolding $\text{ifrangeF_body_def _is_ifrangeF_body_def _is_ifrFb_body_def _If_abs}$
by ($\text{cases } i \in \text{range}(s)$; $\text{cases } r = 0$; auto dest:transM)

lemma (**in** M_ZF1_trans) ifrangeF_body_abs :
assumes $(\#\#M)(A) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$
shows $\text{is_ifrangeF_body}(\#\#M, A, r, s, x) \longleftrightarrow \text{ifrangeF_body}(\#\#M, A, r, s, x)$

```

proof -
{
  fix a
  assume  $a \in M$ 
  with assms
  have  $(\mu i. i \in M \wedge is\_ifrFb\_body(\#\#M, r, s, z, i)) = (\mu i. is\_ifrFb\_body(\#\#M, r, s, z, i))$  for z
    using is_ifrFb_body_closed[of r s z]
    by (rule_tac Least_cong[of  $\lambda i. i \in M \wedge is\_ifrFb\_body(\#\#M, r, s, z, i)$ ]) auto
  moreover
  have  $(\mu i. is\_ifrFb\_body(\#\#M, r, s, z, i)) = (\mu i. ifrFb\_body(\#\#M, r, s, z, i))$  for z
  proof (rule_tac Least_cong[of  $\lambda i. is\_ifrFb\_body(\#\#M, r, s, z, i)$   $\lambda i. ifrFb\_body(\#\#M, r, s, z, i)$ ])
    fix y
    from assms  $\langle a \in M \rangle$ 
    show  $is\_ifrFb\_body(\#\#M, r, s, z, y) \longleftrightarrow ifrFb\_body(\#\#M, r, s, z, y)$ 
      using If_abs_apply_0
      unfolding ifrFb_body_def is_ifrFb_body_def
      by (cases  $y \in M$ ; cases  $y \in range(s)$ ; cases converse(s) ' $y \in M$ ';
        auto dest:transM split del: split_if del:iffI)
      (auto simp flip:setclass_iff; (force simp only:setclass_iff))+
    qed
  moreover from  $\langle a \in M \rangle$ 
  have  $least(\#\#M, \lambda i. i \in M \wedge is\_ifrFb\_body(\#\#M, r, s, z, i), a) \longleftrightarrow a = (\mu i. i \in M \wedge is\_ifrFb\_body(\#\#M, r, s, z, i))$  for z
    using If_abs least_abs'[of  $\lambda i. (\#\#M)(i) \wedge is\_ifrFb\_body(\#\#M, r, s, z, i)$ ]
    by simp
  ultimately
  have  $least(\#\#M, \lambda i. i \in M \wedge is\_ifrFb\_body(\#\#M, r, s, z, i), a) \longleftrightarrow a = (\mu i. ifrFb\_body(\#\#M, r, s, z, i))$  for z
    by simp
}
with assms
show ?thesis
  using pair_in_M_iff_apply_closed zero_in_M transitivity[of  $\_ A$ ]
  unfolding ifrangeF_body_def is_ifrangeF_body_def
  by (auto dest:transM)
qed

```

```

lemma (in M_ZF1_trans) separation_ifrangeF_body:
 $(\#\#M)(A) \implies (\#\#M)(b) \implies (\#\#M)(f) \implies separation$ 
 $(\#\#M, \lambda y. \exists x \in A. y = \langle x, \mu i. x \in if\_range\_F\_else\_F(\lambda x. if (\#\#M)(x) then x else 0, b, f, i) \rangle)$ 
  using separation_is_ifrangeF_body ifrangeF_body_abs
  separation_cong[where  $P = is\_ifrangeF\_body(\#\#M, A, b, f)$  and  $M = \#\#M$ , THEN iffD1]
  unfolding ifrangeF_body_def if_range_F_def if_range_F_else_F_def ifrFb_body_def
  by simp

```

definition *ifrFb_body2* **where**

ifrFb_body2(M, G, b, f, x, i) $\equiv x \in$
(if $b = 0$ then if $i \in \text{range}(f)$ 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)

relativize functional *ifrFb_body2* *ifrFb_body2_rel*

relationalize *ifrFb_body2_rel* *is_ifrFb_body2*

synthesize *is_ifrFb_body2* **from_definition** **assuming** *nonempty*

arity_theorem **for** *is_ifrFb_body2_fm*

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

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

relativize functional *ifrangeF_body2* *ifrangeF_body2_rel*

relationalize *ifrangeF_body2_rel* *is_ifrangeF_body2*

synthesize *is_ifrangeF_body2* **from_definition** **assuming** *nonempty*

arity_theorem **for** *is_ifrangeF_body2_fm*

lemma (in *M_Z_trans*) *separation_is_ifrangeF_body2*:

$(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(s) \Longrightarrow \text{separation}(\#\#M,$
 $\text{is_ifrangeF_body2}(\#\#M, A, G, r, s))$

using *separation_in_ctm* [**where** $\varphi = \text{is_ifrangeF_body2_fm}(1, 2, 3, 4, 0)$ **and** $\text{env} = [A, G, r, s]$
zero_in_M *arity_is_ifrangeF_body2_fm* *ord_simp_union* *is_ifrangeF_body2_fm_type*
by *simp*

lemma (in *M_basic*) *is_ifrFb_body2_closed*: $M(G) \Longrightarrow M(r) \Longrightarrow M(s) \Longrightarrow$
 $\text{is_ifrFb_body2}(M, G, r, s, x, i) \Longrightarrow M(i)$

unfolding *ifrangeF_body2_def* *is_ifrangeF_body2_def* *is_ifrFb_body2_def* *If_abs*
by (*cases* $i \in \text{range}(s)$); *cases* $r=0$; *auto* *dest:transM*)

lemma (in *M_ZF1_trans*) *ifrangeF_body2_abs*:

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

shows $\text{is_ifrangeF_body2}(\#\#M, A, G, r, s, x) \longleftrightarrow \text{ifrangeF_body2}(\#\#M, A, G, r, s, x)$

proof -

{

fix a

assume $a \in M$

with *assms*

have $(\mu i. i \in M \wedge \text{is_ifrFb_body2}(\#\#M, G, r, s, z, i)) = (\mu i. \text{is_ifrFb_body2}(\#\#M,$
 $G, r, s, z, i))$ **for** z

using *is_ifrFb_body2_closed* [*of* G r s z]

by (*rule_tac* *Least_cong* [*of* $\lambda i. i \in M \wedge \text{is_ifrFb_body2}(\#\#M, G, r, s, z, i)$])

auto

moreover

```

have ( $\mu$   $i$ .  $is\_ifrFb\_body2(\#\#M, G, r, s, z, i)$ ) = ( $\mu$   $i$ .  $ifrFb\_body2(\#\#M, G,$ 
 $r, s, z, i)$ ) for  $z$ 
proof ( $rule\_tac$   $Least\_cong$ [ $of$   $\lambda i$ .  $is\_ifrFb\_body2(\#\#M, G, r, s, z, i)$   $\lambda i$ .  $ifrFb\_body2(\#\#M, G, r, s, z, i)$ ])
  fix  $y$ 
  from  $assms \langle a \in M \rangle$ 
  show  $is\_ifrFb\_body2(\#\#M, G, r, s, z, y) \longleftrightarrow ifrFb\_body2(\#\#M, G, r, s,$ 
 $z, y)$ 
    using  $If\_abs$   $apply\_0$ 
    unfolding  $ifrFb\_body2\_def$   $is\_ifrFb\_body2\_def$ 
    by ( $cases$   $y \in M$ ;  $cases$   $y \in range(s)$ ;  $cases$   $converse(s)$ ) ' $y \in M$ ;
       $auto$   $dest:transM$   $split$   $del: split\_if$   $del:iffI$ )
      ( $auto$   $simp$   $flip:setclass\_iff$ ; ( $force$   $simp$   $only:setclass\_iff$ ))+
  qed
moreover from  $\langle a \in M \rangle$ 
have  $least(\#\#M, \lambda i. i \in M \wedge is\_ifrFb\_body2(\#\#M, G, r, s, z, i), a)$ 
 $\longleftrightarrow a = (\mu$   $i. i \in M \wedge is\_ifrFb\_body2(\#\#M, G, r, s, z, i))$  for  $z$ 
  using  $If\_abs$   $least\_abs'$ [ $of$   $\lambda i. (\#\#M)(i) \wedge is\_ifrFb\_body2(\#\#M, G, r, s, z, i)$ 
 $a$ ]
  by  $simp$ 
ultimately
have  $least(\#\#M, \lambda i. i \in M \wedge is\_ifrFb\_body2(\#\#M, G, r, s, z, i), a)$ 
 $\longleftrightarrow a = (\mu$   $i. ifrFb\_body2(\#\#M, G, r, s, z, i))$  for  $z$ 
  by  $simp$ 
}
with  $assms$ 
show  $?thesis$ 
  using  $pair\_in\_M\_iff$   $apply\_closed$   $zero\_in\_M$   $transitivity$ [ $of$   $A$ ]
  unfolding  $ifrangeF\_body2\_def$   $is\_ifrangeF\_body2\_def$ 
  by ( $auto$   $dest:transM$ )
qed

lemma (in  $M\_ZF1\_trans$ )  $separation\_ifrangeF\_body2$ :
   $(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(b) \implies (\#\#M)(f) \implies$ 
 $separation$ 
  ( $\#\#M,$ 
   $\lambda y. \exists x \in A.$ 
   $y =$ 
   $\langle x, \mu i. x \in$ 
   $if\_range\_F\_else\_F(\lambda a. if (\#\#M)(a) then G \text{ ' } a \text{ else } 0, b, f,$ 
 $i) \rangle$ )
  using  $separation\_is\_ifrangeF\_body2$   $ifrangeF\_body2\_abs$ 
 $separation\_cong$ [where  $P = is\_ifrangeF\_body2(\#\#M, A, G, b, f)$  and  $M = \#\#M$ , THEN
 $iffD1$ ]
  unfolding  $ifrangeF\_body2\_def$   $if\_range\_F\_def$   $if\_range\_F\_else\_F\_def$   $ifrFb\_body2\_def$ 
  by  $simp$ 

```

definition $ifrFb_body3$ **where**

$ifrFb_body3(M, G, b, f, x, i) \equiv x \in$
 (if $b = 0$ then if $i \in \text{range}(f)$ then
 if $M(\text{converse}(f) \text{ ' } i)$ then $G\text{-}\{\text{converse}(f) \text{ ' } i\}$ else 0 else 0 else if $M(i)$ then
 $G\text{-}\{i\}$ else 0)

relativize functional $ifrFb_body3$ $ifrFb_body3_rel$
relationalize $ifrFb_body3_rel$ is_ifrFb_body3

synthesize is_ifrFb_body3 **from definition** assuming *nonempty*
arity theorem for $is_ifrFb_body3_fm$

definition $ifrFb_body3 :: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$ **where**
 $ifrFb_body3(M, A, G, b, f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb_body3(M, G, b, f, x, i) \rangle$

relativize functional $ifrFb_body3$ $ifrFb_body3_rel$
relationalize $ifrFb_body3_rel$ is_ifrFb_body3

synthesize is_ifrFb_body3 **from definition** assuming *nonempty*
arity theorem for $is_ifrFb_body3_fm$

lemma (in M_Z_trans) $separation_is_ifrFb_body3$:
 $(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(s) \Longrightarrow separation(\#\#M,$
 $is_ifrFb_body3(\#\#M, A, G, r, s))$
using $separation_in_ctm$ [where $\varphi = is_ifrFb_body3_fm(1, 2, 3, 4, 0)$ and $env = [A, G, r, s]$
 $zero_in_M$ $arity_is_ifrFb_body3_fm$ ord_simp_union $is_ifrFb_body3_fm_type$
by *simp*

lemma (in M_basic) $is_ifrFb_body3_closed$: $M(G) \Longrightarrow M(r) \Longrightarrow M(s) \Longrightarrow$
 $is_ifrFb_body3(M, G, r, s, x, i) \Longrightarrow M(i)$
unfolding $ifrFb_body3_def$ $is_ifrFb_body3_def$ $is_ifrFb_body3_def$ If_abs
by (*cases* $i \in \text{range}(s)$); *cases* $r=0$; *auto* *dest:transM*)

lemma (in M_ZF1_trans) $ifrFb_body3_abs$:
assumes $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$
shows $is_ifrFb_body3(\#\#M, A, G, r, s, x) \longleftrightarrow ifrFb_body3(\#\#M, A, G, r, s, x)$
proof -

{
fix a
assume $a \in M$
with *assms*
have $(\mu i. i \in M \wedge is_ifrFb_body3(\#\#M, G, r, s, z, i)) = (\mu i. is_ifrFb_body3(\#\#M,$
 $G, r, s, z, i))$ **for** z
using $is_ifrFb_body3_closed$ [of G r s z]
by (*rule_tac* $Least_cong$ [of $\lambda i. i \in M \wedge is_ifrFb_body3(\#\#M, G, r, s, z, i)$])

auto

moreover
have $(\mu i. is_ifrFb_body3(\#\#M, G, r, s, z, i)) = (\mu i. ifrFb_body3(\#\#M, G,$
 $r, s, z, i))$ **for** z
proof (*rule_tac* $Least_cong$ [of $\lambda i. is_ifrFb_body3(\#\#M, G, r, s, z, i)$ $\lambda i. ifrFb_body3(\#\#M, G, r, s, z, i)$])

```

fix  $y$ 
from  $assms \langle a \in M \rangle$ 
show  $is\_ifrFb\_body3(\#\#M, G, r, s, z, y) \longleftrightarrow ifrFb\_body3(\#\#M, G, r, s,$ 
 $z, y)$ 
  using  $If\_abs\ apply\_0$ 
  unfolding  $ifrFb\_body3\_def\ is\_ifrFb\_body3\_def$ 
  by ( $cases\ y \in M;$   $cases\ y \in range(s);$   $cases\ converse(s) 'y \in M;$ 
     $auto\ dest:transM\ split\ del:split\_if\ del:iffI$ )
    ( $auto\ simp\ flip:setclass\_iff;$   $(force\ simp\ only:setclass\_iff)$ ) +
qed
moreover from  $\langle a \in M \rangle$ 
have  $least(\#\#M, \lambda i. i \in M \wedge is\_ifrFb\_body3(\#\#M, G, r, s, z, i), a)$ 
 $\longleftrightarrow a = (\mu i. i \in M \wedge is\_ifrFb\_body3(\#\#M, G, r, s, z, i))$  for  $z$ 
  using  $If\_abs\ least\_abs'$  [of  $\lambda i. (\#\#M)(i) \wedge is\_ifrFb\_body3(\#\#M, G, r, s, z, i)$ 
 $a]$ 
  by  $simp$ 
  ultimately
have  $least(\#\#M, \lambda i. i \in M \wedge is\_ifrFb\_body3(\#\#M, G, r, s, z, i), a)$ 
 $\longleftrightarrow a = (\mu i. ifrFb\_body3(\#\#M, G, r, s, z, i))$  for  $z$ 
  by  $simp$ 
}
with  $assms$ 
show  $?thesis$ 
  using  $pair\_in\_M\_iff\ apply\_closed\ zero\_in\_M\ transitivity[of\_A]$ 
  unfolding  $ifrangeF\_body3\_def\ is\_ifrangeF\_body3\_def$ 
  by ( $auto\ dest:transM$ )
qed

```

```

lemma (in  $M\_ZF1\_trans$ )  $separation\_ifrangeF\_body3$ :
 $(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(b) \implies (\#\#M)(f) \implies$ 
 $separation$ 
 $(\#\#M,$ 
 $\lambda y. \exists x \in A.$ 
 $y =$ 
 $\langle x, \mu i. x \in$ 
 $if\_range\_F\_else\_F(\lambda a. if\ (\#\#M)(a)\ then\ G\ \{a\}\ else\ 0, b,$ 
 $f, i) \rangle)$ 
  using  $separation\_is\_ifrangeF\_body3\ ifrangeF\_body3\_abs$ 
 $separation\_cong$  [where  $P = is\_ifrangeF\_body3(\#\#M, A, G, b, f)$  and  $M = \#\#M,$  THEN
 $iffD1]$ 
  unfolding  $ifrangeF\_body3\_def\ if\_range\_F\_def\ if\_range\_F\_else\_F\_def\ ifrFb\_body3\_def$ 
  by  $simp$ 

```

definition $ifrFb_body4$ **where**
 $ifrFb_body4(G, b, f, x, i) \equiv x \in$
 $(if\ b = 0\ then\ if\ i \in range(f)\ then\ G'(converse(f)\ 'i)\ else\ 0\ else\ G'i)$

relativize functional *ifrFb_body4 ifrFb_body4_rel*
relationalize *ifrFb_body4_rel is_ifrFb_body4*

synthesize *is_ifrFb_body4* **from_definition** **assuming** *nonempty*
arity_theorem **for** *is_ifrFb_body4_fm*

definition *ifrangeF_body4* :: $[i \Rightarrow o, i, i, i, i] \Rightarrow o$ **where**
ifrangeF_body4(M, A, G, b, f) \equiv $\lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb_body4(G, b, f, x, i) \rangle$

relativize functional *ifrangeF_body4 ifrangeF_body4_rel*
relationalize *ifrangeF_body4_rel is_ifrangeF_body4*

synthesize *is_ifrangeF_body4* **from_definition** **assuming** *nonempty*
arity_theorem **for** *is_ifrangeF_body4_fm*

lemma (**in** *M_Z_trans*) *separation_is_ifrangeF_body4*:
 $(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(s) \Longrightarrow \text{separation}(\#\#M,$
is_ifrangeF_body4(\#\#M, A, G, r, s))
using *separation_in_ctm* [**where** $\varphi = \text{is_ifrangeF_body4_fm}(1, 2, 3, 4, 0)$ **and** $\text{env} = [A, G, r, s]$
zero_in_M_arity_is_ifrangeF_body4_fm ord_simp_union is_ifrangeF_body4_fm_type
by *simp*

lemma (**in** *M_basic*) *is_ifrFb_body4_closed*: $M(G) \Longrightarrow M(r) \Longrightarrow M(s) \Longrightarrow$
is_ifrFb_body4(M, G, r, s, x, i) $\Longrightarrow M(i)$
using *If_abs*
unfolding *ifrangeF_body4_def is_ifrangeF_body4_def is_ifrFb_body4_def fun_apply_def*
by (*cases i \in range(s); cases r=0; auto dest:transM*)

lemma (**in** *M_ZF1_trans*) *ifrangeF_body4_abs*:
assumes $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$
shows $\text{is_ifrangeF_body4}(\#\#M, A, G, r, s, x) \longleftrightarrow \text{ifrangeF_body4}(\#\#M, A, G, r, s, x)$
proof -
 {
 fix *a*
 assume $a \in M$
 with *assms*
 have $(\mu i. i \in M \wedge \text{is_ifrFb_body4}(\#\#M, G, r, s, z, i)) = (\mu i. \text{is_ifrFb_body4}(\#\#M,$
G, r, s, z, i)) **for** *z*
 using *is_ifrFb_body4_closed* [*of G r s z*]
 by (*rule_tac Least_cong* [*of $\lambda i. i \in M \wedge \text{is_ifrFb_body4}(\#\#M, G, r, s, z, i)$*])
auto
 moreover
 have $(\mu i. \text{is_ifrFb_body4}(\#\#M, G, r, s, z, i)) = (\mu i. \text{ifrFb_body4}(G, r, s, z,$
i)) **if** $z \in M$ **for** *z*
 proof (*rule_tac Least_cong* [*of $\lambda i. \text{is_ifrFb_body4}(\#\#M, G, r, s, z, i)$ $\lambda i. \text{ifrFb_body4}(G, r, s, z, i)$*])
 fix *y*
 from *assms* $\langle a \in M \rangle \langle z \in M \rangle$
 show $\text{is_ifrFb_body4}(\#\#M, G, r, s, z, y) \longleftrightarrow \text{ifrFb_body4}(G, r, s, z, y)$
 using *If_abs apply_0*

```

unfolding ifrFb_body4_def is_ifrFb_body4_def
apply (cases y∈M; cases y∈range(s); cases r=0; cases y∈domain(G);
        auto dest:transM split del: split_if del:iffI)
by (auto simp flip:setclass_iff; (force simp only: fun_apply_def setclass_iff)
      (auto simp flip:setclass_iff simp: fun_apply_def )
qed
moreover from ⟨a∈M⟩
have least(##M, λi. i ∈ M ∧ is_ifrFb_body4(##M, G, r, s, z, i), a)
  ↔ a = (μ i. i ∈ M ∧ is_ifrFb_body4(##M, G, r, s, z, i)) for z
using If_abs least_abs[of λi. (##M)(i) ∧ is_ifrFb_body4(##M, G, r, s, z, i)
a]
  by simp
ultimately
have z∈M ⇒ least(##M, λi. i ∈ M ∧ is_ifrFb_body4(##M, G, r, s, z, i),
a)
  ↔ a = (μ i. ifrFb_body4(G, r, s, z, i)) for z
  by simp
}
with assms
show ?thesis
  using pair_in_M_iff apply_closed zero_in_M transitivity[of _ A]
  unfolding ifrangeF_body4_def is_ifrangeF_body4_def
  by (auto dest:transM)
qed

lemma (in M_ZF1_trans) separation_ifrangeF_body4:
  (##M)(A) ⇒ (##M)(G) ⇒ (##M)(b) ⇒ (##M)(f) ⇒
    separation(##M, λy. ∃ x∈A. y = ⟨x, μ i. x ∈ if_range_F_else_F((·)(G),
b, f, i)⟩)
  using separation_is_ifrangeF_body4 ifrangeF_body4_abs
    separation_cong[where P=is_ifrangeF_body4(##M,A,G,b,f) and M=##M, THEN
iffD1]
  unfolding ifrangeF_body4_def if_range_F_def if_range_F_else_F_def ifrFb_body4_def
  by simp

definition ifrFb_body5 where
  ifrFb_body5(G,b,f,x,i) ≡ x ∈
  (if b = 0 then if i ∈ range(f) then {xa ∈ G . converse(f) ‘ i ∈ xa} else 0 else {xa
∈ G . i ∈ xa})

relativize functional ifrFb_body5 ifrFb_body5_rel
relationalize ifrFb_body5_rel is_ifrFb_body5

synthesize is_ifrFb_body5 from definition assuming nonempty
arity_theorem for is_ifrFb_body5_fm

definition ifrangeF_body5 :: [i⇒o,i,i,i,i,i] ⇒ o where

```

$ifrangef_body5(M,A,G,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb_body5(G,b,f,x,i) \rangle$

relativize functional $ifrangef_body5$ $ifrangef_body5_rel$
relationalize $ifrangef_body5_rel$ $is_ifrangef_body5$

synthesize $is_ifrangef_body5$ **from definition assuming** $nonempty$
arity theorem for $is_ifrangef_body5_fm$

lemma (in M_Z_trans) $separation_is_ifrangef_body5$:
 $(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(r) \implies (\#\#M)(s) \implies separation(\#\#M,$
 $is_ifrangef_body5(\#\#M,A,G,r,s))$
using $separation_in_ctm$ [where $\varphi = is_ifrangef_body5_fm(1,2,3,4,0)$ and $env = [A,G,r,s]$
 $zero_in_M$ $arity_is_ifrangef_body5_fm$ ord_simp_union $is_ifrangef_body5_fm_type$
by $simp$

lemma (in M_basic) $is_ifrFb_body5_closed$: $M(G) \implies M(r) \implies M(s) \implies$
 $is_ifrFb_body5(M, G, r, s, x, i) \implies M(i)$
using If_abs
unfolding $ifrangef_body5_def$ $is_ifrangef_body5_def$ $is_ifrFb_body5_def$ fun_apply_def
by (cases $i \in range(s)$; cases $r=0$; auto $dest:transM$)

lemma (in M_ZF1_trans) $ifrangef_body5_abs$:
assumes $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$
shows $is_ifrangef_body5(\#\#M,A,G,r,s,x) \longleftrightarrow ifrangef_body5(\#\#M,A,G,r,s,x)$
proof -

{
fix a
assume $a \in M$
with $assms$
have $(\mu i. i \in M \wedge is_ifrFb_body5(\#\#M, G, r, s, z, i)) = (\mu i. is_ifrFb_body5(\#\#M,$
 $G, r, s, z, i))$ **for** z
using $is_ifrFb_body5_closed$ [of G r s z]
by (rule tac $Least_cong$ [of $\lambda i. i \in M \wedge is_ifrFb_body5(\#\#M, G, r, s, z, i)$])
auto
moreover
have $(\mu i. is_ifrFb_body5(\#\#M, G, r, s, z, i)) = (\mu i. ifrFb_body5(G, r, s,$
 $z, i))$ **if** $z \in M$ **for** z
proof (rule tac $Least_cong$ [of $\lambda i. is_ifrFb_body5(\#\#M, G, r, s, z, i)$ $\lambda i. ifrFb_body5(G, r, s, z, i)$])
fix y
from $assms$ $\langle a \in M \rangle$ $\langle z \in M \rangle$
show $is_ifrFb_body5(\#\#M, G, r, s, z, y) \longleftrightarrow ifrFb_body5(G, r, s, z, y)$
using If_abs $apply_0$ $separation_in_constant$ $separation_in_rev$
unfolding $ifrFb_body5_def$ $is_ifrFb_body5_def$
apply (cases $y \in M$; cases $y \in range(s)$; cases $r=0$; cases $y \in domain(G)$;
 $auto$ $dest:transM$ $split$ $del: split_if$ $del: iffI$)
apply (auto $simp$ $flip:setclass_iff$; (force $simp$ only: fun_apply_def
 $setclass_iff$))
apply (auto $simp$ $flip:setclass_iff$ $simp: fun_apply_def$)
apply (auto $dest:transM$)

```

done
qed
moreover from ‹a∈M›
have least(##M, λi. i ∈ M ∧ is_ifrFb_body5(##M, G, r, s, z, i), a)
  ‹↔ a = (μ i. i ∈ M ∧ is_ifrFb_body5(##M, G, r, s, z, i)) for z
  using If_abs least_abs[of λi. (##M)(i) ∧ is_ifrFb_body5(##M, G, r, s, z, i)
a]
  by simp
ultimately
have z∈M ⇒ least(##M, λi. i ∈ M ∧ is_ifrFb_body5(##M, G, r, s, z, i),
a)
  ‹↔ a = (μ i. ifrFb_body5(G, r, s, z, i)) for z
  by simp
}
with assms
show ?thesis
  using pair_in_M_iff_apply_closed zero_in_M transitivity[of _ A]
  unfolding ifrangeF_body5_def is_ifrangeF_body5_def
  by (auto dest:transM)
qed

```

```

lemma (in M_ZF1_trans) separation_ifrangeF_body5:
  (##M)(A) ⇒ (##M)(G) ⇒ (##M)(b) ⇒ (##M)(f) ⇒
  separation(##M, λy. ∃ x∈A. y = ⟨x, μ i. x ∈ if_range_F_else_F(λx. {xa
∈ G . x ∈ xa}, b, f, i)⟩)
  using separation_is_ifrangeF_body5 ifrangeF_body5_abs
  separation_cong[where P=is_ifrangeF_body5(##M, A, G, b, f) and M=##M, THEN
iffD1]
  unfolding ifrangeF_body5_def if_range_F_def if_range_F_else_F_def ifrFb_body5_def
  by simp

```

```

definition ifrFb_body6 where
  ifrFb_body6(G, b, f, x, i) ≡ x ∈
  (if b = 0 then if i ∈ range(f) then {p∈G . domain(p) = converse(f) ‘ i} else 0
  else {p∈G . domain(p) = i})

```

```

relativize functional ifrFb_body6 ifrFb_body6_rel
relationalize ifrFb_body6_rel is_ifrFb_body6

```

```

synthesize is_ifrFb_body6 from_definition assuming nonempty
arity_theorem for is_ifrFb_body6_fm

```

```

definition ifrangeF_body6 :: [i⇒o, i, i, i, i] ⇒ o where
  ifrangeF_body6(M, A, G, b, f) ≡ λy. ∃ x∈A. y = ⟨x, μ i. ifrFb_body6(G, b, f, x, i)⟩

```

```

relativize functional ifrangeF_body6 ifrangeF_body6_rel
relationalize ifrangeF_body6_rel is_ifrangeF_body6

```

synthesize *is_ifrangeF_body6* from definition assuming *nonempty*
arity_theorem for *is_ifrangeF_body6_fm*

lemma (in *M_Z_trans*) *separation_is_ifrangeF_body6*:
 $(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(r) \implies (\#\#M)(s) \implies \text{separation}(\#\#M,$
 $\text{is_ifrangeF_body6}(\#\#M,A,G,r,s))$
using *separation_in_ctm*[**where** $\varphi = \text{is_ifrangeF_body6_fm}(1,2,3,4,0)$ **and** $\text{env} = [A,G,r,s]$
 $\text{zero_in_M_arity_is_ifrangeF_body6_fm}$ ord_simp_union $\text{is_ifrangeF_body6_fm_type}$
by *simp*

lemma (in *M_basic*) *ifrFb_body6_closed*: $M(G) \implies M(r) \implies M(s) \implies \text{ifrFb_body6}(G,$
 $r, s, x, i) \longleftrightarrow M(i) \wedge \text{ifrFb_body6}(G, r, s, x, i)$
using *If_abs*
unfolding *ifrangeF_body6_def* *is_ifrangeF_body6_def* *ifrFb_body6_def* *fun_apply_def*
by (cases *i* in *range*(*s*); cases *r*=0; auto *dest:transM*)

lemma (in *M_basic*) *is_ifrFb_body6_closed*: $M(G) \implies M(r) \implies M(s) \implies$
 $\text{is_ifrFb_body6}(M, G, r, s, x, i) \implies M(i)$
using *If_abs*
unfolding *ifrangeF_body6_def* *is_ifrangeF_body6_def* *is_ifrFb_body6_def* *fun_apply_def*
by (cases *i* in *range*(*s*); cases *r*=0; auto *dest:transM*)

lemma (in *M_ZF1_trans*) *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 -
{
fix *a*
assume $a \in M$
with *assms*
have $(\mu i. i \in M \wedge \text{is_ifrFb_body6}(\#\#M, G, r, s, z, i)) = (\mu i. \text{is_ifrFb_body6}(\#\#M,$
 $G, r, s, z, i))$ **for** *z*
using *is_ifrFb_body6_closed*[*of* *G r s z*]
by (*rule_tac* *Least_cong*[*of* $\lambda i. i \in M \wedge \text{is_ifrFb_body6}(\#\#M, G, r, s, z, i)$])
auto
moreover
have $(\mu i. i \in M \wedge \text{is_ifrFb_body6}(\#\#M, G, r, s, z, i)) = (\mu i. i \in M \wedge$
 $\text{ifrFb_body6}(G, r, s, z, i))$ **if** $z \in M$ **for** *z*
proof (*rule_tac* *Least_cong*[*of* $\lambda i. i \in M \wedge \text{is_ifrFb_body6}(\#\#M, G, r, s, z, i)$ $\lambda i.$
 $i \in M \wedge \text{ifrFb_body6}(G, r, s, z, i)$])
fix *y*
from *assms* $\langle a \in M \rangle \langle z \in M \rangle$
show $y \in M \wedge \text{is_ifrFb_body6}(\#\#M, G, r, s, z, y) \longleftrightarrow y \in M \wedge \text{ifrFb_body6}(G,$
 $r, s, z, y)$
using *If_abs* *apply_0* *separation_in_constant* *transitivity*[*of* $_ G$]
 separation_closed *converse_closed* *apply_closed* *range_closed* *zero_in_M*
 separation_cong [*OF* *eq_commute*, *THEN* *iffD1*, *OF* *domain_eq_separation*]
unfolding *ifrFb_body6_def* *is_ifrFb_body6_def*

```

    by auto
  qed
  moreover from ⟨a∈M⟩
  have least(##M, λi. i ∈ M ∧ is_ifrFb_body6(##M, G, r, s, z, i), a)
    ←→ a = (μ i. i ∈ M ∧ is_ifrFb_body6(##M, G, r, s, z, i)) for z
    using If_abs_least_abs'[of λi. (##M)(i) ∧ is_ifrFb_body6(##M, G, r, s, z, i)
a]
    by simp
  ultimately
  have z∈M ⇒ least(##M, λi. i ∈ M ∧ is_ifrFb_body6(##M, G, r, s, z, i),
a)
    ←→ a = (μ i. ifrFb_body6(G, r, s, z, i)) for z
    using Least_cong[OF ifrFb_body6_closed[of G r s]] assms
    by simp
}
with assms
show ?thesis
  using pair_in_M_iff_apply_closed zero_in_M transitivity[of _ A]
  unfolding ifrangeF_body6_def is_ifrangeF_body6_def
  by (auto dest:transM)
qed

```

```

lemma (in M_ZF1_trans) separation_ifrangeF_body6:
  (##M)(A) ⇒ (##M)(G) ⇒ (##M)(b) ⇒ (##M)(f) ⇒
    separation(##M,
      λy. ∃ x∈A. y = ⟨x, μ i. x ∈ if_range_F_else_F(λa. {p ∈ G . domain(p) =
a}, b, f, i)⟩)
  using separation_is_ifrangeF_body6 ifrangeF_body6_abs
    separation_cong[where P=is_ifrangeF_body6(##M, A, G, b, f) and M=##M, THEN
iffD1]
  unfolding ifrangeF_body6_def if_range_F_def if_range_F_else_F_def ifrFb_body6_def
  by simp

```

definition *ifrFb_body7* **where**

```

ifrFb_body7(B, D, A, b, f, x, i) ≡ x ∈
  (if b = 0 then if i ∈ range(f) then
    {d ∈ D . ∃ r∈A. restrict(r, B) = converse(f) ‘ i ∧ d = domain(r)} else 0
  else {d ∈ D . ∃ r∈A. restrict(r, B) = i ∧ d = domain(r)})

```

relativize functional *ifrFb_body7* *ifrFb_body7_rel*

relationalize *ifrFb_body7_rel* *is_ifrFb_body7*

synthesize *is_ifrFb_body7* **from definition** **assuming** *nonempty*

arity_theorem **for** *is_ifrFb_body7_fm*

definition *ifrangeF_body7* :: [$i \Rightarrow o, i, i, i, i, i, i$] $\Rightarrow o$ **where**
ifrangeF_body7(M, A, B, D, G, b, f) $\equiv \lambda y. \exists x \in A. y = \langle x, \mu i. \text{ifrFb_body7}(B, D, G, b, f, x, i) \rangle$

relativize functional *ifrangeF_body7 ifrangeF_body7_rel*
relationalize *ifrangeF_body7_rel is_ifrangeF_body7*

synthesize *is_ifrangeF_body7* **from_definition** **assuming** *nonempty*
arity_theorem **for** *is_ifrangeF_body7_fm*

lemma (**in** *M_Z_trans*) *separation is_ifrangeF_body7*:
 $(\#\#M)(A) \Longrightarrow (\#\#M)(B) \Longrightarrow (\#\#M)(D) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(r)$
 $\Longrightarrow (\#\#M)(s) \Longrightarrow \text{separation}(\#\#M, \text{is_ifrangeF_body7}(\#\#M, A, B, D, G, r, s))$
using *separation_in_ctm*[**where** $\varphi = \text{is_ifrangeF_body7_fm}(1, 2, 3, 4, 5, 6, 0)$ **and**
 $\text{env} = [A, B, D, G, r, s]$
zero_in_M_arity_is_ifrangeF_body7_fm ord_simp_union is_ifrangeF_body7_fm_type
by *simp*

lemma (**in** *M_basic*) *ifrFb_body7_closed*: $M(B) \Longrightarrow M(D) \Longrightarrow M(G) \Longrightarrow M(r)$
 $\Longrightarrow M(s) \Longrightarrow$
 $\text{ifrFb_body7}(B, D, G, r, s, x, i) \longleftrightarrow M(i) \wedge \text{ifrFb_body7}(B, D, G, r, s, x, i)$
using *If_abs*
unfolding *ifrangeF_body7_def is_ifrangeF_body7_def ifrFb_body7_def fun_apply_def*
by (*cases i_in_range(s); cases r=0; auto dest:transM*)

lemma (**in** *M_basic*) *is_ifrFb_body7_closed*: $M(B) \Longrightarrow M(D) \Longrightarrow M(G) \Longrightarrow$
 $M(r) \Longrightarrow M(s) \Longrightarrow$
 $\text{is_ifrFb_body7}(M, B, D, G, r, s, x, i) \Longrightarrow M(i)$
using *If_abs*
unfolding *ifrangeF_body7_def is_ifrangeF_body7_def is_ifrFb_body7_def fun_apply_def*
by (*cases i_in_range(s); cases r=0; auto dest:transM*)

lemma (**in** *M_ZF1_trans*) *ifrangeF_body7_abs*:
assumes $(\#\#M)(A) (\#\#M)(B) (\#\#M)(D) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s)$
 $(\#\#M)(x)$
shows $\text{is_ifrangeF_body7}(\#\#M, A, B, D, G, r, s, x) \longleftrightarrow \text{ifrangeF_body7}(\#\#M, A, B, D, G, r, s, x)$
proof -
from *assms*
have *sep_dr*: $y \in M \Longrightarrow \text{separation}(\#\#M, \lambda d. \exists r \in M. r \in G \wedge y = \text{restrict}(r, B) \wedge d = \text{domain}(r))$ **for** y
by(*rule_tac separation_cong*[**where** $P' = \lambda d. \exists r \in M. r \in G \wedge y = \text{restrict}(r, B) \wedge d = \text{domain}(r)$, *THEN iffD1, OF* $\text{separation_restrict_eq_dom_eq}$ [*rule_format, of G B y*]], *auto simp:transitivity*[*of* $_ G$])
from *assms*
have *sep_dr''*: $y \in M \Longrightarrow \text{separation}(\#\#M, \lambda d. \exists r \in M. r \in G \wedge d = \text{domain}(r) \wedge \text{converse}(s) \text{ ' } y = \text{restrict}(r, B))$ **for** y
by(*rule_tac separation_cong*[*THEN iffD1, OF* $\text{separation_restrict_eq_dom_eq}$ [*rule_format, of G B converse(s) ' y*]], *auto simp:transitivity*[*of* $_ G$] *apply_closed*[*simplified*] *converse_closed*[*simplified*])

```

{
  fix a
  assume a ∈ M
  with assms
  have (μ i. i ∈ M ∧ is_ifrFb_body7(##M, B, D, G, r, s, z, i)) = (μ i. is_ifrFb_body7(##M, B, D,
G, r, s, z, i)) for z
    using is_ifrFb_body7_closed[of B D G r s z]
    by (rule_tac Least_cong[of λi. i ∈ M ∧ is_ifrFb_body7(##M, B, D, G, r, s, z, i)])
  auto
  moreover from this
    have (μ i. i ∈ M ∧ is_ifrFb_body7(##M, B, D, G, r, s, z, i)) = (μ i. i ∈ M ∧
ifrFb_body7(B, D, G, r, s, z, i)) if z ∈ M for z
      proof (rule_tac Least_cong[of λi. i ∈ M ∧ is_ifrFb_body7(##M, B, D, G, r, s, z, i)]
λi. i ∈ M ∧ ifrFb_body7(B, D, G, r, s, z, i))
        from assms ⟨a ∈ M⟩ ⟨z ∈ M⟩
        have is_ifrFb_body7(##M, B, D, G, r, s, z, y) ⟷ ifrFb_body7(B, D, G, r,
s, z, y) if y ∈ M for y
          using If_abs apply_0
          separation_closed converse_closed apply_closed range_closed zero_in_M
          transitivity[of _ D] transitivity[of _ G] that sep_dr sep_dr''
          unfolding ifrFb_body7_def is_ifrFb_body7_def
          by auto
        then
          show y ∈ M ∧ is_ifrFb_body7(##M, B, D, G, r, s, z, y) ⟷ y ∈ M ∧
ifrFb_body7(B, D, G, r, s, z, y) for y
            using conj_cong
            by simp
          qed
        moreover from ⟨a ∈ M⟩
          have least(##M, λi. i ∈ M ∧ is_ifrFb_body7(##M, B, D, G, r, s, z, i), a)
            ⟷ a = (μ i. i ∈ M ∧ is_ifrFb_body7(##M, B, D, G, r, s, z, i)) for z
          using If_abs least_abs'[of λi. (##M)(i) ∧ is_ifrFb_body7(##M, B, D, G, r, s, z, i)]
a]
            by simp
          ultimately
            have z ∈ M ⟹ least(##M, λi. i ∈ M ∧ is_ifrFb_body7(##M, B, D, G, r, s,
z, i), a)
              ⟷ a = (μ i. ifrFb_body7(B, D, G, r, s, z, i)) for z
            using Least_cong[OF ifrFb_body7_closed[of B D G r s]] assms
            by simp
        }
  with assms
  show ?thesis
    using pair_in_M_iff apply_closed zero_in_M transitivity[of _ A]
    unfolding ifrangeF_body7_def is_ifrangeF_body7_def
    by (auto dest: transM)
  qed

```

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( $\#\#M$ ,
     $\lambda y. \exists x \in A. y = \langle x, \mu i. x \in \text{if\_range\_F\_else\_F}(drSR\_Y(B, D, G), b, f, i) \rangle$ )
  using separation_is_ifrangeF_body7 ifrangeF_body7_abs drSR_Y_equality
  separation_cong[where  $P = \text{is\_ifrangeF\_body7}(\#\#M, A, B, D, G, b, f)$  and  $M = \#\#M$ , THEN
iffD1]
  unfolding ifrangeF_body7_def if_range_F_def if_range_F_else_F_def ifrFb_body7_def
  by simp

```

```

definition omfunospace :: [ $i, i$ ]  $\Rightarrow$   $o$  where
  omfunospace( $B$ )  $\equiv$   $\lambda z. \exists x. \exists n \in \omega. z \in x \wedge x = n \rightarrow B$ 
relativize functional omfunospace omfunospace_rel
relationalize omfunospace_rel is_omfunospace
synthesize is_omfunospace from definition assuming nonempty
arity_theorem for is_omfunospace_fm

```

```

context  $M\_pre\_seqspace$ 
begin

```

```

is_iff_rel for omfunospace
  using is_function_space_iff
  unfolding omfunospace_rel_def is_omfunospace_def
  by (simp add:absolut)

```

```

end —  $M\_pre\_seqspace$ 

```

```

context  $M\_ZF1\_trans$ 
begin

```

```

lemma separation_omfunospace:
  assumes ( $\#\#M$ )( $B$ )
  shows separation( $\#\#M, \lambda z. \exists x[\#\#M]. \exists n[\#\#M]. n \in \omega \wedge z \in x \wedge x = n \rightarrow^M B$ )
  using assms separation_in_ctm[where  $env = [B]$  and  $\varphi = \text{is\_omfunospace\_fm}(1, 0)$ 
and  $Q = \text{is\_omfunospace}(\#\#M, B)$ ]
  nonempty is_omfunospace_iff[of  $B$ , THEN separation_cong, of  $\#\#M$ ]
  arity_is_omfunospace_fm is_omfunospace_fm_type
  unfolding omfunospace_rel_def
  by (auto simp add:ord_simp_union)

```

```

end —  $M\_ZF1\_trans$ 

```

```

sublocale  $M\_ZF1\_trans \subseteq M\_seqspace \#\#M$ 
  using separation_omfunospace by unfold_locales

```

```

definition cdltgamma :: [ $i, i$ ]  $\Rightarrow$   $o$  where
  cdltgamma( $\gamma$ )  $\equiv$   $\lambda Z. |Z| < \gamma$ 
relativize functional cdltgamma cdltgamma_rel

```

```

relationalize cdltgamma_rel is_cdltgamma
synthesize is_cdltgamma from_definition assuming nonempty
arity_theorem for is_cdltgamma_fm

definition cdeggamma :: [i]  $\Rightarrow$  o where
  cdeggamma  $\equiv$   $\lambda Z . |fst(Z)| = snd(Z)$ 
relativize functional cdeggamma cdeggamma_rel
relationalize cdeggamma_rel is_cdeggamma
synthesize is_cdeggamma from_definition assuming nonempty
arity_theorem for is_cdeggamma_fm

context M_Perm
begin

is_iff_rel for cdltgamma
  using is_cardinal_iff
  unfolding cdltgamma_rel_def is_cdltgamma_def
  by (simp add:absolut)

is_iff_rel for cdeggamma
  using is_cardinal_iff fst_rel_abs snd_rel_abs
  unfolding cdeggamma_rel_def is_cdeggamma_def
  by (auto simp add:absolut)

lemma is_cdeggamma_iff_split:  $M(Z) \Longrightarrow cdeggamma\_rel(M, Z) \longleftrightarrow (\lambda \langle x, y \rangle .$ 
 $|x|^M = y)(Z)$ 
  using fst_rel_abs snd_rel_abs
  unfolding cdeggamma_rel_def split_def
  by simp

end

context M_ZF1_trans
begin

lemma separation_cdltgamma:
  assumes ( $\#\#M$ )( $\gamma$ )
  shows separation( $\#\#M, \lambda Z . cardinal\_rel(\#\#M, Z) < \gamma$ )
  using assms separation_in_ctm [where env=[ $\gamma$ ] and  $\varphi = is\_cdltgamma\_fm(1, 0)$ ]
  and  $Q = cdltgamma\_rel(\#\#M, \gamma)$ ]
  nonempty is_cdltgamma_iff [of  $\gamma$ ] arity_is_cdltgamma_fm is_cdltgamma_fm_type
  unfolding cdltgamma_rel_def
  by (auto simp add:ord_simp_union)

lemma separation_cdeggamma:
  shows separation( $\#\#M, \lambda Z . (\lambda \langle x, y \rangle . cardinal\_rel(\#\#M, x) = y)(Z)$ )
  using separation_in_ctm [where env=[] and  $\varphi = is\_cdeggamma\_fm(0)$ ]
  and  $Q = cdeggamma\_rel(\#\#M)$ ] is_cdeggamma_iff_split
  nonempty is_cdeggamma_iff arity_is_cdeggamma_fm is_cdeggamma_fm_type

```

```

    separation_cong[OF is_cdeggamma_iff_split, of ##M]
  unfolding cdeggamma_rel_def
  by (simp add:ord_simp_union)

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 ≡
  { 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*:

$B \in M \implies strong_replacement(\#\#M, HAleph_wfrec_repl_body(\#\#M, B))$

using *strong_replacement_rel_in_ctm* [where $\varphi = HAleph_wfrec_repl_body_fm(2, 0, 1)$
and $env = [B]$]

zero_in_M_arity_HAleph_wfrec_repl_body_fm replacement_ax2(3) ord_simp_union
by *simp*

lemma *HAleph_wfrec_repl*:

$(\#\#M)(sa) \implies$

$(\#\#M)(esa) \implies$

$(\#\#M)(mesa) \implies$

strong_replacement

$(\#\#M,$

$\lambda x z. \exists y[\#\#M].$

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

$(\exists f[\#\#M].$

$(\forall z[\#\#M].$

$z \in f \longleftrightarrow$

$(\exists xa[\#\#M].$

$\exists y[\#\#M].$

$\exists xaa[\#\#M].$

$\exists sx[\#\#M].$

$\exists r_sx[\#\#M].$

$\exists f_r_sx[\#\#M].$

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

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

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

$pre_image(\#\#M, mesa, sx, r_sx) \wedge$

$restriction(\#\#M, f, r_sx, f_r_sx) \wedge xaa \in mesa \wedge is_HAleph(\#\#M, xa, f_r_sx, y)) \wedge$
 $is_HAleph(\#\#M, x, f, y))$
using *replacement_HAleph_wfrec_repl_body unfolding HAleph_wfrec_repl_body_def*
by *simp*

lemma *replacement_is_order_eq_map*:
 $A \in M \implies r \in M \implies strong_replacement(\#\#M, order_eq_map(\#\#M, A, r))$
using *strong_replacement_rel_in_ctm[where $\varphi = order_eq_map_fm(2, 3, 0, 1)$*
and $env = [A, r]$ **and** $f = order_eq_map(\#\#M, A, r)$
 $order_eq_map_iff_sats[where\ env = [_, _, A, r]]\ zero_in_M\ fst_snd_closed$
 $pair_in_M_iff$
 $arity_order_eq_map_fm\ ord_simp_union\ replacement_ax2(4)$
by *simp*

end — *M_ZF2_trans*

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

lemma *type_omap_wfrec_body_fm* : $A \in nat \implies r \in nat \implies omap_wfrec_body(A, r) \in formula$
unfolding *omap_wfrec_body_def* **by** *simp*

lemma *arity_aux* : $A \in nat \implies r \in nat \implies arity(omap_wfrec_body(A, r)) = (9 +_\omega A) \cup (9 +_\omega r)$
unfolding *omap_wfrec_body_def*
using *arity_image_fm arity_pred_set_fm pred_Un_distrib union_abs2[of 3] union_abs1*
by (*simp add:FOL_aritys, auto simp add:Un_assoc[symmetric] union_abs1*)

lemma *arity_omap_wfrec* : $A \in nat \implies r \in nat \implies$
 $arity(is_wfrec_fm(omap_wfrec_body(A, r), succ(succ(succ(r))), 1, 0)) =$
 $(4 +_\omega A) \cup (4 +_\omega r)$
using *Arities.arity_is_wfrec_fm[OF _ _ _ _ _ arity_aux, of A r 3 +_\omega r 1 0]*
pred_Un_distrib
 $union_abs1\ union_abs2\ type_omap_wfrec_body_fm$
by *auto*

lemma *arity_isordermap* : $A \in nat \implies r \in nat \implies d \in nat \implies$
 $arity(is_ordermap_fm(A, r, d)) = succ(d) \cup (succ(A) \cup succ(r))$
unfolding *is_ordermap_fm_def*
using *arity_lambda_fm[where $i = (4 +_\omega A) \cup (4 +_\omega r)$, OF _ _ _ _ _ arity_omap_wfrec,*
 $unfolded\ omap_wfrec_body_def]$ *pred_Un_distrib union_abs1*
by *auto*

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

```

using arity_isordermap arity_image_fm pred_Un_distrib FOL_arities
by auto

lemma arity_is_order_body: arity(is_order_body_fm(1,0)) = 2
  using arity_is_order_body_fm arity_is_ordertype ord_simp_union
  by (simp add:FOL_arities)

lemma (in M_ZF2_trans) replacement_is_order_body:
  strong_replacement(##M,  $\lambda x z . \exists y[\#M]. \text{is\_order\_body}(\#M,x,y) \wedge z = \langle x,y \rangle$ )
  apply(rule_tac strong_replacement_cong[
    where  $P = \lambda x f. M, [x,f] \models (\exists y. \text{is\_order\_body\_fm}(1,0) \wedge \text{pair\_fm}(1,0,2))$ 
    ..), THEN iffD1)
  apply(simp add: is_order_body_iff_sats[where env=[_,_], symmetric])
  apply(simp_all add:zero_in_M)
  apply(rule_tac replacement_ax2(1)[unfolded replacement_assm_def, rule_format,
where env=[], simplified])
  apply(simp_all add:arity_is_order_body arity pred_Un_distrib ord_simp_union)
  done

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

relationalize H_order_pred is_H_order_pred

lemma (in M_basic) H_order_pred_abs :
   $M(A) \implies M(r) \implies M(x) \implies M(f) \implies M(z) \implies$ 
   $\text{is\_H\_order\_pred}(M,A,r,x,f,z) \longleftrightarrow z = \text{H\_order\_pred}(A,r,x,f)$ 
  unfolding is_H_order_pred_def H_order_pred_def
  by simp

synthesize is_H_order_pred from _definition assuming nonempty

lemma (in M_ZF2_trans) wfrec_replacement_order_pred:
   $A \in M \implies r \in M \implies \text{wfrec\_replacement}(\#M, \lambda x g z. \text{is\_H\_order\_pred}(\#M,A,r,x,g,z), r)$ 
  unfolding wfrec_replacement_def is_wfrec_def M_is_recfun_def is_H_order_pred_def
  apply(rule_tac strong_replacement_cong[
    where  $P = \lambda x f. M, [x,f,r,A] \models \text{order\_pred\_wfrec\_body\_fm}(3,2,1,0)$ , THEN
    iffD1)
  apply(subst order_pred_wfrec_body_def[symmetric])
  apply(rule_tac order_pred_wfrec_body_iff_sats[where env=[_,_,r,A], symmetric])
  apply(simp_all add:zero_in_M)
  apply(rule_tac replacement_ax2(2)[unfolded replacement_assm_def, rule_format,
where env=[r,A], simplified])
  apply(simp_all add: arity_order_pred_wfrec_body_fm ord_simp_union)
  done

```

lemma (in M_ZF2_trans) $wfrec_replacement_order_pred'$:
 $A \in M \implies r \in M \implies wfrec_replacement(\#\#M, \lambda x g z. z = H_order_pred(A, r, x, g)$
 $, r)$
using $wfrec_replacement_cong$ [$OF H_order_pred_abs$ [of $A r, rule_format$] $refl$, $THEN$
 $iffD1$,
 $OF _ _ _ _ wfrec_replacement_order_pred$ [of $A r$]]
by $simp$

sublocale $M_ZF2_trans \subseteq M_pre_cardinal_arith \#\#M$
using $separation_instances wfrec_replacement_order_pred'$ [$unfolded H_order_pred_def$]
 $replacement_is_order_eq_map$ [$unfolded order_eq_map_def$]
by $unfold_locales simp_all$

definition $is_well_ord_fst_snd$ **where**
 $is_well_ord_fst_snd(A, x) \equiv (\exists a[A]. \exists b[A]. is_well_ord(A, a, b) \wedge is_snd(A, x,$
 $b) \wedge is_fst(A, x, a))$

synthesize $is_well_ord_fst_snd$ **from** **definition** **assuming** $nonempty$
arity_theorem **for** $is_well_ord_fst_snd_fm$

lemma (in M_ZF2_trans) $separation_well_ord$: $separation(\#\#M, \lambda x. is_well_ord(\#\#M, fst(x),$
 $snd(x)))$
using $arity_is_well_ord_fst_snd_fm is_well_ord_iff_sats$ [$symmetric$] $nonempty$
 $fst_closed snd_closed fst_abs snd_abs$
 $separation_in_ctm$ [**where** $env=[]$ **and** $\varphi=is_well_ord_fst_snd_fm(0)$]
by ($simp_all$ $add: is_well_ord_fst_snd_def$)

sublocale $M_ZF2_trans \subseteq M_pre_aleph \#\#M$
using $HAleph_wfrec_repl replacement_is_order_body$
 $separation_well_ord separation_Pow_rel$
by $unfold_locales (simp_all$ $add: transrec_replacement_def$
 $wfrec_replacement_def is_wfrec_def M_is_recfun_def flip:setclass_iff)$

arity_theorem **intermediate** **for** is_HAleph_fm
lemma $arity_is_HAleph_fm$: $arity(is_HAleph_fm(2, 1, 0)) = 3$
using $arity_fun_apply_fm$ [of $11\ 0\ 1, simplified$]
 $arity_is_HAleph_fm'$ $arity_ordinal_fm$ $arity_is_If_fm$
 $arity_empty_fm$ $arity_is_Limit_fm$
 $arity_is_If_fm$
 $arity_is_Limit_fm$ $arity_empty_fm$
 $arity_Replace_fm$ [**where** $i=12$ **and** $v=10$ **and** $n=3$]
 $pred_Un_distrib$ ord_simp_union
by ($simp$ $add:FOL_arities$)

lemma $arity_is_Aleph$ [$arity$]: $arity(is_Aleph_fm(0, 1)) = 2$
unfolding $is_Aleph_fm_def$
using $arity_transrec_fm$ [$OF _ _ _ _ arity_is_HAleph_fm$] ord_simp_union

by *simp*

definition *bex_Aleph_rel* :: $[i \Rightarrow o, i, i] \Rightarrow o$ **where**
 $bex_Aleph_rel(M, x) \equiv \lambda y. \exists z \in x. y = \aleph_z^M$

relationalize *bex_Aleph_rel is_bex_Aleph*

schematic_goal *sats_is_bex_Aleph_fm_auto*:

$a \in nat \Longrightarrow c \in nat \Longrightarrow env \in list(A) \Longrightarrow$

$a < length(env) \Longrightarrow c < length(env) \Longrightarrow 0 \in A \Longrightarrow$

$is_bex_Aleph(\#\#A, nth(a, env), nth(c, env)) \longleftrightarrow A, env \models ?fm(a, c)$

unfolding *is_bex_Aleph_def*

by (*rule iff_sats | simp*)⁺

synthesize_notc *is_bex_Aleph from_schematic*

lemma *is_bex_Aleph_fm_type* [TC]:

$x \in \omega \Longrightarrow z \in \omega \Longrightarrow is_bex_Aleph_fm(x, z) \in formula$

unfolding *is_bex_Aleph_fm_def* **by** *simp*

lemma *sats_is_bex_Aleph_fm*:

$x \in \omega \Longrightarrow$

$z \in \omega \Longrightarrow x < length(env) \Longrightarrow z < length(env) \Longrightarrow$

$env \in list(Aa) \Longrightarrow$

$0 \in Aa \Longrightarrow$

$(Aa, env \models is_bex_Aleph_fm(x, z)) \longleftrightarrow$

$is_bex_Aleph(\#\#Aa, nth(x, env), nth(z, env))$

using *sats_is_bex_Aleph_fm_auto* **unfolding** *is_bex_Aleph_def is_bex_Aleph_fm_def*

by *simp*

lemma *is_bex_Aleph_iff_sats* [*iff_sats*]:

$nth(x, env) = xa \Longrightarrow$

$nth(z, env) = za \Longrightarrow$

$x \in \omega \Longrightarrow$

$z \in \omega \Longrightarrow x < length(env) \Longrightarrow z < length(env) \Longrightarrow$

$env \in list(Aa) \Longrightarrow$

$0 \in Aa \Longrightarrow$

$is_bex_Aleph(\#\#Aa, xa, za) \longleftrightarrow$

$Aa, env \models is_bex_Aleph_fm(x, z)$

using *sats_is_bex_Aleph_fm* **by** *simp*

arity_theorem for *is_bex_Aleph_fm*

lemma (in *M_ZF1_trans*) *separation_is_bex_Aleph*:

assumes $(\#\#M)(A)$

shows $separation(\#\#M, is_bex_Aleph(\#\#M, A))$

using *assms separation_in_ctm* [**where** $env=[A]$ **and** $\varphi=is_bex_Aleph_fm(1,0)$,

OF $is_bex_Aleph_iff_sats$ [*symmetric*],

of $\lambda_.A]$

nonempty arity_is_bex_Aleph_fm is_bex_Aleph_fm_type
by (*simp add:ord_simp_union*)

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)*
unfolding *is_bex_Aleph_def bex_Aleph_rel_def*
using *assms is_Aleph_iff transM[of _ u] Ord_in_Ord*
by *simp*

lemma (**in** *M_ZF2_trans*) *separation_bex_Aleph_rel*:
Ord(x) \implies (##M)(x) \implies separation(##M, bex_Aleph_rel(##M,x))
using *separation_is_bex_Aleph bex_Aleph_rel_abs*
separation_cong[where P=is_bex_Aleph(##M,x) and M=##M, THEN iffD1]
unfolding *bex_Aleph_rel_def*
by *simp*

sublocale *M_ZF2_trans \subseteq M_aleph ##M*
using *separation_bex_Aleph_rel[unfolded bex_Aleph_rel_def]*
by *unfold_locales*

sublocale *M_ZF1_trans \subseteq M_FiniteFun ##M*
using *separation_is_function separation_omfunspace*
by *unfold_locales simp*

sublocale *M_ZFC2_trans \subseteq M_cardinal_AC ##M*
using *lam_replacement_minimum*
by *unfold_locales simp*

lemma (**in** *M_ZF1_trans*) *separation_cardinal_rel_lesspoll_rel*:
(##M)(κ) \implies separation(##M, $\lambda x. x \prec^M \kappa$)
using *separation_in_ctm[where $\varphi=(\cdot 0 \prec 1 \cdot)$ and env= $[\kappa]$]*
is_lesspoll_iff nonempty
arity_is_cardinal_fm arity_is_lesspoll_fm arity_is_bij_fm ord_simp_union
by (*simp add:FOL_aritys*)

sublocale *M_ZFC2_trans \subseteq M_library ##M*
using *separation_cardinal_rel_lesspoll_rel lam_replacement_minimum*
by *unfold_locales simp_all*

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 $instances2_fms_type[TC] : instances2_fms \subseteq formula$
unfolding $instances2_fms_def replacement_instances2_defs$
by ($auto simp del: Lambda_in_M_fm_def$)

lemma $overhead_type: overhead \subseteq formula$
using $instances1_fms_type instances_ground_fms_type$
unfolding $overhead_def replacement_instances1_defs$
by $simp$

lemma $overhead_notCH_type: overhead_notCH \subseteq formula$
using $overhead_type$
unfolding $overhead_notCH_def rec_constr_abs_fm_def$
 $rec_constr_fm_def instances_ground_notCH_fms_def$
 $instances2_fms_def instances3_fms_def$
by ($auto simp: replacement_instances1_defs$
 $replacement_instances2_defs simp del: Lambda_in_M_fm_def$)

lemma $overhead_CH_type: overhead_CH \subseteq formula$

```

using overhead_notCH_type unfolding overhead_CH_def dc_abs_fm_def
by auto

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 ⊨ ZF) ⇒ M_ZF1(M)
proof -
  assume M ⊨ ZF
  then
  have fin: upair_ax(##M) Union_ax(##M) power_ax(##M)
    extensionality(##M) foundation_ax(##M) infinity_ax(##M)
  unfolding ZF_def ZF_fm_def ZFC_fm_defs satT_def
  using ZFC_fm_sats[of M] by simp_all
  {
    fix φ env
    assume φ ∈ formula env ∈ list(M)
    moreover from ⟨M ⊨ ZF⟩
    have ∀ p ∈ formula. (M, [] ⊨ (ZF_separation_fm(p)))
      ∀ p ∈ formula. (M, [] ⊨ (ZF_replacement_fm(p)))
    unfolding ZF_def ZF_schemes_def by auto
    moreover from calculation
    have arity(φ) ≤ succ(length(env)) ⇒ separation(##M, λx. (M, Cons(x, env)
    ⊨ φ))
      arity(φ) ≤ succ(succ(length(env))) ⇒ strong_replacement(##M, λx y.
    sats(M, φ, Cons(x, Cons(y, env))))
    using sats_ZF_separation_fm_iff sats_ZF_replacement_fm_iff
    unfolding replacement_assm_def by simp_all
  }
  with fin
  show M_ZF1(M)
  by unfold_locales (simp_all add: replacement_assm_def ground_replacement_assm_def)
qed

lemma M_satT_imp_M_ZFC1:
  shows (M ⊨ ZFC) → M_ZFC1(M)
proof -
  have (M ⊨ ZF) ∧ choice_ax(##M) → M_ZFC1(M)
  using M_satT_imp_M_ZF2[of M]
  unfolding M_ZFC1_def M_ZC_basic_def M_ZF1_def M_AC_def
  by auto
  then

```

```

show ?thesis
  unfolding ZFC_def by auto
qed

lemma M_satT_instances1_imp_M_ZF1:
  assumes (M ⊨ ·Z · ∪ {·Replacement(p) · . p ∈ instances1_fms })
  shows M_ZF1(M)
proof -
  from assms
  have fin: upair_ax(##M) Union_ax(##M) power_ax(##M)
    extensionality(##M) foundation_ax(##M) infinity_ax(##M)
  unfolding ZF_fin_def Zermelo_fms_def ZFC_fm_defs satT_def
  using ZFC_fm_sats[of M] by simp_all
  moreover
  {
    fix φ env
    from ⟨M ⊨ ·Z · ∪ {·Replacement(p) · . p ∈ instances1_fms }⟩
    have ∀ p ∈ formula. (M, [] ⊨ (ZF_separation_fm(p)))
    unfolding Zermelo_fms_def ZF_def instances1_fms_def
    by auto
    moreover
    assume φ ∈ formula env ∈ list(M)
    ultimately
    have arity(φ) ≤ succ(length(env)) ⇒ separation(##M, λx. (M, Cons(x, env)
    ⊨ φ))
    using sats_ZF_separation_fm_iff by simp_all
  }
  moreover
  {
    fix φ env
    assume φ ∈ instances1_fms env ∈ list(M)
    moreover from this and ⟨M ⊨ ·Z · ∪ {·Replacement(p) · . p ∈ instances1_fms
  }⟩
    have M, [] ⊨ ·Replacement(φ) · by auto
    ultimately
    have arity(φ) ≤ succ(succ(length(env))) ⇒ strong_replacement(##M, λx y.
    sats(M, φ, Cons(x, Cons(y, env))))
    using sats_ZF_replacement_fm_iff[of φ] instances1_fms_type
    unfolding replacement_assm_def by auto
  }
  ultimately
  show ?thesis
  unfolding instances1_fms_def
  by unfold_locales (simp_all add: replacement_assm_def ground_replacement_assm_def)
qed

theorem M_satT_imp_M_ZF_ground_trans:
  assumes Transset(M) M ⊨ ·Z · ∪ {·Replacement(p) · . p ∈ overhead}
  shows M_ZF_ground_trans(M)

```

```

proof -
  from  $\langle M \models \cdot Z \cdot \cup \_ \rangle$ 
  have  $M \models \cdot Z \cdot \cup \{ \cdot \text{Replacement}(p) \cdot \mid p \in \text{instances1\_fms} \}$ 
     $M \models \{ \cdot \text{Replacement}(p) \cdot \mid p \in \text{instances\_ground\_fms} \}$ 
  unfolding overhead_def by auto
  then
  interpret M_ZF1 M
    using M_satT_instances1_imp_M_ZF1
    by simp
  from  $\langle \text{Transset}(M) \rangle$ 
  interpret M_ZF1_trans M
    using M_satT_imp_M_ZF2
    by unfold_locales
  {
    fix  $\varphi \text{ env}$ 
    assume  $\varphi \in \text{instances\_ground\_fms env} \in \text{list}(M)$ 
    moreover from this and  $\langle M \models \{ \cdot \text{Replacement}(p) \cdot \mid p \in \text{instances\_ground\_fms} \} \rangle$ 
    have  $M, [] \models \cdot \text{Replacement}(\varphi) \cdot$  by auto
    ultimately
    have  $\text{arity}(\varphi) \leq \text{succ}(\text{succ}(\text{length}(\text{env}))) \implies \text{strong\_replacement}(\#\#M, \lambda x y.$ 
sats( $M, \varphi, \text{Cons}(x, \text{Cons}(y, \text{env}))$ ))
    using sats_ZF_replacement_fm_iff[of  $\varphi$ ] instances\_ground\_fms\_type
    unfolding replacement_assm_def by auto
  }
  then
  show ?thesis
    unfolding instances\_ground\_fms_def
    by unfold_locales (simp_all add:replacement_assm_def)
qed

theorem M_satT_imp_M_ZF_ground_notCH_trans:
  assumes
    Transset(M)
     $M \models \cdot Z \cdot \cup \{ \cdot \text{Replacement}(p) \cdot \mid p \in \text{overhead\_notCH} \}$ 
  shows M_ZF_ground_notCH_trans(M)
proof -
  from assms
  interpret M_ZF_ground_trans M
    using M_satT_imp_M_ZF_ground_trans unfolding overhead_notCH_def
by force
  {
    fix  $\varphi \text{ env}$ 
    assume  $\varphi \in \text{instances\_ground\_notCH\_fms env} \in \text{list}(M)$ 
    moreover from this and assms
    have  $M, [] \models \cdot \text{Replacement}(\varphi) \cdot$ 
    unfolding overhead_notCH_def by auto
    ultimately
    have  $\text{arity}(\varphi) \leq \text{succ}(\text{succ}(\text{length}(\text{env}))) \implies \text{strong\_replacement}(\#\#M, \lambda x y.$ 
sats( $M, \varphi, \text{Cons}(x, \text{Cons}(y, \text{env}))$ ))
  }

```

```

    using sats_ZF_replacement_fm_iff[of  $\varphi$ ] instances_ground_notCH_fms_type
    unfolding replacement_assm_def by auto
  }
  then
  show ?thesis
  by unfold_locales (simp_all add:replacement_assm_def instances_ground_notCH_fms_def)
qed

```

theorem $M_satT_imp_M_ZF_ground_CH_trans$:

```

  assumes
    Transset(M)
     $M \models \cdot Z \cup \{ \cdot Replacement(p) \cdot \mid p \in overhead\_CH \}$ 
  shows  $M\_ZF\_ground\_CH\_trans(M)$ 
  proof -
    from assms
    interpret  $M\_ZF\_ground\_notCH\_trans\ M$ 
    using  $M\_satT\_imp\_M\_ZF\_ground\_notCH\_trans$  unfolding overhead_CH_def
  by auto
  {
    fix env
    assume  $env \in list(M)$ 
    moreover from assms
    have  $M, [] \models \cdot Replacement(dc\_abs\_fm) \cdot$ 
      unfolding overhead_CH_def by auto
    ultimately
    have  $arity(dc\_abs\_fm) \leq succ(succ(length(env)))$ 
       $\implies strong\_replacement(\#\#M, \lambda x y. sats(M, dc\_abs\_fm, Cons(x, Cons(y, env))))$ 
    using sats_ZF_replacement_fm_iff[of  $dc\_abs\_fm$ ]
    unfolding replacement_assm_def
    by (auto simp:dc_abs_fm_def)
  }
  then
  show ?thesis
  by unfold_locales (simp_all add:replacement_assm_def)
qed

```

lemma (in M_Z_basic) $M_satT_Zermelo_fms$: $M \models \cdot Z \cdot$
 using $upair_ax$ $Union_ax$ $power_ax$ $extensionality$ $foundation_ax$
 $infinity_ax$ $separation_ax$ $sats_ZF_separation_fm_iff$
 unfolding $Zermelo_fms_def$ ZF_fin_def
 by auto

lemma (in M_ZFC1) M_satT_ZC : $M \models ZC$
 using $upair_ax$ $Union_ax$ $power_ax$ $extensionality$ $foundation_ax$
 $infinity_ax$ $separation_ax$ $sats_ZF_separation_fm_iff$ $choice_ax$
 unfolding ZC_def $Zermelo_fms_def$ ZF_fin_def
 by auto

```

locale  $M\_ZF = M\_Z\_basic +$ 
  assumes
     $replacement\_ax: replacement\_assm(M, env, \varphi)$ 

sublocale  $M\_ZF \subseteq M\_ZF3$ 
  using  $replacement\_ax$ 
  by  $unfold\_locales (simp\_all add: ground\_replacement\_assm\_def)$ 

lemma  $M\_satT\_imp\_M\_ZF: M \models ZF \implies M\_ZF(M)$ 
proof -
  assume  $M \models ZF$ 
  then
  have  $fin: upair\_ax(\#\#M) \ Union\_ax(\#\#M) \ power\_ax(\#\#M) \$ 
     $extensionality(\#\#M) \ foundation\_ax(\#\#M) \ infinity\_ax(\#\#M)$ 
  unfolding  $ZF\_def \ ZF\_fin\_def \ ZFC\_fm\_defs \ satT\_def$ 
  using  $ZFC\_fm\_sats[of \ M]$  by  $simp\_all$ 
  {
    fix  $\varphi \ env$ 
    assume  $\varphi \in formula \ env \in list(M)$ 
    moreover from  $\langle M \models ZF \rangle$ 
    have  $\forall p \in formula. (M, [] \models (ZF\_separation\_fm(p)))$ 
       $\forall p \in formula. (M, [] \models (ZF\_replacement\_fm(p)))$ 
    unfolding  $ZF\_def \ ZF\_schemes\_def$  by  $auto$ 
    moreover from  $calculation$ 
    have  $arity(\varphi) \leq succ(length(env)) \implies separation(\#\#M, \lambda x. (M, Cons(x, env)$ 
       $\models \varphi))$ 
       $arity(\varphi) \leq succ(succ(length(env))) \implies strong\_replacement(\#\#M, \lambda x \ y.$ 
       $sats(M, \varphi, Cons(x, Cons(y, env))))$ 
    using  $sats\_ZF\_separation\_fm\_iff \ sats\_ZF\_replacement\_fm\_iff$ 
    unfolding  $replacement\_assm\_def$  by  $simp\_all$ 
  }
  with  $fin$ 
  show  $M\_ZF(M)$ 
  unfolding  $M\_ZF\_def \ M\_Z\_basic\_def \ M\_ZF\_axioms\_def \ replacement\_assm\_def$ 
by  $simp$ 
qed

lemma (in  $M\_ZF$ )  $M\_satT\_ZF: M \models ZF$ 
  using  $upair\_ax \ Union\_ax \ power\_ax \ extensionality \ foundation\_ax$ 
   $infinity\_ax \ separation\_ax \ sats\_ZF\_separation\_fm\_iff$ 
   $replacement\_ax \ sats\_ZF\_replacement\_fm\_iff$ 
  unfolding  $ZF\_def \ ZF\_schemes\_def \ ZF\_fin\_def \ replacement\_assm\_def$ 
by  $auto$ 

lemma  $M\_ZF\_iff\_M\_satT: M\_ZF(M) \longleftrightarrow (M \models ZF)$ 
  using  $M\_ZF.M\_satT\_ZF \ M\_satT\_imp\_M\_ZF$ 
by  $auto$ 

locale  $M\_ZFC = M\_ZF + M\_ZC\_basic$ 

```

```

sublocale  $M\_ZFC \subseteq M\_ZFC3$ 
  by unfold_locales

lemma  $M\_ZFC\_iff\_M\_satT$ :
  notes iff_trans[trans]
  shows  $M\_ZFC(M) \longleftrightarrow (M \models ZFC)$ 
proof -
  have  $M\_ZFC(M) \longleftrightarrow (M \models ZF) \wedge choice\_ax(\#\#M)$ 
    using  $M\_ZF\_iff\_M\_satT$ 
    unfolding  $M\_ZFC\_def\ M\_ZC\_basic\_def\ M\_AC\_def\ M\_ZF\_def$  by auto
  also
  have  $\dots \longleftrightarrow M \models ZFC$ 
    unfolding  $ZFC\_def$  by auto
  ultimately
  show ?thesis by simp
qed

lemma  $M\_satT\_imp\_M\_ZF3$ :  $(M \models ZF) \longrightarrow M\_ZF3(M)$ 
proof
  assume  $M \models ZF$ 
  then
  interpret  $M\_ZF\ M$ 
    using  $M\_satT\_imp\_M\_ZF$  by simp
  show  $M\_ZF3(M)$ 
    by unfold_locales
qed

lemma  $M\_satT\_imp\_M\_ZFC3$ :
  shows  $(M \models ZFC) \longrightarrow M\_ZFC3(M)$ 
proof
  assume  $M \models ZFC$ 
  then
  interpret  $M\_ZFC\ M$ 
    using  $M\_ZFC\_iff\_M\_satT$  by simp
  show  $M\_ZFC3(M)$ 
    by unfold_locales
qed

lemma  $M\_satT\_overhead\_imp\_M\_ZF3$ :
   $(M \models ZC \cup \{\cdot Replacement(p) \cdot \mid p \in overhead\_notCH\}) \longrightarrow M\_ZF3(M)$ 
proof
  assume  $M \models ZC \cup \{\cdot Replacement(p) \cdot \mid p \in overhead\_notCH\}$ 
  then
  have fin:  $upair\_ax(\#\#M)\ Union\_ax(\#\#M)\ power\_ax(\#\#M)\ choice\_ax(\#\#M)$ 
     $extensionality(\#\#M)\ foundation\_ax(\#\#M)\ infinity\_ax(\#\#M)$ 
    unfolding  $ZC\_def\ ZF\_fin\_def\ Zermelo\_fms\_def\ ZFC\_fm\_defs\ satT\_def$ 
    using  $ZFC\_fm\_sats[of\ M]$  by simp_all
  moreover

```

```

{
  fix  $\varphi$  env
  from  $\langle M \models ZC \cup \{\cdot\text{Replacement}(p) \cdot p \in \text{overhead\_notCH}\} \rangle$ 
  have  $\forall p \in \text{formula}. (M, [] \models (ZF\_separation\_fm(p)))$ 
    unfolding ZC_def Zermelo_fms_def ZF_def by auto
  moreover
  assume  $\varphi \in \text{formula env} \in \text{list}(M)$ 
  ultimately
  have  $\text{arity}(\varphi) \leq \text{succ}(\text{length}(\text{env})) \implies \text{separation}(\#\#M, \lambda x. (M, \text{Cons}(x, \text{env})$ 
 $\models \varphi))$ 
    using sats_ZF_separation_fm_iff by simp_all
}
moreover
{
  fix  $\varphi$  env
  assume  $\varphi \in \text{overhead\_notCH env} \in \text{list}(M)$ 
  moreover from this and  $\langle M \models ZC \cup \{\cdot\text{Replacement}(p) \cdot p \in \text{overhead\_notCH}\} \rangle$ 
  have  $M, [] \models \cdot\text{Replacement}(\varphi) \cdot$  by auto
  ultimately
  have  $\text{arity}(\varphi) \leq \text{succ}(\text{succ}(\text{length}(\text{env}))) \implies \text{strong\_replacement}(\#\#M, \lambda x y.$ 
 $\text{sats}(M, \varphi, \text{Cons}(x, \text{Cons}(y, \text{env}))))$ 
    using sats_ZF_replacement_fm_iff[of  $\varphi$ ] overhead_notCH_type
    unfolding replacement_assm_def by auto
}
ultimately
show  $M\_ZFC3(M)$ 
  unfolding overhead_def overhead_notCH_def instances1_fms_def
  instances2_fms_def instances3_fms_def
  by unfold_locales (simp_all add: replacement_assm_def ground_replacement_assm_def)
qed

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 ( $\langle \aleph \_ \rangle$  [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 :  $\mathbb{P} \subseteq M$ 
  using transitivity P_in_M by auto

```

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)  $\implies$  filter(G)
  unfolding M_generic_def by blast

```

lemma generic_filter_existence:

```

 $p \in \mathbb{P} \implies \exists G. p \in G \wedge M\_generic(G)$ 

```

proof -

```

assume p  $\in \mathbb{P}$ 
let ?D =  $\lambda n \in nat. (if (enum\ 'n \subseteq \mathbb{P} \wedge$  dense(enum\ 'n)) then enum\ 'n else  $\mathbb{P})$ 
have  $\forall n \in nat. ?D\ 'n \in Pow(\mathbb{P})$ 
  by auto
then
have ?D :  $nat \rightarrow Pow(\mathbb{P})$ 
  using lam_type by auto
have  $\forall n \in nat. dense(?D\ 'n)$ 
proof(intro ballI)
  fix n
  assume n  $\in nat$ 
  then
  have dense(?D\ 'n)  $\longleftrightarrow$  dense(if enum\ 'n  $\subseteq \mathbb{P} \wedge$  dense(enum\ 'n) then enum\ 'n
else  $\mathbb{P})$ 
  by simp
  also
  have ...  $\longleftrightarrow$  ( $\neg$ (enum\ 'n  $\subseteq \mathbb{P} \wedge$  dense(enum\ 'n))  $\longrightarrow$  dense( $\mathbb{P}$ ))

```

```

    using split_if by simp
  finally
  show dense(?D'n)
    using P_dense ⟨n∈nat⟩ by auto
qed
with ⟨?D∈_⟩
interpret cg: countable_generic ℙ leq 1 ?D
  by (unfold_locales, auto)
from ⟨p∈ℙ⟩
obtain G where 1: p∈G ∧ filter(G) ∧ (∀ n∈nat.(?D'n)∩G≠0)
  using cg.countable_rasiowa_sikorski[where M=λ_. M] P_sub_M
  M_countable[THEN bij_is_fun] M_countable[THEN bij_is_surj, THEN
surj_range]
  unfolding cg.D_generic_def by blast
then
have (∀ D∈M. D⊆ℙ ∧ dense(D)⟶D∩G≠0)
proof (intro ballI impI)
  fix D
  assume D∈M and 2: D ⊆ ℙ ∧ dense(D)
  moreover
  have ∀ y∈M. ∃ x∈nat. enum'x = y
    using M_countable and bij_is_surj unfolding surj_def by (simp)
  moreover from calculation
  obtain n where Eq10: n∈nat ∧ enum'n = D
    by auto
  moreover from calculation if_P
  have ?D'n = D
    by simp
  moreover
  note 1
  ultimately
  show D∩G≠0
    by auto
qed
with 1
show ?thesis
  unfolding M_generic_def by auto
qed

lemma one_in_M: 1 ∈ M
  using one_in_P P_in_M transitivity
  by simp

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  $G\_nonempty: G \neq 0$ 
  using  $generic\ subset\_refl[of\ \mathbb{P}]\ P\_dense$ 
  unfolding  $M\_generic\_def$ 
  by  $auto$ 

lemma  $M\_genericD\ [dest]: x \in G \implies x \in \mathbb{P}$ 
  using  $generic$ 
  by  $(blast\ dest:filterD)$ 

lemma  $M\_generic\_leqD\ [dest]: p \in G \implies q \in \mathbb{P} \implies p \preceq q \implies q \in G$ 
  using  $generic$ 
  by  $(blast\ dest:filter\_leqD)$ 

lemma  $M\_generic\_compatD\ [dest]: p \in G \implies r \in G \implies \exists q \in G. q \preceq p \wedge q \preceq r$ 
  using  $generic$ 
  by  $(blast\ dest:low\_bound\_filter)$ 

lemma  $M\_generic\_denseD\ [dest]: dense(D) \implies D \subseteq \mathbb{P} \implies D \in M \implies \exists q \in G. q \in D$ 
  using  $generic$ 
  unfolding  $M\_generic\_def$  by  $blast$ 

lemma  $G\_subset\_P: G \subseteq \mathbb{P}$ 
  using  $generic$  by  $auto$ 

lemma  $one\_in\_G : 1 \in G$ 
proof -
  have  $increasing(G)$ 
  using  $generic$ 
  unfolding  $M\_generic\_def\ filter\_def$  by  $simp$ 
  then
  show  $?thesis$ 
  using  $G\_nonempty\ one\_max$ 
  unfolding  $increasing\_def$  by  $blast$ 
qed

lemma  $G\_subset\_M: G \subseteq M$ 
  using  $generic\ transitivity[OF\ _\ P\_in\_M]$  by  $auto$ 

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
   $x = \langle f, n1, n2, p \rangle$   $f \in 2$   $n1 \in \text{eclose}N(z)$   $n2 \in \text{eclose}N(z)$   $p \in P$ 
using assms unfolding names\_below\_def by auto
```

```
context forcing_data1
begin
```

```
lemma ftype_abs:
 $\llbracket x \in M; y \in M \rrbracket \implies \text{is\_ftype}(\#\#M, x, y) \longleftrightarrow y = \text{ftype}(x)$ 
unfolding ftype\_def is\_ftype\_def by (simp add: absolut)
```

```
lemma name1_abs:
 $\llbracket x \in M; y \in M \rrbracket \implies \text{is\_name1}(\#\#M, x, y) \longleftrightarrow y = \text{name1}(x)$ 
unfolding name1\_def is\_name1\_def
by (rule is_hcomp_abs[OF fst_abs], simp_all add: fst_snd_closed[simplified] absolut)
```

```
lemma snd_snd_abs:
 $\llbracket x \in M; y \in M \rrbracket \implies \text{is\_snd\_snd}(\#\#M, x, y) \longleftrightarrow y = \text{snd}(\text{snd}(x))$ 
unfolding is_snd_snd_def
by (rule is_hcomp_abs[OF snd_abs],
  simp_all add: conjunct2[OF fst_snd_closed, simplified] absolut)
```

```
lemma name2_abs:
 $\llbracket x \in M; y \in M \rrbracket \implies \text{is\_name2}(\#\#M, x, y) \longleftrightarrow y = \text{name2}(x)$ 
unfolding name2\_def is\_name2\_def
by (rule is_hcomp_abs[OF fst_abs snd_snd_abs], simp_all add: absolut conjunct2[OF fst_snd_closed, simplified])
```

```
lemma cond_of_abs:
 $\llbracket x \in M; y \in M \rrbracket \implies \text{is\_cond\_of}(\#\#M, x, y) \longleftrightarrow y = \text{cond\_of}(x)$ 
unfolding cond\_of\_def is\_cond\_of\_def
by (rule is_hcomp_abs[OF snd_abs snd_snd_abs]; simp_all add: fst_snd_closed[simplified])
```

```
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 = ⟨z,t1,t2,p⟩
unfolding is_tuple_def using pair_in_M_iff by simp

lemmas components_abs = ftype_abs name1_abs name2_abs cond_of_abs
tuple_abs

lemma comp_in_M:
p  $\preceq$  q  $\implies$  p ∈ M
p  $\preceq$  q  $\implies$  q ∈ M
using transitivity[of _ leq] pair_in_M_iff by auto

lemma eq_case_abs [simp]:
assumes t1 ∈ M t2 ∈ M p ∈ M f ∈ M
shows is_eq_case(##M,t1,t2,p,ℙ,leq,f)  $\longleftrightarrow$  eq_case(t1,t2,p,ℙ,leq,f)
proof -
have q  $\preceq$  p  $\implies$  q ∈ M for q
using comp_in_M by simp
moreover
have ⟨s,y⟩ ∈ t  $\implies$  s ∈ domain(t) if t ∈ M for s y t
using that unfolding domain_def by auto
ultimately
have
(∀ s ∈ M. s ∈ domain(t1) ∨ s ∈ domain(t2)  $\longrightarrow$  (∀ q ∈ M. q ∈ ℙ ∧ q  $\preceq$  p  $\longrightarrow$ 
(f ' ⟨1, s, t1, q⟩ = 1  $\longleftrightarrow$  f ' ⟨1, s, t2, q⟩ = 1)))  $\longleftrightarrow$ 
(∀ s. s ∈ domain(t1) ∨ s ∈ domain(t2)  $\longrightarrow$  (∀ q. q ∈ ℙ ∧ q  $\preceq$  p  $\longrightarrow$ 
(f ' ⟨1, s, t1, q⟩ = 1  $\longleftrightarrow$  f ' ⟨1, s, t2, q⟩ = 1)))
using assms domain_trans[OF trans_M,of t1] domain_trans[OF trans_M,of
t2]
by auto
then
show ?thesis
unfolding eq_case_def is_eq_case_def
using assms pair_in_M_iff nat_into_M domain_closed apply_closed zero_in_M
Un_closed
by (simp add:components_abs)
qed

lemma mem_case_abs [simp]:
assumes t1 ∈ M t2 ∈ M p ∈ M f ∈ M
shows is_mem_case(##M,t1,t2,p,ℙ,leq,f)  $\longleftrightarrow$  mem_case(t1,t2,p,ℙ,leq,f)
proof
{
fix v
assume v ∈ ℙ v  $\preceq$  p is_mem_case(##M,t1,t2,p,ℙ,leq,f)
moreover
from this
have v ∈ M ⟨v,p⟩ ∈ M (##M)(v)

```

```

    using transitivity[OF _ P_in_M, of v] transitivity[OF _ leq_in_M]
    by simp_all
  moreover
  from calculation assms
  obtain q r s where
    r ∈ P ∧ q ∈ P ∧ ⟨q, v⟩ ∈ M ∧ ⟨s, r⟩ ∈ M ∧ ⟨q, r⟩ ∈ M ∧ 0 ∈ M ∧
    ⟨0, t1, s, q⟩ ∈ M ∧ q ≼ v ∧ ⟨s, r⟩ ∈ t2 ∧ q ≼ r ∧ f ' ⟨0, t1, s, q⟩ = 1
    unfolding is_mem_case_def by (auto simp add: components_abs)
  then
  have ∃ q s r. r ∈ P ∧ q ∈ P ∧ q ≼ v ∧ ⟨s, r⟩ ∈ t2 ∧ q ≼ r ∧ f ' ⟨0, t1, s, q⟩ = 1
    by auto
}
then
show mem_case(t1, t2, p, P, leq, f) if is_mem_case(##M, t1, t2, p, P, leq, f)
  unfolding mem_case_def using that assms by auto
next
{ fix v
  assume v ∈ M v ∈ P ⟨v, p⟩ ∈ M v ≼ p mem_case(t1, t2, p, P, leq, f)
  moreover
  from this
  obtain q s r where r ∈ P ∧ q ∈ P ∧ q ≼ v ∧ ⟨s, r⟩ ∈ t2 ∧ q ≼ r ∧ f ' ⟨0, t1,
s, q⟩ = 1
    unfolding mem_case_def by auto
  moreover
  from this ⟨t2 ∈ M⟩
  have r ∈ M q ∈ M s ∈ M r ∈ P ∧ q ∈ P ∧ q ≼ v ∧ ⟨s, r⟩ ∈ t2 ∧ q ≼ r ∧ f ' ⟨0,
t1, s, q⟩ = 1
    using transitivity domainI[of s r] domain_closed
    by auto
  moreover
  note ⟨t1 ∈ M⟩
  ultimately
  have ∃ q ∈ M . ∃ s ∈ M. ∃ r ∈ M.
    r ∈ P ∧ q ∈ P ∧ ⟨q, v⟩ ∈ M ∧ ⟨s, r⟩ ∈ M ∧ ⟨q, r⟩ ∈ M ∧ 0 ∈ M ∧
    ⟨0, t1, s, q⟩ ∈ M ∧ q ≼ v ∧ ⟨s, r⟩ ∈ t2 ∧ q ≼ r ∧ f ' ⟨0, t1, s, q⟩ = 1
    using pair_in_M_iff zero_in_M by auto
}
then
show is_mem_case(##M, t1, t2, p, P, leq, f) if mem_case(t1, t2, p, P, leq, f)
  unfolding is_mem_case_def
  using assms that zero_in_M pair_in_M_iff apply_closed nat_into_M
  by (auto simp add: components_abs)
qed

```

lemma *Hfrc_abs*:

$\llbracket fnc \in M; f \in M \rrbracket \implies$

$is_Hfrc(\#\#M, P, leq, fnc, f) \longleftrightarrow Hfrc(P, leq, fnc, f)$

unfolding *is_Hfrc_def* *Hfrc_def* **using** *pair_in_M_iff zero_in_M*
by (*auto simp add: components_abs*)

lemma *Hfrc_at_abs*:
 $\llbracket fnc \in M; f \in M; z \in M \rrbracket \implies$
 $is_Hfrc_at(\#\#M, \mathbb{P}, leq, fnc, f, z) \longleftrightarrow z = bool_of_o(Hfrc(\mathbb{P}, leq, fnc, f))$
unfolding *is_Hfrc_at_def* **using** *Hfrc_abs*
by *auto*

lemma *components_closed* :
 $x \in M \implies (\#\#M)(ftype(x))$
 $x \in M \implies (\#\#M)(name1(x))$
 $x \in M \implies (\#\#M)(name2(x))$
 $x \in M \implies (\#\#M)(cond_of(x))$
unfolding *ftype_def name1_def name2_def cond_of_def* **using** *fst_snd_closed*
by *simp_all*

lemma *ecloseN_closed*:
 $(\#\#M)(A) \implies (\#\#M)(ecloseN(A))$
 $(\#\#M)(A) \implies (\#\#M)(eclose_n(name1, A))$
 $(\#\#M)(A) \implies (\#\#M)(eclose_n(name2, A))$
unfolding *ecloseN_def eclose_n_def*
using *components_closed eclose_closed singleton_closed Un_closed* **by** *auto*

lemma *eclose_n_abs* :
assumes $x \in M$ $ec \in M$
shows $is_eclose_n(\#\#M, is_name1, ec, x) \longleftrightarrow ec = eclose_n(name1, x)$
 $is_eclose_n(\#\#M, is_name2, ec, x) \longleftrightarrow ec = eclose_n(name2, x)$
unfolding *is_eclose_n_def eclose_n_def*
using *assms name1_abs name2_abs eclose_abs singleton_closed components_closed*
by *auto*

lemma *ecloseN_abs* :
 $\llbracket x \in M; ec \in M \rrbracket \implies is_ecloseN(\#\#M, x, ec) \longleftrightarrow ec = ecloseN(x)$
unfolding *is_ecloseN_def ecloseN_def*
using *eclose_n_abs Un_closed union_abs ecloseN_closed*
by *auto*

lemma *frecR_abs* :
 $x \in M \implies y \in M \implies frecR(x, y) \longleftrightarrow is_frecR(\#\#M, x, y)$
unfolding *frecR_def is_frecR_def*
using *zero_in_M domain_closed Un_closed components_closed nat_into_M*
by (*auto simp add: components_abs*)

lemma *frecrelP_abs* :
 $z \in M \implies frecrelP(\#\#M, z) \longleftrightarrow (\exists x y. z = \langle x, y \rangle \wedge frecR(x, y))$
using *pair_in_M_iff frecR_abs* **unfolding** *frecrelP_def* **by** *auto*

lemma *frecrel_abs*:
assumes $A \in M$ $r \in M$

shows $is_frecrel(\#\#M,A,r) \longleftrightarrow r = frecrel(A)$
proof -
from $\langle A \in M \rangle$
have $z \in M$ **if** $z \in A \times A$ **for** z
using *cartprod_closed* *transitivity* **that** **by** *simp*
then
have $Collect(A \times A, frecrelP(\#\#M)) = Collect(A \times A, \lambda z. (\exists x y. z = \langle x, y \rangle \wedge frecR(x, y)))$
using *Collect_cong*[of $A \times A$ $A \times A$ *frecrelP*($\#\#M$)] *assms* *frecrelP_abs* **by** *simp*
with *assms*
show *?thesis*
unfolding *is_frecrel_def* *def_frecrel* **using** *cartprod_closed*
by *simp*
qed

lemma *frecrel_closed*:
assumes $x \in M$
shows $frecrel(x) \in M$
proof -
have $Collect(x \times x, \lambda z. (\exists x y. z = \langle x, y \rangle \wedge frecR(x, y))) \in M$
using *Collect_in_M*[of *frecrelP_fm*(0) []] *arity_frecrelP_fm* *sats_frecrelP_fm*
frecrelP_abs $\langle x \in M \rangle$ *cartprod_closed*
by *simp*
then
show *?thesis*
unfolding *frecrel_def* *Rrel_def* *frecrelP_def* **by** *simp*
qed

lemma *field_frecrel* : $field(frecrel(names_below(\mathbb{P}, x))) \subseteq names_below(\mathbb{P}, x)$
unfolding *frecrel_def*
using *field_Rrel* **by** *simp*

lemma *forcerelD* : $wv \in forcerel(\mathbb{P}, x) \implies uv \in names_below(\mathbb{P}, x) \times names_below(\mathbb{P}, x)$
unfolding *forcerel_def*
using *trancl_type* *field_frecrel* **by** *blast*

lemma *wf_forcerel* :
 $wf(forcerel(\mathbb{P}, x))$
unfolding *forcerel_def* **using** *wf_trancl* *wf_frecrel* .

lemma *restrict_trancl_forcerel*:
assumes $frecR(w, y)$
shows $restrict(f, frecrel(names_below(\mathbb{P}, x)) - \{y\}) 'w$
 $= restrict(f, forcerel(\mathbb{P}, x) - \{y\}) 'w$
unfolding *forcerel_def* *frecrel_def* **using** *assms* *restrict_trancl_Rrel*[of *frecR*]
by *simp*

lemma *names_belowI* :
assumes $frecR(\langle ft, n1, n2, p \rangle, \langle a, b, c, d \rangle)$ $p \in \mathbb{P}$

```

shows ⟨ft,n1,n2,p⟩ ∈ names_below(ℙ,⟨a,b,c,d⟩) (is ?x ∈ names_below(⟦_,?y⟧))
proof -
  from assms
  have ft ∈ 2 a ∈ 2
    unfolding frecR_def by (auto simp add:components_simp)
  from assms
  consider (eq) n1 ∈ domain(b) ∪ domain(c) ∧ (n2 = b ∨ n2 = c)
    | (mem) n1 = b ∧ n2 ∈ domain(c)
    unfolding frecR_def by (auto simp add:components_simp)
  then show ?thesis
  proof cases
    case eq
    then
    have n1 ∈ eclose(b) ∨ n1 ∈ eclose(c)
      using Un_iff in_dom_in_eclose by auto
    with eq
    have n1 ∈ ecloseN(?y) n2 ∈ ecloseN(?y)
      using ecloseNI components_in_eclose by auto
    with ⟨ft∈2⟩ ⟨p∈ℙ⟩
    show ?thesis
      unfolding names_below_def by auto
  next
  case mem
  then
  have n1 ∈ ecloseN(?y) n2 ∈ ecloseN(?y)
    using mem_eclose_trans ecloseNI in_dom_in_eclose components_in_eclose
    by auto
  with ⟨ft∈2⟩ ⟨p∈ℙ⟩
  show ?thesis
    unfolding names_below_def
    by auto
qed
qed

lemma names_below_tr :
  assumes x ∈ names_below(ℙ,y) y ∈ names_below(ℙ,z)
  shows x ∈ names_below(ℙ,z)
proof -
  let ?A = λy . names_below(ℙ,y)
  note assms
  moreover from this
  obtain fx x1 x2 px where x = ⟨fx,x1,x2,px⟩ fx ∈ 2 x1 ∈ ecloseN(y) x2 ∈ ecloseN(y)
  px ∈ ℙ
  unfolding names_below_def by auto
  moreover from calculation
  obtain fy y1 y2 py where y = ⟨fy,y1,y2,py⟩ fy ∈ 2 y1 ∈ ecloseN(z) y2 ∈ ecloseN(z)
  py ∈ ℙ
  unfolding names_below_def by auto
  moreover from calculation

```

```

have  $x1 \in \text{eclose}N(z)$   $x2 \in \text{eclose}N(z)$ 
  using ecloseN_mono names_simp by auto
ultimately
have  $x \in ?A(z)$ 
  unfolding names_below_def by simp
then
show ?thesis using subsetI by simp
qed

```

```

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 -
  from assms
  have  $x \in \text{names\_below}(\mathbb{P}, z)$   $y \in \text{names\_below}(\mathbb{P}, z)$  frecR(x,y)
    unfolding frecrel_def Rrel_def
    by auto
  obtain  $f\ n1\ n2\ p$  where  $x = \langle f, n1, n2, p \rangle$   $f \in 2$   $n1 \in \text{eclose}N(z)$   $n2 \in \text{eclose}N(z)$   $p \in \mathbb{P}$ 
    using  $\langle x \in \text{names\_below}(\mathbb{P}, z) \rangle$ 
    unfolding names_below_def by auto
  moreover
  obtain  $fy\ m1\ m2\ q$  where  $q \in \mathbb{P}$   $y = \langle fy, m1, m2, q \rangle$ 
    using  $\langle y \in \text{names\_below}(\mathbb{P}, z) \rangle$ 
    unfolding names_below_def by auto
  moreover
  note  $\langle \text{frecR}(x, y) \rangle$ 
  ultimately
  show ?thesis
    using names_belowI by simp
qed

```

```

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 -
  from assms
  have  $x \in \text{names\_below}(\mathbb{P}, z)$ 
    unfolding frecrel_def Rrel_def
    by auto
  from  $\langle x \in \text{names\_below}(\mathbb{P}, z) \rangle$ 
  obtain  $f\ n1\ n2\ p$  where
     $x = \langle f, n1, n2, p \rangle$   $f \in 2$   $n1 \in \text{eclose}N(z)$   $n2 \in \text{eclose}N(z)$   $p \in \mathbb{P}$ 
    unfolding names_below_def by auto
  then
  have  $n1 \in \text{eclose}N(x)$   $n2 \in \text{eclose}N(x)$ 
    using components_in_eclose by simp_all
  with  $\langle f \in 2 \rangle$   $\langle p \in \mathbb{P} \rangle$   $\langle x = \langle f, n1, n2, p \rangle \rangle$ 
  show ?thesis
    unfolding names_below_def by simp

```

qed

lemma *forcereel_arg_into_names_below* :
 assumes $\langle x,y \rangle \in \text{forcereel}(\mathbb{P},z)$
 shows $x \in \text{names_below}(\mathbb{P},x)$
 using *assms*
 unfolding *forcereel_def*
 by(*rule trancl_induct;auto simp add: arg_into_names_below*)

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 -
 from *assms*
 have $x \in \text{names_below}(\mathbb{P},y)$
 using *arg_into_names_below2 by simp*
 then
 show *?thesis*
 using *names_below_tr subsetI by simp*
qed

lemma *frecrel_mono* :
 assumes $\langle x,y \rangle \in \text{frecrel}(\text{names_below}(\mathbb{P},z))$
 shows $\text{frecrel}(\text{names_below}(\mathbb{P},x)) \subseteq \text{frecrel}(\text{names_below}(\mathbb{P},y))$
 unfolding *frecrel_def*
 using *Rrel_mono names_below_mono assms by simp*

lemma *forcereel_mono2* :
 assumes $\langle x,y \rangle \in \text{frecrel}(\text{names_below}(\mathbb{P},z))$
 shows $\text{forcereel}(\mathbb{P},x) \subseteq \text{forcereel}(\mathbb{P},y)$
 unfolding *forcereel_def*
 using *trancl_mono frecrel_mono assms by simp*

lemma *forcereel_mono_aux* :
 assumes $\langle x,y \rangle \in \text{frecrel}(\text{names_below}(\mathbb{P}, w))^+$
 shows $\text{forcereel}(\mathbb{P},x) \subseteq \text{forcereel}(\mathbb{P},y)$
 using *assms*
 by (*rule trancl_induct,simp_all add: subset_trans forcereel_mono2*)

lemma *forcereel_mono* :
 assumes $\langle x,y \rangle \in \text{forcereel}(\mathbb{P},z)$
 shows $\text{forcereel}(\mathbb{P},x) \subseteq \text{forcereel}(\mathbb{P},y)$
 using *forcereel_mono_aux assms unfolding forcereel_def by simp*

lemma *forcereel_eq_aux*: $x \in \text{names_below}(\mathbb{P}, w) \implies \langle x,y \rangle \in \text{forcereel}(\mathbb{P},z) \implies$
 $(y \in \text{names_below}(\mathbb{P}, w) \longrightarrow \langle x,y \rangle \in \text{forcereel}(\mathbb{P},w))$
 unfolding *forcereel_def*
proof (*rule_tac a=x and b=y and*
 $P=\lambda y . y \in \text{names_below}(\mathbb{P}, w) \longrightarrow \langle x,y \rangle \in \text{frecrel}(\text{names_below}(\mathbb{P},w))^+ \text{ in$

```

trancl_induct,simp)
  let ?A= $\lambda$  a . names_below( $\mathbb{P}$ , a)
  let ?R= $\lambda$  a . frecrel(?A(a))
  let ?fR= $\lambda$  a . forcerel(a)
  show  $u \in ?A(w) \longrightarrow \langle x, u \rangle \in ?R(w)^{\wedge+}$  if  $x \in ?A(w) \langle x, y \rangle \in ?R(z)^{\wedge+} \langle x, u \rangle \in ?R(z)$ 
for u
  using that frecrelD frecrelI r_into_trancl
  unfolding names_below_def by simp
{
  fix u v
  assume  $x \in ?A(w)$ 
   $\langle x, y \rangle \in ?R(z)^{\wedge+}$ 
   $\langle x, u \rangle \in ?R(z)^{\wedge+}$ 
   $\langle u, v \rangle \in ?R(z)$ 
   $u \in ?A(w) \implies \langle x, u \rangle \in ?R(w)^{\wedge+}$ 
  then
  have  $v \in ?A(w) \implies \langle x, v \rangle \in ?R(w)^{\wedge+}$ 
  proof -
    assume  $v \in ?A(w)$ 
    from  $\langle \langle u, v \rangle \in \_ \rangle$ 
    have  $u \in ?A(v)$ 
      using arg_into_names_below2 by simp
    with  $\langle v \in ?A(w) \rangle$ 
    have  $u \in ?A(w)$ 
      using names_below_tr by simp
    with  $\langle v \in \_ \rangle \langle \langle u, v \rangle \in \_ \rangle$ 
    have  $\langle u, v \rangle \in ?R(w)$ 
      using frecrelD frecrelI r_into_trancl unfolding names_below_def by simp
    with  $\langle u \in ?A(w) \implies \langle x, u \rangle \in ?R(w)^{\wedge+} \rangle \langle u \in ?A(w) \rangle$ 
    have  $\langle x, u \rangle \in ?R(w)^{\wedge+}$ 
      by simp
    with  $\langle \langle u, v \rangle \in ?R(w) \rangle$ 
    show  $\langle x, v \rangle \in ?R(w)^{\wedge+}$  using trancl_trans r_into_trancl
      by simp
  qed
}
then
show  $v \in ?A(w) \longrightarrow \langle x, v \rangle \in ?R(w)^{\wedge+}$ 
  if  $x \in ?A(w)$ 
   $\langle x, y \rangle \in ?R(z)^{\wedge+}$ 
   $\langle x, u \rangle \in ?R(z)^{\wedge+}$ 
   $\langle u, v \rangle \in ?R(z)$ 
   $u \in ?A(w) \longrightarrow \langle x, u \rangle \in ?R(w)^{\wedge+}$  for u v
  using that
  by simp
qed

lemma forcerel_eq :
  assumes  $\langle z, x \rangle \in \text{forcerel}(\mathbb{P}, x)$ 

```

```

shows forcere( $\mathbb{P}, z$ ) = forcere( $\mathbb{P}, x$ )  $\cap$  names_below( $\mathbb{P}, z$ ) $\times$ names_below( $\mathbb{P}, z$ )
using assms forcere_eq_aux forcereD forcere_mono[of  $z\ x$ ] subsetI
by auto

lemma forcere_below_aux :
assumes  $\langle z, x \rangle \in$  forcere( $\mathbb{P}, x$ )  $\langle u, z \rangle \in$  forcere( $\mathbb{P}, x$ )
shows  $u \in$  names_below( $\mathbb{P}, z$ )
using assms(2)
unfolding forcere_def
proof(rule trancl_induct)
show  $u \in$  names_below( $\mathbb{P}, y$ ) if  $\langle u, y \rangle \in$  frecrel(names_below( $\mathbb{P}, x$ )) for  $y$ 
using that vimage_singleton_iff arg_into_names_below2 by simp
next
show  $u \in$  names_below( $\mathbb{P}, z$ )
if  $\langle u, y \rangle \in$  frecrel(names_below( $\mathbb{P}, x$ )) $^+$ 
 $\langle y, z \rangle \in$  frecrel(names_below( $\mathbb{P}, x$ ))
 $u \in$  names_below( $\mathbb{P}, y$ )
for  $y\ z$ 
using that arg_into_names_below2[of  $y\ z\ x$ ] names_below_tr by simp
qed

lemma forcere_below :
assumes  $\langle z, x \rangle \in$  forcere( $\mathbb{P}, x$ )
shows forcere( $\mathbb{P}, x$ ) -“  $\{z\} \subseteq$  names_below( $\mathbb{P}, z$ )
using vimage_singleton_iff assms forcere_below_aux by auto

lemma relation_forcere :
shows relation(forcere( $\mathbb{P}, z$ )) trans(forcere( $\mathbb{P}, z$ ))
unfolding forcere_def using relation_trancl trans_trancl by simp_all

lemma Hfrc_restrict_trancl: bool_of_o(Hfrc( $\mathbb{P},$  leq,  $y,$  restrict( $f,$  frecrel(names_below( $\mathbb{P}, x$ ))-“ $\{y\}$ ”)))
= bool_of_o(Hfrc( $\mathbb{P},$  leq,  $y,$  restrict( $f,$  (frecrel(names_below( $\mathbb{P}, x$ )) $^+$ )-“ $\{y\}$ ”)))
unfolding Hfrc_def bool_of_o_def eq_case_def mem_case_def
using restrict_trancl_forcere frecRI1 frecRI2 frecRI3
unfolding forcere_def
by simp

lemma frc_at_trancl: frc_at( $\mathbb{P},$  leq,  $z$ ) = wfrec(forcere( $\mathbb{P}, z$ ),  $z,$   $\lambda x\ f.$  bool_of_o(Hfrc( $\mathbb{P},$  leq,  $x,$   $f$ )))
unfolding frc_at_def forcere_def using wf_eq_trancl Hfrc_restrict_trancl by
simp

lemma forcereI1 :
assumes  $n1 \in$  domain( $b$ )  $\vee$   $n1 \in$  domain( $c$ )  $p \in \mathbb{P}$   $d \in \mathbb{P}$ 
shows  $\langle \langle 1, n1, b, p \rangle, \langle 0, b, c, d \rangle \rangle \in$  forcere( $\mathbb{P}, \langle 0, b, c, d \rangle$ )
proof -
let ? $x = \langle 1, n1, b, p \rangle$ 
let ? $y = \langle 0, b, c, d \rangle$ 
from assms

```

```

have frecR(?x,?y)
  using frecRI1 by simp
then
have ?x∈names_below(ℙ,?y) ?y ∈ names_below(ℙ,?y)
  using names_belowI assms components_in_eclose
  unfolding names_below_def by auto
with ⟨frecR(?x,?y)⟩
show ?thesis
  unfolding forcereI_def frecrel_def
  using subsetD[OF r_subset_trancl[OF relation_Rrel]] RrelI
  by auto
qed

```

```

lemma forcereI2 :
  assumes n1 ∈ domain(b) ∨ n1 ∈ domain(c) p∈ℙ d∈ℙ
  shows ⟨⟨1, n1, c, p⟩, ⟨0,b,c,d⟩⟩ ∈ forcereI(ℙ,⟨0,b,c,d⟩)
proof -
  let ?x=⟨1, n1, c, p⟩
  let ?y=⟨0,b,c,d⟩
  note assms
  moreover from this
  have frecR(?x,?y)
    using frecRI2 by simp
  moreover from calculation
  have ?x∈names_below(ℙ,?y) ?y ∈ names_below(ℙ,?y)
    using names_belowI components_in_eclose
    unfolding names_below_def by auto
  ultimately
  show ?thesis
    unfolding forcereI_def frecrel_def
    using subsetD[OF r_subset_trancl[OF relation_Rrel]] RrelI
    by auto
qed

```

```

lemma forcereI3 :
  assumes ⟨n2, r⟩ ∈ c p∈ℙ d∈ℙ r ∈ ℙ
  shows ⟨⟨0, b, n2, p⟩, ⟨1, b, c, d⟩⟩ ∈ forcereI(ℙ,⟨1,b,c,d⟩)
proof -
  let ?x=⟨0, b, n2, p⟩
  let ?y=⟨1, b, c, d⟩
  note assms
  moreover from this
  have frecR(?x,?y)
    using frecRI3 by simp
  moreover from calculation
  have ?x∈names_below(ℙ,?y) ?y ∈ names_below(ℙ,?y)
    using names_belowI components_in_eclose
    unfolding names_below_def by auto
  ultimately

```

show *?thesis*
unfolding *forcereI_def frecereI_def*
using *subsetD[OF r_subset_trancl[OF relation_Rrel]] RreI*
by *auto*
qed

lemmas *forcereI = forcereII[THEN vimage_singleton_iff[THEN iffD2]]*
forcereII[THEN vimage_singleton_iff[THEN iffD2]]
forcereI3[THEN vimage_singleton_iff[THEN iffD2]]

lemma *aux_def_frc_at:*
assumes $z \in \text{forcereI}(\mathbb{P}, x) - \{x\}$
shows $\text{wfrec}(\text{forcereI}(\mathbb{P}, x), z, H) = \text{wfrec}(\text{forcereI}(\mathbb{P}, z), z, H)$
proof -
let $?A = \text{names_below}(\mathbb{P}, z)$
from *assms*
have $\langle z, x \rangle \in \text{forcereI}(\mathbb{P}, x)$
using *vimage_singleton_iff by simp*
moreover from this
have $z \in ?A$
using *forcereI_arg_into_names_below by simp*
moreover from calculation
have $\text{forcereI}(\mathbb{P}, z) = \text{forcereI}(\mathbb{P}, x) \cap (?A \times ?A)$
 $\text{forcereI}(\mathbb{P}, x) - \{z\} \subseteq ?A$
using *forcereI_eq forcereI_below*
by *auto*
moreover from calculation
have $\text{wfrec}(\text{forcereI}(\mathbb{P}, x), z, H) = \text{wfrec}[?A](\text{forcereI}(\mathbb{P}, x), z, H)$
using *wfrec_trans_restr[OF relation_forcereI(1) wf_forcereI_relation_forcereI(2),*
of x z ?A]
by *simp*
ultimately
show *?thesis*
using *wfrec_restr_eq by simp*
qed

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 -
  let ?r= $\lambda y. \text{forcerel}(\mathbb{P}, y)$  and ?Hf= $\lambda x f. \text{bool\_of\_o}(\text{Hfrc}(\mathbb{P}, \text{leq}, x, f))$ 
  let ?t= $\lambda y. ?r(y) -\{\{y\}\}$ 
  let ?arg= $\langle ft, n1, n2, p \rangle$ 
  from wf_forcerel
  have wfr:  $\forall w. \text{wf}(?r(w)) ..$ 
  with wfrec [of ?r(?arg) ?arg ?Hf]
  have frc_at( $\mathbb{P}, \text{leq}, ?arg$ ) = ?Hf( ?arg,  $\lambda x \in ?r(?arg) -\{\{?arg\}\}. \text{wfrec}(?r(?arg), x,$ 
    ?Hf))
    using frc_at_trancl by simp
  also
  have ... = ?Hf( ?arg,  $\lambda x \in ?r(?arg) -\{\{?arg\}\}. \text{frc\_at}(\mathbb{P}, \text{leq}, x)$ )
    using aux_def_frc_at_frc_at_trancl by simp
  finally
  show ?thesis
    unfolding Hfrc_def mem_case_def eq_case_def
    using forcerelI assms
    by auto
qed

```

11.3 Absoluteness of frc_at

```

lemma forcerel_in_M :
  assumes  $x \in M$ 
  shows  $\text{forcerel}(\mathbb{P}, x) \in M$ 
  unfolding forcerel_def def_frecrel_names_below_def
proof -
  let ?Q =  $2 \times \text{ecloseN}(x) \times \text{ecloseN}(x) \times \mathbb{P}$ 
  have ?Q  $\times$  ?Q  $\in M$ 
    using  $\langle x \in M \rangle \text{nat\_into\_M ecloseN\_closed cartprod\_closed}$  by simp
  moreover
  have separation( $\#\#M, \lambda z. \text{frecrelP}(\#\#M, z)$ )
    using separation_in_ctm[of frecrelP_fm(0), OF  $\_\_\_\_ \text{sats\_frecrelP\_fm}$ ]
    arity_frecrelP_fm frecrelP_fm_type
    by auto
  moreover from this
  have separation( $\#\#M, \lambda z. \exists x y. z = \langle x, y \rangle \wedge \text{frecR}(x, y)$ )
    using separation_cong[OF frecrelP_abs]
    by force
  ultimately
  show  $\{z \in ?Q \times ?Q. \exists x y. z = \langle x, y \rangle \wedge \text{frecR}(x, y)\}^{\wedge+} \in M$ 
    using separation_closed_frecrelP_abs trancl_closed
    by simp
qed

```

```

lemma relation2_Hfrc_at_abs:
   $\text{relation2}(\#\#M, \text{is\_Hfrc\_at}(\#\#M, \mathbb{P}, \text{leq}), \lambda x f. \text{bool\_of\_o}(\text{Hfrc}(\mathbb{P}, \text{leq}, x, f)))$ 
  unfolding relation2_def using Hfrc_at_abs
  by simp

```

```

lemma Hfrc_at_closed :
   $\forall x \in M. \forall g \in M. \text{function}(g) \longrightarrow \text{bool\_of\_o}(\text{Hfrc}(\mathbb{P}, \text{leq}, x, g)) \in M$ 
  unfolding bool_of_o_def using zero_in_M nat_into_M [of 1] by simp

lemma wfrec_Hfrc_at :
  assumes  $X \in M$ 
  shows wfrec_replacement( $\#\#M, \text{is\_Hfrc\_at}(\#\#M, \mathbb{P}, \text{leq}), \text{forcerel}(\mathbb{P}, X)$ )
proof -
  have  $0: \text{is\_Hfrc\_at}(\#\#M, \mathbb{P}, \text{leq}, a, b, c) \longleftrightarrow$ 
     $\text{sats}(M, \text{Hfrc\_at\_fm}(8, 9, 2, 1, 0), [c, b, a, d, e, y, x, z, \mathbb{P}, \text{leq}, \text{forcerel}(\mathbb{P}, X)])$ 
  if  $a \in M \ b \in M \ c \in M \ d \in M \ e \in M \ y \in M \ x \in M \ z \in M$ 
  for  $a \ b \ c \ d \ e \ y \ x \ z$ 
  using that  $\langle X \in M \rangle \text{forcerel\_in\_M}$ 
   $\text{Hfrc\_at\_iff\_sats}$ [of concl:  $M \ \mathbb{P} \ \text{leq} \ a \ b \ c \ 8 \ 9 \ 2 \ 1 \ 0$ ]
  by simp
  have  $1: \text{sats}(M, \text{is\_wfrec\_fm}(\text{Hfrc\_at\_fm}(8, 9, 2, 1, 0), 5, 1, 0), [y, x, z, \mathbb{P}, \text{leq}, \text{forcerel}(\mathbb{P}, X)])$ 
 $\longleftrightarrow$ 
     $\text{is\_wfrec}(\#\#M, \text{is\_Hfrc\_at}(\#\#M, \mathbb{P}, \text{leq}), \text{forcerel}(\mathbb{P}, X), x, y)$ 
  if  $x \in M \ y \in M \ z \in M$  for  $x \ y \ z$ 
  using that  $\langle X \in M \rangle \text{forcerel\_in\_M}$  sats_is_wfrec_fm[OF 0]
  by simp
  let
     $?f = \text{Exists}(\text{And}(\text{pair\_fm}(1, 0, 2), \text{is\_wfrec\_fm}(\text{Hfrc\_at\_fm}(8, 9, 2, 1, 0), 5, 1, 0)))$ 
  have satsf:  $\text{sats}(M, ?f, [x, z, \mathbb{P}, \text{leq}, \text{forcerel}(\mathbb{P}, X)]) \longleftrightarrow$ 
     $(\exists y \in M. \text{pair}(\#\#M, x, y, z) \ \& \ \text{is\_wfrec}(\#\#M, \text{is\_Hfrc\_at}(\#\#M, \mathbb{P}, \text{leq}), \text{forcerel}(\mathbb{P}, X),$ 
 $x, y))$ 
  if  $x \in M \ z \in M$  for  $x \ z$ 
  using that  $1 \ \langle X \in M \rangle \text{forcerel\_in\_M}$  by (simp del: pair_abs)
  have artyf: arity(?f) = 5
  using arity_wfrec_replacement_fm[where  $p = \text{Hfrc\_at\_fm}(8, 9, 2, 1, 0)$  and
 $i = 10$ ]
   $\text{arity\_Hfrc\_at\_fm}$  ord_simp_union
  by simp
  moreover
  have ?f formula by simp
  ultimately
  have strong_replacement( $\#\#M, \lambda x \ z. \text{sats}(M, ?f, [x, z, \mathbb{P}, \text{leq}, \text{forcerel}(\mathbb{P}, X)])$ )
  using ZF_ground_replacements(1) 1 artyf  $\langle X \in M \rangle \text{forcerel\_in\_M}$ 
  unfolding replacement_assm_def wfrec_Hfrc_at_fm_def by simp
  then
  have strong_replacement( $\#\#M, \lambda x \ z.$ 
     $\exists y \in M. \text{pair}(\#\#M, x, y, z) \ \& \ \text{is\_wfrec}(\#\#M, \text{is\_Hfrc\_at}(\#\#M, \mathbb{P}, \text{leq}), \text{forcerel}(\mathbb{P}, X),$ 
 $x, y))$ 
  using repl_sats[of  $M \ ?f \ [\mathbb{P}, \text{leq}, \text{forcerel}(\mathbb{P}, X)]$ ] satsf by (simp del: pair_abs)
  then
  show thesis unfolding wfrec_replacement_def by simp
qed

```

lemma *names_below_abs* :
 $\llbracket Q \in M; x \in M; nb \in M \rrbracket \implies is_names_below(\#\#M, Q, x, nb) \longleftrightarrow nb = names_below(Q, x)$
unfolding *is_names_below_def names_below_def*
using *succ_in_M_iff zero_in_M cartprod_closed ecloseN_abs ecloseN_closed*
by *auto*

lemma *names_below_closed*:
 $\llbracket Q \in M; x \in M \rrbracket \implies names_below(Q, x) \in M$
unfolding *names_below_def*
using *zero_in_M cartprod_closed ecloseN_closed succ_in_M_iff*
by *simp*

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 $R(names_below(Q, x))$
unfolding *names_below_def* **using** *assms nat_into_M ecloseN_closed[of x]* **by**
auto

lemma *forcereL_abs* :
 $\llbracket x \in M; z \in M \rrbracket \implies is_forcereL(\#\#M, \mathbb{P}, x, z) \longleftrightarrow z = forcereL(\mathbb{P}, x)$
unfolding *is_forcereL_def forcereL_def*
using *frecreL_abs names_below_abs trancl_abs ecloseN_closed names_below_closed*
 $names_below_productE[of\ concl:\lambda p. is_frecreL(\#\#M, p, _) \longleftrightarrow _ = frecreL(p)]$
frecreL_closed
by *simp*

lemma *frc_at_abs*:
assumes $fnc \in M \ z \in M$
shows $is_frc_at(\#\#M, \mathbb{P}, leq, fnc, z) \longleftrightarrow z = frc_at(\mathbb{P}, leq, fnc)$
proof -
from *assms*
have $(\exists r \in M. is_forcereL(\#\#M, \mathbb{P}, fnc, r) \wedge is_wfrec(\#\#M, is_Hfrc_at(\#\#M,$
 $\mathbb{P}, leq), r, fnc, z))$
 $\longleftrightarrow is_wfrec(\#\#M, is_Hfrc_at(\#\#M, \mathbb{P}, leq), forcereL(\mathbb{P}, fnc), fnc, z)$
using *forcereL_abs forcereL_in_M* **by** *simp*
then
show *?thesis*
unfolding *frc_at_trancl is_frc_at_def*
using *assms wfrec_Hfrc_at[of fnc] wf_forcereL_relation_forcereL forcereL_in_M*
 $Hfrc_at_closed_relation2_Hfrc_at_abs$
 $trans_wfrec_abs[of\ forcereL(\mathbb{P}, fnc) \ fnc \ z \ is_Hfrc_at(\#\#M, \mathbb{P}, leq) \ \lambda x \ f.$
 $bool_of_o(Hfrc(\mathbb{P}, leq, x, f))]$
by *(simp flip:setclass_iff)*
qed

lemma *forces_eq'_abs* :
 $\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \implies is_forces_eq'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces_eq'(\mathbb{P}, leq, p, t1, t2)$

unfolding $is_forces_eq'_def$ $forces_eq'_def$
using frc_at_abs nat_into_M $pair_in_M_iff$ **by** (*auto simp add:components_abs*)

lemma $forces_mem'_abs$:
 $\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \implies is_forces_mem'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces_mem'(\mathbb{P}, leq, p, t1, t2)$
unfolding $is_forces_mem'_def$ $forces_mem'_def$
using frc_at_abs nat_into_M $pair_in_M_iff$ **by** (*auto simp add:components_abs*)

lemma $forces_neq'_abs$:
assumes $p \in M$ $t1 \in M$ $t2 \in M$
shows $is_forces_neq'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces_neq'(\mathbb{P}, leq, p, t1, t2)$
proof -
have $q \in M$ **if** $q \in \mathbb{P}$ **for** q
using *that transitivity by simp*
with *assms*
show *?thesis*
unfolding $is_forces_neq'_def$ $forces_neq'_def$
using $forces_eq'_abs$ $pair_in_M_iff$
by (*auto simp add:components_abs,blast*)
qed

lemma $forces_nmem'_abs$:
assumes $p \in M$ $t1 \in M$ $t2 \in M$
shows $is_forces_nmem'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces_nmem'(\mathbb{P}, leq, p, t1, t2)$
proof -
have $q \in M$ **if** $q \in \mathbb{P}$ **for** q
using *that transitivity by simp*
with *assms*
show *?thesis*
unfolding $is_forces_nmem'_def$ $forces_nmem'_def$
using $forces_mem'_abs$ $pair_in_M_iff$
by (*auto simp add:components_abs,blast*)
qed

lemma leq_abs :
 $\llbracket l \in M ; q \in M ; p \in M \rrbracket \implies is_leq(\#\#M, l, q, p) \longleftrightarrow \langle q, p \rangle \in l$
unfolding is_leq_def **using** $pair_in_M_iff$ **by** *simp*

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 $def_forces_eq: p \in \mathbb{P} \Longrightarrow p \text{ forces}_a (t1 = t2) \longleftrightarrow$
 $(\forall s \in \text{domain}(t1) \cup \text{domain}(t2). \forall q. q \in \mathbb{P} \wedge q \preceq p \longrightarrow$
 $(q \text{ forces}_a (s \in t1) \longleftrightarrow q \text{ forces}_a (s \in t2)))$
unfolding $forces_eq_def \text{ forces_mem_def } forces_eq'_def \text{ forces_mem}'_def$
using $def_frc_at[of p 0 t1 t2]$
unfolding $bool_of_o_def$
by $auto$

lemma $def_forces_mem: p \in \mathbb{P} \Longrightarrow p \text{ forces}_a (t1 \in t2) \longleftrightarrow$
 $(\forall v \in \mathbb{P}. v \preceq p \longrightarrow$
 $(\exists q. \exists s. \exists r. r \in \mathbb{P} \wedge q \in \mathbb{P} \wedge q \preceq v \wedge \langle s, r \rangle \in t2 \wedge q \preceq r \wedge q \text{ forces}_a (t1 = s)))$
unfolding $forces_eq'_def \text{ forces_mem}'_def \text{ forces_eq_def } \text{ forces_mem_def}$
using $def_frc_at[of p 1 t1 t2]$
unfolding $bool_of_o_def$
by $auto$

lemma $forces_eq_abs :$
 $\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \Longrightarrow is_forces_eq(p, t1, t2) \longleftrightarrow p \text{ forces}_a (t1 = t2)$
unfolding $forces_eq_def$
using $forces_eq'_abs$ **by** $simp$

lemma $forces_mem_abs :$
 $\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \Longrightarrow is_forces_mem(p, t1, t2) \longleftrightarrow p \text{ forces}_a (t1 \in t2)$
unfolding $forces_mem_def$
using $forces_mem'_abs$
by $simp$

definition

$forces_neq :: [i, i, i] \Rightarrow o$ $(\preceq \text{ forces}_a '(_ \neq _)' [36, 1, 1] 60)$ **where**
 $p \text{ forces}_a (t1 \neq t2) \equiv \neg (\exists q \in \mathbb{P}. q \preceq p \wedge q \text{ forces}_a (t1 = t2))$

definition

$forces_nmem :: [i, i, i] \Rightarrow o$ $(\preceq \text{ forces}_a '(_ \notin _)' [36, 1, 1] 60)$ **where**
 $p \text{ forces}_a (t1 \notin t2) \equiv \neg (\exists q \in \mathbb{P}. q \preceq p \wedge q \text{ forces}_a (t1 \in t2))$

lemma $forces_neq :$

$p \text{ forces}_a (t1 \neq t2) \longleftrightarrow forces_neq'(\mathbb{P}, leq, p, t1, t2)$
unfolding $forces_neq_def \text{ forces_neq}'_def \text{ forces_eq_def}$ **by** $simp$

lemma $forces_nmem :$

$p \text{ forces}_a (t1 \notin t2) \longleftrightarrow forces_nmem'(\mathbb{P}, leq, p, t1, t2)$

unfolding *forces_nmemb_def forces_nmemb'_def forces_mem_def* **by** *simp*

abbreviation *Forces* :: $[i, i, i] \Rightarrow o$ ($\langle _ \Vdash _ \rangle$ [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 *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 $q \Vdash \cdot x \in y \cdot \text{env} \longleftrightarrow q \in \mathbb{P} \wedge \text{is_forces_mem}(q, xx, yy)$
unfolding *forces_def*
using *assms*
by *simp*

lemma *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 $q \Vdash \cdot a = b \cdot \text{env} \longleftrightarrow q \in \mathbb{P} \wedge \text{is_forces_eq}(q, x, y)$
unfolding *forces_def*
using *assms*
by *simp*

lemma *sats_forces_Nand* :
assumes $\varphi \in \text{formula } \psi \in \text{formula } \text{env} \in \text{list}(M) \ p \in M$
shows $p \Vdash \cdot \neg(\varphi \wedge \psi) \cdot \text{env} \longleftrightarrow$
 $p \in \mathbb{P} \wedge \neg(\exists q \in M. q \in \mathbb{P} \wedge \text{is_leq}(\#\#M, \text{leq}, q, p) \wedge (q \Vdash \varphi \text{ env}) \wedge (q \Vdash \psi \text{ env}))$
unfolding *forces_def*
using *sats_is_leq_fm_auto assms sats_ren_forces_nand zero_in_M*
by *simp*

lemma *sats_forces_Neg* :
assumes $\varphi \in \text{formula } \text{env} \in \text{list}(M) \ p \in M$
shows $p \Vdash \cdot \neg \varphi \cdot \text{env} \longleftrightarrow$
 $(p \in \mathbb{P} \wedge \neg(\exists q \in M. q \in \mathbb{P} \wedge \text{is_leq}(\#\#M, \text{leq}, q, p) \wedge (q \Vdash \varphi \text{ env})))$
unfolding *Neg_def* **using** *assms sats_forces_Nand*
by *simp*

lemma *sats_forces_Forall* :
assumes $\varphi \in \text{formula } \text{env} \in \text{list}(M) \ p \in M$
shows $p \Vdash \cdot (\forall \varphi) \cdot \text{env} \longleftrightarrow p \in \mathbb{P} \wedge (\forall x \in M. p \Vdash \varphi ([x] @ \text{env}))$
unfolding *forces_def* **using** *assms sats_ren_forces_forall*
by *simp*

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

$val :: [i, i] \Rightarrow i$ **where**
 $val(G, \tau) \equiv wfrec(\text{edrel}(\text{eclose}(\{\tau\})), \tau, Hv(G))$

definition

$GenExt :: [i, i] \Rightarrow i$ ($\langle _ _ \rangle [71, 1]$)
where $M[G] \equiv \{ val(G, \tau) . \tau \in M \}$

lemma *map_val_in_MG*:

assumes
 $env \in \text{list}(M)$
shows
 $\text{map}(val(G), env) \in \text{list}(M[G])$
unfolding *GenExt_def* **using** *assms map_type2* **by** *simp*

12.1 Values and check-names

context *forcing_data1*
begin

lemma *name_components_in_M*:

assumes $\langle \sigma, p \rangle \in \vartheta \ \vartheta \in M$
shows $\sigma \in M \ p \in M$
using *assms transitivity pair_in_M_iff*
by *auto*

definition

$Hcheck :: [i, i] \Rightarrow i$ **where**
 $Hcheck(z, f) \equiv \{ \langle f'y, 1 \rangle . y \in z \}$

definition

$check :: i \Rightarrow i$ **where**
 $check(x) \equiv \text{transrec}(x, Hcheck)$

lemma *checkD*:

$check(x) = wfrec(\text{Memrel}(\text{eclose}(\{x\})), x, Hcheck)$
unfolding *check_def transrec_def* **..**

```

lemma Hcheck_trancl: Hcheck(y, restrict(f, Memrel(eclose({x}))-“{y}”))
    = Hcheck(y, restrict(f, (Memrel(eclose({x}))+)-“{y}”))
unfolding Hcheck_def
using restrict_trans_eq by simp

lemma check_trancl: check(x) = wfrec(rcheck(x), x, Hcheck)
using checkD wf_eq_trancl Hcheck_trancl unfolding rcheck_def by simp

lemma rcheck_in_M : x ∈ M ⇒ rcheck(x) ∈ M
unfolding rcheck_def by (simp flip: setclass_iff)

lemma rcheck_subset_M : x ∈ M ⇒ field(rcheck(x)) ⊆ eclose({x})
unfolding rcheck_def using field_Memrel field_trancl by auto

lemma aux_def_check: x ∈ y ⇒
    wfrec(Memrel(eclose({y})), x, Hcheck) =
    wfrec(Memrel(eclose({x})), x, Hcheck)
by (rule wfrec_eclose_eq, auto simp add: arg_into_eclose eclose_sing)

lemma def_check : check(y) = { ⟨check(w), 1⟩ . w ∈ y }
proof -
  let
    ?r = λy. Memrel(eclose({y}))
  have wfr: ∀ w . wf(?r(w))
    using wf_Memrel ..
  then
  have check(y) = Hcheck( y, λx ∈ ?r(y) . “{y}”. wfrec(?r(y), x, Hcheck))
    using wfrec[of ?r(y) y Hcheck] checkD by simp
  also
  have ... = Hcheck( y, λx ∈ y . wfrec(?r(y), x, Hcheck))
    using under_Memrel_eclose arg_into_eclose by simp
  also
  have ... = Hcheck( y, λx ∈ y . check(x))
    using aux_def_check checkD by simp
  finally
  show ?thesis
    using Hcheck_def by simp
qed

lemma def_checkS :
  fixes n
  assumes n ∈ nat
  shows check(succ(n)) = check(n) ∪ {⟨check(n), 1⟩}
proof -
  have check(succ(n)) = {⟨check(i), 1⟩ . i ∈ succ(n)}
    using def_check by blast
  also
  have ... = {⟨check(i), 1⟩ . i ∈ n} ∪ {⟨check(n), 1⟩}
    by blast

```

```

also
have ... =  $check(n) \cup \{ \langle check(n), 1 \rangle \}$ 
  using def_check[of n,symmetric] by simp
finally
show ?thesis .
qed

```

```

lemma field_Memrel2 :
  assumes  $x \in M$ 
  shows  $field(Memrel(eclose(\{x\}))) \subseteq M$ 
proof -
  have  $field(Memrel(eclose(\{x\}))) \subseteq eclose(\{x\})$   $eclose(\{x\}) \subseteq M$ 
    using Ordinal.Memrel_type field_rel_subset assms eclose_least[OF trans_M]
by auto
  then
  show ?thesis
    using subset_trans by simp
qed

```

```

lemma aux_def_val:
  assumes  $z \in domain(x)$ 
  shows  $wfrec(edrel(eclose(\{z\})),z,Hv(G)) = wfrec(edrel(eclose(\{z\})),z,Hv(G))$ 
proof -
  let  $?r = \lambda x . edrel(eclose(\{x\}))$ 
  have  $z \in eclose(\{z\})$ 
    using arg_in_eclose_sing .
  moreover
  have relation( $?r(x)$ )
    using relation_edrel .
  moreover
  have wf( $?r(x)$ )
    using wf_edrel .
  moreover from assms
  have  $tr\_down(?r(x),z) \subseteq eclose(\{z\})$ 
    using tr_edrel_subset by simp
  ultimately
  have  $wfrec(?r(x),z,Hv(G)) = wfrec[eclose(\{z\})](?r(x),z,Hv(G))$ 
    using wfrec_restr by simp
  also from  $\langle z \in domain(x) \rangle$ 
  have ... =  $wfrec(?r(z),z,Hv(G))$ 
    using restrict_edrel_eq wfrec_restr_eq by simp
  finally
  show ?thesis .
qed

```

The next lemma provides the usual recursive expression for the definition of *val*.

```

lemma def_val:  $val(G,x) = \{ z . t \in domain(x) , (\exists p \in G . \langle t,p \rangle \in x) \wedge z = val(G,t) \}$ 
proof -

```

```

let
  ?r = λτ . edrel(eclose({τ}))
let
  ?f = λz ∈ ?r(x) . {x}. wfrec(?r(x), z, Hv(G))
have ∀τ. wf(?r(τ))
  using wf_edrel by simp
with wfrec [of _ x]
have val(G, x) = Hv(G, x, ?f)
  using val_def by simp
also
have ... = Hv(G, x, λz ∈ domain(x). wfrec(?r(x), z, Hv(G)))
  using dom_under_edrel_eclose by simp
also
have ... = Hv(G, x, λz ∈ domain(x). val(G, z))
  using aux_def_val val_def by simp
finally
show ?thesis
  using Hv_def by simp
qed

```

```

lemma val_mono : x ⊆ y ⇒ val(G, x) ⊆ val(G, y)
  by (subst (1 2) def_val, force)

```

Check-names are the canonical names for elements of the ground model.
Here we show that this is the case.

```

lemma val_check : 1 ∈ G ⇒ 1 ∈ P ⇒ val(G, check(y)) = y
proof (induct rule:eps_induct)
  case (1 y)
  then show ?case
  proof -
    have check(y) = { ⟨check(w), 1⟩ . w ∈ y } (is _ = ?C)
      using def_check .
    then
    have val(G, check(y)) = val(G, {⟨check(w), 1⟩ . w ∈ y})
      by simp
    also
    have ... = {z . t ∈ domain(?C) , (∃ p ∈ G . ⟨t, p⟩ ∈ ?C) ∧ z = val(G, t) }
      using def_val by blast
    also
    have ... = {z . t ∈ domain(?C) , (∃ w ∈ y. t = check(w)) ∧ z = val(G, t) }
      using 1 by simp
    also
    have ... = {val(G, check(w)) . w ∈ y }
      by force
    finally
    show val(G, check(y)) = y
      using 1 by simp
  qed
qed

```

lemma *val_of_name* :

$$val(G, \{x \in A \times \mathbb{P}. Q(x)\}) = \{z . t \in A, (\exists p \in \mathbb{P} . Q(\langle t, p \rangle) \wedge p \in G) \wedge z = val(G, t)\}$$

proof -

let

$$?n = \{x \in A \times \mathbb{P}. Q(x)\} \text{ and}$$

$$?r = \lambda \tau . edrel(eclose(\{\tau\}))$$

let

$$?f = \lambda z \in ?r(?n) . \text{“}\{?n\} . val(G, z)$$

have

$$wfR : wf(?r(\tau)) \text{ for } \tau$$

by (*simp add: wf_edrel*)

have $domain(?n) \subseteq A$ **by** *auto*

{ fix *t*

assume $H : t \in domain(\{x \in A \times \mathbb{P} . Q(x)\})$

then have $?f \text{ ‘ } t = (if\ t \in ?r(?n) \text{ ‘ } \{?n\} \text{ then } val(G, t) \text{ else } 0)$

by *simp*

moreover have $\dots = val(G, t)$

using *dom_under_edrel_eclose H if_P* **by** *auto*

}

then

have $Eq1 : t \in domain(\{x \in A \times \mathbb{P} . Q(x)\}) \implies val(G, t) = ?f \text{ ‘ } t$ **for** *t*

by *simp*

have $val(G, ?n) = \{z . t \in domain(?n), (\exists p \in G . \langle t, p \rangle \in ?n) \wedge z = val(G, t)\}$

by (*subst def_val, simp*)

also

have $\dots = \{z . t \in domain(?n), (\exists p \in \mathbb{P} . \langle t, p \rangle \in ?n \wedge p \in G) \wedge z = ?f \text{ ‘ } t\}$

unfolding *Hv_def*

by (*auto simp add: Eq1*)

also

have $\dots = \{z . t \in domain(?n), (\exists p \in \mathbb{P} . \langle t, p \rangle \in ?n \wedge p \in G) \wedge z = (if\ t \in ?r(?n) \text{ ‘ } \{?n\} \text{ then } val(G, t) \text{ else } 0)\}$

then $val(G, t) \text{ else } 0\}$

by (*simp*)

also

have $\dots = \{z . t \in domain(?n), (\exists p \in \mathbb{P} . \langle t, p \rangle \in ?n \wedge p \in G) \wedge z = val(G, t)\}$

proof -

have $domain(?n) \subseteq ?r(?n) \text{ ‘ } \{?n\}$

using *dom_under_edrel_eclose* **by** *simp*

then

have $\forall t \in domain(?n) . (if\ t \in ?r(?n) \text{ ‘ } \{?n\} \text{ then } val(G, t) \text{ else } 0) = val(G, t)$

by *auto*

then

show $\{z . t \in domain(?n), (\exists p \in \mathbb{P} . \langle t, p \rangle \in ?n \wedge p \in G) \wedge z = (if\ t \in ?r(?n) \text{ ‘ } \{?n\} \text{ then } val(G, t) \text{ else } 0)\} =$

$\{z . t \in domain(?n), (\exists p \in \mathbb{P} . \langle t, p \rangle \in ?n \wedge p \in G) \wedge z = val(G, t)\}$

by *auto*

qed

also

have $\dots = \{z . t \in A, (\exists p \in \mathbb{P} . \langle t, p \rangle \in ?n \wedge p \in G) \wedge z = val(G, t)\}$

```

    by force
  finally
  show val(G, ?n) = { z . t ∈ A, (∃ p ∈ P . Q(⟨t, p⟩) ∧ p ∈ G) ∧ z = val(G, t) }
    by auto
qed

lemma val_of_name_alt :
  val(G, {x ∈ A × P . Q(x)}) = {z . t ∈ A , (∃ p ∈ P ∩ G . Q(⟨t, p⟩)) ∧ z = val(G, t) }
  using val_of_name by force

lemma val_only_names: val(F, τ) = val(F, {x ∈ τ . ∃ t ∈ domain(τ) . ∃ p ∈ F . x = ⟨t, p⟩})
  (is _ = val(F, ?name))
proof -
  have val(F, ?name) = {z . t ∈ domain(?name), (∃ p ∈ F . ⟨t, p⟩ ∈ ?name) ∧ z = val(F, t)}
  using def_val by blast
  also
  have ... = {val(F, t). t ∈ {y ∈ domain(τ) . ∃ p ∈ F . ⟨y, p⟩ ∈ τ }}
    by blast
  also
  have ... = {z . t ∈ domain(τ), (∃ p ∈ F . ⟨t, p⟩ ∈ τ) ∧ z = val(F, t)}
    by blast
  also
  have ... = val(F, τ)
    using def_val[symmetric] by blast
  finally
  show ?thesis ..
qed

lemma val_only_pairs: val(F, τ) = val(F, {x ∈ τ . ∃ t p . x = ⟨t, p⟩})
proof
  have val(F, τ) = val(F, {x ∈ τ . ∃ t ∈ domain(τ) . ∃ p ∈ F . x = ⟨t, p⟩}) (is _ = val(F, ?name))
    using val_only_names .
  also
  have ... ⊆ val(F, {x ∈ τ . ∃ t p . x = ⟨t, p⟩})
    using val_mono[of ?name {x ∈ τ . ∃ t p . x = ⟨t, p⟩}] by auto
  finally
  show val(F, τ) ⊆ val(F, {x ∈ τ . ∃ t p . x = ⟨t, p⟩}) by simp
next
  show val(F, {x ∈ τ . ∃ t p . x = ⟨t, p⟩}) ⊆ val(F, τ)
    using val_mono[of {x ∈ τ . ∃ t p . x = ⟨t, p⟩}] by auto
qed

lemma val_subset_domain_times_range: val(F, τ) ⊆ val(F, domain(τ) × range(τ))
  using val_only_pairs[THEN equalityD1]
  val_mono[of {x ∈ τ . ∃ t p . x = ⟨t, p⟩} domain(τ) × range(τ)] by blast

lemma val_of_elem: ⟨∅, p⟩ ∈ π ⇒ p ∈ G ⇒ val(G, ∅) ∈ val(G, π)
proof -

```

```

assume  $\langle \vartheta, p \rangle \in \pi$ 
then
have  $\vartheta \in \text{domain}(\pi)$ 
  by auto
assume  $p \in G$ 
with  $\langle \vartheta \in \text{domain}(\pi) \rangle \langle \langle \vartheta, p \rangle \in \pi \rangle$ 
have  $\text{val}(G, \vartheta) \in \{z . t \in \text{domain}(\pi) , (\exists p \in G . \langle t, p \rangle \in \pi) \wedge z = \text{val}(G, t)\}$ 
  by auto
then
show ?thesis
  by (subst def_val)
qed

```

```

lemma elem_of_val:  $x \in \text{val}(G, \pi) \implies \exists \vartheta \in \text{domain}(\pi). \text{val}(G, \vartheta) = x$ 
  by (subst (asm) def_val, auto)

```

```

lemma elem_of_val_pair:  $x \in \text{val}(G, \pi) \implies \exists \vartheta. \exists p \in G. \langle \vartheta, p \rangle \in \pi \wedge \text{val}(G, \vartheta) = x$ 
  by (subst (asm) def_val, auto)

```

```

lemma elem_of_val_pair':
  assumes  $\pi \in M \ x \in \text{val}(G, \pi)$ 
  shows  $\exists \vartheta \in M. \exists p \in G. \langle \vartheta, p \rangle \in \pi \wedge \text{val}(G, \vartheta) = x$ 
proof -
  from assms
  obtain  $\vartheta \ p$  where  $p \in G \ \langle \vartheta, p \rangle \in \pi \ \text{val}(G, \vartheta) = x$ 
    using elem_of_val_pair by blast
  moreover from this  $\langle \pi \in M \rangle$ 
  have  $\vartheta \in M$ 
    using pair_in_M_iff[THEN iffD1, THEN conjunct1, simplified]
    transitivity by blast
  ultimately
  show ?thesis
    by blast
qed

```

```

lemma GenExtD:  $x \in M[G] \implies \exists \tau \in M. x = \text{val}(G, \tau)$ 
  by (simp add: GenExt_def)

```

```

lemma GenExtI:  $x \in M \implies \text{val}(G, x) \in M[G]$ 
  by (auto simp add: GenExt_def)

```

```

lemma Transset_MG : Transset( $M[G]$ )
proof -
  { fix  $vc \ y$ 
    assume  $vc \in M[G]$  and  $y \in vc$ 
    then
    obtain  $c$  where  $c \in M \ \text{val}(G, c) \in M[G] \ y \in \text{val}(G, c)$ 
      using GenExtD by auto
    from  $\langle y \in \text{val}(G, c) \rangle$ 
  }

```

```

obtain  $\vartheta$  where  $\vartheta \in \text{domain}(c)$   $\text{val}(G, \vartheta) = y$ 
  using elem_of_val by blast
with trans_M  $\langle c \in M \rangle$ 
have  $y \in M[G]$ 
  using domain_trans GenExtI by blast
}
then
show ?thesis
  using Transset_def by auto
qed

```

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  $n \in \text{nat}$ 
  shows  $\text{check}(n) \in M$ 
  using assms
proof (induct  $n$ )
  case 0
  then
  show ?case
    using zero_in_M by (subst def_check_simp)
next
  case (succ  $x$ )
  have  $1 \in M$ 
    using one_in_P P_sub_M subsetD by simp
  with  $\langle \text{check}(x) \in M \rangle$ 
  have  $\langle \text{check}(x), 1 \rangle \in M$ 
    using pair_in_M_iff by simp
  then
  have  $\{ \langle \text{check}(x), 1 \rangle \} \in M$ 
    using singleton_closed by simp
  with  $\langle \text{check}(x) \in M \rangle$ 
  have  $\text{check}(x) \cup \{ \langle \text{check}(x), 1 \rangle \} \in M$ 
    using Un_closed by simp
  then
  show ?case
    using  $\langle x \in \text{nat} \rangle$  def_checkS by simp
qed

```

```

lemma def_PHcheck:
  assumes
     $z \in M$   $f \in M$ 
  shows
     $H\text{check}(z, f) = \text{Replace}(z, PH\text{check}(\#\#M, 1, f))$ 
proof -
  from assms

```

```

have  $\langle f'x, \mathbf{1} \rangle \in M \text{ } f'x \in M$  if  $x \in z$  for  $x$ 
  using pair_in_M_iff_transitivity that apply_closed by simp_all
then
have  $\{y . x \in z, y = \langle f'x, \mathbf{1} \rangle\} = \{y . x \in z, y = \langle f'x, \mathbf{1} \rangle \wedge y \in M \wedge f'x \in M\}$ 
  by simp
then
show ?thesis
  using  $\langle z \in M \rangle \langle f \in M \rangle$  transitivity
  unfolding Hcheck_def PHcheck_def RepFun_def
  by auto
qed

```

```

lemma wfrec_Hcheck :
  assumes  $X \in M$ 
  shows wfrec_replacement( $\#\#M, is\_Hcheck(\#\#M, \mathbf{1}), rcheck(X)$ )
proof -
  let  $?f = Exists(And(pair\_fm(1, 0, 2),$ 
     $is\_wfrec\_fm(is\_Hcheck\_fm(8, 2, 1, 0), 4, 1, 0)))$ 
  have  $is\_Hcheck(\#\#M, \mathbf{1}, a, b, c) \longleftrightarrow$ 
     $sats(M, is\_Hcheck\_fm(8, 2, 1, 0), [c, b, a, d, e, y, x, z, \mathbf{1}, rcheck(x)])$ 
    if  $a \in M \ b \in M \ c \in M \ d \in M \ e \in M \ y \in M \ x \in M \ z \in M$ 
    for  $a \ b \ c \ d \ e \ y \ x \ z$ 
    using that  $\langle X \in M \rangle$  rcheck_in_M is_Hcheck_iff_sats zero_in_M
    by simp
  then
  have  $sats(M, is\_wfrec\_fm(is\_Hcheck\_fm(8, 2, 1, 0), 4, 1, 0), [y, x, z, \mathbf{1}, rcheck(X)])$ 
     $\longleftrightarrow is\_wfrec(\#\#M, is\_Hcheck(\#\#M, \mathbf{1}), rcheck(X), x, y)$ 
    if  $x \in M \ y \in M \ z \in M$  for  $x \ y \ z$ 
    using that sats_is_wfrec_fm  $\langle X \in M \rangle$  rcheck_in_M zero_in_M
    by simp
  moreover from this
  have  $satsf : sats(M, ?f, [x, z, \mathbf{1}, rcheck(X)]) \longleftrightarrow$ 
     $(\exists y \in M. pair(\#\#M, x, y, z) \ \& \ is\_wfrec(\#\#M, is\_Hcheck(\#\#M, \mathbf{1}), rcheck(X),$ 
 $x, y))$ 
    if  $x \in M \ z \in M$  for  $x \ z$ 
    using that  $\langle X \in M \rangle$  rcheck_in_M
    by (simp del:pair_abs)
  moreover
  have  $arityf : arity(?f) = 4$ 
    using arity_wfrec_replacement_fm [where  $p = is\_Hcheck\_fm(8, 2, 1, 0)$  and
 $i = 9$ ]
     $arity\_is\_Hcheck\_fm$  ord_simp_union
    by simp
  ultimately
  have strong_replacement( $\#\#M, \lambda x z. sats(M, ?f, [x, z, \mathbf{1}, rcheck(X)])$ )
    using ZF_ground_replacements(2) arityf  $\langle X \in M \rangle$  rcheck_in_M
    unfolding replacement_assm_def wfrec_Hcheck_fm_def by simp
  then

```

```

have strong_replacement(##M,λx z.
  ∃ y∈M. pair(##M,x,y,z) & is_wfrec(##M, is_Hcheck(##M,1),rcheck(X),
x, y))
  using repl_sats[of M ?f [1,rcheck(X)]] satsf by (simp del:pair_abs)
then
show ?thesis
  unfolding wfrec_replacement_def by simp
qed

```

```

lemma Hcheck_closed' : f∈M ⇒ z∈M ⇒ {f ' x . x ∈ z} ∈ M
  using RepFun_closed[OF lam_replacement_imp_strong_replacement]
  lam_replacement_apply_apply_closed transM[of _ z]
by simp

```

```

lemma repl_PHcheck :
  assumes f∈M
  shows lam_replacement(##M,λx. Hcheck(x,f))
proof -
  have Hcheck(x,f) = {f'y . y∈x}×{1} for x
    unfolding Hcheck_def by auto
  moreover
  note assms
  moreover from this
  have 1:lam_replacement(##M, λx . {f'y . y∈x}×{1})
    using lam_replacement_RepFun_apply
    lam_replacement_constant lam_replacement_fst lam_replacement_snd
    singleton_closed cartprod_closed fst_snd_closed Hcheck_closed'
  by (rule_tac lam_replacement_CartProd[THEN [5] lam_replacement_hcomp2],simp_all)
  ultimately
  show ?thesis
    using singleton_closed cartprod_closed Hcheck_closed'
    by(rule_tac lam_replacement_cong[OF 1],auto)
qed

```

```

lemma univ_PHcheck : [ z∈M ; f∈M ] ⇒ univalent(##M,z,PHcheck(##M,1,f))
  unfolding univalent_def PHcheck_def
  by simp

```

```

lemma PHcheck_closed : [ z∈M ; f∈M ; x∈z; PHcheck(##M,1,f,x,y) ] ⇒
(##M)(y)
  unfolding PHcheck_def by simp

```

```

lemma relation2_Hcheck : relation2(##M,is_Hcheck(##M,1),Hcheck)
proof -
  have is_Replace(##M,z,PHcheck(##M,1,f),hc) ↔ hc = Replace(z,PHcheck(##M,1,f))
    if z∈M f∈M hc∈M for z f hc
    using that Replace_abs[OF _ _ univ_PHcheck] PHcheck_closed[of z f]
    by simp
  with def_PHcheck

```

show *?thesis*
unfolding *relation2_def is_Hcheck_def Hcheck_def*
by *simp*
qed

lemma *Hcheck_closed* : $\forall y \in M. \forall g \in M. Hcheck(y,g) \in M$
proof -
have *eq:Hcheck(x,f) = {f'y . y ∈ x} × {1}* **for** *f x*
unfolding *Hcheck_def* **by** *auto*
then
have *Hcheck(y,g) ∈ M* **if** *y ∈ M g ∈ M* **for** *y g*
using *eq that Hcheck_closed' cartprod_closed singleton_closed*
by *simp*
then
show *?thesis*
by *auto*
qed

lemma *wf_rcheck* : $x \in M \implies wf(rcheck(x))$
unfolding *rcheck_def* **using** *wf_trancl[OF wf_Memrel]* .

lemma *trans_rcheck* : $x \in M \implies trans(rcheck(x))$
unfolding *rcheck_def* **using** *trans_trancl* .

lemma *relation_rcheck* : $x \in M \implies relation(rcheck(x))$
unfolding *rcheck_def* **using** *relation_trancl* .

lemma *check_in_M* : $x \in M \implies check(x) \in M$
using *wfrec_Hcheck[of x] check_trancl wf_rcheck trans_rcheck relation_rcheck*
rcheck_in_M
Hcheck_closed relation2_Hcheck trans_wfrec_closed[of rcheck(x)]
by *simp*

lemma *rcheck_abs[Rel]* : $\llbracket x \in M ; r \in M \rrbracket \implies is_rcheck(\#\#M,x,r) \longleftrightarrow r = rcheck(x)$
unfolding *rcheck_def is_rcheck_def*
using *singleton_closed trancl_closed Memrel_closed eclose_closed zero_in_M*
by *simp*

lemma *check_abs[Rel]* :
assumes *x ∈ M z ∈ M*
shows *is_check(\#\#M,1,x,z) ⟷ z = check(x)*
proof -
have *is_check(\#\#M,1,x,z) ⟷ is_wfrec(\#\#M,is_Hcheck(\#\#M,1),rcheck(x),x,z)*
unfolding *is_check_def*
using *assms rcheck_abs rcheck_in_M zero_in_M*
unfolding *check_trancl is_check_def*
by *simp*

```

then
show ?thesis
  unfolding check_trancl
  using assms wfrec_Hcheck[of x] wf_rcheck trans_rcheck relation_rcheck rcheck_in_M
    Hcheck_closed relation2_Hcheck trans_wfrec_abs[of rcheck(x) x z is_Hcheck(##M,1)
Hcheck]
  by (simp flip: setclass_iff)
qed

```

lemma check_lam_replacement: lam_replacement(##M,check)

proof -

```

  have arity(check_fm(2,0,1)) = 3
    by (simp add:ord_simp_union arity)
  then
  have Lambda(A, check) ∈ M if A ∈ M for A
    using that check_in_M transitivity[of _ A]
      sats_check_fm check_abs zero_in_M
      check_fm_type ZF_ground_replacements(3)
    by(rule_tac Lambda_in_M [of check_fm(2,0,1) [1]],simp_all)
  then
  show ?thesis
    using check_in_M lam_replacement_iff_lam_closed[THEN iffD2]
    by simp
qed

```

lemma check_replacement: {check(x). x ∈ ℙ} ∈ M

```

using lam_replacement_imp_strong_replacement_aux[OF check_lam_replacement]
  transitivity check_in_M RepFun_closed
by simp_all

```

lemma M_subset_MG : 1 ∈ G ⇒ M ⊆ M[G]

```

using check_in_M GenExtI
by (intro subsetI, subst val_check [of G,symmetric], auto)

```

The name for the generic filter

definition

```

G_dot :: i where
G_dot ≡ {⟨check(p),p⟩ . p ∈ ℙ}

```

lemma G_dot_in_M : G_dot ∈ M

```

using lam_replacement_Pair[THEN [5] lam_replacement_hcomp2,OF
  check_lam_replacement lam_replacement_identity]
  check_in_M lam_replacement_imp_strong_replacement_aux
  transitivity check_in_M RepFun_closed pair_in_M_iff
unfolding G_dot_def
by simp

```

lemma zero_in_MG : 0 ∈ M[G]

proof -

```

have 0 = val(G,0)
  using zero_in_M elem_of_val by auto
also
have ... ∈ M[G]
  using GenExtI zero_in_M by simp
finally
show ?thesis .
qed

declare check_in_M [simp,intro]

end — forcing_data1

context G_generic1
begin

lemma val_G_dot : val(G,G_dot) = G
proof (intro equalityI subsetI)
  fix x
  assume x ∈ val(G,G_dot)
  then obtain ∅ p where p ∈ G ⟨∅,p⟩ ∈ G_dot val(G,∅) = x ∅ = check(p)
    unfolding G_dot_def using elem_of_val_pair G_dot_in_M
    by force
  then
  show x ∈ G
    using G_subset_P one_in_G val_check P_sub_M by auto
next
  fix p
  assume p ∈ G
  have ⟨check(q),q⟩ ∈ G_dot if q ∈ P for q
    unfolding G_dot_def using that by simp
  with ⟨p ∈ G⟩
  have val(G,check(p)) ∈ val(G,G_dot)
    using val_of_elem G_dot_in_M by blast
  with ⟨p ∈ G⟩
  show p ∈ val(G,G_dot)
    using one_in_G G_subset_P P_sub_M val_check by auto
qed

lemma G_in_Gen_Ext : G ∈ M[G]
  using G_subset_P one_in_G val_G_dot GenExtI[of _ G] G_dot_in_M
  by force

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]
  using generic_transitivity_MG zero_in_MG
  by unfold_locales force+

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
     $\text{separation}(\##M, \lambda p. (p \Vdash \varphi \text{ env}))$ 
  using separation_ax arity_forces far fty envty arity_forces_le
    transitivity[of _  $\mathbb{P}$ ]
  by simp

```

```

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
     $\{p \in \mathbb{P} . p \Vdash \varphi \text{ env}\} \in M$ 
  using assms separation_forces separation_closed
  by simp

```

```

lemma forces_mem_iff_dense_below:  $p \in \mathbb{P} \implies p \text{ forces}_a (t1 \in t2) \iff \text{dense\_below}(\{q \in \mathbb{P} . \exists s. \exists r. r \in \mathbb{P} \wedge \langle s, r \rangle \in t2 \wedge q \preceq r \wedge q \text{ forces}_a (t1 = s)\}, p)$ 

```

using *def_forces_mem*[of p $t1$ $t2$] by *blast*

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 *forces* _{a} ($t1 = t2$)
shows r *forces* _{a} ($t1 = t2$)
using *assms* *def_forces_eq*[of $_$ $t1$ $t2$] *leq_transD* by *blast*

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 *forces* _{a} ($t1 \in t2$)
shows r *forces* _{a} ($t1 \in t2$)
using *assms* *forces_mem_iff_dense_below* *dense_below_under* by *auto*

13.4 Kunen 2013, Lemma IV.2.37(b)

lemma *density_mem*:
assumes $p \in \mathbb{P}$
shows p *forces* _{a} ($t1 \in t2$) \longleftrightarrow *dense_below*($\{q \in \mathbb{P}. q$ *forces* _{a} ($t1 \in t2$) $\}, p$)
proof
assume p *forces* _{a} ($t1 \in t2$)
with *assms*
show *dense_below*($\{q \in \mathbb{P}. q$ *forces* _{a} ($t1 \in t2$) $\}, p$)
using *forces_mem_iff_dense_below* *strengthening_mem*[of p] *ideal_dense_below*
by *auto*
next
assume *dense_below*($\{q \in \mathbb{P}. q$ *forces* _{a} ($t1 \in t2$) $\}, p$)
with *assms*
have *dense_below*($\{q \in \mathbb{P}.$
dense_below($\{q' \in \mathbb{P}. \exists s r. r \in \mathbb{P} \wedge \langle s, r \rangle \in t2 \wedge q' \preceq r \wedge q'$ *forces* _{a} ($t1 = s$) $\}, q$
 $\}, p$)
using *forces_mem_iff_dense_below* by *simp*
with *assms*
show p *forces* _{a} ($t1 \in t2$)
using *dense_below_dense_below* *forces_mem_iff_dense_below*[of p $t1$ $t2$] by
blast
qed

lemma *aux_density_eq*:
assumes
dense_below(
 $\{q' \in \mathbb{P}. \forall q. q \in \mathbb{P} \wedge q \preceq q' \longrightarrow q$ *forces* _{a} ($s \in t1$) \longleftrightarrow q *forces* _{a} ($s \in t2$) $\}$
 $.p$)
 q *forces* _{a} ($s \in t1$) $q \in \mathbb{P}$ $p \in \mathbb{P}$ $q \preceq p$
shows
dense_below($\{r \in \mathbb{P}. r$ *forces* _{a} ($s \in t2$) $\}, q$)
proof
fix r

```

assume  $r \in \mathbb{P}$   $r \preceq q$ 
moreover from this and  $\langle p \in \mathbb{P} \rangle \langle q \preceq p \rangle \langle q \in \mathbb{P} \rangle$ 
have  $r \preceq p$ 
  using leq_transD by simp
moreover
note  $\langle q \text{ forces}_a (s \in t1) \rangle \langle \text{dense\_below}(\_, p) \rangle \langle q \in \mathbb{P} \rangle$ 
ultimately
obtain  $q1$  where  $q1 \preceq r$   $q1 \in \mathbb{P}$   $q1 \text{ forces}_a (s \in t2)$ 
  using strengthening_mem[of  $q \_ s t1$ ] refl_leq leq_transD[of  $\_ r q$ ] by blast
then
show  $\exists d \in \{r \in \mathbb{P} . r \text{ forces}_a (s \in t2)\} . d \in \mathbb{P} \wedge d \preceq r$ 
  by blast
qed

```

lemma *density_eq*:

```

assumes  $p \in \mathbb{P}$ 
shows  $p \text{ forces}_a (t1 = t2) \iff \text{dense\_below}(\{q \in \mathbb{P} . q \text{ forces}_a (t1 = t2)\}, p)$ 
proof
assume  $p \text{ forces}_a (t1 = t2)$ 
with  $\langle p \in \mathbb{P} \rangle$ 
show  $\text{dense\_below}(\{q \in \mathbb{P} . q \text{ forces}_a (t1 = t2)\}, p)$ 
  using strengthening_eq_ideal_dense_below by auto
next
assume  $\text{dense\_below}(\{q \in \mathbb{P} . q \text{ forces}_a (t1 = t2)\}, p)$ 
{
  fix  $s q$ 
let  $?D1 = \{q' \in \mathbb{P} . \forall s \in \text{domain}(t1) \cup \text{domain}(t2) . \forall q . q \in \mathbb{P} \wedge q \preceq q' \implies$ 
     $q \text{ forces}_a (s \in t1) \iff q \text{ forces}_a (s \in t2)\}$ 
let  $?D2 = \{q' \in \mathbb{P} . \forall q . q \in \mathbb{P} \wedge q \preceq q' \implies q \text{ forces}_a (s \in t1) \iff q \text{ forces}_a (s \in t2)\}$ 
assume  $s \in \text{domain}(t1) \cup \text{domain}(t2)$ 
then
have  $?D1 \subseteq ?D2$  by blast
with  $\langle \text{dense\_below}(\_, p) \rangle$ 
have  $\text{dense\_below}(\{q' \in \mathbb{P} . \forall s \in \text{domain}(t1) \cup \text{domain}(t2) . \forall q . q \in \mathbb{P} \wedge q \preceq q' \implies$ 
     $q \text{ forces}_a (s \in t1) \iff q \text{ forces}_a (s \in t2)\}, p)$ 
  using dense_below_cong'[OF  $\langle p \in \mathbb{P} \rangle$  def_forces_eq[of  $\_ t1 t2$ ]] by simp
with  $\langle p \in \mathbb{P} \rangle \langle ?D1 \subseteq ?D2 \rangle$ 
have  $\text{dense\_below}(\{q' \in \mathbb{P} . \forall q . q \in \mathbb{P} \wedge q \preceq q' \implies$ 
     $q \text{ forces}_a (s \in t1) \iff q \text{ forces}_a (s \in t2)\}, p)$ 
  using dense_below_mono by simp
moreover from this
have  $\text{dense\_below}(\{q' \in \mathbb{P} . \forall q . q \in \mathbb{P} \wedge q \preceq q' \implies$ 
     $q \text{ forces}_a (s \in t2) \iff q \text{ forces}_a (s \in t1)\}, p)$ 
  by blast
moreover
assume  $q \in \mathbb{P}$   $q \preceq p$ 
moreover
note  $\langle p \in \mathbb{P} \rangle$ 

```

```

ultimately
have q forces_a (s ∈ t1) ⇒ dense_below({r ∈ ℙ. r forces_a (s ∈ t2)}, q)
  q forces_a (s ∈ t2) ⇒ dense_below({r ∈ ℙ. r forces_a (s ∈ t1)}, q)
  using aux_density_eq by simp_all
then
have q forces_a (s ∈ t1) ↔ q forces_a (s ∈ t2)
  using density_mem[OF ⟨q ∈ ℙ⟩] by blast
}
with ⟨p ∈ ℙ⟩
show p forces_a (t1 = t2) using def_forces_eq by blast
qed

```

13.5 Kunen 2013, Lemma IV.2.38

```

lemma not_forces_neq:
  assumes p ∈ ℙ
  shows p forces_a (t1 = t2) ↔ ¬ (∃ q ∈ ℙ. q ≤ p ∧ q forces_a (t1 ≠ t2))
  using assms density_eq unfolding forces_neq_def by blast

```

```

lemma not_forces_nmem:
  assumes p ∈ ℙ
  shows p forces_a (t1 ∈ t2) ↔ ¬ (∃ q ∈ ℙ. q ≤ p ∧ q forces_a (t1 ∉ t2))
  using assms density_mem unfolding forces_nmem_def by blast

```

13.6 The relation of forcing and atomic formulas

```

lemma Forces_Equal:
  assumes
    p ∈ ℙ t1 ∈ M t2 ∈ M env ∈ list(M) nth(n, env) = t1 nth(m, env) = t2 n ∈ nat m ∈ nat
  shows
    (p ⊩ Equal(n, m) env) ↔ p forces_a (t1 = t2)
  using assms sats_forces_Equal forces_eq_abs transitivity
  by simp

```

```

lemma Forces_Member:
  assumes
    p ∈ ℙ t1 ∈ M t2 ∈ M env ∈ list(M) nth(n, env) = t1 nth(m, env) = t2 n ∈ nat m ∈ nat
  shows
    (p ⊩ Member(n, m) env) ↔ p forces_a (t1 ∈ t2)
  using assms sats_forces_Member forces_mem_abs transitivity
  by simp

```

```

lemma Forces_Neg:
  assumes
    p ∈ ℙ env ∈ list(M) φ ∈ formula
  shows
    (p ⊩ Neg(φ) env) ↔ ¬ (∃ q ∈ M. q ∈ ℙ ∧ q ≤ p ∧ (q ⊩ φ env))
  using assms sats_forces_Neg transitivity pair_in_M_iff leq_abs
  by simp

```

13.7 The relation of forcing and connectives

lemma *Forces_Nand*:

assumes

$p \in \mathbb{P}$ $env \in list(M)$ $\varphi \in formula$ $\psi \in formula$

shows

$(p \Vdash Nand(\varphi, \psi) \ env) \longleftrightarrow \neg(\exists q \in M. q \in \mathbb{P} \wedge q \preceq p \wedge (q \Vdash \varphi \ env) \wedge (q \Vdash \psi \ env))$

using *assms sats_forces_Nand transitivity pair_in_M_iff leq_abs* **by** *simp*

lemma *Forces_And_aux*:

assumes

$p \in \mathbb{P}$ $env \in list(M)$ $\varphi \in formula$ $\psi \in formula$

shows

$p \Vdash And(\varphi, \psi) \ env \longleftrightarrow$

$(\forall q \in M. q \in \mathbb{P} \wedge q \preceq p \longrightarrow (\exists r \in M. r \in \mathbb{P} \wedge r \preceq q \wedge (r \Vdash \varphi \ env) \wedge (r \Vdash \psi \ env)))$

unfolding *And_def* **using** *assms Forces_Neg Forces_Nand* **by** (*auto simp only*):

lemma *Forces_And_iff_dense_below*:

assumes

$p \in \mathbb{P}$ $env \in list(M)$ $\varphi \in formula$ $\psi \in formula$

shows

$(p \Vdash And(\varphi, \psi) \ env) \longleftrightarrow dense_below(\{r \in \mathbb{P}. (r \Vdash \varphi \ env) \wedge (r \Vdash \psi \ env)\}, p)$

unfolding *dense_below_def* **using** *Forces_And_aux assms*

by (*auto dest:transitivity[OF _ P_in_M]; rename_tac q; drule_tac x=q in bspec*)⁺

lemma *Forces_Forall*:

assumes

$p \in \mathbb{P}$ $env \in list(M)$ $\varphi \in formula$

shows

$(p \Vdash Forall(\varphi) \ env) \longleftrightarrow (\forall x \in M. (p \Vdash \varphi \ ([x] \ @ \ env)))$

using *sats_forces_Forall assms transitivity[OF _ P_in_M]*

by *simp*

bundle *some_rules* = *elem_of_val_pair* [*dest*]

context

includes *some_rules*

begin

lemma *elem_of_valI*: $\exists \vartheta. \exists p \in \mathbb{P}. p \in G \wedge \langle \vartheta, p \rangle \in \pi \wedge val(G, \vartheta) = x \implies x \in val(G, \pi)$

by (*subst def_val, auto*)

lemma *GenExt_iff*: $x \in M[G] \longleftrightarrow (\exists \tau \in M. x = val(G, \tau))$

unfolding *GenExt_def* **by** *simp*

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 -

let $?D = \{q \in \mathbb{P}. p \perp q \vee q \in D\}$

have *dense*($?D$)

proof

fix r

assume $r \in \mathbb{P}$

show $\exists d \in \{q \in \mathbb{P}. p \perp q \vee q \in D\}. d \preceq r$

proof (cases $p \perp r$)

case *True*

with $\langle r \in \mathbb{P} \rangle$

show *?thesis* using *refl_leq*[of r] by (intro *bestI*) (*blast+*)

next

case *False*

then

obtain s where $s \in \mathbb{P}$ $s \preceq p$ $s \preceq r$ by *blast*

with *assms* $\langle r \in \mathbb{P} \rangle$

show *?thesis*

using *dense_belowD*[OF *assms*(2), of s] *leq_transD*[of $_ s r$]

by *blast*

qed

qed

have $?D \subseteq \mathbb{P}$ by *auto*

let $?d_fm = \dots \neg \text{compat_in_fm}(1, 2, 3, 0) \cdot \vee \cdot 0 \in 4 \cdot$

from $\langle p \in G \rangle$

have $p \in M$

using *G_subset_M subsetD* by *simp*

moreover

have $?d_fm \in \text{formula}$ by *simp*

moreover

have *arity*($?d_fm$) = 5

by (auto *simp* add: *arity*)

moreover from $\langle D \in M \rangle$ $\langle p \in M \rangle$

have $(M, [q, \mathbb{P}, \text{leq}, p, D] \models ?d_fm) \longleftrightarrow (\neg \text{is_compat_in}(\#\#M, \mathbb{P}, \text{leq}, p, q) \vee q \in D)$

if $q \in M$ for q

using *that_sats_compat_in_fm zero_in_M*

by *simp*

moreover from $\langle p \in M \rangle$

have $(\neg \text{is_compat_in}(\#\#M, \mathbb{P}, \text{leq}, p, q) \vee q \in D) \longleftrightarrow p \perp q \vee q \in D$ if $q \in M$ for q

unfolding *compat_def*

using *that_compat_in_abs*

```

  by simp
ultimately
have ?D∈M
  using Collect_in_M[of ?d_fm [ℙ,leq,p,D]] ‹D∈M›
  by simp
note asm = ‹dense(?D)› ‹?D⊆ℙ› ‹?D∈M›
obtain x where x∈G x∈?D
  using M_generic_denseD[OF asm]
  by force
moreover from this
have x∈D
  using M_generic_compatD[OF _ ‹p∈G›, of x refl_leq_compatI[of _ p x]
  by force
ultimately
show ?thesis by auto
qed

```

13.9 Auxiliary results for Lemma IV.2.40(a)

lemma (in forcing_data1) IV240a_mem_Collect:

```

assumes
  π∈M τ∈M
shows
  {q∈ℙ. ∃σ. ∃r. r∈ℙ ∧ ‹σ,r› ∈ τ ∧ q⊆r ∧ q forces_a (π = σ)}∈M
proof -
let ?rel_pred= λM x a1 a2 a3 a4. ∃σ[M]. ∃r[M]. ∃σr[M].
  r∈a1 ∧ pair(M,σ,r,σr) ∧ σr∈a4 ∧ is_leq(M,a2,x,r) ∧ is_forces_eq'(M,a1,a2,x,a3,σ)
let ?φ=Exists(Exists(Exists(And(Member(1,4),And(pair_fm(2,1,0),
  And(Member(0,7),And(is_leq_fm(5,3,1),forces_eq_fm(4,5,3,6,2))))))))))
have σ∈M ∧ r∈M if ‹σ, r› ∈ τ for σ r
  using that ‹τ∈M› pair_in_M_iff_transitivity[of ‹σ,r› τ] by simp
then
have ?rel_pred(##M,q,ℙ,leq,π,τ) ‹longleftrightarrow› (∃σ. ∃r. r∈ℙ ∧ ‹σ,r› ∈ τ ∧ q⊆r ∧ q
forces_a (π = σ))
  if q∈M for q
  unfolding forces_eq_def
  using assms that leq_abs forces_eq'_abs pair_in_M_iff
  by auto
moreover
have (M, [q,ℙ,leq,π,τ] ⊨ ?φ) ‹longleftrightarrow› ?rel_pred(##M,q,ℙ,leq,π,τ) if q∈M for q
  using assms that sats_forces_eq_fm sats_is_leq_fm zero_in_M
  by simp
moreover
have ?φ∈formula by simp
moreover
have arity(?φ)=5
  using arity_forces_eq_fm
  by (simp add:ord_simp_union arity)
ultimately

```

show *?thesis*
unfolding *forces_eq_def* **using** *assms Collect_in_M*[*of ?φ [P,leq,π,τ]*]
by *simp*
qed

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 (*intro elem_of_valI*)
let $?D = \{q \in \mathbb{P}. \exists \sigma. \exists r. r \in \mathbb{P} \wedge \langle \sigma, r \rangle \in \tau \wedge q \preceq r \wedge q \text{ forces}_a (\pi = \sigma)\}$
from $\langle p \in G \rangle$
have $p \in \mathbb{P}$ **by** *blast*
moreover
note $\langle \pi \in M \rangle \ \langle \tau \in M \rangle$
ultimately
have $?D \in M$ **using** *IV240a_mem_Collect* **by** *simp*
moreover from *assms* $\langle p \in \mathbb{P} \rangle$
have *dense_below*($?D, p$)
using *forces_mem_iff_dense_below* **by** *simp*
moreover
note $\langle p \in G \rangle$
ultimately
obtain q **where** $q \in G \ q \in ?D$
using *generic_inter_dense_below*[*of ?D p*] **by** *blast*
then
obtain $\sigma \ r$ **where** $r \in \mathbb{P} \ \langle \sigma, r \rangle \in \tau \ q \preceq r \ q \text{ forces}_a (\pi = \sigma)$ **by** *blast*
moreover from *this* **and** $\langle q \in G \rangle$ *assms*
have $r \in G \ \text{val}(G, \pi) = \text{val}(G, \sigma)$ **by** *blast+*
ultimately
show $\exists \sigma. \exists p \in \mathbb{P}. p \in G \wedge \langle \sigma, p \rangle \in \tau \wedge \text{val}(G, \sigma) = \text{val}(G, \pi)$ **by** *auto*
qed

lemma *refl_forces_eq*: $p \in \mathbb{P} \implies p \text{ forces}_a (x = x)$
using *def_forces_eq* **by** *simp*

lemma *forces_memI*: $\langle \sigma, r \rangle \in \tau \implies p \in \mathbb{P} \implies r \in \mathbb{P} \implies p \preceq r \implies p \text{ forces}_a (\sigma \in \tau)$
using *refl_forces_eq*[*of _ σ*] *leq_transD* *refl_leq*
by (*blast intro:forces_mem_iff_dense_below*[*THEN iffD2*])

lemma *IV240a_eq_1st_incl*:
includes *some_rules*
assumes

$p \in G$ p 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

$\text{val}(G, \tau) \subseteq \text{val}(G, \vartheta)$

proof

fix x

assume $x \in \text{val}(G, \tau)$

then

obtain σr **where** $\langle \sigma, r \rangle \in \tau$ $r \in G$ $\text{val}(G, \sigma) = x$ **by** *blast*

moreover from *this* **and** $\langle p \in G \rangle$

obtain q **where** $q \in G$ $q \preceq p$ $q \preceq r$ **by** *force*

moreover from *this* **and** $\langle p \in G \rangle$

have $q \in \mathbb{P}$ $p \in \mathbb{P}$ **by** *blast+*

moreover from *calculation*

have $q \text{ forces}_a (\sigma \in \tau)$

using *forces_memI* **by** *auto*

moreover

note $\langle p \text{ forces}_a (\tau = \vartheta) \rangle$

ultimately

have $q \text{ forces}_a (\sigma \in \vartheta)$

using *def_forces_eq* **by** *auto*

with $\langle q \in \mathbb{P} \rangle$ $\langle q \in G \rangle$ *IH*[*of* $q \sigma$] $\langle \langle \sigma, r \rangle \in \tau \rangle$ $\langle \text{val}(G, \sigma) = x \rangle$

show $x \in \text{val}(G, \vartheta)$ **by** *blast*

qed

lemma *IV240a_eq_2nd_incl*:

includes *some_rules*

assumes

$p \in G$ p 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

fix x

assume $x \in \text{val}(G, \vartheta)$

then

obtain σr **where** $\langle \sigma, r \rangle \in \vartheta$ $r \in G$ $\text{val}(G, \sigma) = x$ **by** *blast*

moreover from *this* **and** $\langle p \in G \rangle$

obtain q **where** $q \in G$ $q \preceq p$ $q \preceq r$ **by** *force*

moreover from *this* **and** $\langle p \in G \rangle$

```

have  $q \in \mathbb{P}$   $p \in \mathbb{P}$  by blast+
moreover from calculation
have  $q$  forcesa  $(\sigma \in \vartheta)$ 
  using forces_memI by auto
moreover
note  $\langle p$  forcesa  $(\tau = \vartheta) \rangle$ 
ultimately
have  $q$  forcesa  $(\sigma \in \tau)$ 
  using def_forces_eq by auto
with  $\langle q \in \mathbb{P} \rangle$   $\langle q \in G \rangle$  IH[of  $q$   $\sigma$ ]  $\langle \langle \sigma, r \rangle \in \vartheta \rangle$   $\langle \text{val}(G, \sigma) = x \rangle$ 
show  $x \in \text{val}(G, \tau)$  by blast
qed

```

lemma *IV240a_eq*:

```

includes some_rules
assumes
   $p \in G$   $p$  forcesa  $(\tau = \vartheta)$ 
and
  IH:  $\bigwedge q \sigma. q \in \mathbb{P} \implies q \in G \implies \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies$ 
     $(q$  forcesa  $(\sigma \in \tau) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \tau)) \wedge$ 
     $(q$  forcesa  $(\sigma \in \vartheta) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \vartheta))$ 
shows
   $\text{val}(G, \tau) = \text{val}(G, \vartheta)$ 
using IV240a_eq_1st_incl[OF assms] IV240a_eq_2nd_incl[OF assms] IH by
blast

```

13.10 Induction on names

lemma (*in forcing_data1*) *core_induction*:

```

assumes
   $\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in \text{domain}(\vartheta) \rrbracket \implies Q(0, \tau, \sigma, q) \rrbracket \implies Q(1, \tau, \vartheta, p)$ 
   $\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \rrbracket \implies Q(1, \sigma, \tau, q) \rrbracket$ 
 $\wedge Q(1, \sigma, \vartheta, q) \rrbracket \implies Q(0, \tau, \vartheta, p)$ 
   $ft \in \mathcal{2}$   $p \in \mathbb{P}$ 
shows
   $Q(ft, \tau, \vartheta, p)$ 
proof -
  {
    fix  $ft$   $p$   $\tau$   $\vartheta$ 
    have Transset(eclose( $\{\tau, \vartheta\}$ )) (is Transset( $?e$ ))
      using Transset_eclose by simp
    have  $\tau \in ?e$   $\vartheta \in ?e$ 
      using arg_into_eclose by simp_all
    moreover
    assume  $ft \in \mathcal{2}$   $p \in \mathbb{P}$ 
    ultimately
    have  $\langle ft, \tau, \vartheta, p \rangle \in \mathcal{2} \times ?e \times ?e \times \mathbb{P}$  (is  $?a \in \mathcal{2} \times ?e \times ?e \times \mathbb{P}$ ) by simp
    then

```

```

    have  $Q(ftype(?a), name1(?a), name2(?a), cond\_of(?a))$ 
    using  $core\_induction\_aux[of ?e \mathbb{P} Q ?a, OF \langle Transset(?e) \rangle assms(1,2) \langle ?a \in \_ \rangle]$ 
      by  $(clarify) (blast)$ 
    then have  $Q(ft, \tau, \vartheta, p)$  by  $(simp \text{ add:components\_simp})$ 
  }
  then show  $?thesis$  using  $assms$  by  $simp$ 
qed

```

lemma (in $forcing_data1$) $forces_induction_with_conds$:

```

assumes
   $\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in domain(\vartheta) \rrbracket \implies Q(q, \tau, \sigma) \rrbracket \implies R(p, \tau, \vartheta)$ 
   $\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in domain(\tau) \cup domain(\vartheta) \rrbracket \implies R(q, \sigma, \tau) \wedge$ 
 $R(q, \sigma, \vartheta) \rrbracket \implies Q(p, \tau, \vartheta)$ 
   $p \in \mathbb{P}$ 
shows
   $Q(p, \tau, \vartheta) \wedge R(p, \tau, \vartheta)$ 
proof -
  let  $?Q = \lambda ft \tau \vartheta p. (ft = 0 \longrightarrow Q(p, \tau, \vartheta)) \wedge (ft = 1 \longrightarrow R(p, \tau, \vartheta))$ 
  from  $assms(1)$ 
  have  $\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in domain(\vartheta) \rrbracket \implies ?Q(0, \tau, \sigma, q) \rrbracket \implies$ 
 $?Q(1, \tau, \vartheta, p)$ 
    by  $simp$ 
  moreover from  $assms(2)$ 
  have  $\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in domain(\tau) \cup domain(\vartheta) \rrbracket \implies$ 
 $?Q(1, \sigma, \tau, q) \wedge ?Q(1, \sigma, \vartheta, q) \rrbracket \implies ?Q(0, \tau, \vartheta, p)$ 
    by  $simp$ 
  moreover
  note  $\langle p \in \mathbb{P} \rangle$ 
  ultimately
  have  $?Q(ft, \tau, \vartheta, p)$  if  $ft \in 2$  for  $ft$ 
    by  $(rule \text{ core\_induction}[OF \_ \_ \text{that, of } ?Q])$ 
  then
  show  $?thesis$  by  $auto$ 
qed

```

lemma (in $forcing_data1$) $forces_induction$:

```

assumes
   $\bigwedge \tau \vartheta. \llbracket \bigwedge \sigma. \sigma \in domain(\vartheta) \implies Q(\tau, \sigma) \rrbracket \implies R(\tau, \vartheta)$ 
   $\bigwedge \tau \vartheta. \llbracket \bigwedge \sigma. \sigma \in domain(\tau) \cup domain(\vartheta) \implies R(\sigma, \tau) \wedge R(\sigma, \vartheta) \rrbracket \implies Q(\tau, \vartheta)$ 
shows
   $Q(\tau, \vartheta) \wedge R(\tau, \vartheta)$ 
proof (intro  $forces\_induction\_with\_conds[OF \_ \_ \text{one\_in\_P}]$ )
  fix  $\tau \vartheta p$ 
  assume  $q \in \mathbb{P} \implies \sigma \in domain(\vartheta) \implies Q(\tau, \sigma)$  for  $q \sigma$ 
  with  $assms(1)$ 
  show  $R(\tau, \vartheta)$ 
    using  $\text{one\_in\_P}$  by  $simp$ 
next
  fix  $\tau \vartheta p$ 

```

assume $q \in \mathbb{P} \implies \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies R(\sigma, \tau) \wedge R(\sigma, \vartheta)$ **for** $q \sigma$
with $\text{assms}(2)$
show $Q(\tau, \vartheta)$
using $\text{one_in_}P$ **by** simp
qed

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 $?Q(\tau, \vartheta) \wedge ?R(\tau, \vartheta)$

proof ($\text{intro forces_induction[of } ?Q \text{ ?R] impI}$)

fix $\tau \vartheta$

assume $\tau \in M \vartheta \in M \sigma \in \text{domain}(\vartheta) \implies ?Q(\tau, \sigma)$ **for** σ

moreover from this

have $\sigma \in \text{domain}(\vartheta) \implies q \text{ forces}_a (\tau = \sigma) \implies \text{val}(G, \tau) = \text{val}(G, \sigma)$

if $q \in \mathbb{P} q \in G$ **for** $q \sigma$

using $\text{that domain_closed[of } \vartheta] \text{ transitivity by auto}$

ultimately

show $\forall p \in G. p \text{ forces}_a (\tau \in \vartheta) \longrightarrow \text{val}(G, \tau) \in \text{val}(G, \vartheta)$

using $IV240a_mem \text{ domain_closed transitivity by simp}$

next

fix $\tau \vartheta$

assume $\tau \in M \vartheta \in M$ **and** $d: \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies ?R(\sigma, \tau) \wedge ?R(\sigma, \vartheta)$

for σ

moreover from this

have $IH^!:(q \text{ forces}_a (\sigma \in \tau) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \tau)) \wedge$

$(q \text{ forces}_a (\sigma \in \vartheta) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \vartheta))$

if $\sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) q \in G$ **for** $q \sigma$

proof -

from d **that**

have $A: ?R(\sigma, \tau) \ ?R(\sigma, \vartheta)$

by auto

from $\langle \tau \in _ \rangle \langle \vartheta \in M \rangle \langle q \in G \rangle \langle \sigma \in _ \rangle$

show $?thesis$

using $\text{transitivity[of } \sigma] \text{ domain_closed } A[\text{rule_format, of } q]$

by auto

qed

show $\forall p \in G. p \text{ forces}_a (\tau = \vartheta) \longrightarrow \text{val}(G, \tau) = \text{val}(G, \vartheta)$

using $IV240a_eq[OF _ _ IH^!]$ **by** simp

qed

13.12 Lemma IV.2.40(b)

lemma $IV240b_mem$:

includes some_rules

assumes

$\text{val}(G, \pi) \in \text{val}(G, \tau) \ \pi \in M \ \tau \in M$

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
 $\exists p \in G. p \text{ forces}_a (\pi \in \tau)$
proof -
from $\langle \text{val}(G, \pi) \in \text{val}(G, \tau) \rangle$
obtain $\sigma \ r$ **where** $r \in G \ \langle \sigma, r \rangle \in \tau \ \text{val}(G, \pi) = \text{val}(G, \sigma)$ **by** *auto*
moreover from *this* **and** *IH*
obtain p' **where** $p' \in G \ p' \text{ forces}_a (\pi = \sigma)$ **by** *blast*
ultimately
obtain p **where** $p \preceq r \ p \preceq p' \ p \in G \ p \text{ forces}_a (\pi = \sigma)$
using *M_generic_compatD strengthening_eq[of p'] M_genericD* **by** *auto*
moreover from *calculation*
have $q \text{ forces}_a (\pi = \sigma)$ **if** $q \in \mathbb{P} \ q \preceq p$ **for** q
using *that strengthening_eq* **by** *blast*
moreover
note $\langle \sigma, r \rangle \in \tau \ \langle r \in G \rangle$
ultimately
have $r \in \mathbb{P} \wedge \langle \sigma, r \rangle \in \tau \wedge q \preceq r \wedge q \text{ forces}_a (\pi = \sigma)$ **if** $q \in \mathbb{P} \ q \preceq p$ **for** q
using *that leq_transD[of _ p r]* **by** *blast*
then
have *dense_below* $(\{q \in \mathbb{P}. \exists s \ r. r \in \mathbb{P} \wedge \langle s, r \rangle \in \tau \wedge q \preceq r \wedge q \text{ forces}_a (\pi = s)\}, p)$
using *refl_leq* **by** *blast*
moreover
note $\langle p \in G \rangle$
moreover from *calculation*
have $p \text{ forces}_a (\pi \in \tau)$
using *forces_mem_iff_dense_below* **by** *blast*
ultimately
show *?thesis* **by** *blast*
qed
end — *G_generic1*

context *forcing_data1*
begin

lemma *Collect_forces_eq_in_M*:
assumes $\tau \in M \ \vartheta \in M$
shows $\{p \in \mathbb{P}. p \text{ forces}_a (\tau = \vartheta)\} \in M$
using *assms Collect_in_M[of forces_eq_fm(1,2,0,3,4) [P,leq,tau,vartheta]]*
arity_forces_eq_fm sats_forces_eq_fm forces_eq_abs forces_eq_fm_type
by (*simp add: union_abs1 Un_commute*)

lemma *IV240b_eq_Collects*:
assumes $\tau \in M \ \vartheta \in M$
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**

$\{p \in \mathbb{P}. \exists \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta). p \text{ forces}_a (\sigma \notin \tau) \wedge p \text{ forces}_a (\sigma \in \vartheta)\} \in M$
proof -
let $?rel_pred = \lambda M x a1 a2 a3 a4.$
 $\exists \sigma[M]. \exists u[M]. \exists da3[M]. \exists da4[M]. is_domain(M, a3, da3) \wedge is_domain(M, a4, da4)$
 \wedge
 $union(M, da3, da4, u) \wedge \sigma \in u \wedge is_forces_mem'(M, a1, a2, x, \sigma, a3) \wedge$
 $is_forces_nmem'(M, a1, a2, x, \sigma, a4)$
let $? \varphi = Exists(Exists(Exists(Exists(And(domain_fm(7, 1), And(domain_fm(8, 0),$
 $And(union_fm(1, 0, 2), And(Member(3, 2), And(forces_mem_fm(5, 6, 4, 3, 7),$
 $forces_nmem_fm(5, 6, 4, 3, 8))))))))))$
have $1: \sigma \in M$ **if** $\langle \sigma, y \rangle \in \delta$ $\delta \in M$ **for** $\sigma \delta y$
using *that pair in M iff transitivity[of $\langle \sigma, y \rangle \delta$] by simp*
have $abs1: ?rel_pred(\#\#M, p, \mathbb{P}, leq, \tau, \vartheta) \longleftrightarrow$
 $(\exists \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta). forces_mem'(\mathbb{P}, leq, p, \sigma, \tau) \wedge forces_nmem'(\mathbb{P}, leq, p, \sigma, \vartheta))$
if $p \in M$ **for** p
unfolding *forces_mem_def forces_nmem_def*
using *assms that forces_mem'_abs forces_nmem'_abs*
 $domain_closed Un_closed$
by *(auto simp add: 1[of _ _ \tau] 1[of _ _ \vartheta])*
have $abs2: ?rel_pred(\#\#M, p, \mathbb{P}, leq, \vartheta, \tau) \longleftrightarrow (\exists \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta).$
 $forces_nmem'(\mathbb{P}, leq, p, \sigma, \tau) \wedge forces_mem'(\mathbb{P}, leq, p, \sigma, \vartheta))$ **if** $p \in M$ **for** p
unfolding *forces_mem_def forces_nmem_def*
using *assms that forces_mem'_abs forces_nmem'_abs*
 $domain_closed Un_closed$
by *(auto simp add: 1[of _ _ \tau] 1[of _ _ \vartheta])*
have $fsats1: (M, [p, \mathbb{P}, leq, \tau, \vartheta] \models ? \varphi) \longleftrightarrow ?rel_pred(\#\#M, p, \mathbb{P}, leq, \tau, \vartheta)$ **if** $p \in M$ **for**
 p
using *that assms sats_forces_mem_fm sats_forces_nmem_fm zero_in_M*
 $domain_closed Un_closed$ **by** *simp*
have $fsats2: (M, [p, \mathbb{P}, leq, \vartheta, \tau] \models ? \varphi) \longleftrightarrow ?rel_pred(\#\#M, p, \mathbb{P}, leq, \vartheta, \tau)$ **if** $p \in M$ **for**
 p
using *that assms sats_forces_mem_fm sats_forces_nmem_fm zero_in_M*
 $domain_closed Un_closed$ **by** *simp*
have $fty: ? \varphi \in \text{formula}$ **by** *simp*
have $farit: arity(? \varphi) = 5$
by *(simp add: ord_simp_union arity)*
show
 $\{p \in \mathbb{P}. \exists \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta). p \text{ forces}_a (\sigma \in \tau) \wedge p \text{ forces}_a (\sigma \notin \vartheta)\}$
 $\in M$
and $\{p \in \mathbb{P}. \exists \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta). p \text{ forces}_a (\sigma \notin \tau) \wedge p \text{ forces}_a (\sigma \in$
 $\vartheta)\} \in M$
unfolding *forces_mem_def*
using $abs1 fty fsats1 farit$ *assms forces_nmem*
 $Collect_in_M[of ? \varphi [\mathbb{P}, leq, \tau, \vartheta]]$
using $abs2 fty fsats2 farit$ *assms forces_nmem domain_closed Un_closed*
 $Collect_in_M[of ? \varphi [\mathbb{P}, leq, \vartheta, \tau]]$
by *simp_all*
qed

end — *forcing_data1*

context *G_generic1*

begin

lemma *IV240b_eq*:

includes *some_rules*

assumes

$val(G, \tau) = val(G, \vartheta) \quad \tau \in M \quad \vartheta \in M$

and

$IH: \bigwedge \sigma. \sigma \in domain(\tau) \cup domain(\vartheta) \implies$

$(val(G, \sigma) \in val(G, \tau) \longrightarrow (\exists q \in G. q \text{ forces}_a (\sigma \in \tau))) \wedge$

$(val(G, \sigma) \in val(G, \vartheta) \longrightarrow (\exists q \in G. q \text{ forces}_a (\sigma \in \vartheta)))$

shows

$\exists p \in G. p \text{ forces}_a (\tau = \vartheta)$

proof -

let $?D1 = \{p \in \mathbb{P}. p \text{ forces}_a (\tau = \vartheta)\}$

let $?D2 = \{p \in \mathbb{P}. \exists \sigma \in domain(\tau) \cup domain(\vartheta). p \text{ forces}_a (\sigma \in \tau) \wedge p \text{ forces}_a (\sigma \notin \vartheta)\}$

let $?D3 = \{p \in \mathbb{P}. \exists \sigma \in domain(\tau) \cup domain(\vartheta). p \text{ forces}_a (\sigma \notin \tau) \wedge p \text{ forces}_a (\sigma \in \vartheta)\}$

let $?D = ?D1 \cup ?D2 \cup ?D3$

note *assms*

moreover from *this*

have $domain(\tau) \cup domain(\vartheta) \in M$ (**is** $?B \in M$) **using** *domain_closed Un_closed*
by *auto*

moreover from *calculation*

have $?D2 \in M$ **and** $?D3 \in M$ **using** *IV240b_eq_Collects* **by** *simp_all*

ultimately

have $?D \in M$ **using** *Collect_forces_eq_in_M Un_closed* **by** *auto*

moreover

have *dense*($?D$)

proof

fix p

assume $p \in \mathbb{P}$

have $\exists d \in \mathbb{P}. (d \text{ forces}_a (\tau = \vartheta) \vee$

$(\exists \sigma \in domain(\tau) \cup domain(\vartheta). d \text{ forces}_a (\sigma \in \tau) \wedge d \text{ forces}_a (\sigma \notin \vartheta)) \vee$

$(\exists \sigma \in domain(\tau) \cup domain(\vartheta). d \text{ forces}_a (\sigma \notin \tau) \wedge d \text{ forces}_a (\sigma \in \vartheta))) \wedge$

$d \preceq p$

proof (*cases* $p \text{ forces}_a (\tau = \vartheta)$)

case *True*

with $\langle p \in \mathbb{P} \rangle$

show *?thesis* **using** *refl_leq* **by** *blast*

next

case *False*

moreover note $\langle p \in \mathbb{P} \rangle$

moreover from *calculation*

obtain σ q **where** $\sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta)$ $q \in \mathbb{P}$ $q \preceq p$
 $(q \text{ forces}_a (\sigma \in \tau) \wedge \neg q \text{ forces}_a (\sigma \in \vartheta)) \vee$
 $(\neg q \text{ forces}_a (\sigma \in \tau) \wedge q \text{ forces}_a (\sigma \in \vartheta))$
using *def_forces_eq* **by** *blast*
moreover from this
obtain r **where** $r \preceq q$ $r \in \mathbb{P}$
 $(r \text{ forces}_a (\sigma \in \tau) \wedge r \text{ forces}_a (\sigma \notin \vartheta)) \vee$
 $(r \text{ forces}_a (\sigma \notin \tau) \wedge r \text{ forces}_a (\sigma \in \vartheta))$
using *not_forces_nmem strengthening_mem* **by** *blast*
ultimately
show *?thesis* **using** *leq_transD* **by** *blast*
qed
then
show $\exists d \in ?D . d \preceq p$ **by** *blast*
qed
moreover
have $?D \subseteq \mathbb{P}$
by *auto*
ultimately
obtain p **where** $p \in G$ $p \in ?D$
using *M_generic_denseD[of ?D]* **by** *blast*
then
consider
 $(1) p \text{ forces}_a (\tau = \vartheta) \mid$
 $(2) \exists \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta). p \text{ forces}_a (\sigma \in \tau) \wedge p \text{ forces}_a (\sigma \notin \vartheta) \mid$
 $(3) \exists \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta). p \text{ forces}_a (\sigma \notin \tau) \wedge p \text{ forces}_a (\sigma \in \vartheta)$
by *blast*
then
show *?thesis*
proof (*cases*)
case 1
with $\langle p \in G \rangle$
show *?thesis* **by** *blast*
next
case 2
then
obtain σ **where** $\sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta)$ $p \text{ forces}_a (\sigma \in \tau)$ $p \text{ forces}_a (\sigma \notin \vartheta)$
 $\vartheta)$
by *blast*
moreover from this and $\langle p \in G \rangle$ **and** *assms*
have $\text{val}(G, \sigma) \in \text{val}(G, \tau)$
using *IV240a[of $\sigma \tau$] transitivity[OF _ domain_closed[simplified]]* **by** *force*
moreover note $\langle \text{val}(G, \tau) = _ \rangle$
ultimately
obtain q **where** $q \in G$ $q \text{ forces}_a (\sigma \in \vartheta)$
using *IH[OF $\langle \sigma \in _ \rangle$]*
by *auto*
moreover from this and $\langle p \in G \rangle$
obtain r **where** $r \in \mathbb{P}$ $r \preceq p$ $r \preceq q$

```

    by blast
  ultimately
  have r forces_a (σ ∈ ∅)
    using strengthening_mem by blast
  with ⟨r ≤ p⟩ ⟨p forces_a (σ ∉ ∅)⟩ ⟨r ∈ ℙ⟩
  have False
    unfolding forces_nmem_def by blast
  then
  show ?thesis by simp
next
case 3
then
obtain σ where σ ∈ domain(τ) ∪ domain(∅) p forces_a (σ ∈ ∅) p forces_a (σ ∉
τ)
  by blast
moreover from this and ⟨p ∈ G⟩ and assms
have val(G,σ) ∈ val(G,∅)
  using IV240a[of σ ∅] transitivity[OF _ domain_closed[simplified]] by force
moreover note ⟨val(G,τ) = _⟩
ultimately
obtain q where q ∈ G q forces_a (σ ∈ τ)
  using IH[OF ⟨σ ∈ _⟩]
  by auto
moreover from this and ⟨p ∈ G⟩
obtain r where r ∈ ℙ r ≤ p r ≤ q
  by blast
ultimately
have r forces_a (σ ∈ τ)
  using strengthening_mem by blast
with ⟨r ≤ p⟩ ⟨p forces_a (σ ∉ τ)⟩ ⟨r ∈ ℙ⟩
have False
  unfolding forces_nmem_def by blast
then
show ?thesis by simp
qed
qed

```

lemma IV240b:

$$(\tau \in M \longrightarrow \vartheta \in M \longrightarrow \text{val}(G, \tau) = \text{val}(G, \vartheta) \longrightarrow (\exists p \in G. p \text{ forces}_a (\tau = \vartheta))) \wedge \\
(\tau \in M \longrightarrow \vartheta \in M \longrightarrow \text{val}(G, \tau) \in \text{val}(G, \vartheta) \longrightarrow (\exists p \in G. p \text{ forces}_a (\tau \in \vartheta))) \\
(\text{is } ?Q(\tau, \vartheta) \wedge ?R(\tau, \vartheta))$$

proof (*intro forces_induction*)

fix τ ∅ p

assume σ ∈ domain(∅) ⇒ ?Q(τ, σ) **for** σ

then show ?R(τ, ∅)

using IV240b_mem domain_closed transitivity **by** simp

next

fix τ ∅ p

assume $\sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies ?R(\sigma, \tau) \wedge ?R(\sigma, \vartheta)$ **for** σ
moreover from this
have $IH': \tau \in M \implies \vartheta \in M \implies \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies$
 $(\text{val}(G, \sigma) \in \text{val}(G, \tau) \longrightarrow (\exists q \in G. q \text{ forces}_a (\sigma \in \tau))) \wedge$
 $(\text{val}(G, \sigma) \in \text{val}(G, \vartheta) \longrightarrow (\exists q \in G. q \text{ forces}_a (\sigma \in \vartheta)))$ **for** σ
using $\text{domain_trans}[OF \text{ trans_}M]$
by *blast*
ultimately
show $?Q(\tau, \vartheta)$
using $IV240b_eq$ **by** *auto*
qed

lemma *truth_lemma_mem*:

assumes
 $\text{env} \in \text{list}(M)$
 $n \in \text{nat } m \in \text{nat } n < \text{length}(\text{env}) \ m < \text{length}(\text{env})$
shows
 $(\exists p \in G. p \Vdash \text{Member}(n, m) \ \text{env}) \longleftrightarrow M[G], \text{map}(\text{val}(G), \text{env}) \models \text{Member}(n, m)$
using $\text{assms } IV240a[\text{of } nth(n, \text{env}) \ nth(m, \text{env})]$
 $IV240b[\text{of } nth(n, \text{env}) \ nth(m, \text{env})]$
 $M_genericD$
 $\text{Forces_Member}[\text{of } _ \ nth(n, \text{env}) \ nth(m, \text{env}) \ \text{env } n \ m] \ \text{map_val_in_}MG$
by *auto*

lemma *truth_lemma_eq*:

assumes
 $\text{env} \in \text{list}(M)$
 $n \in \text{nat } m \in \text{nat } n < \text{length}(\text{env}) \ m < \text{length}(\text{env})$
shows
 $(\exists p \in G. p \Vdash \text{Equal}(n, m) \ \text{env}) \longleftrightarrow M[G], \text{map}(\text{val}(G), \text{env}) \models \text{Equal}(n, m)$
using $\text{assms } IV240a(1)[\text{of } nth(n, \text{env}) \ nth(m, \text{env})]$
 $IV240b(1)[\text{of } nth(n, \text{env}) \ nth(m, \text{env})]$
 $M_genericD$
 $\text{Forces_Equal}[\text{of } _ \ nth(n, \text{env}) \ nth(m, \text{env}) \ \text{env } n \ m] \ \text{map_val_in_}MG$
by *auto*

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
 $n < \text{length}(\text{env}) \ m < \text{length}(\text{env})$
using $\text{assms } \text{succ_leE}[OF \text{ Un_leD1, of } n \ \text{succ}(m) \ \text{length}(\text{env})]$
 $\text{succ_leE}[OF \text{ Un_leD2, of } \text{succ}(n) \ m \ \text{length}(\text{env})]$ **by** *auto*

13.13 The Strengthening Lemma

context *forcing_data1*

```

begin

lemma strengthening_lemma:
  assumes
    p ∈ P  φ ∈ formula  r ∈ P  r ≤ p
    env ∈ list(M)  arity(φ) ≤ length(env)
  shows
    p ⊢ φ env ⇒ r ⊢ φ env
  using assms(2-)
proof (induct arbitrary: env)
  case (Member n m)
  then
  have n < length(env)  m < length(env)
    using arities_at_aux by simp_all
  moreover
  assume env ∈ list(M)
  moreover
  note assms Member
  ultimately
  show ?case
    using Forces_Member[of _ nth(n,env) nth(m,env) env n m]
      strengthening_mem[of p r nth(n,env) nth(m,env)] by simp
next
  case (Equal n m)
  then
  have n < length(env)  m < length(env)
    using arities_at_aux by simp_all
  moreover
  assume env ∈ list(M)
  moreover
  note assms Equal
  ultimately
  show ?case
    using Forces_Equal[of _ nth(n,env) nth(m,env) env n m]
      strengthening_eq[of p r nth(n,env) nth(m,env)] by simp
next
  case (Nand φ ψ)
  with assms
  show ?case
    using Forces_Nand transitivity[OF P_in_M] pair_in_M_iff
      transitivity[OF leq_in_M] leq_transD by auto
next
  case (Forall φ)
  with assms
  have p ⊢ φ ([x] @ env) if x ∈ M for x
    using that Forces_Forall by simp
  with Forall
  have r ⊢ φ ([x] @ env) if x ∈ M for x
    using that pred_le2 by (simp)

```

```

with assms Forall
show ?case
  using Forces_Forall by simp
qed

```

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  $\text{arity}(\varphi) \leq \text{length}(\text{env})$   $\text{arity}(\psi) \leq \text{length}(\text{env})$ 
using assms
by (rule_tac Un_leD1, rule_tac [5] Un_leD2, auto)

```

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
   $\text{dense\_below}(\{q \in \mathbb{P}. (q \Vdash \varphi \text{ env})\}, p) \implies (p \Vdash \varphi \text{ env})$ 
using assms(2-)
proof (induct arbitrary:env)
  case (Member n m)
  then
  have  $n < \text{length}(\text{env})$   $m < \text{length}(\text{env})$ 
    using arities_at_aux by simp_all
  moreover
  assume  $\text{env} \in \text{list}(M)$ 
  moreover
  note assms Member
  ultimately
  show ?case
    using Forces_Member[of _ nth(n,env) nth(m,env) env n m]
    density_mem[of p nth(n,env) nth(m,env)] by simp
  next
  case (Equal n m)
  then
  have  $n < \text{length}(\text{env})$   $m < \text{length}(\text{env})$ 
    using arities_at_aux by simp_all
  moreover
  assume  $\text{env} \in \text{list}(M)$ 
  moreover
  note assms Equal
  ultimately
  show ?case
    using Forces_Equal[of _ nth(n,env) nth(m,env) env n m]
    density_eq[of p nth(n,env) nth(m,env)] by simp
  next
  case (Nand  $\varphi \psi$ )
  {

```

```

fix q
assume  $q \in M$   $q \in \mathbb{P}$   $q \preceq p$   $q \Vdash \varphi$  env
moreover
note Nand
moreover from calculation
obtain d where  $d \in \mathbb{P}$   $d \Vdash \text{Nand}(\varphi, \psi)$  env  $d \preceq q$ 
  using dense_belowI by auto
moreover from calculation
have  $\neg(d \Vdash \psi)$  env if  $d \Vdash \varphi$  env
  using that Forces_Nand_refl_leq_transitivity[OF _ P_in_M, of d] by auto
moreover
note arity_Nand_le[of  $\varphi$   $\psi$ ]
moreover from calculation
have  $d \Vdash \varphi$  env
  using strengthening_lemma[of  $q$   $\varphi$   $d$  env] Un_leD1 by auto
ultimately
have  $\neg(q \Vdash \psi)$  env
  using strengthening_lemma[of  $q$   $\psi$   $d$  env] by auto
}
with  $\langle p \in \mathbb{P} \rangle$ 
show ?case
  using Forces_Nand[symmetric, OF _ Nand(6,1,3)] by blast
next
case (Forall  $\varphi$ )
have dense_below( $\{q \in \mathbb{P}. q \Vdash \varphi ([a]@env)\}, p)$  if  $a \in M$  for a
proof
fix r
assume  $r \in \mathbb{P}$   $r \preceq p$ 
with  $\langle \text{dense\_below}(\_, p) \rangle$ 
obtain q where  $q \in \mathbb{P}$   $q \preceq r$   $q \Vdash \text{Forall}(\varphi)$  env
  by blast
moreover
note Forall  $\langle a \in M \rangle$ 
moreover from calculation
have  $q \Vdash \varphi ([a]@env)$ 
  using Forces_Forall by simp
ultimately
show  $\exists d \in \{q \in \mathbb{P}. q \Vdash \varphi ([a]@env)\}. d \in \mathbb{P} \wedge d \preceq r$ 
  by auto
qed
moreover
note Forall(2)[of Cons( $\_, env$ )] Forall(1,3-5)
ultimately
have  $p \Vdash \varphi ([a]@env)$  if  $a \in M$  for a
  using that pred_le2 by simp
with assms Forall
show ?case using Forces_Forall by simp
qed

```

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
 $p \Vdash \varphi \ \text{env} \iff \text{dense_below}(\{q \in \mathbb{P}. (q \Vdash \varphi \ \text{env})\}, p)$
proof
assume $\text{dense_below}(\{q \in \mathbb{P}. (q \Vdash \varphi \ \text{env})\}, p)$
with *assms*
show $(p \Vdash \varphi \ \text{env})$
using *dense_below_imp_forces* **by** *simp*
next
assume $p \Vdash \varphi \ \text{env}$
with *assms*
show $\text{dense_below}(\{q \in \mathbb{P}. q \Vdash \varphi \ \text{env}\}, p)$
using *strengthening_lemma refl_leq* **by** *auto*
qed

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
 $p \Vdash \text{And}(\varphi, \psi) \ \text{env} \iff (p \Vdash \varphi \ \text{env}) \wedge (p \Vdash \psi \ \text{env})$
proof
assume $p \Vdash \text{And}(\varphi, \psi) \ \text{env}$
with *assms*
have $\text{dense_below}(\{r \in \mathbb{P}. (r \Vdash \varphi \ \text{env}) \wedge (r \Vdash \psi \ \text{env})\}, p)$
using *Forces_And_iff_dense_below* **by** *simp*
then
have $\text{dense_below}(\{r \in \mathbb{P}. (r \Vdash \varphi \ \text{env})\}, p) \ \text{dense_below}(\{r \in \mathbb{P}. (r \Vdash \psi \ \text{env})\}, p)$
by *blast+*
with *assms*
show $(p \Vdash \varphi \ \text{env}) \wedge (p \Vdash \psi \ \text{env})$
using *density_lemma[symmetric]* **by** *simp*
next
assume $(p \Vdash \varphi \ \text{env}) \wedge (p \Vdash \psi \ \text{env})$
have $\text{dense_below}(\{r \in \mathbb{P}. (r \Vdash \varphi \ \text{env}) \wedge (r \Vdash \psi \ \text{env})\}, p)$
proof (*intro dense_belowI beqI conjI, assumption*)
fix q
assume $q \in \mathbb{P} \ q \preceq p$
with *assms* $\langle (p \Vdash \varphi \ \text{env}) \wedge (p \Vdash \psi \ \text{env}) \rangle$
show $q \in \{r \in \mathbb{P}. (r \Vdash \varphi \ \text{env}) \wedge (r \Vdash \psi \ \text{env})\} \ q \preceq q$
using *strengthening_lemma refl_leq* **by** *auto*
qed
with *assms*
show $p \Vdash \text{And}(\varphi, \psi) \ \text{env}$

using *Forces_And_iff_dense_below* by *simp*
qed

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

$(p \Vdash Nand(\varphi, \psi) \ env) \longleftrightarrow (p \Vdash Neg(And(\varphi, \psi)) \ env)$

using *assms Forces_Nand Forces_And Forces_Neg* by *auto*

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

$(\exists p \in G. p \Vdash Neg(\varphi) \ env) \longleftrightarrow M[G], map(val(G), env) \models Neg(\varphi)$

proof (*intro iffI, elim bexE, rule ccontr*)

fix *p*

assume $p \in G$ $p \Vdash Neg(\varphi) \ env$ $\neg(M[G], map(val(G), env) \models Neg(\varphi))$

moreover

note *assms*

moreover from *calculation*

have $M[G], map(val(G), env) \models \varphi$ $p \in \mathbb{P}$

using *map_val_in_MG* by *auto*

with *IH*

obtain *r* **where** $r \Vdash \varphi \ env$ $r \in G$ $r \in \mathbb{P}$ by *blast*

moreover from *this* **and** $\langle p \in G \rangle$

obtain *q* **where** $q \preceq p$ $q \preceq r$ $q \in G$ $q \in \mathbb{P}$ $q \in M$

using *transitivity[OF P_in_M]*

by *blast*

moreover from *calculation*

have $q \Vdash \varphi \ env$

using *strengthening_lemma*

by *simp*

with *assms* $\langle p \Vdash _ _ \rangle$ $\langle q \preceq p \rangle$ $\langle q \in M \rangle$ $\langle p \in \mathbb{P} \rangle$ $\langle q \in \mathbb{P} \rangle$

show *False*

using *Forces_Neg*

by *auto*

next

assume $M[G], map(val(G), env) \models Neg(\varphi)$

with *assms*

have $\neg(M[G], map(val(G), env) \models \varphi)$

```

    using map_val_in_MG by simp
  let ?D={p∈ℙ. (p ⊨ φ env) ∨ (p ⊨ Neg(φ) env)}
  from assms
  have ?D ∈ M
    using separation_disj separation_closed separation_forces by simp
  moreover
  have ?D ⊆ ℙ by auto
  moreover
  have dense(?D)
  proof
    fix q
    assume q∈ℙ
    with assms
    show ∃ d∈{p ∈ ℙ . (p ⊨ φ env) ∨ (p ⊨ Neg(φ) env)}. d ≤ q
      using refl_leq Forces_Neg by (cases q ⊨ Neg(φ) env, auto)
  qed
  ultimately
  obtain p where p∈G (p ⊨ φ env) ∨ (p ⊨ Neg(φ) env)
    by blast
  then
  consider (1) p ⊨ φ env | (2) p ⊨ Neg(φ) env by blast
  then
  show ∃ p∈G. (p ⊨ Neg(φ) env)
  proof (cases)
    case 1
    with ⟨¬ (M[G],map(val(G),env) ⊨ φ)⟩ ⟨p∈G⟩ IH
    show ?thesis
      by blast
    next
    case 2
    with ⟨p∈G⟩
    show ?thesis by blast
  qed
  qed

```

lemma *truth_lemma_And*:

```

  assumes
    env∈list(M) φ∈formula ψ∈formula
    arity(φ)≤length(env) arity(ψ) ≤ length(env)
  and
    IH: (∃ p∈G. p ⊨ φ env) ↔ M[G], map(val(G),env) ⊨ φ
    (∃ p∈G. p ⊨ ψ env) ↔ M[G], map(val(G),env) ⊨ ψ
  shows
    (∃ p∈G. (p ⊨ And(φ,ψ) env)) ↔ M[G], map(val(G),env) ⊨ And(φ,ψ)
  using assms map_val_in_MG Forces_And[OF M_genericD assms(1-5)]
  proof (intro iffI, elim bexE)
    fix p
    assume p∈G p ⊨ And(φ,ψ) env
    with assms

```

```

show  $M[G], \text{map}(\text{val}(G), \text{env}) \models \text{And}(\varphi, \psi)$ 
  using Forces_And[of _ _  $\varphi$   $\psi$ ] map_val_in_MG M_genericD by auto
next
assume  $M[G], \text{map}(\text{val}(G), \text{env}) \models \text{And}(\varphi, \psi)$ 
moreover
note assms
moreover from calculation
obtain  $q \ r$  where  $q \Vdash \varphi \ \text{env} \ r \Vdash \psi \ \text{env} \ q \in G \ r \in G \ r \in \mathbb{P} \ q \in \mathbb{P}$ 
  using map_val_in_MG Forces_And[OF M_genericD assms(1-5)] M_genericD
by auto
moreover from calculation
obtain  $p$  where  $p \preceq q \ p \preceq r \ p \in G$ 
  by auto
moreover from calculation
have  $(p \Vdash \varphi \ \text{env}) \wedge (p \Vdash \psi \ \text{env})$ 
  using strengthening_lemma[OF M_genericD] by force
ultimately
show  $\exists p \in G. (p \Vdash \text{And}(\varphi, \psi) \ \text{env})$ 
  using Forces_And[OF M_genericD assms(1-5)] by auto
qed

end

```

definition

```

ren_truth_lemma ::  $i \Rightarrow i$  where
ren_truth_lemma( $\varphi$ )  $\equiv$ 
  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( $\lambda p. \text{incr\_bv}(p)$  '5 , 6,  $\varphi$ ))))))))))

```

```

lemma ren_truth_lemma_type[TC] :
 $\varphi \in \text{formula} \Longrightarrow \text{ren\_truth\_lemma}(\varphi) \in \text{formula}$ 
unfolding ren_truth_lemma_def
by simp

```

```

lemma arity_ren_truth :
  assumes  $\varphi \in \text{formula}$ 
  shows  $\text{arity}(\text{ren\_truth\_lemma}(\varphi)) \leq 6 \cup \text{succ}(\text{arity}(\varphi))$ 
proof -
  consider ( $lt$ )  $5 < \text{arity}(\varphi) \mid$  ( $ge$ )  $\neg 5 < \text{arity}(\varphi)$ 
  by auto
  then
  show ?thesis
proof cases
  case  $lt$ 
  consider ( $a$ )  $5 < \text{arity}(\varphi) +_\omega 5 \mid$  ( $b$ )  $\text{arity}(\varphi) +_\omega 5 \leq 5$ 
  using not_lt_iff_le  $\langle \varphi \in \_ \rangle$  by force
  then
  show ?thesis

```

```

proof cases
  case a
    with ⟨ $\varphi \in \_$ ⟩ lt
    have  $5 < \text{succ}(\text{arity}(\varphi))$   $5 < \text{arity}(\varphi) + \omega^2$   $5 < \text{arity}(\varphi) + \omega^3$   $5 < \text{arity}(\varphi) + \omega^4$ 
      using succ_ltI by auto
    with ⟨ $\varphi \in \_$ ⟩
    have  $c:\text{arity}(\text{iterates}(\lambda p. \text{incr\_bv}(p) '5, 5, \varphi)) = 5 + \omega \text{arity}(\varphi)$  (is  $\text{arity}(\varphi') =$ 
     $\_$ )
      using arity_incr_bv_lemma lt a
      by simp
    with ⟨ $\varphi \in \_$ ⟩
    have  $\text{arity}(\text{incr\_bv}(\varphi') '5) = 6 + \omega \text{arity}(\varphi)$ 
      using arity_incr_bv_lemma[of ? $\varphi'$  5] a by auto
    with ⟨ $\varphi \in \_$ ⟩
    show ?thesis
      unfolding ren_truth_lemma_def
      using pred_Un_distrib union_abs1 Un_assoc[symmetric] a c union_abs2
      by (simp add:arity)
  next
  case b
    with ⟨ $\varphi \in \_$ ⟩ lt
    have  $5 < \text{succ}(\text{arity}(\varphi))$   $5 < \text{arity}(\varphi) + \omega^2$   $5 < \text{arity}(\varphi) + \omega^3$   $5 < \text{arity}(\varphi) + \omega^4$ 
     $5 < \text{arity}(\varphi) + \omega^5$ 
      using succ_ltI by auto
    with ⟨ $\varphi \in \_$ ⟩
    have  $\text{arity}(\text{iterates}(\lambda p. \text{incr\_bv}(p) '5, 6, \varphi)) = 6 + \omega \text{arity}(\varphi)$  (is  $\text{arity}(\varphi') = \_$ )
      using arity_incr_bv_lemma lt
      by simp
    with ⟨ $\varphi \in \_$ ⟩
    show ?thesis
      unfolding ren_truth_lemma_def
      using pred_Un_distrib union_abs1 Un_assoc[symmetric] union_abs2
      by (simp add:arity)
  qed
next
  case ge
    with ⟨ $\varphi \in \_$ ⟩
    have  $\text{arity}(\varphi) \leq 5$   $\text{pred}^5(\text{arity}(\varphi)) \leq 5$ 
      using not_lt_iff_le le_trans[OF le_pred]
      by auto
    with ⟨ $\varphi \in \_$ ⟩
    have  $\text{arity}(\text{iterates}(\lambda p. \text{incr\_bv}(p) '5, 6, \varphi)) = \text{arity}(\varphi)$   $\text{arity}(\varphi) \leq 6$   $\text{pred}^5(\text{arity}(\varphi))$ 
     $\leq 6$ 
      using arity_incr_bv_lemma ge le_trans[OF ⟨arity( $\varphi$ )  $\leq$  5⟩] le_trans[OF
     $\langle \text{pred}^5(\text{arity}(\varphi)) \leq 5 \rangle$ 
      by auto
    with  $\langle \text{arity}(\varphi) \leq 5 \rangle$   $\langle \varphi \in \_ \rangle$   $\langle \text{pred}^5(\_) \leq 5 \rangle$ 
    show ?thesis
      unfolding ren_truth_lemma_def

```

```

    using pred_Un_distrib union_abs1 Un_assoc[symmetric] union_abs2
    by (simp add:arity)
qed
qed

lemma sats_ren_truth_lemma:
  [q,b,d,a1,a2,a3] @ env ∈ list(M) ⇒ φ ∈ formula ⇒
  (M, [q,b,d,a1,a2,a3] @ env ⊨ ren_truth_lemma(φ) ) ↔
  (M, [q,a1,a2,a3,b] @ env ⊨ φ)
  unfolding ren_truth_lemma_def
  by (insert sats_incr_bv_iff [of _ _ M _ [q,a1,a2,a3,b]], simp)

context forcing_data1
begin

lemma truth_lemma' :
  assumes
    φ∈formula env∈list(M) arity(φ) ≤ succ(length(env))
  shows
    separation(##M,λd. ∃ b∈M. ∀ q∈P. q⊆d → ¬(q ⊨ φ ([b]@env)))
proof -
  let ?rel_pred=λM x a1 a2 a3. ∃ b∈M. ∀ q∈M. q∈a1 ∧ is_leq(##M,a2,q,x) →
    ¬(M, [q,a1,a2,a3,b] @ env ⊨ forces(φ))
  let ?ψ=Exists(Forall(Implies(And(Member(0,3),is_leq_fm(4,0,2)),
    Neg(ren_truth_lemma(forces(φ))))))
  have q∈M if q∈P for q using that transitivity[OF _ P_in_M] by simp
  then
  have 1:∀ q∈M. q∈P ∧ R(q) → Q(q) ⇒ (∀ q∈P. R(q) → Q(q)) for R Q
    by auto
  then
  have [[b ∈ M; ∀ q∈M. q ∈ P ∧ q ⊆ d → ¬(q ⊨ φ ([b]@env))] ⇒
    ∃ c∈M. ∀ q∈P. q ⊆ d → ¬(q ⊨ φ ([c]@env))] for b d
    by (rule bexI,simp_all)
  then
  have ?rel_pred(M,d,P,leq,1) ↔ (∃ b∈M. ∀ q∈P. q⊆d → ¬(q ⊨ φ ([b]@env)))
if d∈M for d
  using that leq_abs assms
  by auto
  moreover
  have ?ψ∈formula using assms by simp
  moreover
  have (M, [d,P,leq,1]@env ⊨ ?ψ) ↔ ?rel_pred(M,d,P,leq,1) if d∈M for d
    using assms that sats_is_leq_fm sats_ren_truth_lemma zero_in_M
    by simp
  moreover
  have arity(?ψ) ≤ 4+ω length(env)
proof -
  have eq:arity(is_leq_fm(4, 0, 2)) = 5
    using arity_is_leq_fm succ_Un_distrib ord_simp_union

```

```

    by simp
  with ⟨φ∈_⟩
  have arity(?ψ) = 3 ∪ (pred2(arity(ren_truth_lemma(forces(φ))))))
    using union_abs1 pred_Un_distrib by (simp add:arity)
  moreover
  have ... ≤ 3 ∪ (pred(pred(6 ∪ succ(arity(forces(φ)))))) (is _ ≤ ?r)
    using ⟨φ∈_⟩ Un_le_compat[OF le_refl[of 3]]
      le_imp_subset arity_ren_truth[of forces(φ)]
      pred_mono
    by auto
  finally
  have arity(?ψ) ≤ ?r by simp
  have i:?r ≤ 4 ∪ pred(arity(forces(φ)))
  using pred_Un_distrib pred_succ_eq ⟨φ∈_⟩ Un_assoc[symmetric] union_abs1
by simp
  have h:4 ∪ pred(arity(forces(φ))) ≤ 4 ∪ (4+ωlength(env))
    using ⟨env∈_⟩ add_commute ⟨φ∈_⟩
      Un_le_compat[of 4 4,OF _ pred_mono[OF _ arity_forces_le[OF _ _
⟨arity(φ)≤_⟩]] ]
      ⟨env∈_⟩ by auto
  with ⟨φ∈_⟩ ⟨env∈_⟩
  show ?thesis
    using le_trans[OF ⟨arity(?ψ) ≤ ?r⟩ le_trans[OF i h]] ord_simp_union by
simp
  qed
  ultimately
  show ?thesis using assms
    separation_ax[of ?ψ [ℙ,leq,1]@env]
    separation_cong[of ##M λy. (M, [y,ℙ,leq,1]@env ⊨ ?ψ)]
  by simp
qed
end

context G_generic1
begin

lemma truth_lemma:
  assumes
    φ∈formula
    env∈list(M) arity(φ)≤length(env)
  shows
    (∃ p∈G. p ⊨ φ env) ↔ M[G], map(val(G),env) ⊨ φ
  using assms
proof (induct arbitrary:env)
  case (Member x y)
  then
  show ?case
    using truth_lemma_mem[OF ⟨env∈list(M)⟩ ⟨x∈nat⟩ ⟨y∈nat⟩] arities_at_aux

```

```

    by simp
next
case (Equal x y)
then
show ?case
  using truth_lemma_eq[OF ‹env∈list(M)› ‹x∈nat› ‹y∈nat›] arities_at_aux by
simp
next
case (Nand φ ψ)
then
show ?case
  using truth_lemma_And truth_lemma_Neg[of ·φ ∧ ψ.] Forces_Nand_alt
  M_genericD map_val_in_MG arity_Nand_le[of φ ψ] FOL_arities by auto
next
case (Forall φ)
then
show ?case
proof (intro iffI)
  assume ∃p∈G. (p ⊢ Forall(φ) env)
  then
  obtain p where p∈G p∈M p∈ℙ p ⊢ Forall(φ) env
    using transitivity[OF _ P_in_M] by auto
  with ‹env∈list(M)› ‹φ∈formula›
  have p ⊢ φ ([x]@env) if x∈M for x
    using that Forces_Forall by simp
  with ‹p∈G› ‹φ∈formula› ‹env∈_› ‹arity(Forall(φ)) ≤ length(env)›
  Forall(2)[of Cons(_,env)]
  show M[G], map(val(G),env) ⊨ Forall(φ)
    using pred_le2 map_val_in_MG
    by (auto iff:GenExt_iff)
next
assume M[G], map(val(G),env) ⊨ Forall(φ)
let ?D1={d∈ℙ. (d ⊢ Forall(φ) env)}
let ?D2={d∈ℙ. ∃ b∈M. ∀ q∈ℙ. q ≤ d ⟶ ¬(q ⊢ φ ([b]@env))}
define D where D ≡ ?D1 ∪ ?D2
note ‹arity(Forall(φ)) ≤ length(env)› ‹φ∈formula› ‹env∈list(M)›
moreover from this
have arφ:arity(φ)≤succ(length(env))
  using pred_le2 by simp
moreover from calculation
have ?D1∈M using Collect_forces by simp
moreover from ‹env∈list(M)› ‹φ∈formula›
have ?D2∈M
  using truth_lemma'[of φ] separation_closed arφ
  by simp
ultimately
have D∈M unfolding D_def using Un_closed by simp
moreover

```

```

have  $D \subseteq \mathbb{P}$  unfolding  $D\_def$  by auto
moreover
have dense( $D$ )
proof
  fix  $p$ 
  assume  $p \in \mathbb{P}$ 
  show  $\exists d \in D. d \preceq p$ 
  proof (cases  $p \Vdash \text{Forall}(\varphi) \text{ env}$ )
    case True
    with  $\langle p \in \mathbb{P} \rangle$ 
    show ?thesis unfolding  $D\_def$  using refl_leq by blast
  next
  case False
  with Forall  $\langle p \in \mathbb{P} \rangle$ 
  obtain  $b$  where  $b \in M \neg(p \Vdash \varphi ([b]@env))$ 
    using Forces_Forall by blast
  moreover from this  $\langle p \in \mathbb{P} \rangle$  Forall
  have  $\neg \text{dense\_below}(\{q \in \mathbb{P}. q \Vdash \varphi ([b]@env)\}, p)$ 
    using density_lemma_pred_le2 by auto
  moreover from this
  obtain  $d$  where  $d \preceq p \forall q \in \mathbb{P}. q \preceq d \longrightarrow \neg(q \Vdash \varphi ([b] @ env))$ 
     $d \in \mathbb{P}$  by blast
  ultimately
  show ?thesis unfolding  $D\_def$  by auto
  qed
qed
moreover
note generic
ultimately
obtain  $d$  where  $d \in D \ d \in G$  by blast
then
consider (1)  $d \in ?D1$  | (2)  $d \in ?D2$  unfolding  $D\_def$  by blast
then
show  $\exists p \in G. (p \Vdash \text{Forall}(\varphi) \text{ env})$ 
proof (cases)
  case 1
  with  $\langle d \in G \rangle$ 
  show ?thesis by blast
next
case 2
then
obtain  $b$  where  $b \in M \forall q \in \mathbb{P}. q \preceq d \longrightarrow \neg(q \Vdash \varphi ([b] @ env))$ 
  by blast
moreover from this(1) and  $\langle M[G], \_ \models \text{Forall}(\varphi) \rangle$  and
  Forall(2)[of Cons( $b, env$ )] Forall(1,3-)
obtain  $p$  where  $p \in G \ p \in \mathbb{P} \ p \Vdash \varphi ([b] @ env)$ 
  using pred_le2_map_val_in_MG M_genericD by (auto iff:GenExt_iff)
moreover
note  $\langle d \in G \rangle$ 

```

```

ultimately
obtain  $q$  where  $q \in G$   $q \in \mathbb{P}$   $q \leq d$   $q \leq p$ 
  using  $M\_genericD$  by force
moreover from this and  $\langle p \Vdash \varphi ([b] @ env) \rangle$ 
  Forall  $\langle b \in M \rangle \langle p \in \mathbb{P} \rangle$ 
have  $q \Vdash \varphi ([b] @ env)$ 
  using  $pred\_le2$  strengthening_lemma by simp
moreover
note  $\langle \forall q \in \mathbb{P}. q \leq d \longrightarrow \neg(q \Vdash \varphi ([b] @ env)) \rangle$ 
ultimately
show ?thesis by simp
qed
qed
qed

end

```

```

context forcing_data1
begin

```

13.16 The “Definition of forcing”

lemma definition_of_forcing:

```

assumes
   $p \in \mathbb{P}$   $\varphi \in \text{formula}$   $env \in \text{list}(M)$   $\text{arity}(\varphi) \leq \text{length}(env)$ 
shows
   $(p \Vdash \varphi env) \longleftrightarrow$ 
   $(\forall G. M\_generic(G) \wedge p \in G \longrightarrow M[G], \text{map}(\text{val}(G), env) \models \varphi)$ 
proof (intro iffI allI impI, elim conjE)
fix  $G$ 
assume  $(p \Vdash \varphi env)$   $M\_generic(G)$   $p \in G$ 
moreover from this
interpret  $G\_generic1$   $\mathbb{P}$  leq 1  $M$  enum  $G$ 
  by (unfold_locales, simp)
from calculation assms
show  $M[G], \text{map}(\text{val}(G), env) \models \varphi$ 
  using truth_lemma[of  $\varphi$ ] by auto
next
assume 1:  $\forall G. (M\_generic(G) \wedge p \in G) \longrightarrow M[G], \text{map}(\text{val}(G), env) \models \varphi$ 
{
  fix  $r$ 
  assume 2:  $r \in \mathbb{P}$   $r \leq p$ 
  then
  obtain  $G$  where  $r \in G$   $M\_generic(G)$ 

```

Here we’re using countability (via the existence of generic filters) of M as a shortcut.

```

  using generic_filter_existence by auto
  moreover from this

```

```

interpret  $G\_generic1$   $\mathbb{P}$   $leq$  1  $M$   $enum$   $G$ 
  by ( $unfold\_locales, simp$ )
from  $calculation$  2  $\langle p \in \mathbb{P} \rangle$ 
have  $p \in G$ 
  using  $filter\_leqD$  by  $auto$ 
moreover note 1
ultimately
have  $M[G], map(val(G), env) \models \varphi$ 
  by  $simp$ 
moreover
note  $assms$ 
moreover from  $calculation$ 
obtain  $s$  where  $s \in G$  ( $s \Vdash \varphi$   $env$ )
  using  $truth\_lemma[of \varphi]$  by  $blast$ 
moreover from  $this$   $\langle r \in G \rangle$ 
obtain  $q$  where  $q \in G$   $q \preceq s$   $q \preceq r$   $s \in \mathbb{P}$   $q \in \mathbb{P}$ 
  by  $blast$ 
ultimately
have  $\exists q \in \mathbb{P}. q \preceq r \wedge (q \Vdash \varphi$   $env)$ 
  using  $strengthening\_lemma[of s]$  by  $auto$ 
}
then
have  $dense\_below(\{q \in \mathbb{P}. (q \Vdash \varphi$   $env)\}, p)$ 
  unfolding  $dense\_below\_def$  by  $blast$ 
with  $assms$ 
show ( $p \Vdash \varphi$   $env$ )
  using  $density\_lemma$  by  $blast$ 
qed

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  $rank\_val: rank(val(G, x)) \leq rank(x)$  (is  $?Q(x)$ )
proof ( $induct$   $rule:ed\_induction[of ?Q]$ )
  case ( $1$   $x$ )
  have  $val(G, x) = \{val(G, u). u \in \{t \in domain(x). \exists p \in G . \langle t, p \rangle \in x\}\}$ 

```

```

    using def_val[of G x] by auto
  then
  have rank(val(G,x)) = ( $\bigcup u \in \{t \in \text{domain}(x). \exists p \in G. \langle t,p \rangle \in x\}. \text{succ}(\text{rank}(\text{val}(G,u)))$ )
    using rank[of val(G,x)] by simp
  moreover
  have succ(rank(val(G,y)))  $\leq$  rank(x) if ed(y,x) for y
    using 1[OF that] rank_ed[OF that] by (auto intro:lt_trans1)
  moreover from this
  have ( $\bigcup u \in \{t \in \text{domain}(x). \exists p \in G. \langle t,p \rangle \in x\}. \text{succ}(\text{rank}(\text{val}(G,u)))$ )  $\leq$  rank(x)
    by (rule_tac UN_least_le) (auto)
  ultimately
  show ?case
    by simp
qed

```

```

lemma Ord_MG_iff:
  assumes Ord( $\alpha$ )
  shows  $\alpha \in M \longleftrightarrow \alpha \in M[G]$ 
proof
  show  $\alpha \in M[G]$  if  $\alpha \in M$ 
    using M_subset_MG[OF one_in_G] that ..
next
  assume  $\alpha \in M[G]$ 
  then
  obtain x where  $x \in M$  val(G,x) =  $\alpha$ 
    using GenExtD by auto
  then
  have rank( $\alpha$ )  $\leq$  rank(x)
    using rank_val by blast
  with assms
  have  $\alpha \leq$  rank(x)
    using rank_of_Ord by simp
  then
  have  $\alpha \in \text{succ}(\text{rank}(x))$ 
    using ltD by simp
  with  $\langle x \in M \rangle$ 
  show  $\alpha \in M$ 
    using cons_closed_transitivity[of  $\alpha$  succ(rank(x))] rank_closed
    unfolding succ_def by simp
qed

```

end — *G_generic1*

end

15 Auxiliary renamings for Separation

```

theory Separation_Rename
  imports

```

```

    Interface
begin

no_notation Aleph (‹ℵ› [90] 90)

lemmas apply_fun = apply_iff[THEN iffD1]

lemma nth_concat : [p,t] ∈ list(A) ⇒ env ∈ list(A) ⇒ nth(1 +ω length(env), [p]@
env @ [t]) = t
  by(auto simp add: nth_append)

lemma nth_concat2 : env ∈ list(A) ⇒ nth(length(env), env @ [p,t]) = p
  by(auto simp add: nth_append)

lemma nth_concat3 : env ∈ list(A) ⇒ u = nth(succ(length(env)), env @ [pi, u])
  by(auto simp add: nth_append)

definition
  sep_var :: i ⇒ i where
  sep_var(n) ≡ {⟨0,1⟩,⟨1,3⟩,⟨2,4⟩,⟨3,5⟩,⟨4,0⟩,⟨5+ωn,6⟩,⟨6+ωn,2⟩}

definition
  sep_env :: i ⇒ i where
  sep_env(n) ≡ λ i ∈ (5+ωn)-5 . i+ω2

definition weak :: [i, i] ⇒ i where
  weak(n,m) ≡ {i+ωm . i ∈ n}

lemma weakD :
  assumes n ∈ nat k ∈ nat x ∈ weak(n,k)
  shows ∃ i ∈ n . x = i+ωk
  using assms unfolding weak_def by blast

lemma weak_equal :
  assumes n ∈ nat m ∈ nat
  shows weak(n,m) = (m+ωn) - m
proof -
  have weak(n,m) ⊆ (m+ωn)-m
  proof(intro subsetI)
    fix x
    assume x ∈ weak(n,m)
    with assms
    obtain i where
      i ∈ n x = i+ωm
      using weakD by blast
    then
    have m ≤ i+ωm i < n
      using add_le_self2[of m i] ‹m ∈ nat› ‹n ∈ nat› ltI[OF ‹i ∈ n›] by simp_all
    then

```

```

have  $\neg i +_{\omega} m < m$ 
  using not_lt_iff_le_in_n_in_nat[OF  $\langle n \in \text{nat} \rangle \langle i \in n \rangle$ ]  $\langle m \in \text{nat} \rangle$  by simp
with  $\langle x = i +_{\omega} m \rangle$ 
have  $x \notin m$ 
  using ltI  $\langle m \in \text{nat} \rangle$  by auto
moreover
from assms  $\langle x = i +_{\omega} m \rangle \langle i < n \rangle$ 
have  $x < m +_{\omega} n$ 
  using add_lt_mono1[OF  $\langle i < n \rangle \langle n \in \text{nat} \rangle$ ] by simp
ultimately
show  $x \in (m +_{\omega} n) - m$ 
  using ltD DiffI by simp
qed
moreover
have  $(m +_{\omega} n) - m \subseteq \text{weak}(n, m)$ 
proof (intro subsetI)
  fix  $x$ 
  assume  $x \in (m +_{\omega} n) - m$ 
  then
  have  $x \in m +_{\omega} n$   $x \notin m$ 
    using DiffD1[of  $x$   $n +_{\omega} m$   $m$ ] DiffD2[of  $x$   $n +_{\omega} m$   $m$ ] by simp_all
  then
  have  $x < m +_{\omega} n$   $x \in \text{nat}$ 
    using ltI in_n_in_nat[OF add_type[of  $m$   $n$ ]] by simp_all
  then
  obtain  $i$  where
     $m +_{\omega} n = \text{succ}(x +_{\omega} i)$ 
    using less_iff_succ_add[OF  $\langle x \in \text{nat} \rangle$ , of  $m +_{\omega} n$ ] add_type by auto
  then
  have  $x +_{\omega} i < m +_{\omega} n$  using succ_le_iff by simp
  with  $\langle x \notin m \rangle$ 
  have  $\neg x < m$  using ltD by blast
  with  $\langle m \in \text{nat} \rangle \langle x \in \text{nat} \rangle$ 
  have  $m \leq x$  using not_lt_iff_le by simp
  with  $\langle x < m +_{\omega} n \rangle \langle n \in \text{nat} \rangle$ 
  have  $x -_{\omega} m < m +_{\omega} n -_{\omega} m$ 
    using diff_mono[OF  $\langle x \in \text{nat} \rangle \_ \langle m \in \text{nat} \rangle$ ] by simp
  have  $m +_{\omega} n -_{\omega} m = n$  using diff_cancel2  $\langle m \in \text{nat} \rangle \langle n \in \text{nat} \rangle$  by simp
  with  $\langle x -_{\omega} m < m +_{\omega} n -_{\omega} m \rangle \langle x \in \text{nat} \rangle$ 
  have  $x -_{\omega} m \in n$   $x = x -_{\omega} m +_{\omega} m$ 
    using ltD add_diff_inverse2[OF  $\langle m \leq x \rangle$ ] by simp_all
  then
  show  $x \in \text{weak}(n, m)$ 
    unfolding weak_def by auto
qed
ultimately
show ?thesis by auto
qed

```

```

lemma weak_zero:
  shows  $weak(0,n) = 0$ 
  unfolding weak_def by simp

lemma weakening_diff :
  assumes  $n \in nat$ 
  shows  $weak(n,7) - weak(n,5) \subseteq \{5+\omega n, 6+\omega n\}$ 
  unfolding weak_def using assms
proof(auto)
{
  fix  $i$ 
  assume  $i \in n \text{ succ}(\text{succ}(\text{nativify}(i))) \neq n \ \forall w \in n. \text{succ}(\text{succ}(\text{nativify}(i))) \neq \text{nativify}(w)$ 
  then
  have  $i < n$ 
    using ltI  $\langle n \in nat \rangle$  by simp
  from  $\langle n \in nat \rangle \ \langle i \in n \rangle \ \langle \text{succ}(\text{succ}(\text{nativify}(i))) \neq n \rangle$ 
  have  $i \in nat \ \text{succ}(\text{succ}(i)) \neq n$  using in_n_in_nat by simp_all
  from  $\langle i < n \rangle$ 
  have  $\text{succ}(i) \leq n$  using succ_leI by simp
  with  $\langle n \in nat \rangle$ 
  consider  $(a) \ \text{succ}(i) = n \mid (b) \ \text{succ}(i) < n$ 
    using leD by auto
  then have  $\text{succ}(i) = n$ 
  proof cases
    case a
    then show ?thesis .
  next
    case b
    then
    have  $\text{succ}(\text{succ}(i)) \leq n$  using succ_leI by simp
    with  $\langle n \in nat \rangle$ 
    consider  $(a) \ \text{succ}(\text{succ}(i)) = n \mid (b) \ \text{succ}(\text{succ}(i)) < n$ 
      using leD by auto
    then have  $\text{succ}(i) = n$ 
    proof cases
      case a
      with  $\langle \text{succ}(\text{succ}(i)) \neq n \rangle$  show ?thesis by blast
    next
      case b
      then
      have  $\text{succ}(\text{succ}(i)) \in n$  using ltD by simp
      with  $\langle i \in nat \rangle$ 
      have  $\text{succ}(\text{succ}(\text{nativify}(i))) \neq \text{nativify}(\text{succ}(\text{succ}(i)))$ 
        using  $\langle \forall w \in n. \text{succ}(\text{succ}(\text{nativify}(i))) \neq \text{nativify}(w) \rangle$  by auto
      then
      have False using  $\langle i \in nat \rangle$  by auto
      then show ?thesis by blast
    qed
  then show ?thesis .
}

```

```

qed
with ⟨i∈nat⟩ have succ(natify(i)) = n by simp
}
then
show n ∈ nat ⇒
  succ(succ(natify(y))) ≠ n ⇒
  ∀ x∈n. succ(succ(natify(y))) ≠ natify(x) ⇒
  y ∈ n ⇒ succ(natify(y)) = n for y
  by blast
qed

```

```

lemma in_add_del :
  assumes x∈j+ωn n∈nat j∈nat
  shows x < j ∨ x ∈ weak(n,j)
proof (cases x<j)
  case True
  then show ?thesis ..
next
  case False
  have x∈nat j+ωn∈nat
    using in_n_in_nat[OF _ ⟨x∈j+ωn⟩] assms by simp_all
  then
  have j ≤ x x < j+ωn
    using not_lt_iff_le False ⟨j∈nat⟩ ⟨n∈nat⟩ lt[OF ⟨x∈j+ωn⟩] by auto
  then
  have x-ωj < (j + ω n) -ω j x = j + ω (x -ω j)
    using diff_mono ⟨x∈nat⟩ ⟨j+ωn∈nat⟩ ⟨j∈nat⟩ ⟨n∈nat⟩
    add_diff_inverse[OF ⟨j≤x⟩] by simp_all
  then
  have x-ωj < n x = (x -ω j) + ω j
    using diff_add_inverse ⟨n∈nat⟩ add_commute by simp_all
  then
  have x-ωj ∈ n using ltD by simp
  then
  have x ∈ weak(n,j)
    unfolding weak_def
    using ⟨x = (x-ωj) + ω j⟩ RepFunI[OF ⟨x-ωj∈n⟩] add_commute by force
  then show ?thesis ..
qed

```

```

lemma sep_env_action:
  assumes
    [t,p,u,P,leq,o,pi] ∈ list(M)
    env ∈ list(M)
  shows ∀ i . i ∈ weak(length(env),5) ⇒
    nth(sep_env(length(env)) 'i,[t,p,u,P,leq,o,pi]@env) = nth(i,[p,P,leq,o,t] @ env
  @ [pi,u])
proof -

```

```

from assms
have A:  $5 +_{\omega} \text{length}(env) \in \text{nat}$   $[p, P, \text{leq}, o, t] \in \text{list}(M)$ 
  by simp_all
let ?f = sep_env( $\text{length}(env)$ )
have EQ:  $\text{weak}(\text{length}(env), 5) = 5 +_{\omega} \text{length}(env) - 5$ 
  using weak_equal_length_type[OF  $\langle env \in \text{list}(M) \rangle$ ] by simp
let ?tgt =  $[t, p, u, P, \text{leq}, o, pi] @ env$ 
let ?src =  $[p, P, \text{leq}, o, t] @ env @ [pi, u]$ 
have  $\text{nth}(\text{nth}(i, [t, p, u, P, \text{leq}, o, pi] @ env)) = \text{nth}(i, [p, P, \text{leq}, o, t] @ env @ [pi, u])$ 
  if  $i \in (5 +_{\omega} \text{length}(env) - 5)$  for i
proof -
  from that
  have 2:  $i \in 5 +_{\omega} \text{length}(env)$   $i \notin 5$   $i \in \text{nat}$   $i -_{\omega} 5 \in \text{nat}$   $i +_{\omega} 2 \in \text{nat}$ 
    using in_n_in_nat[OF  $\langle 5 +_{\omega} \text{length}(env) \in \text{nat} \rangle$ ] by simp_all
  then
  have 3:  $\neg i < 5$  using ltD by force
  then
  have  $5 \leq i$   $2 \leq 5$ 
    using not_lt_iff_le  $\langle i \in \text{nat} \rangle$  by simp_all
  then have  $2 \leq i$  using le_trans[OF  $\langle 2 \leq 5 \rangle$ ] by simp
  from A  $\langle i \in 5 +_{\omega} \text{length}(env) \rangle$ 
  have  $i < 5 +_{\omega} \text{length}(env)$  using ltI by simp
  with  $\langle i \in \text{nat} \rangle$   $\langle 2 \leq i \rangle$  A
  have  $C: i +_{\omega} 2 < 7 +_{\omega} \text{length}(env)$  by simp
  with that
  have B:  $?f^i = i +_{\omega} 2$  unfolding sep_env_def by simp
  from 3 assms(1)  $\langle i \in \text{nat} \rangle$ 
  have  $\neg i +_{\omega} 2 < 7$  using not_lt_iff_le add_le_mono by simp
  from  $\langle i < 5 +_{\omega} \text{length}(env) \rangle$  3  $\langle i \in \text{nat} \rangle$ 
  have  $i -_{\omega} 5 < 5 +_{\omega} \text{length}(env) -_{\omega} 5$ 
    using diff_mono[of  $i$   $5 +_{\omega} \text{length}(env)$   $5, OF$   $\_\_\_\_ \langle i < 5 +_{\omega} \text{length}(env) \rangle$ ]
    not_lt_iff_le[THEN iffD1] by force
  with assms(2)
  have  $i -_{\omega} 5 < \text{length}(env)$  using diff_add_inverse length_type by simp
  have  $\text{nth}(i, ?src) = \text{nth}(i -_{\omega} 5, env @ [pi, u])$ 
    using nth_append[OF A(2)  $\langle i \in \text{nat} \rangle$ ] 3 by simp
  also
  have  $\dots = \text{nth}(i -_{\omega} 5, env)$ 
    using nth_append[OF  $\langle env \in \text{list}(M) \rangle$   $\langle i -_{\omega} 5 \in \text{nat} \rangle$ ]  $\langle i -_{\omega} 5 < \text{length}(env) \rangle$  by
simp
  also
  have  $\dots = \text{nth}(i +_{\omega} 2, ?tgt)$ 
    using nth_append[OF assms(1)  $\langle i +_{\omega} 2 \in \text{nat} \rangle$ ]  $\langle \neg i +_{\omega} 2 < 7 \rangle$  by simp
  ultimately
  have  $\text{nth}(i, ?src) = \text{nth}(\text{nth}(i, ?tgt))$ 
    using B by simp
  then show ?thesis using that by simp
qed
then show ?thesis using EQ by force

```

qed

lemma *sep_env_type* :

assumes $n \in \text{nat}$

shows $\text{sep_env}(n) : (5+\omega n)-5 \rightarrow (7+\omega n)-7$

proof -

let $?h = \text{sep_env}(n)$

from $\langle n \in \text{nat} \rangle$

have $(5+\omega n)+\omega 2 = 7+\omega n$ $7+\omega n \in \text{nat}$ $5+\omega n \in \text{nat}$ by *simp_all*

have

D : $\text{sep_env}(n) 'x \in (7+\omega n)-7$ if $x \in (5+\omega n)-5$ for x

proof -

from $\langle x \in 5+\omega n-5 \rangle$

have $?h 'x = x+\omega 2$ $x < 5+\omega n$ $x \in \text{nat}$

unfolding *sep_env_def* using *ltI_in_n_in_nat[OF $\langle 5+\omega n \in \text{nat} \rangle$]* by *simp_all*

then

have $x+\omega 2 < 7+\omega n$ by *simp*

then

have $x+\omega 2 \in 7+\omega n$ using *ltD* by *simp*

from $\langle x \in 5+\omega n-5 \rangle$

have $x \notin 5$ by *simp*

then have $\neg x < 5$ using *ltD* by *blast*

then have $5 \leq x$ using *not_lt_iff_le* $\langle x \in \text{nat} \rangle$ by *simp*

then have $7 \leq x+\omega 2$ using *add_le_mono* $\langle x \in \text{nat} \rangle$ by *simp*

then have $\neg x+\omega 2 < 7$ using *not_lt_iff_le* $\langle x \in \text{nat} \rangle$ by *simp*

then have $x+\omega 2 \notin 7$ using *ltI* $\langle x \in \text{nat} \rangle$ by *force*

with $\langle x+\omega 2 \in 7+\omega n \rangle$ show $?thesis$ using $\langle ?h 'x = x+\omega 2 \rangle$ *DiffI* by *simp*

qed

then show $?thesis$ unfolding *sep_env_def* using *lam_type* by *simp*

qed

lemma *sep_var_fin_type* :

assumes $n \in \text{nat}$

shows $\text{sep_var}(n) : 7+\omega n \dashv\vdash 7+\omega n$

unfolding *sep_var_def*

using *consI ltD emptyI* by *force*

lemma *sep_var_domain* :

assumes $n \in \text{nat}$

shows $\text{domain}(\text{sep_var}(n)) = 7+\omega n - \text{weak}(n,5)$

proof -

let $?A = \text{weak}(n,5)$

have $A : \text{domain}(\text{sep_var}(n)) \subseteq (7+\omega n)$

unfolding *sep_var_def*

by(*auto simp add: le_natE*)

have $C : x = 5+\omega n \vee x = 6+\omega n \vee x \leq 4$ if $x \in \text{domain}(\text{sep_var}(n))$ for x

using *that* unfolding *sep_var_def* by *auto*

have $D : x < n+\omega 7$ if $x \in 7+\omega n$ for x

using *that* $\langle n \in \text{nat} \rangle$ *ltI* by *simp*

```

have  $\neg 5+\omega n < 5+\omega n$  using  $\langle n \in \text{nat} \rangle$  lt_irrefl[of False] by force
have  $\neg 6+\omega n < 5+\omega n$  using  $\langle n \in \text{nat} \rangle$  by force
have  $R: x < 5+\omega n$  if  $x \in ?A$  for  $x$ 
proof -
  from that
  obtain  $i$  where
     $i < n$   $x = 5+\omega i$ 
  unfolding weak_def
  using ltI  $\langle n \in \text{nat} \rangle$  RepFun_iff by force
  with  $\langle n \in \text{nat} \rangle$ 
  have  $5+\omega i < 5+\omega n$  using add_lt_mono2 by simp
  with  $\langle x = 5+\omega i \rangle$ 
  show  $x < 5+\omega n$  by simp
qed
then
have  $1: x \notin ?A$  if  $\neg x < 5+\omega n$  for  $x$  using that by blast
have  $5+\omega n \notin ?A$   $6+\omega n \notin ?A$ 
proof -
  show  $5+\omega n \notin ?A$  using 1  $\langle \neg 5+\omega n < 5+\omega n \rangle$  by blast
  with 1 show  $6+\omega n \notin ?A$  using  $\langle \neg 6+\omega n < 5+\omega n \rangle$  by blast
qed
then
have  $E: x \notin ?A$  if  $x \in \text{domain}(\text{sep\_var}(n))$  for  $x$ 
  unfolding weak_def
  using C that by force
then
have  $F: \text{domain}(\text{sep\_var}(n)) \subseteq 7+\omega n - ?A$  using A by auto
from assms
have  $x < 7 \vee x \in \text{weak}(n, 7)$  if  $x \in 7+\omega n$  for  $x$ 
  using in_add_del[OF  $\langle x \in 7+\omega n \rangle$ ] by simp
moreover
{
  fix  $x$ 
  assume asm:  $x \in 7+\omega n$   $x \notin ?A$   $x \in \text{weak}(n, 7)$ 
  then
  have  $x \in \text{domain}(\text{sep\_var}(n))$ 
  proof -
    from  $\langle n \in \text{nat} \rangle$ 
    have  $\text{weak}(n, 7) - \text{weak}(n, 5) \subseteq \{n+\omega 5, n+\omega 6\}$ 
      using weakening_diff by simp
    with  $\langle x \notin ?A \rangle$  asm
    have  $x \in \{n+\omega 5, n+\omega 6\}$  using subsetD DiffI by blast
    then
    show thesis unfolding sep_var_def by simp
  qed
}
}
moreover
{
  fix  $x$ 

```

```

assume  $asm: x \in \gamma +_{\omega} n \quad x \notin ?A \quad x < \gamma$ 
then have  $x \in \text{domain}(\text{sep\_var}(n))$ 
proof (cases  $2 \leq n$ )
  case True
  moreover
  have  $0 < n$  using  $leD[OF \langle n \in \text{nat} \rangle \langle 2 \leq n \rangle] \text{lt\_imp\_0\_lt}$  by auto
  ultimately
  have  $x < 5$ 
    using  $\langle x < \gamma \rangle \langle x \notin ?A \rangle \langle n \in \text{nat} \rangle \text{in\_n\_in\_nat}$ 
    unfolding weak_def
    by (clarsimp simp add: not_lt_iff_le, auto simp add: lt_def)
  then
  show ?thesis unfolding sep_var_def
    by (clarsimp simp add: not_lt_iff_le, auto simp add: lt_def)
next
  case False
  then
  show ?thesis
  proof (cases  $n=0$ )
    case True
    then show ?thesis
      unfolding sep_var_def using  $ltD \text{asm} \langle n \in \text{nat} \rangle$  by auto
  next
  case False
  then
  have  $n < 2$  using  $\langle n \in \text{nat} \rangle \text{not\_lt\_iff\_le} \langle \neg 2 \leq n \rangle$  by force
  then
  have  $\neg n < 1$  using  $\langle n \neq 0 \rangle$  by simp
  then
  have  $n=1$  using  $\text{not\_lt\_iff\_le} \langle n < 2 \rangle \text{le\_iff}$  by auto
  then show ?thesis
    using  $\langle x \notin ?A \rangle$ 
    unfolding weak_def sep_var_def
    using  $ltD \text{asm} \langle n \in \text{nat} \rangle$  by force
  qed
qed
}
ultimately
have  $w \in \text{domain}(\text{sep\_var}(n))$  if  $w \in \gamma +_{\omega} n - ?A$  for  $w$ 
  using that by blast
then
have  $\gamma +_{\omega} n - ?A \subseteq \text{domain}(\text{sep\_var}(n))$  by blast
with  $F$ 
show ?thesis by auto
qed

lemma sep_var_type :
assumes  $n \in \text{nat}$ 
shows  $\text{sep\_var}(n) : (\gamma +_{\omega} n)\text{-weak}(n, 5) \rightarrow \gamma +_{\omega} n$ 

```

```

using FiniteFun_is_fun[OF sep_var_fin_type[OF  $\langle n \in \text{nat} \rangle$ ]]
      sep_var_domain[OF  $\langle n \in \text{nat} \rangle$ ] by simp

lemma sep_var_action :
  assumes
     $[t, p, u, P, \text{leq}, o, pi] \in \text{list}(M)$ 
     $env \in \text{list}(M)$ 
  shows  $\forall i . i \in (7 +_{\omega} \text{length}(env)) - \text{weak}(\text{length}(env), 5) \longrightarrow$ 
     $\text{nth}(\text{sep\_var}(\text{length}(env)) \text{ ' } i, [t, p, u, P, \text{leq}, o, pi] @ env) = \text{nth}(i, [p, P, \text{leq}, o, t] @ env$ 
  @  $[pi, u])$ 
  using assms
proof (subst sep_var_domain[OF length_type[OF  $\langle env \in \text{list}(M) \rangle$ ], symmetric], auto)
  fix  $i y$ 
  assume  $\langle i, y \rangle \in \text{sep\_var}(\text{length}(env))$ 
  with assms
  show  $\text{nth}(\text{sep\_var}(\text{length}(env)) \text{ ' } i,$ 
     $\text{Cons}(t, \text{Cons}(p, \text{Cons}(u, \text{Cons}(P, \text{Cons}(\text{leq}, \text{Cons}(o, \text{Cons}(pi, env))))))))$ 
  =
     $\text{nth}(i, \text{Cons}(p, \text{Cons}(P, \text{Cons}(\text{leq}, \text{Cons}(o, \text{Cons}(t, env @ [pi, u]))))))$ 
  using apply_fun[OF sep_var_type] assms
  unfolding sep_var_def
  using nth_concat2[OF  $\langle env \in \text{list}(M) \rangle$ ] nth_concat3[OF  $\langle env \in \text{list}(M) \rangle$ , symmetric]
  by force
qed

definition
  rensep ::  $i \Rightarrow i$  where
  rensep( $n$ )  $\equiv \text{union\_fun}(\text{sep\_var}(n), \text{sep\_env}(n), 7 +_{\omega} n - \text{weak}(n, 5), \text{weak}(n, 5))$ 

lemma rensep_aux :
  assumes  $n \in \text{nat}$ 
  shows  $(7 +_{\omega} n - \text{weak}(n, 5)) \cup \text{weak}(n, 5) = 7 +_{\omega} n \cup (7 +_{\omega} n - 7) = 7 +_{\omega} n$ 
proof -
  from  $\langle n \in \text{nat} \rangle$ 
  have  $\text{weak}(n, 5) = n +_{\omega} 5 - 5$ 
  using weak_equal by simp
  with  $\langle n \in \text{nat} \rangle$ 
  show  $(7 +_{\omega} n - \text{weak}(n, 5)) \cup \text{weak}(n, 5) = 7 +_{\omega} n \cup (7 +_{\omega} n - 7) = 7 +_{\omega} n$ 
  using Diff_partition le_imp_subset by auto
qed

lemma rensep_type :
  assumes  $n \in \text{nat}$ 
  shows  $\text{rensep}(n) \in 7 +_{\omega} n \rightarrow 7 +_{\omega} n$ 
proof -
  from  $\langle n \in \text{nat} \rangle$ 
  have  $\text{rensep}(n) \in (7 +_{\omega} n - \text{weak}(n, 5)) \cup \text{weak}(n, 5) \rightarrow 7 +_{\omega} n \cup (7 +_{\omega} n - 7)$ 
  unfolding rensep_def
  using union_fun_type sep_var_type  $\langle n \in \text{nat} \rangle$  sep_env_type weak_equal

```

by force
 then
 show *?thesis* using *rensep_aux* $\langle n \in \text{nat} \rangle$ by auto
 qed

lemma *rensep_action* :

assumes $[t, p, u, P, \text{leq}, o, pi] @ \text{env} \in \text{list}(M)$
 shows $\forall i . i < 7 + \omega \text{length}(\text{env}) \longrightarrow \text{nth}(\text{rensep}(\text{length}(\text{env})) 'i, [t, p, u, P, \text{leq}, o, pi] @ \text{env})$
 $= \text{nth}(i, [p, P, \text{leq}, o, t] @ \text{env} @ [pi, u])$

proof -

let $?tgt = [t, p, u, P, \text{leq}, o, pi] @ \text{env}$
 let $?src = [p, P, \text{leq}, o, t] @ \text{env} @ [pi, u]$
 let $?m = 7 + \omega \text{length}(\text{env}) - \text{weak}(\text{length}(\text{env}), 5)$
 let $?p = \text{weak}(\text{length}(\text{env}), 5)$
 let $?f = \text{sep_var}(\text{length}(\text{env}))$
 let $?g = \text{sep_env}(\text{length}(\text{env}))$
 let $?n = \text{length}(\text{env})$
 from *assms*
 have $1 : [t, p, u, P, \text{leq}, o, pi] \in \text{list}(M) \ \text{env} \in \text{list}(M)$
 $?src \in \text{list}(M) \ ?tgt \in \text{list}(M)$
 $7 + \omega ?n = (7 + \omega ?n - \text{weak}(?n, 5)) \cup \text{weak}(?n, 5)$
 $\text{length}(?src) = (7 + \omega ?n - \text{weak}(?n, 5)) \cup \text{weak}(?n, 5)$
 using *Diff_partition le_imp_subset rensep_aux* by auto
 then
 have $\text{nth}(i, ?src) = \text{nth}(\text{union_fun} (?f, ?g, ?m, ?p) 'i, ?tgt)$ if $i < 7 + \omega \text{length}(\text{env})$
 for i

proof -

from $\langle i < 7 + \omega ?n \rangle$
 have $i \in (7 + \omega ?n - \text{weak}(?n, 5)) \cup \text{weak}(?n, 5)$
 using *ltD by simp*
 then show *?thesis*
 unfolding *rensep_def* using
 $\text{union_fun_action}[OF \langle ?src \in \text{list}(M) \rangle \langle ?tgt \in \text{list}(M) \rangle \langle \text{length}(?src) = (7 + \omega ?n - \text{weak}(?n, 5)) \cup \text{weak}(?n, 5) \rangle$
 $\text{sep_var_action}[OF \langle [t, p, u, P, \text{leq}, o, pi] \in \text{list}(M) \rangle \langle \text{env} \in \text{list}(M) \rangle]$
 $\text{sep_env_action}[OF \langle [t, p, u, P, \text{leq}, o, pi] \in \text{list}(M) \rangle \langle \text{env} \in \text{list}(M) \rangle]$
 $] \text{that}$
 by *simp*
 qed
 then show *?thesis* unfolding *rensep_def* by *simp*
 qed

definition *sep_ren* :: $[i, i] \Rightarrow i$ where

$\text{sep_ren}(n, \varphi) \equiv \text{ren}(\varphi) '(7 + \omega n) '(7 + \omega n) 'rensep(n)$

lemma *arity_resep*: assumes $\varphi \in \text{formula} \ \text{env} \in \text{list}(M)$

$\text{arity}(\varphi) \leq 7 + \omega \text{length}(\text{env})$

shows $\text{arity}(\text{sep_ren}(\text{length}(\text{env}), \varphi)) \leq 7 + \omega \text{length}(\text{env})$

unfolding *sep_ren_def*

```

using arity_ren renssep_type assms
by simp

lemma type_renssep [TC]:
assumes  $\varphi \in \text{formula}$   $env \in \text{list}(M)$ 
shows  $\text{sep\_ren}(\text{length}(env), \varphi) \in \text{formula}$ 
unfolding sep_ren_def
using ren_tc renssep_type assms
by simp

lemma sepren_action:
assumes  $\text{arity}(\varphi) \leq 7 +_{\omega} \text{length}(env)$ 
 $[t, p, u, P, \text{leq}, o, pi] \in \text{list}(M)$ 
 $env \in \text{list}(M)$ 
 $\varphi \in \text{formula}$ 
shows  $\text{sats}(M, \text{sep\_ren}(\text{length}(env), \varphi), [t, p, u, P, \text{leq}, o, pi] @ env) \longleftrightarrow \text{sats}(M,$ 
 $\varphi, [p, P, \text{leq}, o, t] @ env @ [pi, u])$ 
proof -
from assms
have 1:  $[t, p, u, P, \text{leq}, o, pi] @ env \in \text{list}(M)$ 
by simp_all
then
have 2:  $[p, P, \text{leq}, o, t] @ env @ [pi, u] \in \text{list}(M)$ 
using app_type by simp
show ?thesis
unfolding sep_ren_def
using sats_iff_sats_ren[OF  $\langle \varphi \in \text{formula} \rangle$ ]
add_type[of 7 length(env)]
add_type[of 7 length(env)]
2 1
renssep_type[OF length_type[OF  $\langle env \in \text{list}(M) \rangle$ ]]
 $\langle \text{arity}(\varphi) \leq 7 +_{\omega} \text{length}(env) \rangle$ ]
renssep_action[OF 1, rule_format, symmetric]
by simp
qed

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  $env \in \text{list}(M[G])$ 

```

```

shows  $\exists env \in list(M). env = map(val(G), env)$ 
using assms
proof(induct env)
  case Nil
  have  $map(val(G), Nil) = Nil$  by simp
  then show ?case by force
next
  case (Cons a l)
  then obtain a' l' where
     $l' \in list(M) \ l = map(val(G), l') \ a = val(G, a')$ 
     $Cons(a, l) = map(val(G), Cons(a', l')) \ Cons(a', l') \in list(M)$ 
  using GenExtD
  by force
  then show ?case by force
qed

```

lemma *Collect_sats_in_MG* :

```

assumes
   $A \in M[G]$ 
   $\varphi \in formula \ env \in list(M[G]) \ arity(\varphi) \leq 1 +_{\omega} length(env)$ 
shows
   $\{x \in A . (M[G], [x] @ env \models \varphi)\} \in M[G]$ 
proof -
  from  $\langle A \in M[G] \rangle$ 
  obtain  $\pi$  where  $\pi \in M \ val(G, \pi) = A$ 
  using GenExt_def by auto
  then
  have  $domain(\pi) \in M \ domain(\pi) \times \mathbb{P} \in M$ 
  using cartprod_closed[of  $\_ \mathbb{P}$ , simplified]
  by (simp_all flip:setclass_iff)
  let  $?X = \dots 0 \in (1 +_{\omega} length(env)) \cdot \wedge \varphi \cdot$ 
  let  $?new\_form = sep\_ren(length(env), forces(?X))$ 
  let  $?\psi = (\cdot \exists (\cdot \exists \cdot (0, 1) \text{ is } 2 \cdot \wedge ?new\_form \cdot \cdot))$ 
  note  $\phi = \langle \varphi \in formula \rangle \langle arity(\varphi) \leq 1 +_{\omega} length(env) \rangle$ 
  then
  have  $?X \in formula \ forces(?X) \in formula \ arity(\varphi) \leq 2 +_{\omega} length(env)$ 
  using definability le_trans[OF  $\langle arity(\varphi) \leq \_ \rangle$ ] add_le_mono[of 1 2, OF  $\_ le_refl$ ]
  by simp_all
  with  $\langle env \in \_ \rangle \phi$ 
  have  $arity(?X) \leq 2 +_{\omega} length(env)$ 
  using ord_simp_union leI FOL_aritys by simp
  with  $\langle env \in list(\_) \rangle \phi$ 
  have  $arity(forces(?X)) \leq 6 +_{\omega} length(env)$ 
  using arity_forces_le by simp
  then
  have  $arity(forces(?X)) \leq 7 +_{\omega} length(env)$ 
  using ord_simp_union arity_forces leI by simp
  with  $\langle arity(forces(?X)) \leq 7 +_{\omega} \_ \rangle \langle env \in \_ \rangle \langle \varphi \in formula \rangle$ 
  have  $arity(?new\_form) \leq 7 +_{\omega} length(env) \ ?new\_form \in formula \ ?\psi \in formula$ 

```

```

    using arity_resep[OF definability[of ?χ]]
  by auto
then
have arity(?ψ) ≤ 5 +ω length(env)
  using ord_simp_union arity_forces pred_mono[OF _ pred_mono[OF _ ⟨arity(?new_form) ≤ _⟩]]
  by (auto simp:arity)
from ⟨env ∈ _⟩
  obtain nenv where nenv∈list(M) env = map(val(G),nenv) length(nenv) =
length(env)
  using map_val by auto
from phi ⟨nenv∈_⟩ ⟨env∈_⟩ ⟨π∈M⟩ ⟨φ∈_⟩ ⟨length(nenv) = length(env)⟩
have arity(?χ) ≤ length([∅] @ nenv @ [π]) for ∅
  using union_abs2[OF ⟨arity(φ) ≤ 2+ω _⟩] ord_simp_union FOL_aritys
  by simp
note in_M = ⟨π∈M⟩ ⟨domain(π) × ℙ ∈ M⟩
have Equivalence:
  (M, [u,ℙ,leq,1,π] @ nenv ⊨ ?ψ) ↔
  (∃ ∅∈M. ∃ p∈ℙ. u = ⟨∅, p⟩ ∧
  (∀ F. M_generic(F) ∧ p ∈ F → M[F], map(val(F), [∅] @ nenv @ [π]) ⊨
?χ))
  if u ∈ domain(π) × ℙ
  for u
proof -
  from ⟨u ∈ domain(π) × ℙ⟩ ⟨domain(π) × ℙ ∈ M⟩
  have u∈M by (simp add:transitivity)
  have (M, [∅, p, u, ℙ, leq, 1, π] @ nenv ⊨ ?new_form) ↔
  (∀ F. M_generic(F) ∧ p ∈ F → (M[F], map(val(F), [∅] @ nenv @ [π]) ⊨
?χ))
  if ∅∈M p∈ℙ
  for ∅ p
proof -
  from ⟨p∈ℙ⟩
  have p∈M by (simp add:transitivity)
  let ?env=[p,ℙ,leq,1,∅] @ nenv @ [π, u]
  let ?new_env= [∅, p, u, ℙ, leq, 1, π] @ nenv
  note types = in_M ⟨∅ ∈ M⟩ ⟨p∈M⟩ ⟨u ∈ domain(π) × ℙ⟩ ⟨u ∈ M⟩ ⟨nenv∈_⟩
  then
  have tyenv:?env ∈ list(M) ?new_env ∈ list(M)
    by simp_all
  from types
  have eq_env:[p, ℙ, leq, 1] @ ([∅] @ nenv @ [π, u]) =
    ([p, ℙ, leq, 1] @ ([∅] @ nenv @ [π])) @ [u]
    using app_assoc by simp
  then
  have (M, [∅, p, u, ℙ, leq, 1, π] @ nenv ⊨ ?new_form) ↔ (M, ?new_env ⊨
?new_form)
    by simp
  from tyenv ⟨length(nenv) = length(env)⟩ ⟨arity(forces(?χ)) ≤ 7 +ω length(env)⟩

```

$\langle \text{forces}(\ ?\chi) \in \text{formula} \rangle$
have ... $\longleftrightarrow p \Vdash \ ?\chi \ ([\vartheta] @ \text{nenv} @ [\pi, u])$
using *sepren_action*[of *forces*($\ ?\chi$) *nenv*, *OF* $_ _ _ \langle \text{nenv} \in \text{list}(M) \rangle$]
by *simp*
also from *types phi* $\langle \text{env} \in _ \rangle \langle \text{length}(\text{nenv}) = \text{length}(\text{env}) \rangle \langle \text{arity}(\text{forces}(\ ?\chi)) \leq 6 + \omega \text{length}(\text{env}) \rangle$
have ... $\longleftrightarrow p \Vdash \ ?\chi \ ([\vartheta] @ \text{nenv} @ [\pi])$
by (*subst eq_env*, *rule_tac arity_sats_iff*, *auto*)
also from *types phi* $\langle p \in \mathbb{P} \rangle \langle \text{arity}(\text{forces}(\ ?\chi)) \leq 6 + \omega \text{length}(\text{env}) \rangle \langle \text{arity}(\ ?\chi) \leq \text{length}([\vartheta] @ \text{nenv} @ [\pi]) \rangle$
have ... $\longleftrightarrow (\forall F . M_generic(F) \wedge p \in F \longrightarrow M[F], \text{map}(\text{val}(F), [\vartheta] @ \text{nenv} @ [\pi]) \models \ ?\chi)$
using *definition_of_forcing*[**where** $\varphi = \dots \ 0 \in (1 + \omega \text{length}(\text{env})) \cdot \wedge \varphi \cdot$]
by *auto*
finally
show *?thesis*
by *simp*
qed
with *in_M* $\langle \ ?\text{new_form} \in \text{formula} \rangle \langle \ ?\psi \in \text{formula} \rangle \langle \text{nenv} \in _ \rangle \langle u \in \text{domain}(\pi) \times \mathbb{P} \rangle$
show *?thesis*
by (*auto simp add: transitivity*)
qed
moreover from $\langle \text{env} = _ \rangle \langle \pi \in M \rangle \langle \text{nenv} \in \text{list}(M) \rangle$
have *map_nenv*: $\text{map}(\text{val}(G), \text{nenv} @ [\pi]) = \text{env} @ [\text{val}(G, \pi)]$
using *map_app_distrib append1_eq_iff* **by** *auto*
ultimately
have *aux*: $(\exists \vartheta \in M. \exists p \in \mathbb{P}. u = \langle \vartheta, p \rangle \wedge (p \in G \longrightarrow M[G], [\text{val}(G, \vartheta)] @ \text{env} @ [\text{val}(G, \pi)] \models \ ?\chi))$
(is $(\exists \vartheta \in M. \exists p \in \mathbb{P}. _ \ (_ \longrightarrow M[G], \ ?\text{vals}(\vartheta) \models _))$)
if $u \in \text{domain}(\pi) \times \mathbb{P} \ M, [u, \mathbb{P}, \text{leq}, \mathbf{1}, \pi] @ \text{nenv} \models \ ?\psi$ **for** u
using *Equivalence*[*THEN iffD1*, *OF that*] **generic by force**
moreover
have $[\text{val}(G, \vartheta)] @ \text{env} @ [\text{val}(G, \pi)] \in \text{list}(M[G])$ **if** $\vartheta \in M$ **for** ϑ
using $\langle \pi \in M \rangle \langle \text{env} \in \text{list}(M[G]) \rangle$ *GenExtI* **that by force**
ultimately
have $(\exists \vartheta \in M. \exists p \in \mathbb{P}. u = \langle \vartheta, p \rangle \wedge (p \in G \longrightarrow \text{val}(G, \vartheta) \in \text{nth}(1 + \omega \text{length}(\text{env}), [\text{val}(G, \vartheta)] @ \text{env} @ [\text{val}(G, \pi)]))$
 $\wedge (M[G], \ ?\text{vals}(\vartheta) \models \varphi))$
if $u \in \text{domain}(\pi) \times \mathbb{P} \ M, [u, \mathbb{P}, \text{leq}, \mathbf{1}, \pi] @ \text{nenv} \models \ ?\psi$ **for** u
using *aux*[*OF that*] **by** *simp*
moreover from $\langle \text{env} \in _ \rangle \langle \pi \in M \rangle$
have *nth*: $\text{nth}(1 + \omega \text{length}(\text{env}), [\text{val}(G, \vartheta)] @ \text{env} @ [\text{val}(G, \pi)]) = \text{val}(G, \pi)$
if $\vartheta \in M$ **for** ϑ
using *nth_concat*[of $\text{val}(G, \vartheta)$ $\text{val}(G, \pi)$ $M[G]$] **that GenExtI by simp**
ultimately
have $(\exists \vartheta \in M. \exists p \in \mathbb{P}. u = \langle \vartheta, p \rangle \wedge (p \in G \longrightarrow \text{val}(G, \vartheta) \in \text{val}(G, \pi) \wedge (M[G], \ ?\text{vals}(\vartheta) \models \varphi))$
 $\models \varphi$)
if $u \in \text{domain}(\pi) \times \mathbb{P} \ M, [u, \mathbb{P}, \text{leq}, \mathbf{1}, \pi] @ \text{nenv} \models \ ?\psi$ **for** u

```

using that  $\langle \pi \in M \rangle \langle env \in \_ \rangle$  by simp
with  $\langle domain(\pi) \times \mathbb{P} \in M \rangle$ 
have  $\forall u \in domain(\pi) \times \mathbb{P} . (M, [u, \mathbb{P}, leq, \mathbf{1}, \pi] @ nenv \models ?\psi) \longrightarrow (\exists \vartheta \in M . \exists p \in \mathbb{P} .$ 
 $u = \langle \vartheta, p \rangle \wedge$ 
 $(p \in G \longrightarrow val(G, \vartheta) \in val(G, \pi) \wedge (M[G], ?vals(\vartheta) \models \varphi)))$ 
by (simp add:transitivity)
then
have  $\{u \in domain(\pi) \times \mathbb{P} . (M, [u, \mathbb{P}, leq, \mathbf{1}, \pi] @ nenv \models ?\psi)\} \subseteq$ 
 $\{u \in domain(\pi) \times \mathbb{P} . \exists \vartheta \in M . \exists p \in \mathbb{P} . u = \langle \vartheta, p \rangle \wedge$ 
 $(p \in G \longrightarrow val(G, \vartheta) \in val(G, \pi) \wedge (M[G], ?vals(\vartheta) \models \varphi))\}$ 
(is ?n $\subseteq$ ?m)
by auto
then
have first_incl:  $val(G, ?n) \subseteq val(G, ?m)$ 
using val_mono by simp
note  $\langle val(G, \pi) = A \rangle$ 
with  $\langle ?\psi \in formula \rangle \langle arity(?\psi) \leq \_ \rangle$  in  $_M \langle nenv \in \_ \rangle \langle env \in \_ \rangle \langle length(nenv)$ 
 $= \_ \rangle$ 
have  $?n \in M$ 
using separation_ax leI separation_iff by auto
from generic
have filter( $G$ )  $G \subseteq \mathbb{P}$ 
by auto
from  $\langle val(G, \pi) = A \rangle$ 
have  $val(G, ?m) =$ 
 $\{z . t \in domain(\pi) , (\exists q \in \mathbb{P} .$ 
 $(\exists \vartheta \in M . \exists p \in \mathbb{P} . \langle t, q \rangle = \langle \vartheta, p \rangle \wedge$ 
 $(p \in G \longrightarrow val(G, \vartheta) \in A \wedge (M[G], [val(G, \vartheta)] @ env @ [A] \models \varphi)) \wedge$ 
 $q \in G)\} \wedge$ 
 $z = val(G, t)\}$ 
using val_of_name by auto
also
have  $\dots = \{z . t \in domain(\pi) , (\exists q \in \mathbb{P} .$ 
 $val(G, t) \in A \wedge (M[G], [val(G, t)] @ env @ [A] \models \varphi) \wedge q \in G)$ 
 $\wedge z = val(G, t)\}$ 
using  $\langle domain(\pi) \in M \rangle$  by (auto simp add:transitivity)
also
have  $\dots = \{x \in A . \exists q \in \mathbb{P} . x \in A \wedge (M[G], [x] @ env @ [A] \models \varphi) \wedge q \in G\}$ 
proof(intro equalityI, auto)

{
fix  $x q$ 
assume  $M[G], Cons(x, env @ [A]) \models \varphi$   $x \in A$   $q \in \mathbb{P}$   $q \in G$ 
from this  $\langle val(G, \pi) = A \rangle$ 
show  $x \in \{y . x \in domain(\pi), val(G, x) \in A \wedge (M[G], Cons(val(G, x), env$ 
 $@ [A]) \models \varphi) \wedge (\exists q \in \mathbb{P} . q \in G) \wedge y = val(G, x)\}$ 
using elem_of_val by force
}
qed

```

```

also
have ... = {x ∈ A. (M[G], [x] @ env @ [A] ⊨ φ)}
  using ‹G ⊆ P› G_nonempty by force
finally
have val_m: val(G, ?m) = {x ∈ A. (M[G], [x] @ env @ [A] ⊨ φ)} by simp
have val(G, ?m) ⊆ val(G, ?n)
proof
  fix x
  assume x ∈ val(G, ?m)
  with val_m
  have x ∈ {x ∈ A. (M[G], [x] @ env @ [A] ⊨ φ)} by simp
  with ‹val(G, π) = A›
  have x ∈ val(G, π) by simp
  then
  obtain ϑ q where ‹ϑ, q⟩ ∈ π q ∈ G val(G, ϑ) = x ϑ ∈ M
    using elem_of_val_pair domain_trans[OF trans_M ‹π ∈ _›]
    by force
  with ‹π ∈ M› ‹nenv ∈ _› ‹env = _›
  have [val(G, ϑ), val(G, π)] @ env ∈ list(M[G]) [ϑ] @ nenv @ [π] ∈ list(M)
    using GenExt_def by auto
  with ‹val(G, ϑ) = x› ‹val(G, π) = A› ‹x ∈ val(G, π)› nth ‹ϑ ∈ M› ‹x ∈ {x ∈ A .
  _}›
  have M[G], [val(G, ϑ)] @ env @ [val(G, π)] ⊨ .. 0 ∈ (1 +ω length(env)) · ∧ φ ·
    by auto
    — Recall ..0 ∈ 1 +ω length(env) · ∧ φ = ..0 ∈ 1 +ω length(env) · ∧ φ ·
  with ‹[_] @ nenv @ [_] ∈ _› map_nenv ‹arity(?χ) ≤ length(_› ‹length(nenv)
  = _›
  obtain r where r ∈ G r ⊨ ?χ ([ϑ] @ nenv @ [π])
    using truth_lemma[OF ‹?χ ∈ _›, of [ϑ] @ nenv @ [π]]
    by auto
  with ‹filter(G)› and ‹q ∈ G›
  obtain p where p ∈ G p ≤ q p ≤ r
    unfolding filter_def compat_in_def by force
  with ‹r ∈ G› ‹q ∈ G› ‹G ⊆ P›
  have p ∈ P r ∈ P q ∈ P p ∈ M
    using transitivity[OF _ P_in_M] subsetD
    by simp_all
  with ‹φ ∈ formula› ‹ϑ ∈ M› ‹π ∈ M› ‹p ≤ r› ‹nenv ∈ _› ‹arity(?χ) ≤ length(_›
  ‹r ⊨ ?χ _› ‹env ∈ _›
  have p ⊨ ?χ ([ϑ] @ nenv @ [π])
    using strengthening_lemma
    by simp
  with ‹p ∈ P› ‹φ ∈ formula› ‹ϑ ∈ M› ‹π ∈ M› ‹nenv ∈ _› ‹arity(?χ) ≤ length(_›
  have ∀ F. M_generic(F) ∧ p ∈ F ⟶
    M[F], map(val(F), [ϑ] @ nenv @ [π]) ⊨ ?χ
    using definition_of_forcing[where φ = .. 0 ∈ (1 +ω length(env)) · ∧ φ · ]
    by simp
  with ‹p ∈ P› ‹ϑ ∈ M›
  have Eq6: ∃ ϑ' ∈ M. ∃ p' ∈ P. ‹ϑ, p⟩ = ‹ϑ', p'⟩ ∧ (∀ F. M_generic(F) ∧ p' ∈ F

```

→

```

      M[F], map(val(F), [∅] @ nenv @ [π]) ⊨ ?χ) by auto
from ⟨π∈M⟩ ⟨∅,q⟩∈π ⟨∅∈M⟩ ⟨p∈ℙ⟩ ⟨p∈M⟩
have ⟨∅,q⟩ ∈ M ⟨∅,p⟩∈M ⟨∅,p⟩∈domain(π)×ℙ
      using pair_in_M_iff transitivity
      by auto
with ⟨∅∈M⟩ Eq6 ⟨p∈ℙ⟩
have M, [⟨∅,p⟩,ℙ,leq,1,π] @ nenv ⊨ ?ψ
      using Equivalence by auto
with ⟨∅,p⟩∈domain(π)×ℙ
have ⟨∅,p⟩∈?n by simp
with ⟨p∈G⟩ ⟨p∈ℙ⟩
have val(G,∅)∈val(G,?n)
      using val_of_elem[of ∅ p] by simp
with ⟨val(G,∅)=x⟩
show x∈val(G,?n) by simp
qed
with val_m_first_incl
have val(G,?n) = {x ∈ A. (M[G], [x] @ env @ [A] ⊨ φ)} by auto
also from ⟨A∈_⟩ phi ⟨env ∈ _⟩
have ... = {x ∈ A. (M[G], [x] @ env ⊨ φ)}
      using arity_sats_iff[where env=[_]@env] transitivity_MG
      by auto
finally
show {x ∈ A. (M[G], [x] @ env ⊨ φ)} ∈ M[G]
      using ⟨?n∈M⟩ GenExt_def by force
qed

theorem separation_in_MG:
  assumes
    φ∈formula and arity(φ) ≤ 1 + ω length(env) and env∈list(M[G])
  shows
    separation(##M[G],λx. (M[G], [x] @ env ⊨ φ))
proof -
  {
    fix A
    assume A∈M[G]
    moreover from ⟨env ∈ _⟩
      obtain nenv where nenv∈list(M)env = map(val(G),nenv) length(env) =
length(nenv)
      using GenExt_def map_val[of env] by auto
    moreover note ⟨φ ∈ _⟩ ⟨arity(φ) ≤ _⟩ ⟨env ∈ _⟩
    ultimately
    have {x ∈ A . (M[G], [x] @ env ⊨ φ)} ∈ M[G]
      using Collect_sats_in_MG by auto
  }
then
show ?thesis
  using separation_iff rev_bexI unfolding is_Collect_def by force

```

```

qed

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 :
   $1 \in G \implies \text{val}(G, \{\langle \tau, 1 \rangle, \langle \rho, 1 \rangle\}) = \{\text{val}(G, \tau), \text{val}(G, \rho)\}$ 
  by (rule trans, subst def_val, auto)

lemma pairing_in_MG : upair_ax(## $M[G]$ )
proof -
  {
    fix x y
    assume  $x \in M[G]$   $y \in M[G]$ 
    moreover from this
    obtain  $\tau \ \rho$  where  $\text{val}(G, \tau) = x$   $\text{val}(G, \rho) = y$   $\rho \in M$   $\tau \in M$ 
      using GenExtD by blast
    moreover from this
    have  $\langle \tau, 1 \rangle \in M$   $\langle \rho, 1 \rangle \in M$ 
      using pair_in_M_iff by auto
    moreover from this
    have  $\{\langle \tau, 1 \rangle, \langle \rho, 1 \rangle\} \in M$  (is  $?\sigma \in \_$ )
      using upair_in_M_iff by simp
    moreover from this
    have  $\text{val}(G, ?\sigma) \in M[G]$ 
      using GenExtI by simp
    moreover from calculation
    have  $\{\text{val}(G, \tau), \text{val}(G, \rho)\} \in M[G]$ 
      using val_Upair one_in_G by simp
    ultimately
    have  $\{x, y\} \in M[G]$ 
      by simp
  }
then
show ?thesis
  unfolding upair_ax_def upair_def by auto
qed

```

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**

$Union_name_body(P, leq, \tau, x) \equiv \exists \sigma \in domain(\tau) . \exists q \in P . \exists r \in P .$
 $\langle \sigma, q \rangle \in \tau \wedge \langle fst(x), r \rangle \in \sigma \wedge \langle snd(x), r \rangle \in leq \wedge \langle snd(x), q \rangle \in leq$

definition *Union_name* :: $[i, i, i] \Rightarrow i$ **where**

$Union_name(P, leq, \tau) \equiv \{u \in domain(\bigcup (domain(\tau))) \times P . Union_name_body(P, leq, \tau, u)\}$

context *forcing_data1*

begin

lemma *Union_name_closed* :

assumes $\tau \in M$

shows $Union_name(\mathbb{P}, leq, \tau) \in M$

proof -

let $?Q = Union_name_body(\mathbb{P}, leq, \tau)$

note $lr_fst2 = lam_replacement_hcomp[OF lam_replacement_fst lam_replacement_fst]$
and $lr_fst3 = lam_replacement_hcomp[OF lr_fst2] lam_replacement_hcomp[OF$
 $lr_fst2 lr_fst2]$

note $\langle \tau \in M \rangle$

moreover from *this*

have $domain(\bigcup (domain(\tau))) \in M$ (**is** $?d \in _$)

using *domain_closed Union_closed* **by** *simp*

moreover from *this*

have $?d \times \mathbb{P} \in M$

using *cartprod_closed* **by** *simp*

note *types = assms* $\langle ?d \times \mathbb{P} \in M \rangle \langle ?d \in M \rangle$

ultimately

show *?thesis*

using *domain_closed pair_in_M_iff fst_closed snd_closed separation_closed*

lam_replacement_constant lam_replacement_hcomp

lam_replacement_fst lam_replacement_snd lam_replacement_product

separation_bex separation_conj separation_in lr_fst2 lr_fst3

lam_replacement_hcomp[OF lr_fst3(1) lam_replacement_snd]

unfolding *Union_name_body_def Union_name_def*

by *simp*

qed

lemma *Union_MG_Eq* :

assumes $a \in M[G]$ **and** $a = val(G, \tau)$ **and** *filter(G)* **and** $\tau \in M$

```

shows  $\bigcup a = \text{val}(G, \text{Union\_name}(\mathbb{P}, \text{leq}, \tau))$ 
proof (intro equalityI subsetI)
  fix x
  assume  $x \in \bigcup a$ 
  with  $\langle a = \_ \rangle$ 
  have  $x \in \bigcup (\text{val}(G, \tau))$ 
    by simp
  then
  obtain i where  $i \in \text{val}(G, \tau)$   $x \in i$ 
    by blast
  with  $\langle \tau \in M \rangle$ 
  obtain  $\sigma$  q where  $q \in G$   $\langle \sigma, q \rangle \in \tau$   $\text{val}(G, \sigma) = i$   $\sigma \in M$ 
    using elem_of_val_pair domain_trans[OF trans_M] by blast
  moreover from this  $\langle x \in i \rangle$ 
  obtain  $\vartheta$  r where  $r \in G$   $\langle \vartheta, r \rangle \in \sigma$   $\text{val}(G, \vartheta) = x$   $\vartheta \in M$ 
    using elem_of_val_pair domain_trans[OF trans_M] by blast
  moreover from calculation
  have  $\vartheta \in \text{domain}(\bigcup (\text{domain}(\tau)))$ 
    by auto
  moreover from calculation  $\langle \text{filter}(G) \rangle$ 
  obtain p where  $p \in G$   $\langle p, r \rangle \in \text{leq}$   $\langle p, q \rangle \in \text{leq}$   $p \in \mathbb{P}$   $r \in \mathbb{P}$   $q \in \mathbb{P}$ 
    using low_bound_filter filterD by blast
  moreover from this
  have  $p \in M$   $q \in M$   $r \in M$ 
    by (auto dest:transitivity)
  moreover from calculation
  have  $\langle \vartheta, p \rangle \in \text{Union\_name}(\mathbb{P}, \text{leq}, \tau)$ 
    unfolding Union_name_def Union_name_body_def
    by auto
  moreover from this  $\langle p \in \mathbb{P} \rangle$   $\langle p \in G \rangle$ 
  have  $\text{val}(G, \vartheta) \in \text{val}(G, \text{Union\_name}(\mathbb{P}, \text{leq}, \tau))$ 
    using val_of_elem by simp
  ultimately
  show  $x \in \text{val}(G, \text{Union\_name}(\mathbb{P}, \text{leq}, \tau))$ 
    by simp
next
fix x
assume  $x \in (\text{val}(G, \text{Union\_name}(\mathbb{P}, \text{leq}, \tau)))$ 
moreover
note  $\langle \text{filter}(G) \rangle$   $\langle a = \text{val}(G, \tau) \rangle$ 
moreover from calculation
obtain  $\vartheta$  p where  $p \in G$   $\langle \vartheta, p \rangle \in \text{Union\_name}(\mathbb{P}, \text{leq}, \tau)$   $\text{val}(G, \vartheta) = x$ 
  using elem_of_val_pair by blast
moreover from calculation
have  $p \in \mathbb{P}$ 
  using filterD by simp
moreover from calculation
obtain  $\sigma$  q r where  $\langle \sigma, q \rangle \in \tau$   $\langle \vartheta, r \rangle \in \sigma$   $\langle p, r \rangle \in \text{leq}$   $\langle p, q \rangle \in \text{leq}$   $r \in \mathbb{P}$   $q \in \mathbb{P}$ 
  unfolding Union_name_def Union_name_body_def

```

```

    by auto
  moreover from calculation
  have  $r \in G \ q \in G$ 
    using filter_leqD by auto
  moreover from this  $\langle \vartheta, r \rangle \in \sigma \ \langle \sigma, q \rangle \in \tau \ \langle q \in \mathbb{P} \rangle \ \langle r \in \mathbb{P} \rangle$ 
  have  $val(G, \sigma) \in val(G, \tau) \ val(G, \vartheta) \in val(G, \sigma)$ 
    using val_of_elem by simp+
  ultimately
  show  $x \in \bigcup a$ 
    by blast
qed

```

```

lemma union_in_MG :
  assumes filter(G)
  shows Union_ax(##M[G])
  unfolding Union_ax_def
proof(clarsimp)
  fix a
  assume  $a \in M[G]$ 
  moreover
  note  $\langle filter(G) \rangle$ 
  moreover from calculation
  interpret mgtrans : M_trans ##M[G]
    using transitivity_MG by (unfold_locales; auto)
  from calculation
  obtain  $\tau$  where  $\tau \in M \ a = val(G, \tau)$ 
    using GenExtD by blast
  moreover from this
  have  $val(G, Union\_name(\mathbb{P}, leq, \tau)) \in M[G]$ 
    using GenExtI Union_name_closed by simp
  ultimately
  show  $\exists z \in M[G] . big\_union(##M[G], a, z)$ 
    using Union_MG_Eq by auto
qed

```

```

theorem Union_MG : M_generic(G)  $\implies$  Union_ax(##M[G])
  by (auto simp: union_in_MG)

```

```

end — forcing_data1

```

```

end

```

19 The Powerset Axiom in $M[G]$

```

theory Powerset_Axiom
  imports
    Separation_Axiom Pairing_Axiom Union_Axiom
begin

```

```

simple_rename perm_pow src [ss,p,l,o,fs,χ] tgt [fs,ss,sp,p,l,o,χ]

context G_generic1
begin

lemma satsfst_snd_in_M:
  assumes
    A∈M B∈M φ ∈ formula p∈M l∈M o∈M χ∈M arity(φ) ≤ 6
  shows {⟨s,q⟩∈A×B . M, [q,p,l,o,s,χ] ⊨ φ} ∈ M (is ?∅ ∈ M)
proof -
  let ?φ' = ren(φ) '6'7'perm_pow_fn
  from ⟨A∈M⟩ ⟨B∈M⟩
  have A×B ∈ M
    using cartprod_closed by simp
  from ⟨arity(φ) ≤ 6⟩ ⟨φ ∈ formula⟩
  have ?φ' ∈ formula arity(?φ') ≤ 7
    unfolding perm_pow_fn_def
    using perm_pow_thm arity_ren ren_tc Nil_type
    by auto
  with ⟨?φ' ∈ formula⟩
  have arty: arity(Exists(Exists(And(pair_fm(0,1,2),?φ')))) ≤ 5 (is arity(?ψ) ≤ 5)
    using ord_simp_union pred_le
    by (auto simp:arity)
  {
    fix sp
    note ⟨A×B ∈ M⟩ ⟨A∈M⟩ ⟨B∈M⟩
    moreover
    assume sp ∈ A×B
    moreover from calculation
    have fst(sp) ∈ A snd(sp) ∈ B
      using fst_type snd_type by simp_all
    ultimately
    have sp ∈ M fst(sp) ∈ M snd(sp) ∈ M
      using transitivity
      by simp_all
    note inM = ⟨A∈M⟩ ⟨B∈M⟩ ⟨p∈M⟩ ⟨l∈M⟩ ⟨o∈M⟩ ⟨χ∈M⟩
      ⟨sp∈M⟩ ⟨fst(sp)∈M⟩ ⟨snd(sp)∈M⟩
    with arty ⟨sp ∈ M⟩ ⟨?φ' ∈ formula⟩
    have (M, [sp,p,l,o,χ]@[p] ⊨ ?ψ) ↔ M, [sp,p,l,o,χ] ⊨ ?ψ (is (M, ?env0@_ ⊨ _)
    ↔ _)
      using arity_sats_iff[of ?ψ [p] M ?env0] by auto
    also from inM ⟨sp ∈ A×B⟩
    have ... ↔ sats(M, ?φ', [fst(sp), snd(sp), sp, p, l, o, χ])
      by auto
    also from inM ⟨φ ∈ formula⟩ ⟨arity(φ) ≤ 6⟩
    have ... ↔ M, [snd(sp), p, l, o, fst(sp), χ] ⊨ φ
      (is sats(____, ?env1) ↔ sats(____, ?env2))
    using sats_iff_sats_ren[of φ 6 7 ?env2 M ?env1 perm_pow_fn] perm_pow_thm
    unfolding perm_pow_fn_def by simp
  }

```

```

finally
  have  $(M, [sp, p, l, o, \chi, p] \models ?\psi) \longleftrightarrow M, [snd(sp), p, l, o, fst(sp), \chi] \models \varphi$ 
    by simp
}
then
have  $?v = \{sp \in A \times B . sats(M, ?\psi, [sp, p, l, o, \chi, p])\}$ 
  by auto
with assms  $\langle A \times B \in M \rangle$ 
show ?thesis
  using separation_ax separation_iff arty leI  $\langle ?\varphi' \in formula \rangle$ 
  by simp
qed

```

```

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 -

```

from assms
obtain  $\tau$  where  $\tau \in M$   $val(G, \tau) = a$ 
  using GenExtD by auto
let  $?Q = Pow^M(domain(\tau) \times \mathbb{P})$ 
let  $?\pi = ?Q \times \{\mathbf{1}\}$ 
let  $?b = val(G, ?\pi)$ 
from  $\langle \tau \in M \rangle$ 
have  $domain(\tau) \times \mathbb{P} \in M$   $domain(\tau) \in M$ 
  by simp_all

```

```

then
have  $?b \in M[G]$ 
  by (auto intro!: GenExtI)
have  $Pow(a) \cap M[G] \subseteq ?b$ 

```

proof

```

fix  $c$ 
assume  $c \in Pow(a) \cap M[G]$ 
then
obtain  $\chi$  where  $c \in M[G]$   $\chi \in M$   $val(G, \chi) = c$ 
  using GenExt_iff by auto
let  $?v = \{ \langle \sigma, p \rangle \in domain(\tau) \times \mathbb{P} . p \Vdash \cdot 0 \in 1 \cdot [\sigma, \chi] \}$ 
have  $arity(forces(\cdot 0 \in 1 \cdot)) = 6$ 
  using arity_forces_at by auto
with  $\langle domain(\tau) \in M \rangle$   $\langle \chi \in M \rangle$ 
have  $?v \in M$ 
  using sats_fst_snd_in_M
  by simp
with  $\langle domain(\tau) \times \mathbb{P} \in M \rangle$ 

```

```

have ? $\vartheta \in ?Q$ 
  using Pow_rel_char by auto
have val(G, ? $\vartheta$ ) = c
proof(intro equalityI subsetI)
  fix x
  assume x  $\in$  val(G, ? $\vartheta$ )
  then
  obtain  $\sigma$  p where 1:  $\langle \sigma, p \rangle \in ?\vartheta$  p  $\in$  G val(G,  $\sigma$ ) = x
    using elem_of_val_pair
    by blast
  moreover from  $\langle \langle \sigma, p \rangle \in ?\vartheta \rangle \langle ?\vartheta \in M \rangle$ 
  have  $\sigma \in M$ 
    using name_components_in_M[of _ _ ? $\vartheta$ ] by auto
  moreover from 1
  have p  $\Vdash$   $\cdot 0 \in 1$ . [ $\sigma, \chi$ ] p  $\in \mathbb{P}$ 
    by simp_all
  moreover
  note  $\langle \text{val}(G, \chi) = c \rangle \langle \chi \in M \rangle$ 
  ultimately
  have M[G], [x, c]  $\models$   $\cdot 0 \in 1$ .
    using generic_definition_of_forcing[where  $\varphi = \cdot 0 \in 1$ ] ord_simp_union
    by auto
  moreover from  $\langle \sigma \in M \rangle \langle \chi \in M \rangle$ 
  have x  $\in$  M[G]
    using  $\langle \text{val}(G, \sigma) = x \rangle$  GenExtI by blast
  ultimately
  show x  $\in$  c
    using  $\langle c \in M[G] \rangle$  by simp
next
fix x
assume x  $\in$  c
with  $\langle c \in \text{Pow}(a) \cap M[G] \rangle$ 
have x  $\in$  a c  $\in$  M[G] x  $\in$  M[G]
  using transitivity_MG by auto
with  $\langle \text{val}(G, \tau) = a \rangle$ 
obtain  $\sigma$  where  $\sigma \in \text{domain}(\tau)$  val(G,  $\sigma$ ) = x
  using elem_of_val by blast
moreover
note  $\langle x \in c \rangle \langle \text{val}(G, \chi) = c \rangle \langle c \in M[G] \rangle \langle x \in M[G] \rangle$ 
moreover from calculation
have val(G,  $\sigma$ )  $\in$  val(G,  $\chi$ )
  by simp
moreover from calculation
have M[G], [x, c]  $\models$   $\cdot 0 \in 1$ .
  by simp
moreover
have  $\sigma \in M$ 
proof -
  from  $\langle \sigma \in \text{domain}(\tau) \rangle$ 

```

```

obtain  $p$  where  $\langle \sigma, p \rangle \in \tau$ 
  by auto
with  $\langle \tau \in M \rangle$ 
show ?thesis
  using name_components_in_M by blast
qed
moreover
note  $\langle \chi \in M \rangle$ 
ultimately
obtain  $p$  where  $p \in G \ p \Vdash \cdot 0 \in 1 \cdot [\sigma, \chi]$ 
  using generic_truth_lemma [of  $\cdot 0 \in 1 \cdot [\sigma, \chi]$  ] ord_simp_union
  by auto
moreover from  $\langle p \in G \rangle$ 
have  $p \in \mathbb{P}$ 
  using generic by blast
ultimately
have  $\langle \sigma, p \rangle \in ?\vartheta$ 
  using  $\langle \sigma \in \text{domain}(\tau) \rangle$  by simp
with  $\langle \text{val}(G, \sigma) = x \rangle \langle p \in G \rangle$ 
show  $x \in \text{val}(G, ?\vartheta)$ 
  using val_of_elem [of  $\_ \_ ?\vartheta$   $G$ ] by auto
qed
with  $\langle ?\vartheta \in ?Q \rangle$ 
show  $c \in ?b$ 
  using one_in_G_generic_val_of_elem [of  $?\vartheta$   $\mathbf{1}$   $?\pi$   $G$ ]
  by auto
qed
then
have  $\text{Pow}(a) \cap M[G] = \{x \in ?b \ . \ x \subseteq a \wedge x \in M[G]\}$ 
  by auto
also from  $\langle a \in M[G] \rangle$ 
have  $\dots = \{x \in ?b \ . \ (M[G], [x, a] \models \cdot 0 \subseteq 1 \cdot )\} \cap M[G]$ 
  using Transset_MG by force
also from  $\langle ?b \in M[G] \rangle$ 
have  $\dots = \{x \in ?b \ . \ (M[G], [x, a] \models \cdot 0 \subseteq 1 \cdot )\}$ 
  by (intro equalityI) (auto dest:ext.transM)
also from  $\langle ?b \in M[G] \rangle \langle a \in M[G] \rangle$ 
have  $\dots \in M[G]$ 
  using Collect_sats_in_MG GenExtI ord_simp_union by (simp add:arity)
finally
show ?thesis .
qed

end — G_generic1

sublocale  $G\_generic1 \subseteq \text{ext: } M\_trivial \ \#\# \ M[G]$ 
  using generic Union_MG_pairing_in_MG
  by unfold_locales (simp; blast)

```

context *G_generic1* **begin**

theorem *power_in_MG* : *power_ax*($\#\#(M[G])$)
unfolding *power_ax_def*
proof (*intro* *rallI*, *simp* *only:setclass_iff rex_setclass_is_bex*)
fix *a*

After simplification, we have to show that for every $a \in M[G]$ there exists some $x \in M[G]$ satisfying *powerset*($\#\#M[G]$, *a*, *x*)

assume $a \in M[G]$
have $\{x \in Pow(a) . x \in M[G]\} = Pow(a) \cap M[G]$
by *auto*
also from $\langle a \in M[G] \rangle$
have $\dots \in M[G]$
using *Pow_inter_MG* **by** *simp*
finally
have $\{x \in Pow(a) . x \in M[G]\} \in M[G]$.
moreover from $\langle a \in M[G] \rangle$ *this*
have *powerset*($\#\#M[G]$, *a*, $\{x \in Pow(a) . x \in M[G]\}$)
using *ext.powerset_abs*
by *simp*
ultimately
show $\exists x \in M[G] . powerset(\#\#M[G], a, x)$
by *auto*
qed

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*($\#\#(M[G])$)
unfolding *extensionality_def*
proof(*clarsimp*)
fix *x y*
assume $x \in M[G]$ $y \in M[G]$ $(\forall w \in M[G] . w \in x \longleftrightarrow w \in y)$
moreover from *this*
have $z \in x \longleftrightarrow z \in M[G] \wedge z \in y$ **for** *z*
using *transitivity_MG* **by** *auto*
moreover from *calculation*

```

    have  $z \in M[G] \wedge z \in x \longleftrightarrow z \in y$  for  $z$ 
      using transitivity_MG by auto
    ultimately
    show  $x=y$ 
      by auto
  qed

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( $\#\#(M[G])$ )
  unfolding foundation_ax_def
  by (rule rallI, cut_tac  $A=x$  in foundation, auto intro: transitivity_MG)

```

```

lemma foundation_ax( $\#\#(M[G])$ )
proof -
  {
    fix  $x$ 
    assume  $x \in M[G] \exists y \in M[G] . y \in x$ 
    then
    have  $\exists y \in M[G] . y \in x \cap M[G]$ 
      by simp
    then
    obtain  $y$  where  $y \in x \cap M[G] \forall z \in y. z \notin x \cap M[G]$ 
      using foundation[of  $x \cap M[G]$ ] by blast
    then
    have  $\exists y \in M[G] . y \in x \wedge (\forall z \in M[G] . z \notin x \vee z \notin y)$ 
      by auto
  }
  then
  show ?thesis
    unfolding foundation_ax_def by auto
qed

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

$\exists t p. x = \langle t, p \rangle \ [x, \alpha, m, \mathbb{P}, leq, \mathbf{1}] \ @ \ nenv \in list(M)$

$\varphi \in formula$

shows

$(\exists \tau \in M. \exists V \in M. is_Vset(\lambda a. (\#\#M)(a), \alpha, V) \wedge \tau \in V \wedge (snd(x) \Vdash \varphi$
 $([fst(x), \tau] @ nenv)))$

$\longleftrightarrow M, [\alpha, x, m, \mathbb{P}, leq, \mathbf{1}] \ @ \ nenv \models body_ground_repl_fm(\varphi)$

unfolding *body_ground_repl_fm_def rename_split_fm_def*

by $((insert_assms, rule_iff_sats \ | \ simp \ add:nonempty[simplified])+,$

insert_sats_incr_bv_iff[where bus=[_,_,_,_,_], simplified], auto del: iffI)

end — *forcing_data1*

context *G_generic1*

begin

lemma *Replace_sats_in_MG*:

assumes

$c \in M[G] \ env \in list(M[G])$

$\varphi \in formula \ arity(\varphi) \leq 2 +_{\omega} length(env)$

$univalent(\#\#M[G], c, \lambda x v. (M[G], [x, v] @ env \models \varphi))$

and

ground_replacement:

$\bigwedge nenv. ground_replacement_assm(M, [\mathbb{P}, leq, \mathbf{1}] \ @ \ nenv, \varphi)$

shows

$\{v. x \in c, v \in M[G] \wedge (M[G], [x, v] @ env \models \varphi)\} \in M[G]$

proof -

let $?R = \lambda x v. v \in M[G] \wedge (M[G], [x, v] @ env \models \varphi)$

from $\langle c \in M[G] \rangle$

obtain π' **where** $val(G, \pi') = c \ \pi' \in M$

```

    using GenExt_def by auto
  then
  have domain( $\pi'$ ) $\times\mathbb{P}\in M$  (is  $? \pi \in M$ )
    using cartprod_closed domain_closed by simp
  from  $\langle \text{val}(G, \pi') = c \rangle$ 
  have  $c \subseteq \text{val}(G, ?\pi)$ 
    using def_val[of  $G$   $? \pi$ ] elem_of_val[of  $G$   $\pi'$ ] one_in_G
      domain_of_prod[OF one_in_P, of domain( $\pi'$ )]
    by (force del:M_genericD)
  from  $\langle \text{env} \in \_ \rangle$ 
  obtain nenv where nenv $\in \text{list}(M)$  env = map(val(G),nenv)
    using map_val by auto
  then
  have length(nenv) = length(env) by simp
  with  $\langle \text{arity}(\varphi) \leq \_ \rangle$ 
  have arity( $\varphi$ )  $\leq 2 + \omega$  length(nenv) by simp
  define f where f( $\varrho p$ )  $\equiv \mu \alpha. \alpha \in M \wedge (\exists \tau \in M. \tau \in \text{Vset}(\alpha) \wedge$ 
    (snd( $\varrho p$ )  $\Vdash \varphi$  ([fst( $\varrho p$ ), $\tau$ ] @ nenv))) (is  $\_ \equiv \mu \alpha. ?P(\varrho p, \alpha)$ ) for  $\varrho p$ 
  have f( $\varrho p$ ) = ( $\mu \alpha. \alpha \in M \wedge (\exists \tau \in M. \exists V \in M. \text{is\_Vset}(\#\#M, \alpha, V) \wedge \tau \in V \wedge$ 
    (snd( $\varrho p$ )  $\Vdash \varphi$  ([fst( $\varrho p$ ), $\tau$ ] @ nenv)))) (is  $\_ = (\mu \alpha. \alpha \in M \wedge ?Q(\varrho p, \alpha))$ ) for
 $\varrho p$ 
  unfolding f_def using Vset_abs Vset_closed Ord_Least_cong[of  $?P(\varrho p) \lambda \alpha.$ 
 $\alpha \in M \wedge ?Q(\varrho p, \alpha)$ ]
    by (simp, simp del:setclass_iff)
  moreover
  note inM =  $\langle \text{nenv} \in \text{list}(M) \rangle \langle ? \pi \in M \rangle$ 
  moreover
  have f( $\varrho p$ )  $\in M$  Ord(f( $\varrho p$ )) for  $\varrho p$ 
    unfolding f_def using Least_closed'[of  $?P(\varrho p)$ ] by simp_all
  ultimately
  have 1:least( $\#\#M, \lambda \alpha. ?Q(\varrho p, \alpha), f(\varrho p)$ ) for  $\varrho p$ 
    using least_abs'[of  $\lambda \alpha. \alpha \in M \wedge ?Q(\varrho p, \alpha) f(\varrho p)$ ] least_conj
    by (simp flip:setclass_iff)
  define QQ where QQ $\equiv ?Q$ 
  from 1
  have least( $\#\#M, \lambda \alpha. QQ(\varrho p, \alpha), f(\varrho p)$ ) for  $\varrho p$ 
    unfolding QQ_def .
  have body:( $M, [\varrho p, m, \mathbb{P}, \text{leq}, \mathbf{1}]$  @ nenv  $\models \text{ground\_repl\_fm}(\varphi) \longleftrightarrow \text{least}(\#\#M,$ 
  QQ( $\varrho p$ ), m)
    if  $\varrho p \in M$   $\varrho p \in ? \pi$   $m \in M$  for  $\varrho p$   $m$ 
  proof -
    note inM that
    moreover from this assms 1
    have ( $M, [\alpha, \varrho p, m, \mathbb{P}, \text{leq}, \mathbf{1}]$  @ nenv  $\models \text{body\_ground\_repl\_fm}(\varphi) \longleftrightarrow ?Q(\varrho p, \alpha)$ )
  if  $\alpha \in M$  for  $\alpha$ 
    using that sats_body_ground_repl_fm[of  $\varrho p$   $\alpha$   $m$  nenv  $\varphi$ ]
    by auto
  moreover from calculation
  have body: $\bigwedge \alpha. \alpha \in M \implies (\exists \tau \in M. \exists V \in M. \text{is\_Vset}(\lambda a. a \in M, \alpha, V) \wedge \tau \in$ 

```

```

V ∧
  (snd(ϱp) ⊨ φ ([fst(ϱp),τ] @ nenv)) ↔
  M, Cons(α, [ϱp, m, ℙ, leq, 1] @ nenv) ⊨ body_ground_repl_fm(φ)
  by simp
  ultimately
  show (M , [ϱp,m,ℙ,leq,1] @ nenv ⊨ ground_repl_fm(φ)) ↔ least(##M,
QQ(ϱp), m)
  using sats_least_fm[OF body,of 1] unfolding QQ_def ground_repl_fm_def
  by (simp, simp flip: setclass_iff)
  qed
  then
  have univalent(##M, ?π, λϱp m. M , [ϱp,m] @ ([ℙ,leq,1] @ nenv) ⊨ ground_repl_fm(φ))
    unfolding univalent_def by (auto intro:unique_least)
  moreover from ⟨length(⋅) = ⋅⟩ ⟨env ∈ ⋅⟩
  have length([ℙ,leq,1] @ nenv) = 3 +ω length(env) by simp
  moreover from ⟨arity(φ) ≤ 2 +ω length(nenv)⟩
    ⟨length(⋅) = length(⋅)⟩[symmetric] ⟨nenv∈⋅⟩ ⟨φ∈⋅⟩
  have arity(ground_repl_fm(φ)) ≤ 5 +ω length(env)
    using arity_ground_repl_fm[of φ] le_trans Un_le by auto
  moreover from ⟨φ∈formula⟩
  have ground_repl_fm(φ)∈formula by simp
  moreover
  note ⟨length(nenv) = length(env)⟩ inM
  ultimately
  obtain Y where Y∈M
    ∀ m∈M. m ∈ Y ↔ (∃ ϱp∈M. ϱp ∈ ?π ∧ (M, [ϱp,m] @ ([ℙ,leq,1] @ nenv) ⊨
ground_repl_fm(φ)))
    using ground_replacement[of nenv]
    unfolding strong_replacement_def ground_replacement_assm_def replace-
ment_assm_def by auto
  with ⟨least(⋅, QQ(⋅), f(⋅))⟩ ⟨f(⋅) ∈ M⟩ ⟨?π∈M⟩ body
  have f(ϱp)∈Y if ϱp∈?π for ϱp
    using that transitivity[OF _ ⟨?π∈M⟩]
    by (clarsimp, rename_tac ϱ p ϱp, rule_tac x=⟨ϱ,p⟩ in bexI, auto)
  from ⟨Y∈M⟩
  have ⋃ {y∈Y. Ord(y)} ∈ M (is ?sup ∈ M)
    using separation_Ord separation_closed Union_closed by simp
  then
  have {x∈Vset(?sup). x ∈ M} × {1} ∈ M (is ?big_name ∈ M)
    using Vset_closed cartprod_closed singleton_closed by simp
  then
  have val(G, ?big_name) ∈ M[G]
    by (blast intro:GenExtI)
  have {v. x∈c, ?R(x,v)} ⊆ val(G, ?big_name) (is ?repl⊆?big)
  proof(intro subsetI)
    fix v
    assume v∈?repl
    moreover from this
    obtain x where x∈c M[G], [x, v] @ env ⊨ φ v∈M[G]

```

by *auto*
moreover
note $\langle \text{val}(G, \pi') = c \rangle \langle \pi' \in M \rangle$
moreover
from *calculation*
obtain ϱp **where** $\langle \varrho, p \rangle \in \pi' \text{ val}(G, \varrho) = x p \in G \varrho \in M$
using *elem_of_val_pair'* **by** *blast*
moreover from *this* $\langle v \in M[G] \rangle$
obtain σ **where** $\text{val}(G, \sigma) = v \sigma \in M$
using *GenExtD* **by** (*force del: M_genericD*)
moreover
note $\langle \varphi \in _ \rangle \langle \text{nenv} \in _ \rangle \langle \text{env} = _ \rangle \langle \text{arity}(\varphi) \leq 2 + \omega \text{ length}(\text{env}) \rangle$
ultimately
obtain q **where** $q \in G q \Vdash \varphi ([\varrho, \sigma] @ \text{nenv}) q \in \mathbb{P}$
using *truth_lemma[OF* $\langle \varphi \in _ \rangle$, *of* $[\varrho, \sigma] @ \text{nenv}$
by *auto*
with $\langle \langle \varrho, p \rangle \in \pi' \rangle \langle \langle \varrho, q \rangle \in ?\pi \implies f(\langle \varrho, q \rangle) \in Y \rangle$
have $f(\langle \varrho, q \rangle) \in Y$
using *generic* **by** *blast*
let $? \alpha = \text{succ}(\text{rank}(\sigma))$
note $\langle \sigma \in M \rangle$
moreover from *this*
have $? \alpha \in M \sigma \in \text{Vset}(? \alpha)$
using *rank_closed cons_closed Vset_Ord_rank_iff*
by (*simp_all flip: setclass_iff*)
moreover
note $\langle q \Vdash \varphi ([\varrho, \sigma] @ \text{nenv}) \rangle$
ultimately
have $?P(\langle \varrho, q \rangle, ? \alpha)$ **by** (*auto simp del: Vset_rank_iff*)
moreover
have $(\mu \alpha. ?P(\langle \varrho, q \rangle, \alpha)) = f(\langle \varrho, q \rangle)$
unfolding *f_def* **by** *simp*
ultimately
obtain τ **where** $\tau \in M \tau \in \text{Vset}(f(\langle \varrho, q \rangle)) q \Vdash \varphi ([\varrho, \tau] @ \text{nenv})$
using *LeastI[of* $\lambda \alpha. ?P(\langle \varrho, q \rangle, \alpha) ? \alpha]$ **by** *auto*
with $\langle q \in G \rangle \langle \varrho \in M \rangle \langle \text{nenv} \in _ \rangle \langle \text{arity}(\varphi) \leq 2 + \omega \text{ length}(\text{nenv}) \rangle$
have $M[G], \text{map}(\text{val}(G), [\varrho, \tau] @ \text{nenv}) \models \varphi$
using *truth_lemma[OF* $\langle \varphi \in _ \rangle$, *of* $[\varrho, \tau] @ \text{nenv}$
by *auto*
moreover from $\langle x \in c \rangle \langle c \in M[G] \rangle$
have $x \in M[G]$ **using** *transitivity_MG* **by** *simp*
moreover
note $\langle M[G], [x, v] @ \text{env} \models \varphi \rangle \langle \text{env} = \text{map}(\text{val}(G), \text{nenv}) \rangle \langle \tau \in M \rangle \langle \text{val}(G, \varrho) = x \rangle$
 $\langle \text{univalent}(\#\#M[G], _, _) \rangle \langle x \in c \rangle \langle v \in M[G] \rangle$
ultimately
have $v = \text{val}(G, \tau)$
using *GenExtI*[*of* $\tau G]$ **unfolding** *univalent_def* **by** (*auto*)
from $\langle \tau \in \text{Vset}(f(\langle \varrho, q \rangle)) \rangle \langle \text{Ord}(f(_)) \rangle \langle f(\langle \varrho, q \rangle) \in Y \rangle$
have $\tau \in \text{Vset}(? \text{sup})$
using *Vset_Ord_rank_iff lt_Union_iff*[*of* $_ \text{rank}(\tau)$] **by** *auto*

```

with  $\langle \tau \in M \rangle$ 
have  $val(G, \tau) \in val(G, ?big\_name)$ 
  using  $domain\_of\_prod[of \mathbf{1} \{ \mathbf{1} \} \{ x \in Vset(?sup). x \in M \} ] def\_val[of G$ 
 $?big\_name]$ 
   $one\_in\_G \ one\_in\_P$  by  $(auto \ simp \ del: Vset\_rank\_iff)$ 
with  $\langle v = val(G, \tau) \rangle$ 
show  $v \in val(G, ?big\_name)$ 
  by  $simp$ 
qed
from  $\langle ?big\_name \in M \rangle$ 
have  $?repl = \{ v \in ?big. \exists x \in c. M[G], [x, v] @ env \models \varphi \}$  (is  $\_ = ?rhs)$ 
proof( $intro \ equalityI \ subsetI$ )
  fix  $v$ 
  assume  $v \in ?repl$ 
  with  $\langle ?repl \subseteq ?big \rangle$ 
  obtain  $x$  where  $x \in c \ M[G], [x, v] @ env \models \varphi \ v \in ?big$ 
  using  $subsetD$  by  $auto$ 
  with  $\langle univalent(\#\#M[G], \_, \_) \rangle \langle c \in M[G] \rangle$ 
  show  $v \in ?rhs$ 
  unfolding  $univalent\_def$ 
  using  $transitivity\_MG \ ReplaceI[of \ \lambda \ x \ v. \ \exists \ x \in \ c. \ M[G], [x, v] @ env \models \varphi]$  by
 $blast$ 
next
  fix  $v$ 
  assume  $v \in ?rhs$ 
  then
  obtain  $x$  where
     $v \in val(G, ?big\_name) \ M[G], [x, v] @ env \models \varphi \ x \in c$ 
    by  $blast$ 
  moreover from this  $\langle c \in M[G] \rangle$ 
  have  $v \in M[G] \ x \in M[G]$ 
  using  $transitivity\_MG \ GenExtI[OF \ \langle ?big\_name \in \_ \rangle, of \ G]$  by  $auto$ 
  moreover from  $calculation \ \langle univalent(\#\#M[G], \_, \_) \rangle$ 
  have  $?R(x, y) \implies y = v$  for  $y$ 
  unfolding  $univalent\_def$  by  $auto$ 
  ultimately
  show  $v \in ?repl$ 
  using  $ReplaceI[of \ ?R \ x \ v \ c]$ 
  by  $blast$ 
qed
moreover
let  $?\psi = (\cdot \exists \cdot \cdot 0 \in 2 +_\omega length(env) \cdot \wedge \varphi \cdot \cdot)$ 
from  $\langle \varphi \in \_ \rangle$ 
have  $? \psi \in formula \ arity(? \psi) \leq 2 +_\omega length(env)$ 
  using  $pred\_mono[OF \ \_ \ \langle arity(\varphi) \leq 2 +_\omega length(env) \rangle] \ lt\_trans[OF \ \_ \ le\_refl]$ 
  by  $(auto \ simp \ add: ord\_simp\_union \ arity)$ 
moreover
from  $\langle \varphi \in \_ \rangle \ \langle arity(\varphi) \leq 2 +_\omega length(env) \rangle \ \langle c \in M[G] \rangle \ \langle env \in \_ \rangle$ 
have  $(\exists x \in c. M[G], [x, v] @ env \models \varphi) \longleftrightarrow M[G], [v] @ env @ [c] \models ?\psi$  if  $v \in M[G]$ 

```

```

for  $v$ 
  using that  $nth\_concat\ transitivity\_MG[OF\ \_ \langle c \in M[G] \rangle] arity\_sats\_iff$   $[of\ \varphi$ 
 $[c] \_ \_ [_, v] @ env]$ 
  by auto
  moreover from this
  have  $\{v \in ?big. \exists x \in c. M[G], [x, v] @ env \models \varphi\} = \{v \in ?big. M[G], [v] @ env @ [c]$ 
 $\models ?\psi\}$ 
  using  $transitivity\_MG[OF\ \_ GenExtI, OF\ \_ \langle ?big\_name \in M \rangle]$ 
  by simp
  moreover from calculation and  $\langle env \in \_ \rangle \langle c \in \_ \rangle \langle ?big \in M[G] \rangle$ 
  have  $\{v \in ?big. M[G], [v] @ env @ [c] \models ?\psi\} \in M[G]$ 
  using  $Collect\_sats\_in\_MG$  by auto
  ultimately
  show  $?thesis$  by simp
qed

```

theorem *strong_replacement_in_MG*:

```

assumes
   $\varphi \in formula$  and  $arity(\varphi) \leq 2 + \omega$   $length(env) \in list(M[G])$ 
and
  ground_replacement:
   $\bigwedge nenv. ground\_replacement\_assm(M, [\mathbb{P}, leq, 1] @ nenv, \varphi)$ 
shows
   $strong\_replacement(\#\#M[G], \lambda x v. M[G], [x, v] @ env \models \varphi)$ 
proof -
  let  $?R = \lambda x y. M[G], [x, y] @ env \models \varphi$ 
  {
    fix  $A$ 
    let  $?Y = \{v. x \in A, v \in M[G] \wedge ?R(x, v)\}$ 
    assume  $1: (\#\#M[G])(A) univalent(\#\#M[G], A, ?R)$ 
    with assms
    have  $(\#\#M[G])(?Y)$ 
      using  $Replace\_sats\_in\_MG\ ground\_replacement\ 1$ 
      unfolding  $ground\_replacement\_assm\_def$  by auto
    have  $b \in ?Y \iff (\exists x[\#\#M[G]]. x \in A \wedge ?R(x, b))$  if  $(\#\#M[G])(b)$  for  $b$ 
    proof(rule)
      from  $\langle (\#\#M[G])(A) \rangle$ 
      show  $\exists x[\#\#M[G]]. x \in A \wedge ?R(x, b)$  if  $b \in ?Y$ 
      using that  $transitivity\_MG$  by auto
    next
    show  $b \in ?Y$  if  $\exists x[\#\#M[G]]. x \in A \wedge ?R(x, b)$ 
    proof -
      from  $\langle (\#\#M[G])(b) \rangle$ 
      have  $b \in M[G]$  by simp
      with that
      obtain  $x$  where  $(\#\#M[G])(x) \ x \in A \ b \in M[G] \wedge ?R(x, b)$ 
      by blast
      moreover from this  $1 \ \langle (\#\#M[G])(b) \rangle$ 
      have  $x \in M[G] \ z \in M[G] \wedge ?R(x, z) \implies b = z$  for  $z$ 

```

```

      unfolding univalent_def
      by auto
    ultimately
    show ?thesis
      using ReplaceI[of  $\lambda x y. y \in M[G] \wedge ?R(x,y)$ ] by blast
  qed
  qed
  then
  have  $\forall b[\#\#M[G]]. b \in ?Y \longleftrightarrow (\exists x[\#\#M[G]]. x \in A \wedge ?R(x,b))$ 
    by simp
  with  $\langle (\#\#M[G])(?Y) \rangle$ 
  have  $(\exists Y[\#\#M[G]]. \forall b[\#\#M[G]]. b \in Y \longleftrightarrow (\exists x[\#\#M[G]]. x \in A \wedge ?R(x,b)))$ 
    by auto
  }
  then show ?thesis unfolding strong_replacement_def
    by simp
  qed

```

```

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,  $\varphi$ )
  using assms strong_replacement_in_MG
  unfolding replacement_assm_def by simp

```

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:  $M\_trivial\#\#M[G]$ 
  using transitivity_MG zero_in_MG[of  $G$ ] generic_Union_MG pairing_in_MG
  by unfold_locales auto

```

```

lemma infinity_in_MG : infinity_ax( $\#\#M[G]$ )

```

```

proof -

```

```

  have  $\omega \in M[G]$ 
    using  $M\_subset\_MG$  one_in_G nat_in_M by auto
  moreover from this

```

```

have  $\text{succ}(y) \in \omega \cap M[G]$  if  $y \in \omega$  for  $y$ 
  using that transitivity_MG by blast
ultimately
show ?thesis
  using transitivity_MG[of 0  $\omega$ ]
  unfolding infinity_ax_def
  by auto
qed

```

```

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( $\tau, \rho, on$ )  $\equiv$  upair_name(upair_name( $\tau, \tau, on$ ), upair_name( $\tau, \rho, on$ ), on)

```

```

definition

```

```

  induced_surj ::  $i \Rightarrow i \Rightarrow i \Rightarrow i$  where
  induced_surj( $f, a, e$ )  $\equiv$   $f^{-1}((\text{range}(f) - a) \times \{e\} \cup \text{restrict}(f, f^{-1}a))$ 

```

```

lemma domain_induced_surj:  $\text{domain}(\text{induced\_surj}(f, a, e)) = \text{domain}(f)$ 
  unfolding induced_surj_def using domain_restrict domain_of_prod by auto

```

```

lemma range_restrict_vimage:

```

```

  assumes function( $f$ )
  shows  $\text{range}(\text{restrict}(f, f^{-1}a)) \subseteq a$ 

```

```

proof

```

```

  from assms
  have function( $\text{restrict}(f, f^{-1}a)$ )
    using function_restrictI by simp
  fix  $y$ 
  assume  $y \in \text{range}(\text{restrict}(f, f^{-1}a))$ 
  then

```

```

obtain  $x$  where  $\langle x, y \rangle \in \text{restrict}(f, f^{-1}a)$   $x \in f^{-1}a$   $x \in \text{domain}(f)$ 
  using domain_restrict domainI[of _ _ restrict(f, f^{-1}a)] by auto
moreover
note  $\langle \text{function}(\text{restrict}(f, f^{-1}a)) \rangle$ 
ultimately
have  $y = \text{restrict}(f, f^{-1}a)'x$ 
  using function_apply_equality by blast
also from  $\langle x \in f^{-1}a \rangle$ 
have  $\text{restrict}(f, f^{-1}a)'x = f'x$ 
  by simp
finally
have  $y = f'x$  .
moreover from assms  $\langle x \in \text{domain}(f) \rangle$ 
have  $\langle x, f'x \rangle \in f$ 
  using function_apply_Pair by auto
moreover
note assms  $\langle x \in f^{-1}a \rangle$ 
ultimately
show  $y \in a$ 
  using function_image_vimage[of f a] by auto
qed

lemma induced_surj_type:
  assumes function(f)
  shows
     $\text{induced\_surj}(f, a, e): \text{domain}(f) \rightarrow \{e\} \cup a$ 
    and
     $x \in f^{-1}a \implies \text{induced\_surj}(f, a, e)'x = f'x$ 
proof -
let  $?f1 = f^{-1}(\text{range}(f) - a) \times \{e\}$  and  $?f2 = \text{restrict}(f, f^{-1}a)$ 
have  $\text{domain}(?f2) = \text{domain}(f) \cap f^{-1}a$ 
  using domain_restrict by simp
moreover from assms
have  $\text{domain}(?f1) = f^{-1}(\text{range}(f)) - f^{-1}a$ 
  using domain_of_prod function_vimage_Diff by simp
ultimately
have  $\text{domain}(?f1) \cap \text{domain}(?f2) = 0$ 
  by auto
moreover
have  $\text{function}(?f1)$   $\text{relation}(?f1)$   $\text{range}(?f1) \subseteq \{e\}$ 
  unfolding function_def relation_def range_def by auto
moreover from this and assms
have  $?f1: \text{domain}(?f1) \rightarrow \text{range}(?f1)$ 
  using function_imp_Pi by simp
moreover from assms
have  $?f2: \text{domain}(?f2) \rightarrow \text{range}(?f2)$ 
  using function_imp_Pi[of restrict(f, f^{-1}a)] function_restrictI by simp
moreover from assms
have  $\text{range}(?f2) \subseteq a$ 

```

```

    using range_restrict_vimage by simp
  ultimately
  have induced_surj(f,a,e): domain(?f1) ∪ domain(?f2) → {e} ∪ a
    unfolding induced_surj_def using fun_is_function fun_disjoint_Un fun_weaken_type
  by simp
  moreover
  have domain(?f1) ∪ domain(?f2) = domain(f)
    using domain_restrict domain_of_prod by auto
  ultimately
  show induced_surj(f,a,e): domain(f) → {e} ∪ a
    by simp
  assume x ∈ f-`a
  then
  have ?f2`x = f`x
    using restrict by simp
  moreover from ⟨x ∈ f-`a⟩ ⟨domain(?f1) = _⟩
  have x ∉ domain(?f1)
    by simp
  ultimately
  show induced_surj(f,a,e)`x = f`x
    unfolding induced_surj_def using fun_disjoint_apply2[of x ?f1 ?f2] by simp
qed

```

lemma induced_surj_is_surj :

```

  assumes
    e ∈ a function(f) domain(f) = α ∧ y. y ∈ a ⇒ ∃ x ∈ α. f ` x = y
  shows induced_surj(f,a,e) ∈ surj(α,a)
  unfolding surj_def
proof (intro CollectI ballI)
  from assms
  show induced_surj(f,a,e): α → a
    using induced_surj_type[of f a e] cons_eq cons_absorb by simp
  fix y
  assume y ∈ a
  with assms
  have ∃ x ∈ α. f ` x = y
    by simp
  then
  obtain x where x ∈ α f ` x = y by auto
  with ⟨y ∈ a⟩ assms
  have x ∈ f-`a
    using vimage_iff function_apply_Pair[of f x] by auto
  with ⟨f ` x = y⟩ assms
  have induced_surj(f, a, e) ` x = y
    using induced_surj_type by simp
  with ⟨x ∈ α⟩ show
    ∃ x ∈ α. induced_surj(f, a, e) ` x = y by auto
qed

```

```

lemma (in M_ZF1_trans) upair_name_closed :
   $\llbracket x \in M; y \in M; o \in M \rrbracket \implies \text{upair\_name}(x, y, o) \in M$ 
  unfolding upair_name_def
  using upair_in_M_iff pair_in_M_iff Upair_eq_cons
  by simp

```

```

context G_generic1
begin

```

```

lemma val_upair_name :  $\text{val}(G, \text{upair\_name}(\tau, \varrho, \mathbf{1})) = \{\text{val}(G, \tau), \text{val}(G, \varrho)\}$ 
  unfolding upair_name_def
  using val_Upair Upair_eq_cons generic_one_in_G
  by simp

```

```

lemma val_opair_name :  $\text{val}(G, \text{opair\_name}(\tau, \varrho, \mathbf{1})) = \langle \text{val}(G, \tau), \text{val}(G, \varrho) \rangle$ 
  unfolding opair_name_def Pair_def
  using val_upair_name by simp

```

```

lemma val_RepFun_one :  $\text{val}(G, \{\langle f(x), \mathbf{1} \rangle . x \in a\}) = \{\text{val}(G, f(x)) . x \in a\}$ 
proof -

```

```

  let ?A =  $\{f(x) . x \in a\}$ 
  let ?Q =  $\lambda \langle x, p \rangle . p = \mathbf{1}$ 
  have  $\mathbf{1} \in \mathbb{P} \cap G$  using generic_one_in_G one_in_P by simp
  have  $\{\langle f(x), \mathbf{1} \rangle . x \in a\} = \{t \in ?A \times \mathbb{P} . ?Q(t)\}$ 
    using one_in_P by force
  then
  have  $\text{val}(G, \{\langle f(x), \mathbf{1} \rangle . x \in a\}) = \text{val}(G, \{t \in ?A \times \mathbb{P} . ?Q(t)\})$ 
    by simp
  also
  have  $\dots = \{z . t \in ?A, (\exists p \in \mathbb{P} \cap G . ?Q(\langle t, p \rangle)) \wedge z = \text{val}(G, t)\}$ 
    using val_of_name_alt by simp
  also from  $\langle \mathbf{1} \in \mathbb{P} \cap G \rangle$ 
  have  $\dots = \{\text{val}(G, t) . t \in ?A\}$ 
    by force
  also
  have  $\dots = \{\text{val}(G, f(x)) . x \in a\}$ 
    by auto
  finally
  show ?thesis
    by simp

```

```

qed

```

```

end— G_generic1

```

24.1 $M[G]$ is a transitive model of ZF

```

sublocale G_generic1  $\subseteq \text{ext}: M_Z\_trans M[G]$ 
  using Transset_MG generic_pairing_in_MG Union_MG
  extensionality_in_MG power_in_MG foundation_in_MG

```

```

    replacement_assm_MG separation_in_MG infinity_in_MG
    replacement_ax1
  by unfold_locales

lemma (in M_replacement) upair_name_lam_replacement :
  M(z)  $\implies$  lam_replacement(M,  $\lambda x . \text{upair\_name}(\text{fst}(x), \text{snd}(x), z)$ )
  using lam_replacement_Upair[THEN [5] lam_replacement_hcomp2]
    lam_replacement_product
    lam_replacement_fst lam_replacement_snd lam_replacement_constant
  unfolding upair_name_def
  by simp

lemma (in forcing_data1) repl_opname_check :
  assumes  $A \in M$   $f \in M$ 
  shows  $\{\text{opair\_name}(\text{check}(x), f'x, \mathbf{1}). x \in A\} \in M$ 
  using assms lam_replacement_constant check_lam_replacement lam_replacement_identity
    upair_name_lam_replacement[THEN [5] lam_replacement_hcomp2]
    lam_replacement_apply2[THEN [5] lam_replacement_hcomp2]
    lam_replacement_imp_strong_replacement_aux
    transitivity RepFun_closed upair_name_closed apply_closed
  unfolding opair_name_def
  by simp

theorem (in G_generic1) choice_in_MG:
  assumes choice_ax( $\#\#M$ )
  shows choice_ax( $\#\#M[G]$ )
proof -
  {
    fix a
    assume  $a \in M[G]$ 
    then
    obtain  $\tau$  where  $\tau \in M$   $\text{val}(G, \tau) = a$ 
      using GenExt_def by auto
    with  $\langle \tau \in M \rangle$ 
    have  $\text{domain}(\tau) \in M$ 
      using domain_closed by simp
    then
    obtain  $s \ \alpha$  where  $s \in \text{surj}(\alpha, \text{domain}(\tau))$   $\text{Ord}(\alpha)$   $s \in M$   $\alpha \in M$ 
      using assms choice_ax_abs
      by auto
    then
    have  $\alpha \in M[G]$ 
      using M_subset_MG generic_one_in_G subsetD
      by blast
    let  $?A = \text{domain}(\tau) \times \mathbb{P}$ 
    let  $?g = \{\text{opair\_name}(\text{check}(\beta), s'\beta, \mathbf{1}). \beta \in \alpha\}$ 
    have  $?g \in M$ 
      using  $\langle s \in M \rangle$   $\langle \alpha \in M \rangle$  repl_opname_check
      by simp
  }

```

```

let ?f_dot = {⟨opair_name(check(β), s'β, 1), 1⟩ . β ∈ α}
have ?f_dot = ?g × {1} by blast
define f where
  f ≡ val(G, ?f_dot)
from ⟨?g ∈ M⟩ ⟨?f_dot = ?g × {1}⟩
have ?f_dot ∈ M
  using cartprod_closed singleton_closed
  by simp
then
have f ∈ M[G]
  unfolding f_def
  by (blast intro: GenExtI)
have f = {val(G, opair_name(check(β), s'β, 1)) . β ∈ α}
  unfolding f_def
  using val_RepFun_one
  by simp
also
have ... = {⟨β, val(G, s'β)⟩ . β ∈ α}
  using val_opair_name val_check generic_one_in_G one_in_P
  by simp
finally
have f = {⟨β, val(G, s'β)⟩ . β ∈ α} .
then
have 1: domain(f) = α function(f)
  unfolding function_def by auto
have 2: y ∈ a ⟹ ∃ x ∈ α. f ' x = y for y
proof -
  fix y
  assume
    y ∈ a
  with ⟨val(G, τ) = a⟩
  obtain σ where σ ∈ domain(τ) val(G, σ) = y
    using elem_of_val[of y _ τ]
    by blast
  with ⟨s ∈ surj(α, domain(τ))⟩
  obtain β where β ∈ α s'β = σ
    unfolding surj_def
    by auto
  with ⟨val(G, σ) = y⟩
  have val(G, s'β) = y
    by simp
  with ⟨f = {⟨β, val(G, s'β)⟩ . β ∈ α}⟩ ⟨β ∈ α⟩
  have ⟨β, y⟩ ∈ f
    by auto
  with ⟨function(f)⟩
  have f'β = y
    using function_apply_equality by simp
  with ⟨β ∈ α⟩ show
    ∃ β ∈ α. f ' β = y

```

```

    by auto
  qed
then
have  $\exists \alpha \in (M[G]). \exists f' \in (M[G]). \text{Ord}(\alpha) \wedge f' \in \text{surj}(\alpha, a)$ 
proof (cases a=0)
  case True
  then
  show ?thesis
    unfolding surj_def
    using zero_in_MG
    by auto
next
  case False
  with  $\langle a \in M[G] \rangle$ 
  obtain e where  $e \in a \wedge e \in M[G]$ 
    using transitivity_MG
    by blast
  with 1 and 2
  have  $\text{induced\_surj}(f, a, e) \in \text{surj}(\alpha, a)$ 
    using induced_surj_is_surj by simp
  moreover from  $\langle f \in M[G] \rangle \langle a \in M[G] \rangle \langle e \in M[G] \rangle$ 
  have  $\text{induced\_surj}(f, a, e) \in M[G]$ 
    unfolding induced_surj_def
    by (simp flip: setclass_iff)
  moreover
  note  $\langle \alpha \in M[G] \rangle \langle \text{Ord}(\alpha) \rangle$ 
  ultimately
  show ?thesis
    by auto
  qed
}
then
show ?thesis
  using ext.choice_ax_abs
  by simp
qed

sublocale  $G\_generic1\_AC \subseteq \text{ext}:M\_ZC\_basic M[G]$ 
  using choice_ax choice_in_MG
  by unfold_locales

```

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

```
  fix  $p$ 
```

```
  assume  $p \in \mathbb{P}$ 
```

```
  show  $\exists d \in \mathbb{P} - G. d \preceq p$ 
```

```
  proof (cases  $p \in G$ )
```

```
    case True
```

```
      note  $\langle p \in \mathbb{P} \rangle$  assms
```

```
      moreover
```

```
      obtain  $q r$  where  $q \preceq p \wedge r \preceq p \wedge q \perp r$   $q \in \mathbb{P} \wedge r \in \mathbb{P}$ 
```

```
        using separative[OF  $\langle p \in \mathbb{P} \rangle$ ]
```

```
        by force
```

```
      with  $\langle$ filter( $G$ ) $\rangle$ 
```

```
      obtain  $s$  where  $s \preceq p \wedge s \notin G \wedge s \in \mathbb{P}$ 
```

```
        using filter_imp_compat[of  $G \ q \ r$ ]
```

```
        by auto
```

```
      then
```

```
        show ?thesis
```

```
          by blast
```

```
    next
```

```
      case False
```

```
        with  $\langle p \in \mathbb{P} \rangle$ 
```

```
        show ?thesis
```

```
          using refl_leq unfolding Diff_def by auto
```

```
  qed
```

```
qed
```

```
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 1 M enum G
  by unfold_locales (simp add:generic)

lemma generic_not_in_M:
  shows  $G \notin M$ 
proof
  assume  $G \in M$ 
  then
  have  $\mathbb{P} - G \in M$ 
    using Diff_closed by simp
  moreover
  have  $\neg(\exists q \in G. q \in \mathbb{P} - G) (\mathbb{P} - G) \subseteq \mathbb{P}$ 
    unfolding Diff_def by auto
  moreover
  note generic
  ultimately
  show False
    using filter_complement_dense[of G] M_generic_denseD[of  $\mathbb{P}-G$ ]
    by auto
qed

theorem proper_extension:  $M \neq M[G]$ 
  using generic G_in_Gen_Ext one_in_G generic_not_in_M
  by force
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 $\omega \text{-}|| > 2$ or equivalently as the set of partial functions $F_n(\omega, \omega, 2)$, 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. $F_n\text{lerel}(A^{<\omega})$), so it could be surprising that we have not used $F_n(\omega, \omega, 2)$ for the set. The only reason why we keep this alternative definition is because we can prove $A^{<\omega} \in M$ (and therefore $F_n\text{lerel}(A^{<\omega}) \in M$) 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  $seq\_upd(f,a) \in succ(n) \rightarrow A$ 
proof -
  from assms
  have equ:  $domain(f) = n$ 
    using domain_of_fun by simp
  {
    fix  $j$ 
    assume  $j \in succ(domain(f))$ 
    with equ  $\langle n \in \_ \rangle$ 
    have  $j \leq n$ 
      using ltI by auto
    with  $\langle n \in \_ \rangle$ 
    consider (lt)  $j < n$  | (eq)  $j = n$ 
      using leD by auto
    then
    have ( $if\ j < n\ then\ f'j\ else\ a$ )  $\in A$ 
    proof cases
      case lt
      with  $\langle f \in \_ \rangle$ 
      show ?thesis
        using apply_type ltD[OF lt] by simp
    next
      case eq
      with  $\langle a \in \_ \rangle$ 
      show ?thesis
        by auto
    qed
  }
  with equ
  show ?thesis
    unfolding seq_upd_def
    using lam_type[of  $succ(domain(f))$ ]
    by auto
qed

```

```

lemma seq_upd_type :
  assumes  $f \in A^{<\omega}\ a \in A$ 
  shows  $seq\_upd(f,a) \in A^{<\omega}$ 
proof -
  from  $\langle f \in \_ \rangle$ 
  obtain  $y$  where  $y \in nat\ f \in y \rightarrow A$ 
    unfolding seqspace_def by blast
  with  $\langle a \in A \rangle$ 
  have  $seq\_upd(f,a) \in succ(y) \rightarrow A$ 
    using seq_upd_succ_type by simp

```

with $\langle y \in _ \rangle$
show *?thesis*
unfolding *seqspace_def* **by** *auto*
qed

lemma *seq_upd_apply_domain* [*simp*]:
assumes $f: n \rightarrow A$ $n \in \text{nat}$
shows $\text{seq_upd}(f, a)^{\langle n \rangle} = a$
unfolding *seq_upd_def* **using** *assms domain_of_fun* **by** *auto*

lemma *zero_in_seqspace* :
shows $0 \in A^{<\omega}$
unfolding *seqspace_def*
by *force*

definition
seqlerel :: $i \Rightarrow i$ **where**
seqlerel(A) $\equiv \text{Fnlerel}(A^{<\omega})$

definition
seqle :: i **where**
seqle $\equiv \text{seqlerel}(2)$

lemma *seqleI*[*intro!*]:
 $\langle f, g \rangle \in 2^{<\omega} \times 2^{<\omega} \implies g \subseteq f \implies \langle f, g \rangle \in \text{seqle}$
unfolding *seqle_def seqlerel_def seqspace_def Rrel_def Fnlerel_def*
by *blast*

lemma *seqleD*[*dest!*]:
 $z \in \text{seqle} \implies \exists x y. \langle x, y \rangle \in 2^{<\omega} \times 2^{<\omega} \wedge y \subseteq x \wedge z = \langle x, y \rangle$
unfolding *Rrel_def seqle_def seqlerel_def Fnlerel_def*
by *blast*

lemma *upd_leI* :
assumes $f \in 2^{<\omega}$ $a \in 2$
shows $\langle \text{seq_upd}(f, a), f \rangle \in \text{seqle}$ (**is** $\langle ?f, _ \rangle \in _$)

proof
show $\langle ?f, f \rangle \in 2^{<\omega} \times 2^{<\omega}$
using *assms seq_upd_type* **by** *auto*

next
show $f \subseteq \text{seq_upd}(f, a)$
proof
fix x
assume $x \in f$
moreover from $\langle f \in 2^{<\omega} \rangle$
obtain n **where** $n \in \text{nat}$ $f : n \rightarrow 2$
by *blast*
moreover from *calculation*
obtain y **where** $y \in n$ $x = \langle y, f^{\langle y \rangle} \rangle$

```

    using Pi_memberD[of f n λ_ . 2]
    by blast
  moreover from ⟨f:n→2⟩
  have domain(f) = n
    using domain_of_fun by simp
  ultimately
  show x ∈ seq_upd(f,a)
    unfolding seq_upd_def lam_def
    by (auto intro:ltI)
qed
qed

lemma preorder_on_seqle: preorder_on(2<sup>ω,seqle)
  unfolding preorder_on_def refl_def trans_on_def by blast

lemma zero_seqle_max: x∈2<sup>ω ⇒ ⟨x,0⟩ ∈ seqle
  using zero_in_seqspace
  by auto

interpretation sp:forcing_notion 2<sup>ω seqle 0
  using preorder_on_seqle zero_seqle_max zero_in_seqspace
  by unfold_locales simp_all

notation sp.Leq (infixl <sup>≤s> 50)
notation sp.Incompatible (infixl <sup>⊥s> 50)

lemma seqspace_separative:
  assumes f∈2<sup>ω
  shows seq_upd(f,0) ⊥s seq_upd(f,1) (is ?f ⊥s ?g)
proof
  assume sp.compat(?f, ?g)
  then
  obtain h where h ∈ 2<sup>ω ?f ⊆ h ?g ⊆ h
    by blast
  moreover from ⟨f∈_⟩
  obtain y where y∈nat f:y→2
    by blast
  moreover from this
  have ?f: succ(y) → 2 ?g: succ(y) → 2
    using seq_upd_succ_type by blast+
  moreover from this
  have ⟨y,?f'y⟩ ∈ ?f ⟨y,?g'y⟩ ∈ ?g
    using apply_Pair by auto
  ultimately
  have ⟨y,0⟩ ∈ h ⟨y,1⟩ ∈ h
    by auto
  moreover from ⟨h ∈ 2<sup>ω⟩
  obtain n where n∈nat h:n→2
    by blast

```

```

ultimately
show False
  using fun_is_function[of h n λ_ 2]
  unfolding seqspace_def function_def by auto
qed

```

```

definition seqleR_fm :: i ⇒ i where
  seqleR_fm(fg) ≡ Exists(Exists(And(pair_fm(0,1,fg+ω 2),subset_fm(1,0))))

```

```

lemma type_seqleR_fm : fg ∈ nat ⇒ seqleR_fm(fg) ∈ formula
  unfolding seqleR_fm_def
  by simp

```

arity_theorem for *seqleR_fm*

```

lemma (in M_ctm1) seqleR_fm_sats :
  assumes fg ∈ nat env ∈ list(M)
  shows (M, env ⊨ seqleR_fm(fg)) ↔ (∃ f[##M]. ∃ g[##M]. pair(##M,f,g,nth(fg,env))
  ∧ f ⊇ g)
  unfolding seqleR_fm_def
  using assms trans_M sats_subset_fm pair_iff_sats
  by auto

```

```

context M_ctm1
begin

```

```

lemma seqle_in_M: seqle ∈ M
  using arity_seqleR_fm seqleR_fm_sats type_seqleR_fm
  cartprod_closed seqspace_closed nat_into_M nat_in_M pair_in_M_iff
  unfolding seqle_def seqlel_def Rrel_def Fnlerel_def
  by (rule_tac Collect_in_M[of seqleR_fm(0) []],auto)

```

26.1 Cohen extension is proper

```

interpretation ctm_separative  $2^{<\omega}$  seqle 0

```

```

proof (unfold locales)

```

```

  fix f

```

```

  let ?q=seq_upd(f,0) and ?r=seq_upd(f,1)

```

```

  assume f ∈  $2^{<\omega}$ 

```

```

  then

```

```

  have ?q ≼s f ∧ ?r ≼s f ∧ ?q ⊥s ?r

```

```

    using upd_leI seqspace_separative by auto

```

```

  moreover from calculation

```

```

  have ?q ∈  $2^{<\omega}$  ?r ∈  $2^{<\omega}$ 

```

```

    using seq_upd_type[of f 2] by auto

```

```

  ultimately

```

```

  show ∃ q ∈  $2^{<\omega}$ . ∃ r ∈  $2^{<\omega}$ . q ≼s f ∧ r ≼s f ∧ q ⊥s r

```

```

    by (rule_tac beXI) + — why the heck auto-tools don't solve this?

```

```

next

```

```

  show  $2^{<\omega} \in M$ 
    using nat_into_M seqspace_closed by simp
next
  show seqle  $\in M$ 
    using seqle_in_M .
qed

lemma cohen_extension_is_proper:  $\exists G. M\_generic(G) \wedge M \neq M[G]$ 
  using proper_extension generic_filter_existence zero_in_seqspace
  by force

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 :  $\exists f. f \in \text{surj}(\omega, M[G])$ 
proof -
  let  $?f = \lambda n \in \omega. \text{val}(G, \text{enum } 'n)$ 
  have  $x \in \omega \implies \text{val}(G, \text{enum } 'x) \in M[G]$  for  $x$ 
    using GenExtI bij_is_fun[OF M_countable]
    by simp
  then
  have  $?f: \omega \rightarrow M[G]$ 
    using lam_type[of \omega \lambda n. \text{val}(G, \text{enum } 'n) \lambda_. M[G]] by simp
  moreover
  have  $\exists n \in \omega. ?f 'n = x$  if  $x \in M[G]$  for  $x$ 
    using that GenExt_iff[of _ G] bij_is_surj[OF M_countable]
    unfolding surj_def by auto
  ultimately
  show thesis
    unfolding surj_def by blast
qed

```

```

lemma (in G_generic1) MG_eqpoll_nat:  $M[G] \approx \omega$ 
proof -
  obtain  $f$  where  $f \in \text{surj}(\omega, M[G])$ 
    using surj_nat_MG by blast

```

```

then
have  $M[G] \lesssim \omega$ 
  using well_ord_surj_imp_lepoll well_ord_Memrel[of  $\omega$ ] by simp
moreover
have  $\omega \lesssim M[G]$ 
  using ext.nat_into_M subset_imp_lepoll by (auto del:lepollI)
ultimately
show ?thesis
  using eqpollI by simp
qed

```

27.2 Extensions of ctms of fragments of ZFC

```

context G_generic1
begin

```

```

lemma sats_ground_repl_fm_imp_sats_ZF_replacement_fm:
  assumes
     $\varphi \in \text{formula } M, [] \models \cdot \text{Replacement}(\text{ground\_repl\_fm}(\varphi)) \cdot$ 
  shows
     $M[G], [] \models \cdot \text{Replacement}(\varphi) \cdot$ 
  using assms sats_ZF_replacement_fm_iff
  by (auto simp:replacement_assm_def ground_replacement_assm_def
      intro:strong_replacement_in_MG[simplified])

```

```

lemma satT_ground_repl_fm_imp_satT_ZF_replacement_fm:
  assumes
     $\Phi \subseteq \text{formula } M \models \{ \cdot \text{Replacement}(\text{ground\_repl\_fm}(\varphi)) \cdot \mid \varphi \in \Phi \}$ 
  shows
     $M[G] \models \{ \cdot \text{Replacement}(\varphi) \cdot \mid \varphi \in \Phi \}$ 
  using assms sats_ground_repl_fm_imp_sats_ZF_replacement_fm
  by auto

```

```

end — G_generic1

```

```

theorem extensions_of_ctms:

```

```

  assumes
     $M \approx \omega \text{ Transset}(M)$ 
     $M \models \cdot Z \cup \{ \cdot \text{Replacement}(p) \cdot \mid p \in \text{overhead} \}$ 
     $\Phi \subseteq \text{formula } M \models \{ \cdot \text{Replacement}(\text{ground\_repl\_fm}(\varphi)) \cdot \mid \varphi \in \Phi \}$ 
  shows
     $\exists N.$ 
     $M \subseteq N \wedge N \approx \omega \wedge \text{Transset}(N) \wedge M \neq N \wedge$ 
     $(\forall \alpha. \text{Ord}(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N)) \wedge$ 
     $((M, [] \models \cdot AC \cdot) \longrightarrow N, [] \models \cdot AC \cdot) \wedge N \models \cdot Z \cup \{ \cdot \text{Replacement}(\varphi) \cdot \mid \varphi \in \Phi \}$ 

```

```

proof -

```

```

  from  $\langle M \models \cdot Z \cup \_ \rangle$   $\langle \text{Transset}(M) \rangle$ 
  interpret M_ZF_ground_trans M
  using M_satT_imp_M_ZF_ground_trans

```

```

  by simp
from ⟨M ≈ ω⟩
obtain enum where enum ∈ bij(ω, M)
  using eqpoll_sym unfolding eqpoll_def by blast
then
interpret M_ctm1 M enum by unfold_locales
interpret forcing_data1 2<sup>ω</sup> seqle 0 M enum
  using nat_into_M seqspace_closed seqle_in_M
  by unfold_locales simp
obtain G where M_generic(G) M ≠ M[G]
  using cohen_extension_is_proper
  by blast

```

Recall that $M[G]$ denotes the generic extension of M using the poset of sequences $2^{<\omega}$.

```

then
interpret G_generic1 2<sup>ω</sup> seqle 0 _ enum G by unfold_locales
interpret MG: M_Z_basic M[G]
  using generic_pairing_in_MG
  Union_MG extensionality_in_MG power_in_MG
  foundation_in_MG replacement_assm_MG
  separation_in_MG infinity_in_MG replacement_ax1
  by unfold_locales simp
have M, [] ⊨ ·AC· ⇒ M[G], [] ⊨ ·AC·
proof -
  assume M, [] ⊨ ·AC·
  then
  have choice_ax(##M)
    unfolding ZF_choice_fm_def using ZF_choice_auto by simp
  then
  have choice_ax(##M[G]) using choice_in_MG by simp
  then
  show M[G], [] ⊨ ·AC·
    using ZF_choice_auto sats_ZFC_iff_sats_ZF_AC
    unfolding ZF_choice_fm_def by simp
qed
moreover
note ⟨M ≠ M[G]⟩ ⟨M ⊨ { ·Replacement(ground_repl_fm(φ)) · φ ∈ Φ }⟩ ⟨Φ ⊆
formula⟩
moreover
have Transset(M[G]) using Transset_MG .
moreover
have M ⊆ M[G] using M_subset_MG[OF one_in_G] generic by simp
ultimately
show ?thesis
  using Ord_MG_iff MG_eqpoll_nat ext.M_satT_Zermelo_fms
  satT_ground_repl_fm_imp_satT_ZF_replacement_fm[of Φ]
  by (rule_tac x=M[G] in exI, auto)
qed

```

```

lemma ZF_replacement_overhead_sub_ZF:  $\{\cdot\text{Replacement}(p)\cdot \ . \ p \in \text{overhead}\}$ 
 $\subseteq \text{ZF}$ 
  using instances1_fms_type instances_ground_fms_type
  unfolding overhead_def ZF_def ZF_schemes_def by auto

theorem extensions_of_ctms_ZF:
  assumes
     $M \approx \omega \text{ Transset}(M) \ M \models \text{ZF}$ 
  shows
     $\exists N.$ 
       $M \subseteq N \wedge N \approx \omega \wedge \text{Transset}(N) \wedge N \models \text{ZF} \wedge M \neq N \wedge$ 
       $(\forall \alpha. \text{Ord}(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N)) \wedge$ 
       $((M, \models \cdot \text{AC}\cdot) \longrightarrow N \models \text{ZFC})$ 
  proof -
    from assms
    have  $\exists N.$ 
       $M \subseteq N \wedge N \approx \omega \wedge \text{Transset}(N) \wedge M \neq N \wedge$ 
       $(\forall \alpha. \text{Ord}(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N)) \wedge$ 
       $((M, \models \cdot \text{AC}\cdot) \longrightarrow N, \models \cdot \text{AC}\cdot) \wedge N \models \cdot \text{Z}\cdot \cup \{\cdot \text{Replacement}(\varphi)\cdot \ . \ \varphi \in$ 
formula
      using extensions_of_ctms[of M formula] satT_ZF_imp_satT_Z[of M]
      satT_mono[OF_ground_repl_fm_sub_ZF, of M]
      satT_mono[OF_ZF_replacement_overhead_sub_ZF, of M]
      by (auto simp: satT_Un_iff)
    then
      obtain  $N$  where  $N \models \cdot \text{Z}\cdot \cup \{\cdot \text{Replacement}(\varphi)\cdot \ . \ \varphi \in \text{formula}\} \ M \subseteq N \ N \approx \omega$ 
      Transset( $N$ )
       $M \neq N \ (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N)$ 
       $(M, \models \cdot \text{AC}\cdot) \longrightarrow N, \models \cdot \text{AC}\cdot$ 
      by blast
      moreover from  $\langle N \models \cdot \text{Z}\cdot \cup \{\cdot \text{Replacement}(\varphi)\cdot \ . \ \varphi \in \text{formula}\} \rangle$ 
      have  $N \models \text{ZF}$ 
      using satT_Z_ZF_replacement_imp_satT_ZF by auto
      moreover from this and  $\langle (M, \models \cdot \text{AC}\cdot) \longrightarrow N, \models \cdot \text{AC}\cdot \rangle$ 
      have  $(M, \models \cdot \text{AC}\cdot) \longrightarrow N \models \text{ZFC}$ 
      using sats_ZFC_iff_sats_ZF_AC by simp
      ultimately
      show ?thesis
      by auto
  qed

end

```

28 Preservation of cardinals in generic extensions

```

theory Cardinal_Preservation
  imports
    Forcing_Main

```

```

begin

context forcing_data1

begin

lemma antichain_abs' [absolut]:
  [ A ∈ M ] ⇒ antichainM(ℙ, leq, A) ↔ antichain(ℙ, leq, A)
  unfolding antichain_rel_def antichain_def compat_def
  using transitivity[of _ A]
  by (auto simp add: absolut)

lemma inconsistent_imp_incompatible:
  assumes p ⊨ φ env q ⊨ Neg(φ) env p ∈ ℙ q ∈ ℙ
    arity(φ) ≤ length(env) φ ∈ formula env ∈ list(M)
  shows p ⊥ q
proof
  assume compat(p, q)
  then
  obtain d where d ≤ p d ≤ q d ∈ ℙ by blast
  moreover
  note assms
  moreover from calculation
  have d ⊨ φ env d ⊨ Neg(φ) env
    using strengthening_lemma by simp_all
  ultimately
  show False
    using Forces_Neg[of d env φ] refl_leq
    by (auto dest: transitivity; drule_tac bspec; auto dest: transitivity)
qed

notation check (⟨_v⟩ [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 ⊆ ext: M_ZFC2_trans M[G]
  using ground_replacements3 replacement_assm_MG
  by unfold_locales simp_all

lemma (in forcing_data1) forces_neq_apply_imp_incompatible:

```

```

assumes
  p ⊢ ·0'1 is 2· [f,a,bv]
  q ⊢ ·0'1 is 2· [f,a,b'v]
  b ≠ b'
  — More general version: taking general names bv and b'v, satisfying p ⊢ ·¬·0 =
1· [bv, b'v] and q ⊢ ·¬·0 = 1· [bv, b'v].
and
  types:f∈M a∈M b∈M b'∈M p∈ℙ q∈ℙ
shows
  p ⊥ q
proof -
  {
    fix G
    assume M_generic(G)
    then
    interpret G_generic1 _ _ _ _ G by unfold_locales
    include G_generic1_lemmas
    assume q∈G
    with assms ⟨M_generic(G)⟩
    have M[G], map(val(G),[f,a,b'v]) ⊨ ·0'1 is 2·
      using truth_lemma[of ·0'1 is 2· [f,a,b'v]]
      by (auto simp add:ord_simp_union arity_fun_apply_fm
        fun_apply_type)
    with ⟨b ≠ b'⟩ types
    have M[G], map(val(G),[f,a,bv]) ⊨ ·¬·0'1 is 2·
      using GenExtI by auto
  }
with types
have q ⊢ ·¬·0'1 is 2· [f,a,bv]
  using definition_of_forcing[where φ=·¬·0'1 is 2· ]
  by (auto simp add:ord_simp_union arity_fun_apply_fm)
with ⟨p ⊢ ·0'1 is 2· [f,a,bv⟩ and types
show p ⊥ q
  using inconsistent_imp_incompatible
  by (simp add:ord_simp_union arity_fun_apply_fm fun_apply_type)
qed

```

```

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.quaselist_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.rer_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 $\exists q \in G. q \preceq p \wedge q \Vdash \varphi \ \text{env}$

proof -

note *assms*

moreover from *this*

obtain r **where** $r \Vdash \varphi \ \text{env} \ r \in G$

using *generic_truth_lemma*[of $\varphi \ \text{env}$]

by *blast*

moreover from *this* **and** $\langle p \in G \rangle$

obtain q **where** $q \preceq p \ q \preceq r \ q \in G$ **by** *auto*

ultimately

show *?thesis*

using *strengthening_lemma*[of $r \ \varphi \ _ \ \text{env}$] **by** *blast*

qed

28.1 Preservation by ccc forcing notions

lemma *ccc_fun_closed_lemma_aux*:

assumes $f_dot \in M \ p \in M \ a \in M \ b \in M$

shows $\{q \in \mathbb{P} . q \preceq p \wedge (M, [q, \mathbb{P}, \text{leq}, \mathbf{1}, f_dot, a^v, b^v] \models \text{forces}(\cdot 0'1 \text{ is } 2.))\} \in M$

using *separation_forces*[**where** $\text{env} = [f_dot, a^v, b^v]$ **and** $\varphi = \cdot 0'1 \text{ is } 2., \text{simplified}$]

assms G_subset_M [*THEN* *subsetD*] *generic*

separation_in *lam_replacement_constant* *lam_replacement_identity*

lam_replacement_product

separation_conj *arity_fun_apply_fm* *union_abs1*

by *simp_all*

lemma *ccc_fun_closed_lemma_aux2*:

assumes $B \in M \ f_dot \in M \ p \in M \ a \in M$

shows $(\#\#M)(\lambda b \in B. \{q \in \mathbb{P} . q \preceq p \wedge (M, [q, \mathbb{P}, \text{leq}, \mathbf{1}, f_dot, a^v, b^v] \models \text{forces}(\cdot 0'1 \text{ is } 2.))\})$

proof -

have *separation*($\#\#M, \lambda z. M, [\text{snd}(z), \mathbb{P}, \text{leq}, \mathbf{1}, f_dot, \tau, \text{fst}(z)^v] \models \text{forces}(\cdot 0'1 \text{ is } 2.))$)

if $\tau \in M$ **for** τ

proof -

let $?f_fm = \text{snd_fm}(1, 0)$

let $?g_fm = \text{hcomp_fm}(\text{check_fm}(6), \text{fst_fm}, 2, 0)$

note *assms*

moreover

have $\text{arity}(\text{forces}(\cdot 0'1 \text{ is } 2.)) \leq 7$

using *arity_fun_apply_fm* *union_abs1* *arity_forces*[of $\cdot 0'1 \text{ is } 2.$]

by *simp*

moreover

```

have ?f_fm ∈ formula arity(?f_fm) ≤ 7 ?g_fm ∈ formula arity(?g_fm) ≤ 8
  using ord_simp_union
  unfolding hcomp_fm_def
  by (simp_all add:arity)
ultimately
show ?thesis
  using separation_sat_after_function assms that sats_fst_fm
    snd_abs sats_snd_fm sats_check_fm check_abs fst_abs
  unfolding hcomp_fm_def
  by simp
qed
with assms
show ?thesis
  using lam_replacement_imp_lam_closed separation_conj separation_in
    lam_replacement_product lam_replacement_constant transitivity[of _ B]
    lam_replacement_snd lam_replacement_Collect' ccc_fun_closed_lemma_aux
  by simp
qed

lemma ccc_fun_closed_lemma:
  assumes A ∈ M B ∈ M f_dot ∈ M p ∈ M
  shows (λa ∈ A. {b ∈ B. ∃ q ∈ P. q ≤ p ∧ (q ⊢ ·0'1 is 2· [f_dot, av, bv])}) ∈ M
proof -
  have separation(###M, λz. M, [snd(z), P, leq, 1, f_dot, fst(fst(z))v, snd(fst(z))v])
  ⊢ forces(·0'1 is 2· )
proof -
  let ?f_fm = snd_fm(1,0)
  let ?g = λz . fst(fst(fst(z)))v
  let ?g_fm = hcomp_fm(check_fm(6), hcomp_fm(fst_fm, fst_fm), 2, 0)
  let ?h_fm = hcomp_fm(check_fm(7), hcomp_fm(snd_fm, fst_fm), 3, 0)
  note assms
  moreover
  have arity(forces(·0'1 is 2· )) ≤ 7
    using arity_fun_apply_fm union_abs1 arity_forces[of ·0'1 is 2· ]
    by simp
  moreover
  have ?f_fm ∈ formula arity(?f_fm) ≤ 6 ?g_fm ∈ formula arity(?g_fm) ≤ 7
    ?h_fm ∈ formula arity(?h_fm) ≤ 8
    using ord_simp_union
    unfolding hcomp_fm_def
    by (simp_all add:arity)
  ultimately
  show ?thesis
    using separation_sat_after_function3 assms sats_check_fm check_abs
      fst_abs snd_abs
    unfolding hcomp_fm_def
    by simp
qed
moreover

```

```

have 1:separation(##M, λz. M, [snd(z), ℙ, leq, 1, f_dot, τ, fst(z)v] ⊨ forces(·0'1
is 2. ))
  if τ∈M for τ
proof -
  let ?f_fm=snd_fm(1,0)
  let ?g_fm=hcomp_fm(check_fm(6),fst_fm,2,0)
  note assms
  moreover
  have arity(forces(·0'1 is 2. )) ≤ 7
    using arity_forces[of ·0'1 is 2. ] arity_fun_apply_fm union_abs1
    by simp
  moreover
  have ?f_fm ∈ formula arity(?f_fm) ≤ 7 ?g_fm ∈ formula arity(?g_fm) ≤ 8
    using ord_simp_union
    unfolding hcomp_fm_def
    by (simp_all add:arity)
  ultimately
  show ?thesis
    using separation_sat_after_function that fst_abs snd_abs sats_check_fm
check_abs
    unfolding hcomp_fm_def
    by simp
qed
moreover note assms
ultimately
show ?thesis
  using lam_replacement_imp_lam_closed lam_replacement_Collect' transitivity[of _ A]
  lam_replacement_constant lam_replacement_identity lam_replacement_snd
  lam_replacement_product separation_conj separation_in separation_bex separation_iff'
  by simp
qed

```

— Kunen IV.3.5

lemma *ccc_fun_approximation_lemma*:

notes *le_trans[trans]*

assumes $ccc^M(\mathbb{P}, leq)$ $A \in M$ $B \in M$ $f \in M[G]$ $f : A \rightarrow B$

shows

$\exists F \in M. F : A \rightarrow Pow^M(B) \wedge (\forall a \in A. f'a \in F'a \wedge |F'a|^M \leq \omega)$

proof -

from $\langle f \in M[G] \rangle$

obtain f_dot **where** $f = val(G, f_dot)$ $f_dot \in M$ **using** *GenExtD* **by** *force*

with *assms*

obtain p **where** $p \Vdash \cdot 0 : 1 \rightarrow 2. [f_dot, A^v, B^v]$ $p \in G$ $p \in M$

using *G_subset_M truth_lemma*[of $\cdot 0 : 1 \rightarrow 2. [f_dot, A^v, B^v]$]

by (*auto simp add:ord_simp_union arity_typed_function_fm*

— NOTE: type-checking is not performed here by the Simplifier

typed_function_type)

```

define  $F$  where  $F \equiv \lambda a \in A. \{b \in B. \exists q \in \mathbb{P}. q \preceq p \wedge (q \Vdash \cdot 0'1 \text{ is } 2 \cdot [f\_dot, a^v, b^v])\}$ 
from  $assms \langle f\_dot \in \_ \rangle \langle p \in M \rangle$ 
have  $F \in M$ 
  unfolding  $F\_def$  using  $ccc\_fun\_closed\_lemma$  by  $simp$ 
moreover from  $calculation$ 
have  $f'a \in F'a$  if  $a \in A$  for  $a$ 
proof -
  note  $\langle f: A \rightarrow B \rangle \langle a \in A \rangle$ 
  moreover from  $this$ 
  have  $f'a \in B$  by  $simp$ 
  moreover
  note  $\langle f \in M[G] \rangle \langle A \in M \rangle$ 
  moreover from  $calculation$ 
  have  $M[G], [f, a, f'a] \models \cdot 0'1 \text{ is } 2 \cdot$ 
    by  $(auto \ dest:transitivity)$ 
  moreover
  note  $\langle B \in M \rangle \langle f = val(G, f\_dot) \rangle$ 
  moreover from  $calculation$ 
  have  $a \in M \ val(G, f\_dot)'a \in M$ 
    by  $(auto \ dest:transitivity)$ 
  moreover
  note  $\langle f\_dot \in M \rangle \langle p \in G \rangle$ 
  ultimately
  obtain  $q$  where  $q \preceq p \ q \Vdash \cdot 0'1 \text{ is } 2 \cdot [f\_dot, a^v, (f'a)^v] \ q \in G$ 
    using  $forces\_below\_filter[of \cdot 0'1 \text{ is } 2 \cdot [f\_dot, a^v, (f'a)^v] \ p]$ 
    by  $(auto \ simp \ add: \ ord\_simp\_union \ arity\_fun\_apply\_fm \ fun\_apply\_type)$ 
  with  $\langle f'a \in B \rangle$ 
  have  $f'a \in \{b \in B. \exists q \in \mathbb{P}. q \preceq p \wedge q \Vdash \cdot 0'1 \text{ is } 2 \cdot [f\_dot, a^v, b^v]\}$ 
    by  $blast$ 
  with  $\langle a \in A \rangle$ 
  show  $?thesis$  unfolding  $F\_def$  by  $simp$ 
qed
moreover
have  $|F'a|^M \leq \omega \wedge F'a \in M$  if  $a \in A$  for  $a$ 
proof -
  let  $?Q = \lambda b. \{q \in \mathbb{P}. q \preceq p \wedge (q \Vdash \cdot 0'1 \text{ is } 2 \cdot [f\_dot, a^v, b^v])\}$ 
from  $\langle F \in M \rangle \langle a \in A \rangle \langle A \in M \rangle$ 
have  $F'a \in M \ a \in M$ 
  using  $transitivity[OF \_ \langle A \in M \rangle]$  by  $simp\_all$ 
moreover
have  $2: \bigwedge x. x \in F'a \implies x \in M$ 
  using  $transitivity[OF \_ \langle F'a \in M \rangle]$  by  $simp$ 
moreover
have  $3: \bigwedge x. x \in F'a \implies (\#\#M)(?Q(x))$ 
  using  $ccc\_fun\_closed\_lemma\_aux[OF \langle f\_dot \in M \rangle \langle p \in M \rangle \langle a \in M \rangle 2]$   $transi-$ 
 $tivity[of \_ F'a]$ 
  by  $simp$ 
moreover

```

```

have  $4$ : lam_replacement(##M,  $\lambda b. \{q \in \mathbb{P} . q \preceq p \wedge (M, [q, \mathbb{P}, \text{leq}, \mathbf{1}, f\_dot, a^v, b^v] \models \text{forces}(\cdot 0'1 \text{ is } 2 \cdot))\}$ )
using ccc_fun_closed_lemma_aux2[OF  $\langle f\_dot \in M \rangle \langle p \in M \rangle \langle a \in M \rangle$ ]
      lam_replacement_iff_lam_closed[THEN iffD2]
      ccc_fun_closed_lemma_aux[OF  $\langle f\_dot \in M \rangle \langle p \in M \rangle \langle a \in M \rangle$ ]
by simp
ultimately
interpret M_Pi_assumptions_choice ##M F'a ?Q
using Pi_replacement1[OF  $\_ 3$ ] lam_replacement_Sigfun[OF  $4$ ]
      lam_replacement_imp_strong_replacement
      ccc_fun_closed_lemma_aux[OF  $\langle f\_dot \in M \rangle \langle p \in M \rangle \langle a \in M \rangle$ ]
      lam_replacement_hcomp2[OF lam_replacement_constant  $4 \_ \_$ ]
      lam_replacement_minimum, unfolded lam_replacement_def]
by unfold_locales simp_all
from  $\langle F'a \in M \rangle$ 
interpret M_Pi_assumptions2 ##M F'a ?Q  $\lambda \_ . \mathbb{P}$ 
using lam_replacement_imp_strong_replacement[OF
      lam_replacement_Sigfun[OF lam_replacement_constant]]
      Pi_replacement1 transitivity[of  $\_ F'a$ ]
by unfold_locales simp_all
from  $\langle p \Vdash \cdot 0:1 \rightarrow 2 \cdot [f\_dot, A^v, B^v] \rangle \langle a \in A \rangle$ 
have  $\exists y. y \in ?Q(b)$  if  $b \in F'a$  for  $b$ 
using that unfolding F_def by auto
then
obtain  $q$  where  $q \in Pi^M(F'a, ?Q)$   $q \in M$  using AC_Pi_rel by auto
moreover
note  $\langle F'a \in M \rangle$ 
moreover from calculation
have  $q : F'a \rightarrow^M \mathbb{P}$ 
using Pi_rel_weaken_type def_function_space_rel by auto
moreover from calculation
have  $q : F'a \rightarrow \text{range}(q)$   $q : F'a \rightarrow \mathbb{P}$   $q : F'a \rightarrow^M \text{range}(q)$ 
using mem_function_space_rel_abs range_of_fun by simp_all
moreover
have  $q'b \perp q'c$  if  $b \in F'a$   $c \in F'a$   $b \neq c$ 
  — For the next step, if the premise  $b \neq c$  is first, the proof breaks down badly
for  $b$   $c$ 
proof -
from  $\langle b \in F'a \rangle \langle c \in F'a \rangle \langle q \in Pi^M(F'a, ?Q) \rangle \langle q \in M \rangle$ 
have  $q'b \Vdash \cdot 0'1 \text{ is } 2 \cdot [f\_dot, a^v, b^v]$ 
       $q'c \Vdash \cdot 0'1 \text{ is } 2 \cdot [f\_dot, a^v, c^v]$ 
using mem_Pi_rel_abs[of  $q$ ] apply_type[of  $\_ \_ ?Q$ ]
by simp_all
with  $\langle b \neq c \rangle \langle q : F'a \rightarrow \mathbb{P} \rangle \langle a \in A \rangle \langle b \in \_ \rangle \langle c \in \_ \rangle$ 
       $\langle A \in M \rangle \langle f\_dot \in M \rangle \langle F'a \in M \rangle$ 
show ?thesis
using forces_neq_apply_imp_incompatible
      transitivity[of  $\_ A$ ] transitivity[of  $\_ F'a$ ]
by auto

```

```

qed
moreover from calculation
have antichain( $\mathbb{P}$ , leq, range(q))
  using Pi_range_eq[of _ _  $\lambda$  .  $\mathbb{P}$ ]
  unfolding antichain_def compat_in_def by auto
moreover from this and  $\langle q \in M \rangle$ 
have antichainM( $\mathbb{P}$ , leq, range(q))
  by (simp add: absolut del: P_in_M)
moreover from calculation
have  $q'b \neq q'c$  if  $b \neq c$   $b \in F'a$   $c \in F'a$  for  $b$   $c$ 
  using that Incompatible_imp_not_eq apply_type
  mem_function_space_rel_abs by simp
ultimately
have  $q \in \text{inj}^M(F'a, \text{range}(q))$ 
  using def_inj_rel by auto
with  $\langle F'a \in M \rangle$   $\langle q \in M \rangle$ 
have  $|F'a|^M \leq |\text{range}(q)|^M$ 
  using def_lepoll_rel
  by (rule_tac lepoll_rel_imp_cardinal_rel_le) auto
also from  $\langle \text{antichain}^M(\mathbb{P}, \text{leq}, \text{range}(q)) \rangle$   $\langle \text{ccc}^M(\mathbb{P}, \text{leq}) \rangle$   $\langle q \in M \rangle$ 
have  $|\text{range}(q)|^M \leq \omega$ 
  using def_ccc_rel by simp
finally
show ?thesis using  $\langle F'a \in M \rangle$  by auto
qed
moreover from this
have  $F'a \in M$  if  $a \in A$  for  $a$ 
  using that by simp
moreover from this  $\langle B \in M \rangle$ 
have  $F : A \rightarrow \text{Pow}^M(B)$ 
  using Pow_rel_char
  unfolding F_def by (rule_tac lam_type) auto
ultimately
show ?thesis by auto
qed

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 F_n . But it would have been more appropriate to do the following using the relative F_n_rel . As it turns out, the present theory was developed prior to having F_n relativized!

We also note that $F_n(\omega, \kappa \times \omega, \mathcal{P})$ is separative, i.e. each $X \in F_n(\omega, \kappa \times \omega, \mathcal{P})$ 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**
 $Add_subs(\kappa) \equiv F_n(\omega, \kappa \times \omega, \mathcal{P})$

abbreviation

$Add_le :: i \Rightarrow i$ **where**
 $Add_le(\kappa) \equiv Fnle(\omega, \kappa \times \omega, \mathcal{P})$

lemma (in M_aleph) *Aleph_rel2_closed[intro,simp]*: $M(\aleph_2^M)$
using *nat_into_Ord* **by** *simp*

locale $M_master = M_cohen + M_library +$
assumes

UN_lepoll_assumptions:
 $M(A) \Longrightarrow M(b) \Longrightarrow M(f) \Longrightarrow M(A') \Longrightarrow separation(M, \lambda y. \exists x \in A'. y = \langle x, \mu i. x \in if_range_F_else_F((\cdot)(A), b, f, i) \rangle)$

29.1 Non-absolute concepts between extensions

sublocale $M_master \subseteq M_Pi_replacement$
by *unfold_locales*

locale $M_master_sub = M_master + N:M_aleph\ N$ **for** $N +$
assumes

$M_imp_N: M(x) \Longrightarrow N(x)$ **and**
 $Ord_iff: Ord(x) \Longrightarrow M(x) \longleftrightarrow N(x)$

sublocale $M_master_sub \subseteq M_N_Perm$
using M_imp_N **by** *unfold_locales*

context M_master_sub
begin

lemma *cardinal_rel_le_cardinal_rel*: $M(X) \Longrightarrow |X|^N \leq |X|^M$

using $M_imp_N N.lepoll_rel_cardinal_rel_le$ [*OF* $lepoll_rel_transfer$ $Card_rel_is_Ord$]
 $cardinal_rel_eqpoll_rel$ [*THEN* $eqpoll_rel_sym$, *THEN* $eqpoll_rel_imp_lepoll_rel$]
by *simp*

lemma $Aleph_rel_sub_closed$: $Ord(\alpha) \implies M(\alpha) \implies N(\aleph_\alpha^M)$
using Ord_iff [*THEN* $iffD1$, *OF* $Card_rel_Aleph_rel$ [*THEN* $Card_rel_is_Ord$]]
by *simp*

lemma $Card_rel_imp_Card_rel$: $Card^N(\kappa) \implies M(\kappa) \implies Card^M(\kappa)$
using $N.Card_rel_is_Ord$ [*of* κ] M_imp_N $Ord_cardinal_rel_le$ [*of* κ]
 $cardinal_rel_le_cardinal_rel$ [*of* κ] le_anti_sym
unfolding $Card_rel_def$ **by** *auto*

lemma $csucc_rel_le_csucc_rel$:
assumes $Ord(\kappa)$ $M(\kappa)$
shows $(\kappa^+)^M \leq (\kappa^+)^N$
proof -
note *assms*
moreover from *this*
have $N(L) \wedge Card^N(L) \wedge \kappa < L \implies M(L) \wedge Card^M(L) \wedge \kappa < L$
(is $?P(L) \implies ?Q(L)$) **for** L
using M_imp_N Ord_iff [*THEN* $iffD2$, *of* L] $N.Card_rel_is_Ord$ *lt_Ord*
 $Card_rel_imp_Card_rel$ **by** *auto*
moreover from *assms*
have $N((\kappa^+)^N) Card^N((\kappa^+)^N) \kappa < (\kappa^+)^N$
using $N.lt_csucc_rel$ [*of* κ] $N.Card_rel_csucc_rel$ [*of* κ] M_imp_N **by** *simp_all*
ultimately
show *?thesis*
using M_imp_N $Least_antitone$ [*of* $_ ?P ?Q$] **unfolding** $csucc_rel_def$ **by**
blast
qed

lemma $Aleph_rel_le_Aleph_rel$: $Ord(\alpha) \implies M(\alpha) \implies \aleph_\alpha^M \leq \aleph_\alpha^N$
proof (*induct rule:trans_induct3*)
case 0
then
show *?case*
using $Aleph_rel_zero$ $N.Aleph_rel_zero$ **by** *simp*
next
case (*succ* x)
then
have $\aleph_x^M \leq \aleph_x^N$ $Ord(x)$ $M(x)$ **by** *simp_all*
moreover from *this*
have $(\aleph_x^{M+})^M \leq (\aleph_x^{N+})^M$
using M_imp_N Ord_iff [*THEN* $iffD2$, *OF* $N.Card_rel_is_Ord$]
by (*intro* $csucc_rel_le_mono$) *simp_all*
moreover from *calculation*
have $(\aleph_x^{N+})^M \leq (\aleph_x^{N+})^N$
using M_imp_N $N.Card_rel_is_Ord$ Ord_iff [*THEN* $iffD2$, *OF* $N.Card_rel_is_Ord$]

```

    by (intro csucc_rel_le_csucc_rel) auto
  ultimately
  show ?case
    using M_imp_N Aleph_rel_succ N.Aleph_rel_succ csucc_rel_le_csucc_rel
      le_trans by auto
next
  case (limit x)
  then
  show ?case
    using M_imp_N Aleph_rel_limit N.Aleph_rel_limit
      by simp (blast dest: transM intro!:le_implies_UN_le_UN)
qed

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_lespoll_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 ⊆ M_master ##M
  using replacement_trans_apply_image
  by unfold_locales (simp_all add:repl_instances sep_instances del:setclass_iff
    add:transrec_replacement_def wfrec_replacement_def)

```

```

sublocale M_ZFC2_trans ⊆ M_Pi_replacement ##M
  by unfold_locales

```

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 -

```

```

  interpret M_add_reals ##M  $\aleph_2^M \times \omega$ 

```

```

  by unfold_locales blast

```

```

  show ?thesis

```

```

    using ccc_rel_Fn_nat by fast

```

```

qed

```

```

end — M_ctm2_AC

```

```

sublocale G_generic3_AC ⊆ M_master_sub ##M ##(M[G])
  using M_subset_MG[OF one_in_G] generic Ord_MG_iff
  by unfold_locales auto

```

```

lemma (in M_trans) mem_F_bound4:
  fixes F A
  defines  $F \equiv (\cdot)$ 
  shows  $x \in F(A, c) \implies c \in (\text{range}(f) \cup \text{domain}(A))$ 
  using apply_0 unfolding F_def
  by (cases M(c), auto simp:F_def)

lemma (in M_trans) mem_F_bound5:
  fixes F A
  defines  $F \equiv \lambda x. A \cdot x$ 
  shows  $x \in F(A, c) \implies c \in (\text{range}(f) \cup \text{domain}(A))$ 
  using apply_0 unfolding F_def
  by (cases M(c), auto simp:F_def drSR_Y_def dC_F_def)

sublocale M_ctm2_AC  $\subseteq$  M_replacement_lepoll ##M ( $\cdot$ )
  using UN_lepoll_assumptions lam_replacement_apply lam_replacement_inj_rel
    mem_F_bound4 apply_0 lam_replacement_minimum
  unfolding lepoll_assumptions_defs
proof (unfold_locales,
  rule_tac [3] lam_Least_assumption_general [where  $U = \text{domain}$ , OF_mem_F_bound4],
simp_all)
  fix A i x
  assume  $A \in M \ x \in M \ x \in A \cdot i$ 
  then
  show  $i \in M$ 
    using apply_0 [of i A] transM [of  $\text{domain}(A)$ , simplified]
    by force
qed

context G_generic3_AC begin

context
  includes G_generic1_lemmas
begin

lemma G_in_MG:  $G \in M[G]$ 
  using G_in_Gen_Ext
  by blast

lemma ccc_preserves_Aleph_succ:
  assumes  $\text{ccc}^M(\mathbb{P}, \text{leq}) \ \text{Ord}(z) \ z \in M$ 
  shows  $\text{Card}^{M[G]}(\aleph_{\text{succ}(z)}^M)$ 
proof (rule ccontr)
  assume  $\neg \text{Card}^{M[G]}(\aleph_{\text{succ}(z)}^M)$ 
  moreover
  note  $\langle z \in M \rangle \langle \text{Ord}(z) \rangle$ 
  moreover from this
  have  $\text{Ord}(\aleph_{\text{succ}(z)}^M)$ 

```

```

    using Card_rel_is_Ord by fastforce
ultimately
obtain  $\alpha f$  where  $\alpha < \aleph_{succ(z)}^M$   $f \in surj^M[G](\alpha, \aleph_{succ(z)}^M)$ 
    using ext.lt_surj_rel_empty_imp_Card_rel M_subset_MG[OF one_in_G]
    by force
moreover from this and  $\langle z \in M \rangle \langle Ord(z) \rangle$ 
have  $\alpha \in M$   $f \in M[G]$ 
    using ext.trans_surj_rel_closed
    by (auto dest:transM ext.transM dest!:ltD)
moreover
note  $\langle ccc^M(\mathbb{P}, leq) \rangle \langle z \in M \rangle$ 
ultimately
obtain  $F$  where  $F: \alpha \rightarrow Pow^M(\aleph_{succ(z)}^M) \forall \beta \in \alpha. f' \beta \in F' \beta \forall \beta \in \alpha. |F' \beta|^M \leq \omega$ 
 $F \in M$ 
    using ccc_fun_approximation_lemma[of  $\alpha \aleph_{succ(z)}^M f$ ]
    ext.mem_surj_abs[of  $f \alpha \aleph_{succ(z)}^M$ ]  $\langle Ord(z) \rangle$ 
    surj_is_fun[of  $f \alpha \aleph_{succ(z)}^M$ ] by auto
then
have  $\beta \in \alpha \implies |F' \beta|^M \leq \aleph_0^M$  for  $\beta$ 
    using Aleph_rel_zero by simp
have  $w \in F' x \implies x \in M$  for  $w x$ 
proof -
  fix  $w x$ 
  assume  $w \in F' x$ 
  then
  have  $x \in domain(F)$ 
    using apply_0 by auto
  with  $\langle F: \alpha \rightarrow Pow^M(\aleph_{succ(z)}^M) \rangle \langle \alpha \in M \rangle$ 
  show  $x \in M$  using domain_of_fun
    by (auto dest:transM)
qed
with  $\langle \alpha \in M \rangle \langle F: \alpha \rightarrow Pow^M(\aleph_{succ(z)}^M) \rangle \langle F \in M \rangle$ 
interpret  $M\_cardinal\_UN\_lepoll \#\#M \lambda \beta. F' \beta \alpha$ 
  using UN_lepoll_assumptions lepoll_assumptions
  lam_replacement_apply lam_replacement_inj_rel lam_replacement_minimum
proof (unfold locales, auto dest:transM simp del:if_range_F_else_F_def)
  fix  $f b$ 
  assume  $b \in M$   $f \in M$ 
  with  $\langle F \in M \rangle$ 
  show lam_replacement( $\#\#M, \lambda x. \mu i. x \in if\_range\_F\_else\_F((\cdot)^{\cdot})(F), b, f,$ 
i))
    using UN_lepoll_assumptions mem_F_bound5
    by (rule_tac lam_Least_assumption_general[where  $U=domain, OF \_$ 
mem_F_bound5])
    simp_all
qed
from  $\langle \alpha < \aleph_{succ(z)}^M \rangle \langle \alpha \in M \rangle \langle Ord(z) \rangle \langle z \in M \rangle$ 
have  $\alpha \lesssim^M \aleph_z^M$ 

```

```

using
  cardinal_rel_lt_csucc_rel_iff[of  $\aleph_z^M \alpha$ ]
  le_Card_rel_iff[of  $\aleph_z^M \alpha$ ]
  Aleph_rel_succ[of z] Card_rel_lt_iff[of  $\alpha \aleph_{succ(z)}^M$ ]
  lt_Ord[of  $\alpha \aleph_{succ(z)}^M$ ]
  Card_rel_csucc_rel[of  $\aleph_z^M$ ]
  Card_rel_Aleph_rel[THEN Card_rel_is_Ord]
by simp
with  $\langle \alpha < \aleph_{succ(z)}^M, \forall \beta \in \alpha. |F'\beta|^M \leq \omega, \langle \alpha \in M \rangle \text{ assms}$ 
have  $|\bigcup \beta \in \alpha. F'\beta|^M \leq \aleph_z^M$ 
  using InfCard_rel_Aleph_rel[of z] Aleph_rel_zero
    subset_imp_lepoll_rel[THEN lepoll_rel_imp_cardinal_rel_le,
      of  $\bigcup \beta \in \alpha. F'\beta \aleph_z^M$ ] Aleph_rel_succ
    Aleph_rel_increasing[THEN leI, THEN [2] le_trans, of  $\_ 0 z$ ]
    Ord_0_lt_iff[THEN iffD1, of z]
  by (cases 0 < z; rule_tac lepoll_rel_imp_cardinal_rel_UN_le) (auto, force)
moreover
note  $\langle z \in M \rangle \langle Ord(z) \rangle$ 
moreover from  $\langle \forall \beta \in \alpha. f'\beta \in F'\beta \rangle \langle f \in \text{surj}^{M[G]}(\alpha, \aleph_{succ(z)}^M) \rangle$ 
   $\langle \alpha \in M \rangle \langle f \in M[G] \rangle$  and this
have  $\aleph_{succ(z)}^M \subseteq (\bigcup \beta \in \alpha. F'\beta)$ 
  using ext.mem_surj_abs by (force simp add:surj_def)
moreover from  $\langle F \in M \rangle \langle \alpha \in M \rangle$ 
have  $(\bigcup x \in \alpha. F'x) \in M$ 
  using j.B_replacement
  by (intro Union_closed[simplified] RepFun_closed[simplified])
    (auto dest:transM)
ultimately
have  $\aleph_{succ(z)}^M \leq \aleph_z^M$ 
  using subset_imp_le_cardinal_rel[of  $\aleph_{succ(z)}^M \bigcup \beta \in \alpha. F'\beta$ ]
    le_trans by auto
with assms
show False
  using Aleph_rel_increasing_not_le_iff_lt[of  $\aleph_{succ(z)}^M \aleph_z^M$ ]
    Card_rel_Aleph_rel[THEN Card_rel_is_Ord]
  by auto
qed

end — bundle G_generic1_lemmas

end — G_generic3_AC

context M_ctm1
begin

abbreviation
  Add :: i where
  Add  $\equiv Fn(\omega, \aleph_2^M \times \omega, 2)$ 

```

```

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 by unfold_locales auto

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)  $\implies$   $z \in M \implies$  Card $^{M[G]}(\aleph_{succ(z)}^M)$ 
  using ccc_preserves_Aleph_succ ccc_Add_subs_Aleph_2
  by auto

lemma Aleph_rel_nats_MG_eq_Aleph_rel_nats_M:
  includes G_generic1_lemmas
  assumes  $z \in \omega$ 
  shows  $\aleph_z^{M[G]} = \aleph_z^M$ 
  using assms
proof (induct)
  case 0
  show ?case
    by(rule trans[OF ext.Aleph_rel_zero Aleph_rel_zero[symmetric]])
next
  case (succ z)
  then
  have  $\aleph_{succ(z)}^M \leq \aleph_{succ(z)}^{M[G]}$ 
    using Aleph_rel_le_Aleph_rel_nat_into_M by simp
  moreover from  $\langle z \in \omega \rangle$ 
  have  $\aleph_z^M \in M[G]$   $\aleph_{succ(z)}^M \in M[G]$ 
    using nat_into_M by simp_all
  moreover from this and  $\langle \aleph_z^{M[G]} = \aleph_z^M \rangle$   $\langle z \in \omega \rangle$ 
  have  $\aleph_{succ(z)}^{M[G]} \leq \aleph_{succ(z)}^M$ 
    using ext.Aleph_rel_succ_nat_into_M
    Add_subs_preserves_Aleph_succ[THEN ext.csucc_rel_le, of z]
    Aleph_rel_increasing[of z succ(z)]
  by simp
  ultimately
  show ?case using le_anti_sym by blast
qed

abbreviation
  f_G ::  $i$  ( $\langle f_G \rangle$ ) where
  f_G  $\equiv \bigcup G$ 

```

abbreviation

$dom_dense :: i \Rightarrow i$ **where**
 $dom_dense(x) \equiv \{p \in Add . x \in domain(p)\}$

declare (in M_ctm2_AC) Fn_nat_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*]: $x \in \aleph_2^M \times \omega \implies dom_dense(x) \in M$
using $separation_in_domain$ [*of x*] nat_into_M
by ($rule_tac$ $separation_closed$ [*simplified*], $blast$ $dest:transM$) $simp$

lemma $domain_f_G$: **assumes** $x \in \aleph_2^M$ $y \in \omega$
shows $\langle x, y \rangle \in domain(f_G)$

proof -

from $assms$
have $Add = Fn^M(\omega, \aleph_2^M \times \omega, 2)$
using Fn_nat_abs **by** $auto$
moreover from $this$
have $Fnle(\omega, \aleph_2^M \times \omega, 2) = Fnle^M(\omega, \aleph_2^M \times \omega, 2)$
unfolding $Fnle_rel_def$ $Fnle_def$ **by** $auto$
moreover from $calculation$ $assms$
have $dense(dom_dense(\langle x, y \rangle))$
using $dense_dom_dense$ [*of $\langle x, y \rangle$ $\aleph_2^M \times \omega$ ω 2*] $InfCard_rel_nat$
unfolding $dense_def$ **by** $auto$
with $assms$
obtain p **where** $p \in dom_dense(\langle x, y \rangle)$ $p \in G$
using $M_generic_denseD$ [*of $dom_dense(\langle x, y \rangle)$*]
by $auto$
then
show $\langle x, y \rangle \in domain(f_G)$ **by** $blast$

qed

lemma $f_G_funtype$:

includes $G_generic1_lemmas$
shows $f_G : \aleph_2^M \times \omega \rightarrow 2$
using $generic$ $domain_f_G$ Pi_iff $Un_filter_is_function$ $generic$
 $subset_trans$ [*OF filter_subset_notion Fn_nat_subset_Pow*]
by $force$

lemma inj_dense_closed [*intro,simp*]:

$w \in \aleph_2^M \implies x \in \aleph_2^M \implies inj_dense(\aleph_2^M, 2, w, x) \in M$
using $transM$ [*OF Aleph_rel2_closed*] $separation_conj$ $separation_ber$
 $lam_replacement_product$
 $separation_in$ $lam_replacement_fst$ $lam_replacement_snd$ $lam_replacement_constant$
 $lam_replacement_hcomp$ [*OF lam_replacement_snd lam_replacement_restrict'*]

separation_bex separation_conj
by simp

lemma *Aleph_rel2_new_reals*:
assumes $w \in \aleph_2^M$ $x \in \aleph_2^M$ $w \neq x$
shows $(\lambda n \in \omega. f_G \langle w, n \rangle) \neq (\lambda n \in \omega. f_G \langle x, n \rangle)$
proof -
have $0 \in 2$ **by auto**
with *assms*
have $dense(inj_dense(\aleph_2^M, 2, w, x))$
unfolding *dense_def* **using** *dense_inj_dense* **by auto**
with *assms*
obtain p **where** $p \in inj_dense(\aleph_2^M, 2, w, x)$ $p \in G$
using *M_generic_denseD* [of *inj_dense*($\aleph_2^M, 2, w, x$)]
by blast
then
obtain n **where** $n \in \omega$ $\langle w, n \rangle, 1 \in p$ $\langle x, n \rangle, 0 \in p$
by blast
moreover from *this* **and** $\langle p \in G \rangle$
have $\langle w, n \rangle, 1 \in f_G$ $\langle x, n \rangle, 0 \in f_G$ **by auto**
moreover from *calculation*
have $f_G \langle w, n \rangle = 1$ $f_G \langle x, n \rangle = 0$
using *f_G_funtype* *apply_equality*
by auto
ultimately
have $(\lambda n \in \omega. f_G \langle w, n \rangle) \neq (\lambda n \in \omega. f_G \langle x, n \rangle)$
by simp
then
show *?thesis* **by fastforce**
qed

definition

$h_G :: i \langle h_G \rangle$ **where**
 $h_G \equiv \lambda \alpha \in \aleph_2^M. \lambda n \in \omega. f_G \langle \alpha, n \rangle$

lemma *h_G_in_MG* [*simp*]:
includes *G_generic1_lemmas*
shows $h_G \in M[G]$
using *ext.curry_closed* [*unfolded curry_def*] *G_in_MG*
unfolding *h_G_def*
by simp

lemma *h_G_inj_Aleph_rel2_reals*: $h_G \in inj^{M[G]}(\aleph_2^M, \omega \rightarrow^{M[G]} 2)$
using *Aleph_rel_sub_closed* *f_G_funtype* *G_in_MG* *Aleph_rel_sub_closed*
ext.curry_rel_exp [*unfolded curry_def*] *ext.curry_closed* [*unfolded curry_def*]
ext.mem_function_space_rel_abs
by (*intro ext.mem_inj_abs* [*THEN iffD2*], *simp_all*)
(auto simp: inj_def h_G_def dest: Aleph_rel2_new_reals)

lemma *Aleph2_extension_le_continuum_rel*:
includes *G_generic1_lemmas*
shows $\aleph_2^{M[G]} \leq 2^{\aleph_0^{M[G],M[G]}}$
proof -
have $\aleph_2^{M[G]} \lesssim^{M[G]} \omega \rightarrow^{M[G]} 2$
using *ext.def_lepoll_rel*[of $\aleph_2^M \omega \rightarrow^{M[G]} 2$]
h_G_inj_Aleph_rel2_reals Aleph_rel_nats_MG_eq_Aleph_rel_nats_M
by auto
moreover from calculation
have $\aleph_2^{M[G]} \lesssim^{M[G]} |\omega \rightarrow^{M[G]} 2|^{M[G]}$
using *ext.lepoll_rel_imp_lepoll_rel_cardinal_rel* **by simp**
ultimately
have $|\aleph_2^{M[G]}|^{M[G]} \leq 2^{\aleph_0^{M[G],M[G]}}$
using *ext.lepoll_rel_imp_cardinal_rel_le*[of $\aleph_2^{M[G]} \omega \rightarrow^{M[G]} 2$,
OF __ ext.function_space_rel_closed]
ext.Aleph_rel_zero
unfolding *cexp_rel_def* **by simp**
then
show $\aleph_2^{M[G]} \leq 2^{\aleph_0^{M[G],M[G]}}$
using *ext.Card_rel_Aleph_rel*[of 2, *THEN ext.Card_rel_cardinal_rel_eq*]
by simp
qed

lemma *Aleph_rel_lt_continuum_rel*: $\aleph_1^{M[G]} < 2^{\aleph_0^{M[G],M[G]}}$
using *Aleph2_extension_le_continuum_rel*
ext.Aleph_rel_increasing[of 1 2] *le_trans* **by auto**

corollary *not_CH*: $\aleph_1^{M[G]} \neq 2^{\aleph_0^{M[G],M[G]}}$
using *Aleph_rel_lt_continuum_rel* **by auto**

end — *add_generic3*

29.3 Models of fragments of $ZFC + \neg CH$

definition

ContHyp :: *o* **where**
ContHyp $\equiv \aleph_1 = 2^{\aleph_0}$

relativize functional *ContHyp ContHyp_rel*

notation *ContHyp_rel* ($\langle CH \rightarrow \rangle$)

relationalize *ContHyp_rel is ContHyp*

context *M_ZF_library*

begin

is_iff_rel for *ContHyp*

using *is_cexp_iff is_Aleph_iff*[of 0] *is_Aleph_iff*[of 1]

unfolding *is_ContHyp_def ContHyp_rel_def*

by (*auto simp del:setclass_iff*) (*rule rexI*[of __ *M*, *OF __ nonempty*], *auto*)

end — $M_ZF_library$

synthesize $is_ContHyp$ **from** $_definition$ **assuming** $nonempty$
arity_theorem **for** $is_ContHyp_fm$

notation $is_ContHyp_fm$ ($\langle \cdot CH \cdot \rangle$)

theorem $ctm_of_not_CH$:

assumes

$M \approx \omega$ $Transset(M)$ $M \models ZC \cup \{ \cdot Replacement(p) \cdot \mid p \in overhead_notCH \}$

$\Phi \subseteq formula$ $M \models \{ \cdot Replacement(ground_repl_fm(\varphi)) \cdot \mid \varphi \in \Phi \}$

shows

$\exists N$.

$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZC \cup \{ \cdot \neg \cdot CH \cdot \} \cup \{ \cdot Replacement(\varphi) \cdot$

$\cdot \varphi \in \Phi \} \wedge$

$(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$

proof -

from $\langle M \models ZC \cup \{ \cdot Replacement(p) \cdot \mid p \in overhead_notCH \} \rangle$

interpret M_ZFC3 M

using $M_satT_overhead_imp_M_ZF3$ **unfolding** $overhead_notCH_def$ **by**

force

from $\langle M \models ZC \cup \{ \cdot Replacement(p) \cdot \mid p \in overhead_notCH \} \rangle$ $\langle Transset(M) \rangle$

interpret $M_ZF_ground_notCH_trans$ M

using $M_satT_imp_M_ZF_ground_notCH_trans$

unfolding ZC_def **by** *auto*

from $\langle M \approx \omega \rangle$

obtain $enum$ **where** $enum \in bij(\omega, M)$

using $eqpoll_sym$ **unfolding** $eqpoll_def$ **by** *blast*

then

interpret M_ctm3_AC M $enum$ **by** $unfold_locales$

interpret $cohen_data$ ω $\aleph_2^M \times \omega$ 2 **by** $unfold_locales$ *auto*

have $Add \in M$ $Add_le(\aleph_2^M) \in M$

using nat_into_M $Aleph_rel_closed$ M_nat $cartprod_closed$ Fn_nat_closed

$Fnle_nat_closed$

by $simp_all$

then

interpret $forcing_data1$ Add $Add_le(\aleph_2^M)$ 0 M $enum$

by $unfold_locales$ $simp_all$

obtain G **where** $M_generic(G)$

using $generic_filter_existence[OF$ $one_in_P]$

by *auto*

moreover **from** *this*

interpret $add_generic3$ M $enum$ G **by** $unfold_locales$

have $\neg (\aleph_1^{M[G]} = 2^{\aleph_0^{M[G], M[G]}})$

using not_CH .

then

have $M[G], [] \models \cdot \neg \cdot CH \cdot$

using $ext.is_ContHyp_iff$

by (*simp add:ContHyp_rel_def*)
 then
 have $M[G] \models ZC \cup \{\cdot \neg \cdot CH \cdot\}$
 using *ext.M_satT_ZC* by *auto*
 moreover
 have *Transset*($M[G]$) using *Transset_MG* .
 moreover
 have $M \subseteq M[G]$ using *M_subset_MG[OF one_in_G]* generic by *simp*
 moreover
 note $\langle M \models \{ \cdot Replacement(ground_repl_fm(\varphi)) \cdot \cdot \varphi \in \Phi \} \rangle \langle \Phi \subseteq formula \rangle$
 ultimately
 show ?thesis
 using *Ord_MG_iff_MG_eqpoll_nat_satT_ground_repl_fm_imp_satT_ZF_replacement_fm*[of
 Φ]
 by (*rule_tac x=M[G] in exI, blast*)
 qed

lemma *ZF_replacement_overhead_sub_ZFC*: $\{ \cdot Replacement(p) \cdot \cdot p \in overhead \}$
 $\subseteq ZFC$
 using *overhead_type unfolding ZFC_def ZF_def ZF_schemes_def* by *auto*

lemma *ZF_replacement_overhead_notCH_sub_ZFC*: $\{ \cdot Replacement(p) \cdot \cdot p \in overhead_notCH \}$
 $\subseteq ZFC$
 using *overhead_notCH_type unfolding ZFC_def ZF_def ZF_schemes_def* by *auto*

lemma *ZF_replacement_overhead_CH_sub_ZFC*: $\{ \cdot Replacement(p) \cdot \cdot p \in overhead_CH \}$
 $\subseteq ZFC$
 using *overhead_CH_type unfolding ZFC_def ZF_def ZF_schemes_def* by *auto*

corollary *ctm_ZFC_imp_ctm_not_CH*:

assumes

$M \approx \omega$ *Transset*(M) $M \models ZFC$

shows

$\exists N.$

$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZFC \cup \{ \cdot \neg \cdot CH \cdot \} \wedge$
 $(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$

proof-

from *assms*

have $\exists N.$

$M \subseteq N \wedge$

$N \approx \omega \wedge$

Transset(N) \wedge

$N \models ZC \wedge N \models \{ \cdot \neg \cdot CH \cdot \} \wedge N \models \{ \cdot Replacement(x) \cdot \cdot x \in formula \} \wedge (\forall \alpha.$

$Ord(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N)$

using *ctm_of_not_CH[of M formula] satT_ZFC_imp_satT_ZC[of M]*

satT_mono[OF ground_repl_fm_sub_ZFC, of M]

satT_mono[OF ZF_replacement_overhead_notCH_sub_ZFC, of M]

```

    satT_mono[OF_ZF_replacement_fms_sub_ZFC, of M]
  by (simp add: satT_Un_iff)
then
obtain N where N ⊨ ZC N ⊨ {·¬·CH·} N ⊨ {·Replacement(x)· . x ∈ formula}
  M ⊆ N N ≈ ω Transset(N) (∀α. Ord(α) ⟶ α ∈ M ⟷ α ∈ N)
  by auto
moreover from this
have N ⊨ ZFC
  using satT_ZC_ZF_replacement_imp_satT_ZFC
  by auto
moreover from this and ⟨N ⊨ {·¬·CH·}⟩
have N ⊨ ZFC ∪ {·¬·CH·}
  by auto
ultimately
show ?thesis by auto
qed

end

```

30 Preservation results for κ -closed forcing notions

```

theory Kappa_Closed_Notions
  imports
    Not_CH
  begin

  definition
    lereI :: i ⇒ i where
    lereI(α) ≡ Memrel(α) ∪ id(α)

  lemma lereI[intro!]: x ≤ y ⟹ y ∈ α ⟹ Ord(α) ⟹ ⟨x,y⟩ ∈ lereI(α)
    using Ord_trans[of x y α] ltD unfolding lereI_def by auto

  lemma lereD[dest]: ⟨x,y⟩ ∈ lereI(α) ⟹ Ord(α) ⟹ x ≤ y
    using ltI[THEN leI] Ord_in_Ord unfolding lereI_def by auto

  definition
    mono_seqspace :: [i,i,i] ⇒ i (⟨_ <→ '(_,_)⟩ [61] 60) where
    α <→ (P,leq) ≡ mono_map(α,Memrel(α),P,leq)

  relativize functional mono_seqspace mono_seqspace_rel
  relationalize mono_seqspace_rel is_mono_seqspace
  synthesize is_mono_seqspace from_definition assuming nonempty

  context M_ZF_library
  begin

  rel_closed for mono_seqspace
    unfolding mono_seqspace_rel_def mono_map_rel_def

```

using *separation_closed separation_ball separation_imp separation_in*
lam_replacementfst lam_replacement_snd lam_replacement_hcomp lam_replacement_constant
lam_replacement_product
lam_replacement_apply2[THEN[5] lam_replacement_hcomp2]
by *simp_all*

end — *M_ZF_library*

abbreviation

mono_seqspace_r ($\langle _ _ \rangle \rightarrow _$ '(_,_)' [61] 60) **where**
 $\alpha _ \rightarrow^M (P, leq) \equiv mono_seqspace_rel(M, \alpha, P, leq)$

abbreviation

mono_seqspace_r_set ($\langle _ _ \rangle \rightarrow _$ '(_,_)' [61] 60) **where**
 $\alpha _ \rightarrow^M (P, leq) \equiv mono_seqspace_rel(\#\#M, \alpha, P, leq)$

lemma *mono_seqspaceI[intro!]*:

includes *mono_map_rules*

assumes $f: A \rightarrow P \wedge x y. x \in A \implies y \in A \implies x < y \implies \langle f'x, f'y \rangle \in leq$ *Ord(A)*

shows $f: A _ \rightarrow (P, leq)$

using *ltI[OF _ Ord_in_Ord[of A], THEN [3] assms(2)] assms(1,3)*

unfolding *mono_seqspace_def* **by** *auto*

lemma (in *M_ZF_library*) *mono_seqspace_rel_char*:

assumes $M(A) M(P) M(leq)$

shows $A _ \rightarrow^M (P, leq) = \{f \in A _ \rightarrow (P, leq). M(f)\}$

using *assms mono_map_rel_char*

unfolding *mono_seqspace_def mono_seqspace_rel_def* **by** *simp*

lemma (in *M_ZF_library*) *mono_seqspace_relI[intro!]*:

assumes $f: A \rightarrow^M P \wedge x y. x \in A \implies y \in A \implies x < y \implies \langle f'x, f'y \rangle \in leq$

Ord(A) M(A) M(P) M(leq)

shows $f: A _ \rightarrow^M (P, leq)$

using *mono_seqspace_rel_char function_space_rel_char assms* **by** *auto*

lemma *mono_seqspace_is_fun[dest]*:

includes *mono_map_rules*

shows $j: A _ \rightarrow (P, leq) \implies j: A \rightarrow P$

unfolding *mono_seqspace_def* **by** *auto*

lemma *mono_map_lt_le_is_mono[dest]*:

includes *mono_map_rules*

assumes $j: A _ \rightarrow (P, leq) a \in A c \in A a \leq c$ *Ord(A) refl(P, leq)*

shows $\langle j'a, j'c \rangle \in leq$

using *assms mono_map_increasing* **unfolding** *mono_seqspace_def refl_def*

by (*cases a=c*) (*auto dest:ltD*)

lemma (in *M_ZF_library*) *mem_mono_seqspace_abs[absolut]*:

assumes $M(f) M(A) M(P) M(leq)$

shows $f:A \xrightarrow{M} (P,leq) \longleftrightarrow f: A \xrightarrow{\ } (P,leq)$
using *assms mono_map_rel_char unfolding mono_seqspace_def mono_seqspace_rel_def*
by (*simp*)

definition

$mono_map_lt_le :: [i,i] \Rightarrow i$ (**infixr** $\langle \xrightarrow{\ } \rangle$ 60) **where**
 $\alpha \xrightarrow{\ } \beta \equiv \alpha \xrightarrow{\ } (\beta,lerel(\beta))$

lemma *mono_map_lt_leI*[*intro!*]:

includes *mono_map_rules*
assumes $f: A \rightarrow B \wedge x y. x \in A \implies y \in A \implies x < y \implies f'x \leq f'y$ *Ord(A) Ord(B)*
shows $f: A \xrightarrow{\ } B$
using *assms*
unfolding *mono_map_lt_le_def* **by** *auto*

— Kunen IV.7.13, with “ κ ” in place of “ λ ”

definition

$kappa_closed :: [i,i,i] \Rightarrow o$ ($\langle \xrightarrow{\ } \text{-closed}'(_,_)' \rangle$) **where**
 $\kappa\text{-closed}(P,leq) \equiv \forall \delta. \delta < \kappa \longrightarrow (\forall f \in \delta \xrightarrow{\ } (P,converse(leq))). \exists q \in P. \forall \alpha \in \delta. \langle q, f'\alpha \rangle \in leq$

relativize functional *kappa_closed kappa_closed_rel*
relationalize *kappa_closed_rel is_kappa_closed*
synthesize *is_kappa_closed from_definition* **assuming** *nonempty*

abbreviation

$kappa_closed_r$ ($\langle \xrightarrow{\ } \text{-closed}'(_,_)' \rangle$) [61] 60) **where**
 $\kappa\text{-closed}^M(P,leq) \equiv kappa_closed_rel(M,\kappa,P,leq)$

abbreviation

$kappa_closed_r_set$ ($\langle \xrightarrow{\ } \text{-closed}'(_,_)' \rangle$) [61] 60) **where**
 $\kappa\text{-closed}^M(P,leq) \equiv kappa_closed_rel(\#\#M,\kappa,P,leq)$

lemma (*in forcing_data3*) *forcing_a_value*:

assumes $p \Vdash \cdot 0:1 \rightarrow 2 \cdot [f_dot, A^v, B^v]$ $a \in A$
 $q \preceq p$ $q \in \mathbb{P}$ $p \in \mathbb{P}$ $f_dot \in M$ $A \in M$ $B \in M$
shows $\exists d \in \mathbb{P}. \exists b \in B. d \preceq q \wedge d \Vdash \cdot 0'1 \text{ is } 2 \cdot [f_dot, a^v, b^v]$

proof -

from *assms*
have $q \Vdash \cdot 0:1 \rightarrow 2 \cdot [f_dot, A^v, B^v]$
using *strengthening_lemma*[*of p · 0:1 → 2 · q [f_dot, A^v, B^v]*]
typed_function_type arity_typed_function_fm
by (*auto simp: union_abs2 union_abs1*)
from $\langle a \in A \rangle \langle A \in M \rangle$
have $a \in M$ **by** (*auto dest:transitivity*)
from $\langle q \in \mathbb{P} \rangle$

Here we’re using countability (via the existence of generic filters) of M as a

shortcut, to avoid a further density argument.

```

obtain  $G$  where  $M\_generic(G)$   $q \in G$ 
  using  $generic\_filter\_existence$  by  $blast$ 
then
interpret  $G\_generic3\_AC$   $---$   $G$  by  $unfold\_locales$ 
include  $G\_generic1\_lemmas$ 
note  $\langle q \in G \rangle$ 
moreover
note  $\langle q \Vdash \cdot 0 : 1 \rightarrow 2 \cdot [f\_dot, A^v, B^v] \rangle \langle M\_generic(G) \rangle$ 
moreover
note  $\langle q \in \mathbb{P} \rangle \langle f\_dot \in M \rangle \langle B \in M \rangle \langle A \in M \rangle$ 
moreover from  $this$ 
have  $map(val(G), [f\_dot, A^v, B^v]) \in list(M[G])$  by  $simp$ 
moreover from  $calculation$ 
have  $val(G, f\_dot) : A \rightarrow^{M[G]} B$ 
  using  $truth\_lemma[of \cdot 0 : 1 \rightarrow 2 \cdot [f\_dot, A^v, B^v], THEN iffD1]$ 
   $typed\_function\_type$   $arity\_typed\_function\_fm$   $val\_check[OF one\_in\_G$ 
 $one\_in\_P]$ 
  by  $(auto simp: union\_abs2 union\_abs1 ext.mem\_function\_space\_rel\_abs)$ 
moreover
note  $\langle a \in M \rangle$ 
moreover from  $calculation$  and  $\langle a \in A \rangle$ 
have  $val(G, f\_dot) \cdot a \in B$  (is  $?b \in B$  )
  by  $(simp add: ext.mem\_function\_space\_rel\_abs)$ 
moreover from  $calculation$ 
have  $?b \in M$  by  $(auto dest:transitivity)$ 
moreover from  $calculation$ 
have  $M[G], map(val(G), [f\_dot, a^v, ?b^v]) \models \cdot 0'1$  is  $2 \cdot$ 
  by  $simp$ 
ultimately
obtain  $r$  where  $r \Vdash \cdot 0'1$  is  $2 \cdot [f\_dot, a^v, ?b^v]$   $r \in G$   $r \in \mathbb{P}$ 
  using  $truth\_lemma[of \cdot 0'1$  is  $2 \cdot [f\_dot, a^v, ?b^v], THEN iffD2]$ 
   $fun\_apply\_type$   $arity\_fun\_apply\_fm$   $val\_check[OF one\_in\_G$ 
 $one\_in\_P]$ 
 $G\_subset\_P$ 
  by  $(auto simp: union\_abs2 union\_abs1 ext.mem\_function\_space\_rel\_abs)$ 
moreover from  $this$  and  $\langle q \in G \rangle$ 
obtain  $d$  where  $d \preceq q$   $d \preceq r$   $d \in \mathbb{P}$  by  $force$ 
moreover
note  $\langle f\_dot \in M \rangle \langle a \in M \rangle \langle ?b \in B \rangle \langle B \in M \rangle$ 
moreover from  $calculation$ 
have  $d \preceq q \wedge d \Vdash \cdot 0'1$  is  $2 \cdot [f\_dot, a^v, ?b^v]$ 
  using  $fun\_apply\_type$   $arity\_fun\_apply\_fm$ 
   $strengthening\_lemma[of r \cdot 0'1$  is  $2 \cdot d [f\_dot, a^v, ?b^v]]$ 
  by  $(auto dest:transitivity simp add: union\_abs2 union\_abs1)$ 
ultimately
show  $?thesis$  by  $auto$ 
qed

```

locale $M_master_CH = M_master + M_library_DC$

```

sublocale M_ZFC2_ground_CH_trans  $\subseteq$  M_master_CH ##M
  using replacement_dcwit_repl_body
  by unfold_locales (simp_all add:sep_instances del:setclass_iff
    add:transrec_replacement_def wfrec_replacement_def dcwit_repl_body_def)

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, 1, f_dot,  $\tau$ , snd(r)v]  $\models$   $\chi$ )
proof -
  let ?f_fm=fst_fm(1,0)
  let ?g_fm=hcomp_fm(check_fm(6),snd_fm,2,0)
  note assms
  moreover
  have ?f_fm  $\in$  formula arity(?f_fm)  $\leq$  7 ?g_fm  $\in$  formula arity(?g_fm)  $\leq$  8
    using ord_simp_union
    unfolding hcomp_fm_def
    by (simp_all add:arity)
  ultimately
  show ?thesis
    using separation_sat_after_function
    using fst_abs snd_abs sats_snd_fm sats_check_fm check_abs
    unfolding hcomp_fm_def
    by simp
qed

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, 1, f_dot, fst(p)v, snd(p)v]  $\models$   $\chi$ )
proof -
  let ? $\rho$ = $\lambda z.$  [r,  $\mathbb{P}$ , leq, 1, f_dot, fst(z)v, snd(z)v]
  let ? $\rho'$ = $\lambda z.$  [fst(z)v,  $\mathbb{P}$ , leq, 1, f_dot, r, snd(z)v]
  let ? $\varphi$ = ( $\cdot$  $\exists$  ( $\cdot$  $\exists$  ( $\cdot$  $\exists$  ( $\cdot$  $\exists$  ( $\cdot$  $\exists$  ( $\cdot$  $\exists$   $\cdot$  0 = 11  $\cdot$   $\wedge$   $\cdot$  1 = 7  $\cdot$   $\wedge$   $\cdot$  2 = 8  $\cdot$   $\wedge$   $\cdot$  3 = 9  $\cdot$   $\wedge$   $\cdot$  4 =
  10  $\cdot$   $\wedge$   $\cdot$  5 = 6  $\cdot$   $\wedge$ 
    ( $\lambda p.$  incr_bv(p)'6)^6 ( $\chi$ ) .....))..))..))
  let ?f_fm=hcomp_fm(check_fm(5),fst_fm,1,0)
  let ?g_fm=hcomp_fm(check_fm(6),snd_fm,2,0)
  note assms
  moreover
  have ?f_fm  $\in$  formula arity(?f_fm)  $\leq$  7 ?g_fm  $\in$  formula arity(?g_fm)  $\leq$  8
    using ord_simp_union
    unfolding hcomp_fm_def
    by (simp_all add:arity)

```

```

moreover from assms
have fn: ?φ ∈ formula by simp
moreover from  $\langle \chi \in \text{formula} \rangle \langle \text{arity}(\chi) \leq 7 \rangle$ 
have  $\text{arity}(\chi) = 0 \vee \text{arity}(\chi) = 1 \vee \text{arity}(\chi) = 2 \vee \text{arity}(\chi) = 3$ 
 $\vee \text{arity}(\chi) = 4 \vee \text{arity}(\chi) = 5 \vee \text{arity}(\chi) = 6 \vee \text{arity}(\chi) = 7$ 
unfolding lt_def by auto
with calculation and  $\langle \chi \in \text{formula} \rangle$ 
have ar: arity(?φ) ≤ 7
using arity_incr_bv_lemma by safe (simp_all add: arity_ord_simp_union)
moreover from calculation
have sep: separation(##M, λz. M, ?ρ'(z) ⊨ ?φ)
using separation_sat_after_function sats_check_fm check_abs
fst_abs snd_abs
unfolding hcomp_fm_def
by simp
moreover from assms
have  $?ρ(z) \in \text{list}(M)$  if  $(##M)(z)$  for z
using that by simp
moreover from calculation and  $\langle r \in M \rangle \langle \chi \in \text{formula} \rangle$ 
have  $(M, ?ρ(z) \models \chi) \longleftrightarrow (M, ?ρ'(z) \models ?φ)$  if  $(##M)(z)$  for z
using that sats_incr_bv_iff[of _ _ M _ [_,_,_,_,_]]
by simp
ultimately
show ?thesis
using separation_cong[THEN iffD1, OF _ sep]
by simp
qed

```

```

lemma separation_leq_and_forces_apply_aux:
assumes f_dot ∈ M B ∈ M
shows  $\forall n \in M. \text{separation}(##M, \lambda x. \text{snd}(x) \preceq \text{fst}(x) \wedge$ 
 $(\exists b \in B. M, [\text{snd}(x), \mathbb{P}, \text{leq}, \mathbf{1}, f\_dot, (\bigcup (n))^v, b^v] \models \text{forces}(\cdot 0'1 \text{ is } 2 \cdot)))$ 

```

proof -

```

have pred_nat_closed: pred(n) ∈ M if  $n \in M$  for n
using nat_case_closed that
unfolding pred_def
by auto

```

```

have separation(##M, λz. M, [snd(fst(z)), ℙ, leq, 1, f_dot, τ, snd(z)v] ⊨ χ)
if  $\chi \in \text{formula}$   $\text{arity}(\chi) \leq 7$   $\tau \in M$  for  $\chi \ \tau$ 

```

proof -

```

let ?f_fm = hcomp_fm(snd_fm, fst_fm, 1, 0)
let ?g_fm = hcomp_fm(check_fm(6), snd_fm, 2, 0)
note assms

```

moreover

```

have  $?f\_fm \in \text{formula}$   $\text{arity}(?f\_fm) \leq 7$   $?g\_fm \in \text{formula}$   $\text{arity}(?g\_fm) \leq 8$ 
using ord_simp_union
unfolding hcomp_fm_def
by (simp_all add: arity)
ultimately

```

```

show ?thesis
  using separation_sat_after_function sats_check_fm check_abs fst_abs
snd_abs that
  unfolding hcomp_fm_def
  by simp
qed
with assms
show ?thesis
  using separation_in lam_replacement_constant lam_replacement_snd lam_replacement_fst
  lam_replacement_product pred_nat_closed
  arity_forces[of ·0'1 is 2.] arity_fun_apply_fm[of 0 1 2] ord_simp_union
  by(clarify,rule_tac separation_conj,simp_all,rule_tac separation_bex,simp_all)
qed

lemma separation_leq_and_forces_apply_aux':
assumes f_dot ∈ M p ∈ M B ∈ M
shows separation
  (##M, λp . snd(snd(p)) ≤ fst(snd(p)) ∧
  (∃ b ∈ B. M, [snd(snd(p)), ℙ, leq, 1, f_dot, (⋃fst(p))v, bv] ⊨ forces(·0'1 is 2.)))
proof -
  have separation(##M, λz. M, [snd(snd(fst(z))), ℙ, leq, 1, f_dot, (⋃fst(fst(z)))v,
  snd(z)v] ⊨ χ)
  if χ ∈ formula arity(χ) ≤ 7 for χ
  proof -
  let ?f_fm = hcomp_fm(snd_fm, hcomp_fm(snd_fm, fst_fm), 1, 0)
  let ?g_fm = λz . (⋃(fst(fst(z))))v
  let ?q_fm = hcomp_fm(check_fm(6), hcomp_fm(big_union_fm, hcomp_fm(fst_fm, fst_fm)), 2, 0)
  let ?h_fm = hcomp_fm(check_fm(7), snd_fm, 3, 0)
  note assms
  moreover
  have f_fm_facts: ?f_fm ∈ formula arity(?f_fm) ≤ 6
  using ord_simp_union
  unfolding hcomp_fm_def
  by (simp_all add:arity)
  moreover from assms
  have ?g_fm ∈ formula arity(?g_fm) ≤ 7 ?h_fm ∈ formula arity(?h_fm) ≤ 8
  using ord_simp_union
  unfolding hcomp_fm_def
  by (simp_all add:arity)
  ultimately
  show ?thesis
  using separation_sat_after_function3[OF _ _ _ f_fm_facts] check_abs
  sats_check_fm that fst_abs snd_abs sats_fst_fm sats_snd_fm
  unfolding hcomp_fm_def
  by simp
qed
with assms
show ?thesis
  using

```

```

    separation_conj separation_bex
    lam_replacement_constant lam_replacement_hcomp
    lam_replacement_fst lam_replacement_snd
    arity_forces[of ·0'1 is 2] arity_fun_apply_fm[of 0 1 2] ord_simp_union
    separation_in[OF lam_replacement_product]
  by simp
qed

```

lemma *separation_closed_leq_and_forces_eq_check_aux* :

```

  assumes  $A \in M$   $r \in G$   $\tau \in M$ 
  shows  $(\#\#M)(\{q \in \mathbb{P}. \exists h \in A. q \preceq r \wedge q \Vdash \cdot 0 = 1 \cdot [\tau, h^v]\})$ 
proof -

```

```

  have  $\text{separation}(\#\#M, \lambda z. M, [\text{fst}(z), \mathbb{P}, \text{leq}, \mathbf{1}, \tau, \text{snd}(z)^v] \models \chi)$  if
 $\chi \in \text{formula}$   $\text{arity}(\chi) \leq 6$  for  $\chi$ 

```

proof -

```

  let  $?f\_fm = \text{fst\_fm}(1, 0)$ 

```

```

  let  $?g\_fm = \text{hcomp\_fm}(\text{check\_fm}(6), \text{snd\_fm}, 2, 0)$ 

```

```

  note assms

```

moreover

```

  have  $?f\_fm \in \text{formula}$   $\text{arity}(?f\_fm) \leq 6$   $?g\_fm \in \text{formula}$   $\text{arity}(?g\_fm) \leq 7$ 

```

```

  using ord_simp_union

```

```

  unfolding hcomp_fm_def

```

```

  by (simp_all add:arity)

```

ultimately

```

  show  $?thesis$ 

```

```

  using separation_sat_after_function_1 sats_fst_fm that

```

```

 $\text{fst\_abs}$   $\text{snd\_abs}$  sats_snd_fm sats_check_fm check_abs

```

```

  unfolding hcomp_fm_def

```

```

  by simp

```

qed

```

with assms

```

```

show  $?thesis$ 

```

```

  using separation_conj separation_in  $G$  subset_M [THEN subsetD]

```

```

 $\text{lam\_replacement\_constant}$   $\text{lam\_replacement\_fst}$   $\text{lam\_replacement\_product}$ 

```

```

 $\text{arity\_forces}$ [of  $\cdot 0 = 1$ , simplified] ord_simp_union

```

```

by(rule_tac separation_closed[OF separation_bex], simp_all)

```

qed

lemma *separation_closed_forces_apply_aux*:

```

  assumes  $B \in M$   $f\_dot \in M$   $r \in M$ 

```

```

  shows  $(\#\#M)(\{(n, b) \in \omega \times B. r \Vdash \cdot 0'1 \text{ is } 2 \cdot [f\_dot, n^v, b^v]\})$ 

```

```

using nat_in_M assms transitivity[OF  $\langle B \in M \rangle$ ] nat_into_M separation_check_fst_snd_aux

```

```

 $\text{arity\_forces}$ [of  $\cdot 0'1 \text{ is } 2 \cdot$ ]  $\text{arity\_fun\_apply\_fm}$ [of 0 1 2] ord_simp_union

```

```

unfolding split_def

```

```

by simp_all

```

— Kunen IV.6.9 (3) \Rightarrow (2), with general domain.

lemma *kunen_IV_6_9_function_space_rel_eq*:

```

  assumes  $\bigwedge p \tau. p \Vdash \cdot 0:1 \rightarrow 2 \cdot [\tau, A^v, B^v] \implies p \in \mathbb{P} \implies \tau \in M \implies$ 

```

$\exists q \in \mathbb{P}. \exists h \in A \rightarrow^M B. q \preceq p \wedge q \Vdash \cdot 0 = 1. [\tau, h^v] A \in M B \in M$
shows
 $A \rightarrow^M B = A \rightarrow^{M[G]} B$
proof (*intro equalityI; clarsimp simp add:*
assms function_space_rel_char ext.function_space_rel_char)
fix f
assume $f \in A \rightarrow B f \in M[G]$
moreover from *this*
obtain τ **where** $val(G, \tau) = f \tau \in M$
using *GenExtD* **by force**
moreover from *calculation and* $\langle A \in M \rangle \langle B \in M \rangle$
obtain r **where** $r \Vdash \cdot 0:1 \rightarrow 2. [\tau, A^v, B^v] r \in G$
using *truth_lemma*[*of* $\cdot 0:1 \rightarrow 2. [\tau, A^v, B^v]$]
typed_function_type arity_typed_function_fm val_check[*OF one_in_G*
one_in_P]
by (*auto simp: union_abs2 union_abs1*)
moreover from $\langle A \in M \rangle \langle B \in M \rangle \langle r \in G \rangle \langle \tau \in M \rangle$
have $\{q \in \mathbb{P}. \exists h \in A \rightarrow^M B. q \preceq r \wedge q \Vdash \cdot 0 = 1. [\tau, h^v]\} \in M$ (**is** $?D \in M$)
using *separation_closed_leq_and_forces_eq_check_aux* **by auto**
moreover from *calculation and assms(2-)*
have *dense_below*($?D, r$)
using *strengthening_lemma*[*of* $r \cdot 0:1 \rightarrow 2. _ [\tau, A^v, B^v]$, *THEN assms(1)*][*of* $_ \tau$]
leq_transD generic_dests(1)][*of* r]
by (*auto simp: union_abs2 union_abs1 typed_function_type arity_typed_function_fm*)
blast
moreover from *calculation*
obtain $q h$ **where** $h \in A \rightarrow^M B q \Vdash \cdot 0 = 1. [\tau, h^v] q \preceq r q \in \mathbb{P} q \in G$
using *generic_inter_dense_below*][*of* $?D r$] **by blast**
note $\langle q \Vdash \cdot 0 = 1. [\tau, h^v] \rangle \langle \tau \in M \rangle \langle h \in A \rightarrow^M B \rangle \langle A \in M \rangle \langle B \in M \rangle \langle q \in G \rangle$
moreover from *this*
have $map(val(G), [\tau, h^v]) \in list(M[G]) h \in M$
by (*auto dest:transitivity*)
ultimately
have $h = f$
using *truth_lemma*][*of* $\cdot 0=1. [\tau, h^v]$] *val_check*][*OF one_in_G one_in_P*]
by (*auto simp: ord_simp_union*)
with $\langle h \in M \rangle$
show $f \in M$ **by simp**
qed

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 -

The next long block proves that the assumptions of Lemma *kunen_IV_6_9_function_space_rel_eq* are satisfied.

```

{
  fix p f_dot
  assume p ⊢ ·0:1→2. [f_dot, ωv, Bv] p∈ℙ f_dot∈M
  let ?subp={q∈ℙ. q ≼ p}
  from ⟨p∈ℙ⟩
  have ?subp ∈ M
    using first_section_closed[of ℙ p converse(leg)]
    by (auto dest:transitivity)
  define S where S ≡ λn∈nat.
  {⟨q,r⟩ ∈ ?subp×?subp. r ≼ q ∧ (∃ b∈B. r ⊢ ·0'1 is 2. [f_dot, (⋃(n))v, bv])}
  (is S ≡ λn∈nat. ?Y(n))
  define S' where S' ≡ λn∈nat.
  {⟨q,r⟩ ∈ ?subp×?subp. r ≼ q ∧ (∃ b∈B. r ⊢ ·0'1 is 2. [f_dot, (pred(n))v, bv])}
  — Towards proving S ∈ M.
  moreover
  have S = S'
    unfolding S_def S'_def using pred_nat_eq lam_cong by auto
  moreover from ⟨B∈M⟩ ⟨?subp∈M⟩ ⟨f_dot∈M⟩
  have {r ∈ ?subp. ∃ b∈B. r ⊢ ·0'1 is 2. [f_dot, (⋃(n))v, bv]} ∈ M (is ?X(n) ∈
M)
    if n∈ω for n
    using that separation_check_snd_aux nat_into_M ord_simp_union
      arity_forces[of ·0'1 is 2.] arity_fun_apply_fm
    by(rule_tac separation_closed[OF separation_bex,simplified], simp_all)
  moreover
  have ?Y(n) = (?subp × ?X(n)) ∩ converse(leg) for n
    by (intro equalityI) auto
  moreover
  note ⟨?subp ∈ M⟩ ⟨B∈M⟩ ⟨p∈ℙ⟩ ⟨f_dot∈M⟩
  moreover from calculation
  have n ∈ ω ⇒ ?Y(n) ∈ M for n
    using nat_into_M by simp
  moreover from calculation
  have S ∈ M
    using separation_leg_and_forces_apply_aux separation_leg_and_forces_apply_aux'
      transitivity[OF ⟨p∈ℙ⟩]
    unfolding S_def split_def
  by(rule_tac lam_replacement_Collect'[THEN lam_replacement_imp_lam_closed,simplified],
simp_all)
  ultimately
  have S' ∈ M
    by simp
  from ⟨p∈ℙ⟩ ⟨f_dot∈M⟩ ⟨p ⊢ ·0:1→2. [f_dot, ωv, Bv]} ⟨B∈M⟩
  have exr:∃ r∈ℙ. r ≼ q ∧ (∃ b∈B. r ⊢ ·0'1 is 2. [f_dot, pred(n))v, bv])
    if q ≼ p q∈ℙ n∈ω for q n
    using that forcing_a_value by (auto dest:transitivity)
  have ∀ q∈?subp. ∀ n∈ω. ∃ r∈?subp. ⟨q,r⟩ ∈ S' n
  proof -
    {

```

```

fix  $q\ n$ 
assume  $q \in ?subp\ n \in \omega$ 
moreover from this
have  $q \preceq p\ q \in \mathbb{P}\ pred(n) = \bigcup n$ 
  using pred_nat_eq by simp_all
moreover from calculation and exr
obtain  $r$  where  $MM:r \preceq q\ \exists b \in B. r \Vdash \cdot 0'1\ is\ 2. [f\_dot, pred(n)^v, b^v]\ r \in \mathbb{P}$ 
  by blast
moreover from calculation  $\langle q \preceq p \rangle\ \langle p \in \mathbb{P} \rangle$ 
have  $r \preceq p$ 
  using leq_transD[of r q p] by auto
ultimately
have  $\exists r \in ?subp. r \preceq q \wedge (\exists b \in B. r \Vdash \cdot 0'1\ is\ 2. [f\_dot, (pred(n))^v, b^v])$ 
  by auto
}
then
show ?thesis
  unfolding S'_def by simp
qed
with  $\langle p \in \mathbb{P} \rangle\ \langle ?subp \in M \rangle\ \langle S' \in M \rangle$ 
obtain  $g$  where  $g \in \omega \rightarrow^M ?subp\ g'0 = p\ \forall n \in nat. \langle g'n, g'succ(n) \rangle \in S'succ(n)$ 
  using sequence_DC[simplified] refl_leq[of p] by blast
moreover from this and  $\langle ?subp \in M \rangle$ 
have  $g : \omega \rightarrow \mathbb{P}\ g \in M$ 
  using fun_weaken_type[of g \omega ?subp \mathbb{P}] function_space_rel_char by auto
ultimately
have  $g : \omega \xrightarrow{M} (\mathbb{P}, converse(leq))$ 
  using decr_succ_decr[of g] leq_preord
  unfolding S'_def by (auto simp:absolut intro:leI)
moreover from  $\langle succ(\omega)\text{-closed}^M(\mathbb{P}, leq) \rangle$  and this
have  $\exists q \in M. q \in \mathbb{P} \wedge (\forall \alpha \in M. \alpha \in \omega \longrightarrow q \preceq g'\alpha)$ 
  using transitivity[simplified, of g] mono_seqspace_rel_closed[of \omega _ converse(leq)]
  unfolding kappa_closed_rel_def
  by auto
ultimately
obtain  $r$  where  $r \in \mathbb{P}\ r \in M\ \forall n \in \omega. r \preceq g'n$ 
  using nat_into_M by auto
with  $\langle g'0 = p \rangle$ 
have  $r \preceq p$ 
  by blast
let  $?h = \{ \langle n, b \rangle \in \omega \times B. r \Vdash \cdot 0'1\ is\ 2. [f\_dot, n^v, b^v] \}$ 
have function(?h)
proof (rule_tac functionI, rule_tac ccontr, auto simp del: app_Cons)
  fix  $n\ b\ b'$ 
  assume  $n \in \omega\ b \neq b'\ b \in B\ b' \in B$ 
  moreover
  assume  $r \Vdash \cdot 0'1\ is\ 2. [f\_dot, n^v, b^v]\ r \Vdash \cdot 0'1\ is\ 2. [f\_dot, n^v, b'^v]$ 
  moreover

```

```

note  $\langle r \in \mathbb{P} \rangle$ 
moreover from this
have  $\neg r \perp r$ 
  by (auto intro! : refl_leq)
moreover
note  $\langle f\_dot \in M \rangle \langle B \in M \rangle$ 
ultimately
show False
  using forces_neq_apply_imp_incompatible[of  $r$   $f\_dot$   $n^v$   $b$   $r$   $b'$ ]
  transitivity[of  $\_ B$ ] by (auto dest : transitivity)
qed
moreover
have  $range(?h) \subseteq B$ 
  by auto
moreover
have  $domain(?h) = \omega$ 
proof -
  {
    fix  $n$ 
    assume  $n \in \omega$ 
    moreover from this
    have  $1 : (\bigcup (n)) = pred(n)$ 
      using pred_nat_eq by simp
    moreover from calculation and  $\langle \forall n \in nat. \langle g'n, g'succ(n) \rangle \in S'succ(n) \rangle$ 
    obtain  $b$  where  $g'(succ(n)) \Vdash \cdot 0'1$  is 2.  $[f\_dot, n^v, b^v]$   $b \in B$ 
      unfolding S'_def by auto
    moreover from  $\langle B \in M \rangle$  and calculation
    have  $b \in M$   $n \in M$ 
      by (auto dest : transitivity)
    moreover
    note  $\langle g : \omega \rightarrow \mathbb{P} \rangle \langle \forall n \in \omega. r \preceq g'n \rangle \langle r \in \mathbb{P} \rangle \langle f\_dot \in M \rangle$ 
    moreover from calculation
    have  $r \Vdash \cdot 0'1$  is 2.  $[f\_dot, n^v, b^v]$ 
      using fun_apply_type arity_fun_apply_fm
      strengthening_lemma[of  $g'succ(n) \cdot 0'1$  is 2.  $r$   $[f\_dot, n^v, b^v]$ ]
      by (simp add : union_abs2 union_abs1)
    ultimately
    have  $\exists b \in B. r \Vdash \cdot 0'1$  is 2.  $[f\_dot, n^v, b^v]$ 
      by auto
  }
then
show ?thesis
  by force
qed
moreover
have relation(?h)
  unfolding relation_def by simp
moreover from  $\langle f\_dot \in M \rangle \langle r \in M \rangle \langle B \in M \rangle$ 
have  $?h \in M$ 

```

```

    using separation_closed_forces_apply_aux by simp
  moreover
  note ⟨B ∈ M⟩
  ultimately
  have ?h: ω →M B
    using function_imp_Pi[THEN fun_weaken_type[of ?h _ range(?h) B]]
      function_space_rel_char by simp
  moreover
  note ⟨p ⊨ ·0:1→2· [f_dot, ωv, Bv]⟩ ⟨r ≼ p⟩ ⟨r ∈ ℙ⟩ ⟨p ∈ ℙ⟩ ⟨f_dot ∈ M⟩ ⟨B ∈ M⟩
  moreover from this
  have r ⊨ ·0:1→2· [f_dot, ωv, Bv]
    using strengthening_lemma[of p ·0:1→2· r [f_dot, ωv, Bv]]
      typed_function_type arity_typed_function_fm
    by (auto simp: union_abs2 union_abs1)
  moreover
  note ⟨?h ∈ M⟩
  moreover from calculation
  have r ⊨ ·0 = 1· [f_dot, ?hv]
  proof (intro definition_of_forcing[THEN iffD2] allI impI,
    simp_all add:union_abs2 union_abs1 del:app_Cons)
    fix H
    let ?f = val(H, f_dot)
    assume M_generic(H) ∧ r ∈ H
    moreover from this
    interpret g: G_generic1 _ _ _ _ H
      by unfold_locales simp
    note ⟨r ∈ ℙ⟩ ⟨f_dot ∈ M⟩ ⟨B ∈ M⟩
    moreover from calculation
    have map(val(H), [f_dot, ωv, Bv]) ∈ list(M[H]) r ∈ H
      by simp_all
    moreover from calculation and ⟨r ∈ H⟩ and ⟨r ⊨ ·0:1→2· [f_dot, ωv, Bv]⟩
    have ?f : ω → B
      using g.truth_lemma[of ·0:1→2· [f_dot, ωv, Bv], THEN iffD1] g.one_in_G
one_in_P
      typed_function_type arity_typed_function_fm val_check
    by (auto simp: union_abs2 union_abs1)
  moreover
  have ?h'n = ?f'n if n ∈ ω for n
  proof -
    note ⟨n ∈ ω⟩ ⟨domain(?h) = ω⟩
    moreover from this
    have n ∈ domain(?h)
      by simp
    moreover from this
    obtain b where r ⊨ ·0'1 is 2· [f_dot, nv, bv] b ∈ B
      by force
    moreover
    note ⟨function(?h)⟩
    moreover from calculation

```

```

have  $b = ?h'n$ 
  using function_apply_equality by simp
moreover
note  $\langle B \in M \rangle$ 
moreover from calculation
have  $?h'n \in M$ 
  by (auto dest:transitivity)
moreover
note  $\langle f\_dot \in M \rangle \langle r \in \mathbb{P} \rangle \langle M\_generic(H) \wedge r \in H \rangle \langle map(val(H), [f\_dot,$ 
 $\omega^v, B^v]) \in list(M[H]) \rangle$ 
moreover from calculation
have  $[?f, n, ?h'n] \in list(M[H])$ 
using M_subset_MG nat_into_M[of n] g.one_in_G by (auto dest:transitivity)
ultimately
show ?thesis
  using definition_of_forcing[of r · 0'1 is 2 · [f_dot, n^v, b^v],
  THEN iffD1, rule_format, of H]— without this line is slower
  val_check g.one_in_G one_in_P nat_into_M
  by (auto dest:transitivity simp add:fun_apply_type
  arity_fun_apply_fm union_abs2 union_abs1)
qed
with calculation and  $\langle B \in M \rangle \langle ?h: \omega \rightarrow^M B \rangle$ 
have  $?h = ?f$ 
  using function_space_rel_char
  by (rule_tac fun_extension[of ?h  $\lambda_. B$  ?f]) auto
ultimately
show  $?f = val(H, ?h^v)$ 
  using val_check g.one_in_G one_in_P generic by simp
qed
ultimately
have  $\exists r \in \mathbb{P}. \exists h \in \omega \rightarrow^M B. r \preceq p \wedge r \Vdash \cdot 0 = 1 \cdot [f\_dot, h^v]$ 
  by blast
}
moreover
note  $\langle B \in M \rangle$  assms
moreover from calculation
have  $f : \omega \rightarrow^M B$ 
  using kunen_IV_6_9_function_space_rel_eq function_space_rel_char
  ext.mem_function_space_rel_abs by auto
ultimately
show ?thesis
  by (auto dest:transitivity)
qed

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( $\omega$ )-closedM( $\mathbb{P}, leq$ )
```

```

shows  $\omega \rightarrow^M 2 = \omega \rightarrow^{M[G]} 2$ 
proof -
  from assms
  have  $\omega \rightarrow^{M[G]} 2 \subseteq \omega \rightarrow^M 2$ 
  using succ_omega_closed_imp_no_new_nat_sequences function_space_rel_char
    ext.function_space_rel_char Aleph_rel_succ Aleph_rel_zero
  by auto
  then
  show ?thesis
    using function_space_rel_transfer by (intro equalityI) auto
qed

lemma succ_omega_closed_imp_Aleph_1_preserved:
  assumes succ( $\omega$ )-closedM( $\mathbb{P}$ ,leq)
  shows  $\aleph_1^M = \aleph_1^{M[G]}$ 
proof -
  have  $\aleph_1^{M[G]} \leq \aleph_1^M$ 
  proof (rule ccontr)
    assume  $\neg \aleph_1^{M[G]} \leq \aleph_1^M$ 
    then
    have  $\aleph_1^M < \aleph_1^{M[G]}$ 
      — Ridiculously complicated proof
    using Card_rel_is_Ord ext.Card_rel_is_Ord
      not_le_iff_lt[THEN iffD1] by auto
    then
    have  $|\aleph_1^M|^{M[G]} \leq \omega$ 
      using ext.Card_rel_lt_succ_rel_iff ext.Aleph_rel_zero
        ext.Aleph_rel_succ ext.Card_rel_nat
      by (auto intro!:ext.lt_succ_rel_iff[THEN iffD1]
        intro:Card_rel_Aleph_rel[THEN Card_rel_is_Ord, of 1])
    then
    obtain f where  $f \in \text{inj}(\aleph_1^M, \omega)$   $f \in M[G]$ 
      using ext.countable_rel_iff_cardinal_rel_le_nat[of  $\aleph_1^M$ , THEN iffD2]
      unfolding countable_rel_def lepoll_rel_def
      by auto
    then
    obtain g where  $g \in \text{surj}^{M[G]}(\omega, \aleph_1^M)$ 
      using ext.inj_rel_imp_surj_rel[of  $f \_ \omega$ , OF _ zero_lt_Aleph_rel1[THEN
ltD]]
      by auto
    moreover from this
    have  $g : \omega \rightarrow \aleph_1^M$   $g \in M[G]$ 
      using ext.surj_rel_char surj_is_fun by simp_all
    moreover
    note  $\langle \text{succ}(\omega)\text{-closed}^M(\mathbb{P}, \text{leq}) \rangle$ 
    ultimately
    have  $g \in \text{surj}^M(\omega, \aleph_1^M)$   $g \in M$ 
      using succ_omega_closed_imp_no_new_nat_sequences
        mem_surj_abs ext.mem_surj_abs by simp_all

```

```

then
show False
  using surj_rel_implies_cardinal_rel_le[of g ω ℕ1M]
    Card_rel_nat[THEN Card_rel_cardinal_rel_eq] Card_rel_is_Ord
    not_le_iff_lt[THEN iffD2, OF _ _ nat_lt_Aleph_rel1]
  by simp
qed
then
show ?thesis
  using Aleph_rel_le_Aleph_rel
  by (rule_tac le_anti_sym) simp
qed

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 Fn_rel_closed[simp del, rule del, simplified setclass_iff, simp, intro]
declare Fnle_rel_closed[simp del, rule del, simplified setclass_iff, simp, intro]

abbreviation
  Coll :: i where
    Coll ≡ FnM(ℕ1M, ℕ1M, ω →M 2)

abbreviation
  Colleq :: i where
    Colleq ≡ FnleM(ℕ1M, ℕ1M, ω →M 2)

lemma Coll_in_M[intro,simp]: Coll ∈ M by simp

lemma Colleq_refl : refl(Coll,Colleq)
  unfolding Fnle_rel_def Fnlerel_def refl_def
  using RrelI by simp

```

31.1 Collapse forcing is sufficiently closed

lemma *succ_omega_closed_Coll*: $\text{succ}(\omega)$ -closed^M(*Coll*,*Colleg*)

proof -

— Case for finite sequences

have $n \in \omega \implies \forall f \in n \langle \rightarrow^M (\text{Coll}, \text{converse}(\text{Colleg})) \rangle$.

$\exists q \in M. q \in \text{Coll} \wedge (\forall \alpha \in M. \alpha \in n \longrightarrow \langle q, f \ ' \ \alpha \rangle \in \text{Colleg})$ **for** n

proof (*induct rule:nat_induct*)

case 0

then

show ?*case*

using *zero_lt_Aleph_rel1 zero_in_Fn_rel*

by (*auto simp del:setclass_iff*) (*rule bexI[OF _ zero_in_M]*, *auto*)

next

case (*succ x*)

then

have $\forall f \in \text{succ}(x) \langle \rightarrow^M (\text{Coll}, \text{converse}(\text{Colleg})) \rangle. \forall \alpha \in \text{succ}(x). \langle f \ ' \ x, f \ ' \ \alpha \rangle \in \text{Colleg}$

proof(*intro ballI*)

fix $f \ \alpha$

assume $f \in \text{succ}(x) \langle \rightarrow^M (\text{Coll}, \text{converse}(\text{Colleg})) \rangle \ \alpha \in \text{succ}(x)$

moreover from $\langle x \in \omega \rangle$ *this*

have $f \in \text{succ}(x) \langle \rightarrow (\text{Coll}, \text{converse}(\text{Colleg})) \rangle$

using *mono_seqspace_rel_char nat_into_M*

by *simp*

moreover from *calculation succ*

consider $\alpha \in x \mid \alpha = x$

by *auto*

then

show $\langle f \ ' \ x, f \ ' \ \alpha \rangle \in \text{Colleg}$

proof(*cases*)

case 1

then

have $\langle \alpha, x \rangle \in \text{Memrel}(\text{succ}(x)) \ x \in \text{succ}(x) \ \alpha \in \text{succ}(x)$

by *auto*

with $\langle f \in \text{succ}(x) \langle \rightarrow (\text{Coll}, \text{converse}(\text{Colleg})) \rangle \rangle$

show ?*thesis*

using *mono_mapD(2)[OF _ $\langle \alpha \in \text{succ}(x) \rangle$ _ $\langle \langle \alpha, x \rangle \in \text{Memrel}(\text{succ}(x)) \rangle$]*

unfolding *mono_seqspace_def*

by *auto*

next

case 2

with $\langle f \in \text{succ}(x) \langle \rightarrow (\text{Coll}, \text{converse}(\text{Colleg})) \rangle \rangle$

show ?*thesis*

using *Colleg_refl mono_seqspace_is_fun[THEN apply_type]*

unfolding *refl_def*

by *simp*

qed

qed

moreover

```

note  $\langle x \in \omega \rangle$ 
moreover from this
have  $f'x \in Coll$  if  $f: succ(x) \xrightarrow{M} (Coll, converse(Colleg))$  for  $f$ 
using that mono_seqspace_rel_char[simplified, of succ(x) Coll converse(Colleg)]
  nat_into_M[simplified] mono_seqspace_is_fun[of converse(Colleg)]
  by (intro apply_type[of _ succ(x)] (auto simp del:setclass_iff))
ultimately
show ?case
  using transM[of _ Coll]
  by (auto dest:transM simp del:setclass_iff, rule_tac x=f'x in bexI)
  (auto simp del:setclass_iff, simp)
qed
moreover
  — Interesting case: Countably infinite sequences.
have  $\forall f \in M. f \in \omega \xrightarrow{M} (Coll, converse(Colleg)) \longrightarrow$ 
  ( $\exists q \in M. q \in Coll \wedge (\forall \alpha \in M. \alpha \in \omega \longrightarrow \langle q, f' \alpha \rangle \in Colleg)$ )
proof(intro ballI impI)
  fix  $f$ 
  let  $?rnf = f' \omega$ 
  assume  $f \in M$   $f \in \omega \xrightarrow{M} (Coll, converse(Colleg))$ 
  moreover from this
  have  $f \in \omega \xrightarrow{M} (Coll, converse(Colleg))$   $f \in \omega \rightarrow Coll$ 
  using mono_seqspace_rel_char mono_mapD(2) nat_in_M
  by auto
  moreover from this
  have countableM(f'x) if  $x \in \omega$  for  $x$ 
  using that Fn_rel_is_function countable_iff_lespoll_rel_Aleph_rel_one
  by auto
  moreover from calculation
  have  $?rnf \in M$   $f \subseteq \omega \times Coll$ 
  using nat_in_M image_closed Pi_iff
  by simp_all
  moreover from calculation
  have  $1: \exists d \in ?rnf. d \supseteq h \wedge d \supseteq g$  if  $h \in ?rnf$   $g \in ?rnf$  for  $h$   $g$ 
  proof -
    from calculation
    have  $?rnf = \{f'x \mid x \in \omega\}$ 
    using image_function[of f ω] Pi_iff domain_of_fun
    by auto
    from  $\langle ?rnf = \_ \rangle$  that
    obtain  $m$   $n$  where  $eq: h = f'm$   $g = f'n$   $n \in \omega$   $m \in \omega$ 
    by auto
    then
    have  $m \cup n \in \omega$   $m \leq m \cup n$   $n \leq m \cup n$ 
    using Un_upper1_le Un_upper2_le nat_into_Ord by simp_all
    with calculation eq  $\langle ?rnf = \_ \rangle$ 
    have  $f'(m \cup n) \in ?rnf$   $f'(m \cup n) \supseteq h$   $f'(m \cup n) \supseteq g$ 
    using Fnl_relD mono_map_lt_le_is_mono converse_refl[OF Colleg_refl]
    by auto

```

```

then
  show ?thesis
    by auto
qed
moreover from calculation
have ?rnf  $\subseteq (\aleph_1^M \multimap \#\#^M (nat \rightarrow^M \mathbb{2}))$ 
using subset_trans[OF image_subset[OF  $\langle f \subseteq \_ \rangle$ , of  $\omega$ ] Fn_rel_subset_PFun_rel]
  by simp
moreover
have  $\bigcup ?rnf \in (\aleph_1^M \multimap \#\#^M (nat \rightarrow^M \mathbb{2}))$ 
  using pfun_Un_filter_closed'[OF  $\langle ?rnf \subseteq \_ \rangle$  1]  $\langle ?rnf \in M \rangle$ 
  by simp
moreover from calculation
have  $\bigcup ?rnf \prec^M \aleph_1^M$ 
  using countable_fun_imp_countable_image[of f]
    mem_function_space_rel_abs[simplified, OF nat_in_M Coll_in_M  $\langle f \in M \rangle$ ]
    countableI[OF lepoll_rel_refl]
    countable_iff_lespoll_rel_Aleph_rel_one[of  $\bigcup ?rnf$ ]
  by auto
moreover from calculation
have  $\bigcup ?rnf \in Coll$ 
  unfolding Fn_rel_def
  by simp
moreover from calculation
have  $\bigcup ?rnf \supseteq f \text{ ' } \alpha$  if  $\alpha \in \omega$  for  $\alpha$ 
  using that_image_function[OF fun_is_function] domain_of_fun
  by auto
ultimately
show  $\exists q \in M. q \in Coll \wedge (\forall \alpha \in M. \alpha \in \omega \longrightarrow \langle q, f \text{ ' } \alpha \rangle \in Colleg)$ 
  using Fn_rel_is_function Fnle_rell
  by auto
qed
ultimately
show ?thesis
  unfolding kappa_closed_rel_def by (auto elim!: leE dest:ltD)
qed

end — M_ctm2_AC

locale collapse_CH = G_generic3_AC_CH FnM( $\aleph_1^{\#\#^M}$ ,  $\aleph_1^M$ ,  $\omega \rightarrow^M \mathbb{2}$ ) FnleM( $\aleph_1^{\#\#^M}$ ,
 $\aleph_1^M$ ,  $\omega \rightarrow^M \mathbb{2}$ ) 0

sublocale collapse_CH  $\subseteq$  forcing_notion Coll Colleg 0
  using zero_lt_Aleph_rel1 by unfold_locales

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 $f_G \in M[G]$
using G *in* MG **by** *simp*

abbreviation

$dom_dense :: i \Rightarrow i$ **where**
 $dom_dense(x) \equiv \{ p \in Coll . x \in domain(p) \}$

lemma dom_dense_closed [*intro, simp*]: $x \in M \implies dom_dense(x) \in M$
using $separation_in_domain$ [*of* x]
by *simp*

lemma $domain_f_G$: **assumes** $x \in \aleph_1^M$
shows $x \in domain(f_G)$

proof -

have $(\lambda n \in \omega. 0) \in \omega \rightarrow^M 2$
using $function_space_rel_nonempty$ [*of* 0 2 ω]
by *auto*

with *assms*

have $dense(dom_dense(x)) x \in M$
using $dense_dom_dense$ $InfCard_rel_Aleph_rel$ [*of* 1] $transitivity$ [*OF* $Aleph_rel_closed$ [*of* 1 , *THEN* $setclass_iff$ [*THEN* $iffD1$]]]
unfolding $dense_def$

by *auto*

with *assms*

obtain p **where** $p \in dom_dense(x)$ $p \in G$
using $M_generic_denseD$ [*of* $dom_dense(x)$]
by *auto*

then

show $x \in domain(f_G)$ **by** *blast*

qed

lemma $Un_filter_is_function$:

assumes $filter(G)$

shows $function(\bigcup G)$

proof -

have $Coll \subseteq \aleph_1^M \rightarrow \aleph_1^M (\omega \rightarrow^M 2)$

using $Fn_rel_subset_PFun_rel$

by *simp*

moreover

have $\exists d \in Coll. d \supseteq f \wedge d \supseteq g$ **if** $f \in G$ $g \in G$ **for** f g

using $filter_imp_compat$ [*OF* *assms* $\langle f \in G \rangle$ $\langle g \in G \rangle$] $filterD$ [*OF* *assms*]

unfolding $compat_def$ $compat_in_def$

by *auto*
 ultimately
 have $\exists d \in \aleph_1^M \rightarrow \#\#^M (\omega \rightarrow^M 2)$. $d \supseteq f \wedge d \supseteq g$ if $f \in G$ $g \in G$ for $f g$
 using *rex_mono*[of *Coll*] that by *simp*
 moreover
 have $G \subseteq \text{Coll}$
 using *assms*
 unfolding *filter_def*
 by *simp*
 moreover from *this*
 have $G \subseteq \aleph_1^M \rightarrow \#\#^M (\omega \rightarrow^M 2)$
 using *assms* **unfolding** *Fn_rel_def*
 by *auto*
 ultimately
 show *?thesis*
 using *pfun_Un_filter_closed*[of *G*]
 by *simp*
 qed

lemma *f_G_funtype*:
 shows $f_G : \aleph_1^M \rightarrow \omega \rightarrow^{M[G]} 2$
proof -
 have $x \in B \implies B \in G \implies x \in \aleph_1^M \times (\omega \rightarrow^{M[G]} 2)$ for $B \in G$
proof -
 assume $x \in B$ $B \in G$
 moreover from *this*
 have $x \in M[G]$
 by (*auto* *dest!*: *ext.transM* *simp* *add:G_in_MG*)
 moreover from *calculation*
 have $x \in \aleph_1^M \times (\omega \rightarrow 2)$
 using *Fn_rel_subset_Pow*[of \aleph_1^M \aleph_1^M $\omega \rightarrow^M 2$,
OF *function_space_rel_closed*] *function_space_rel_char*
 by (*auto* *dest!*: *M_genericD*)
 moreover from *this*
 obtain $z w$ where $x = \langle z, w \rangle$ $z \in \aleph_1^M$ $w : \omega \rightarrow 2$ by *auto*
 moreover from *calculation*
 have $w \in M[G]$ by (*auto* *dest:ext.transM*)
 ultimately
 show *?thesis* using *ext.function_space_rel_char* by *auto*
 qed
 moreover
 have *function*(*f_G*)
 using *Un_filter_is_function* *generic*
 by *fast*
 ultimately
 show *?thesis*
 using *generic_domain_f_G* *Pi_iff* by *auto*
 qed

abbreviation

$surj_dense :: i \Rightarrow i$ **where**
 $surj_dense(x) \equiv \{ p \in Coll . x \in range(p) \}$

lemma $surj_dense_closed$ [intro,simp]:

$x \in \omega \rightarrow^M \mathcal{Q} \implies surj_dense(x) \in M$
using $separation_in_range$ $transM$ [of x] **by** $simp$

lemma $reals_sub_image_f_G$:

assumes $x \in \omega \rightarrow^M \mathcal{Q}$
shows $\exists \alpha \in \aleph_1^M . f_G \alpha = x$

proof -

from $assms$

have $dense(surj_dense(x))$

using $dense_surj_dense$ $lepoll_rel_refl$ $InfCard_rel_Aleph_rel$

unfolding $dense_def$

by $auto$

with $\langle x \in \omega \rightarrow^M \mathcal{Q} \rangle$

obtain p **where** $p \in surj_dense(x)$ $p \in G$

using $M_generic_denseD$ [of $surj_dense(x)$]

by $blast$

then

show $?thesis$

using $succ_omega_closed_Coll$ $f_G_funtype$ $function_apply_equality$ [of $_ x$ f_G]

$succ_omega_closed_imp_no_new_reals$ [symmetric]

by ($auto$, $rule_tac$ $beXI$) ($auto$ $simp$: Pi_def)

qed

lemma $f_G_surj_Aleph_rel1_reals$: $f_G \in surj^{M[G]}(\aleph_1^M, \omega \rightarrow^{M[G]} \mathcal{Q})$

using $Aleph_rel_sub_closed$

proof ($intro$ $ext.mem_surj_abs$ [$THEN$ $iffD2$], $simp_all$)

show $f_G \in surj(\aleph_1^M, \omega \rightarrow^{M[G]} \mathcal{Q})$

using $f_G_funtype$ G_in_MG $ext.nat_into_M$ $f_G_in_MG$

$reals_sub_image_f_G$ $succ_omega_closed_Coll$

$succ_omega_closed_imp_no_new_reals$

unfolding $surj_def$

by $auto$

qed

lemma $continuum_rel_le_Aleph1_extension$:

includes $G_generic1_lemmas$

shows $\mathcal{Q}^{\aleph_0^{M[G]}} \leq \aleph_1^{M[G]}$

proof -

have $\aleph_1^M \in M[G]$ $Ord(\aleph_1^M)$

using $Card_rel_is_Ord$ **by** $auto$

moreover from $this$

have $\omega \rightarrow^{M[G]} \mathcal{Q} \lesssim^{M[G]} \aleph_1^M$

using $ext.surj_rel_implies_inj_rel$ [OF $f_G_surj_Aleph_rel1_reals$]

f_G_in_MG unfolding lepoll_rel_def by auto
with $\langle \text{Ord}(\aleph_1^M) \rangle$
have $|\omega \rightarrow^{M[G]} \mathcal{P}^{M[G]}| \leq |\aleph_1^M|^{M[G]}$
using $M_subset_MG[OF\ one_in_G]$ $Aleph_rel_closed[of\ 1]$
by $(rule_tac\ ext.lepoll_rel_imp_cardinal_rel_le)$ $simp_all$
ultimately
have $\mathcal{P}^{\aleph_0^{M[G]}, M[G]} \leq |\aleph_1^{M[G]}|^{M[G]}$
using $ext.lepoll_rel_imp_cardinal_rel_le[of\ \aleph_1^M\ \omega \rightarrow^{M[G]} \mathcal{P}]$
 $ext.Aleph_rel_zero\ succ_omega_closed_Coll$
 $succ_omega_closed_imp_Aleph_1_preserved$
unfolding $cexp_rel_def$ **by** $simp$
then
show $\mathcal{P}^{\aleph_0^{M[G]}, M[G]} \leq \aleph_1^{M[G]}$ **by** $simp$
qed

theorem CH: $\aleph_1^{M[G]} = \mathcal{P}^{\aleph_0^{M[G]}, M[G]}$
using $continuum_rel_le_Aleph1_extension\ ext.Aleph_rel_succ[of\ 0]$
 $ext.Aleph_rel_zero\ ext.csucc_rel_le[of\ \mathcal{P}^{\aleph_0^{M[G]}, M[G]} \omega]$
 $ext.Card_rel_cexp_rel\ ext.cantor_cexp_rel[of\ \omega]\ ext.Card_rel_nat$
 le_anti_sym
by $auto$

end — $collapse_CH$

31.2 Models of fragments of ZFC + CH

theorem ctm_of_CH :

assumes

$M \approx \omega$ $Transset(M)$

$M \models ZC \cup \{ \cdot Replacement(p) \cdot \mid p \in overhead_CH \}$

$\Phi \subseteq formula$ $M \models \{ \cdot Replacement(ground_repl_fm(\varphi)) \cdot \mid \varphi \in \Phi \}$

shows

$\exists N.$

$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZC \cup \{ \cdot CH \cdot \} \cup \{ \cdot Replacement(\varphi) \cdot$

$\mid \varphi \in \Phi \} \wedge$

$(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$

proof -

from $\langle M \models ZC \cup \{ \cdot Replacement(p) \cdot \mid p \in overhead_CH \} \rangle$

interpret $M_ZFC3\ M$

using $M_satT_overhead_imp_M_ZF3$ **unfolding** $overhead_CH_def$ $overhead_notCH_def$ **by** $auto$

from $\langle M \models ZC \cup \{ \cdot Replacement(p) \cdot \mid p \in overhead_CH \} \rangle$ $\langle Transset(M) \rangle$

interpret $M_ZF_ground_CH_trans\ M$

using $M_satT_imp_M_ZF_ground_CH_trans$

unfolding ZC_def **by** $auto$

from $\langle M \approx \omega \rangle$

obtain $enum$ **where** $enum \in bij(\omega, M)$

using $eqpoll_sym$ **unfolding** $eqpoll_def$ **by** $blast$

then

```

interpret M_ctm2_AC_CH M enum by unfold_locales
interpret forcing_data1 Coll Colleg 0 M enum
  using zero_in_Fn_rel[of  $\aleph_1^M$   $\aleph_1^M$   $\omega \rightarrow^M 2$ ]
  zero_top_Fn_rel[of  $\aleph_1^M$   $\aleph_1^M$   $\omega \rightarrow^M 2$ ]
  preorder_on_Fnle_rel[of  $\aleph_1^M$   $\aleph_1^M$   $\omega \rightarrow^M 2$ ]
  zero_lt_Aleph_rel1
  by unfold_locales simp_all
obtain G where M_generic(G)
  using generic_filter_existence[OF one_in_P]
  by auto
moreover from this
interpret collapse_CH M enum G by unfold_locales
have  $\aleph_1^{M[G]} = 2^{\aleph_0^{M[G]}}$ , M[G]
  using CH .
then
have M[G], []  $\models \cdot CH \cdot$ 
  using ext.is_ContHyp_iff
  by (simp add:ContHyp_rel_def)
then
have M[G]  $\models ZC \cup \{\cdot CH \cdot\}$ 
  using ext.M_satT_ZC by auto
moreover
have Transset(M[G]) using Transset_MG .
moreover
have M  $\subseteq$  M[G] using M_subset_MG[OF one_in_G] generic by simp
moreover
note  $\langle M \models \{ \cdot Replacement(ground\_repl\_fm(\varphi)) \cdot \cdot \varphi \in \Phi \} \rangle \langle \Phi \subseteq formula \rangle$ 
ultimately
show ?thesis
  using Ord_MG_iff_MG_eqpoll_nat_satT_ground_repl_fm_imp_satT_ZF_replacement_fm[of
 $\Phi$ ]
  by (rule_tac x=M[G] in exI,blast)
qed

```

corollary *ctm_ZFC_imp_ctm_CH*:

```

assumes
   $M \approx \omega$  Transset(M) M  $\models ZFC$ 
shows
   $\exists N.$ 
   $M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZFC \cup \{\cdot CH \cdot\} \wedge$ 
   $(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$ 
proof -
  from assms
  have  $\exists N.$ 
   $M \subseteq N \wedge$ 
   $N \approx \omega \wedge$ 
   $Transset(N) \wedge$ 
   $N \models ZC \wedge N \models \{\cdot CH \cdot\} \wedge N \models \{\cdot Replacement(x) \cdot \cdot x \in formula\} \wedge (\forall \alpha.$ 
 $Ord(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N)$ 

```

```

using ctm_of_CH[of M formula] satT_ZFC_imp_satT_ZC[of M]
  satT_mono[OF_ground_repl_fm_sub_ZFC, of M]
  satT_mono[OF_ZF_replacement_overhead_CH_sub_ZFC, of M]
  satT_mono[OF_ZF_replacement_fms_sub_ZFC, of M]
by (simp add: satT_Un_iff)
then
obtain N where  $N \models ZC$   $N \models \{\cdot CH \cdot\}$   $N \models \{\cdot Replacement(x) \cdot \cdot x \in formula\}$ 
   $M \subseteq N$   $N \approx \omega$  Transset(N) ( $\forall \alpha. Ord(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N$ )
by auto
moreover from this
have  $N \models ZFC$ 
  using satT_ZC_ZF_replacement_imp_satT_ZFC
by auto
moreover from this and  $\langle N \models \{\cdot CH \cdot\} \rangle$ 
have  $N \models ZFC \cup \{\cdot CH \cdot\}$ 
  using satT_ZC_ZF_replacement_imp_satT_ZFC
by auto
ultimately
show ?thesis
by auto
qed

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  $\mathcal{V}$ 
  using Union_ax_absolute upair_ax_absolute
  by unfold_locales auto

```

```

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}
using *power_ax_absolute separation_absolute replacement_absolute*
by *unfold_locales auto*

interpretation *V:M_eclose* \mathcal{V}
by *unfold_locales (auto intro:separation_absolute replacement_absolute*
simp:iterates_replacement_def wfrec_replacement_def)

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}
by *unfold_locales (auto intro:separation_absolute replacement_absolute*
simp add:iterates_replacement_def wfrec_replacement_def lam_replacement_def)

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}
by *unfold_locales (auto intro:separation_absolute replacement_absolute*
simp:wfrec_replacement_def)

lemma *choice_ax_Universe: choice_ax*(\mathcal{V})

proof -

{
 fix *x*
 obtain *f* **where** *f* \in *surj(|x|,x)*
 using *cardinal_eqpoll unfolding eqpoll_def bij_def* **by** *fast*
 moreover
 have *Ord(|x|)* **by** *simp*
 ultimately
 have $\exists a. \text{Ord}(a) \wedge (\exists f. f \in \text{surj}(a,x))$
 by *fast*

}

then

show *?thesis* **unfolding** *choice_ax_def rall_def rex_def*

by *simp*

qed

interpretation *V:M_master* \mathcal{V}
using *choice_ax_Universe*
by *unfold_locales (auto intro:separation_absolute replacement_absolute*
simp:lam_replacement_def transrec_replacement_def wfrec_replacement_def
is_wfrec_def M_is_recfun_def)

named_theorems *V_simps*

— To work systematically, ASCII versions of ”_absolute” theorems as those below are preferable.

lemma *eqpoll_rel_absolute*[*V_simps*]: $x \approx^{\mathcal{V}} y \longleftrightarrow x \approx y$
unfolding *eqpoll_def* **using** *V.def_eqpoll_rel* **by** *auto*

lemma *cardinal_rel_absolute*[*V_simps*]: $|x|^{\mathcal{V}} = |x|$
unfolding *cardinal_def* *cardinal_rel_def* **by** (*simp add:V_simps*)

lemma *Card_rel_absolute*[*V_simps*]: $\text{Card}^{\mathcal{V}}(a) \longleftrightarrow \text{Card}(a)$
unfolding *Card_rel_def* *Card_def* **by** (*simp only:V_simps*)

lemma *csucc_rel_absolute*[*V_simps*]: $(a^+)^{\mathcal{V}} = a^+$
unfolding *csucc_rel_def* *csucc_def* **by** (*simp add:V_simps*)

lemma *function_space_rel_absolute*[*V_simps*]: $x \rightarrow^{\mathcal{V}} y = x \rightarrow y$
using *V.function_space_rel_char* **by** (*simp add:V_simps*)

lemma *cexp_rel_absolute*[*V_simps*]: $x^{\uparrow y, \mathcal{V}} = x^{\uparrow y}$
unfolding *cexp_rel_def* *cexp_def* **by** (*simp only:V_simps*)

lemma *HAleph_rel_absolute*[*V_simps*]: $\text{HAleph_rel}(\mathcal{V}, a, b) = \text{HAleph}(a, b)$
unfolding *HAleph_rel_def* *HAleph_def* **by** (*auto simp add:V_simps*)

lemma *Aleph_rel_absolute*[*V_simps*]: $\text{Ord}(x) \implies \aleph_x^{\mathcal{V}} = \aleph_x$

proof -

assume *Ord(x)*

have $\aleph_x^{\mathcal{V}} = \text{transrec}(x, \lambda a b. \text{HAleph_rel}(\mathcal{V}, a, b))$

unfolding *Aleph_rel_def* **by** *simp*

also

have $\dots = \text{transrec}(x, \text{HAleph})$

by (*simp only:V_simps*)

also from $\langle \text{Ord}(x) \rangle$

have $\dots = \aleph_x$

using *Aleph'_eq_Aleph* **unfolding** *Aleph'_def* **by** *simp*

finally

show *?thesis* .

qed

Example of absolute lemmas obtained from the relative versions. Note the *only* declarations

lemma *Ord_cardinal_idem'*: $\text{Ord}(A) \implies ||A|| = |A|$
using *V.Ordinal_cardinal_rel_idem* **by** (*simp only:V_simps*)

lemma *Aleph_succ'*: $\text{Ord}(\alpha) \implies \aleph_{\text{succ}(\alpha)} = \aleph_{\alpha}^+$
using *V.Aleph_rel_succ* **by** (*simp only:V_simps*)

These two results are new, first obtained in relative form (not ported).

```
lemma csucc_cardinal:
  assumes Ord( $\kappa$ ) shows  $|\kappa|^+ = \kappa^+$ 
  using assms V.csucc_rel_cardinal_rel by (simp only: V_simps)
```

```
lemma csucc_le_mono:
  assumes  $\kappa \leq \nu$  shows  $\kappa^+ \leq \nu^+$ 
  using assms V.csucc_rel_le_mono by (simp only: V_simps)
```

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 -
  interpret M_N_Perm M  $\mathcal{V}$ 
  by (unfold locales, simp only: V_simps)
  from assms
  show ?thesis using eqpoll_rel_transfer
  by (simp only: V_simps)
qed
```

The “relationalized” *CH* with respect to \mathcal{V} corresponds to the real *CH*.

```
lemma is_ContHyp_iff_CH:  $is\_ContHyp(\mathcal{V}) \longleftrightarrow ContHyp$ 
  using V.is_ContHyp_iff
  by (auto simp add: ContHyp_rel_def ContHyp_def V_simps)
```

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* $[unfolded\ inj_def\ surj_def]$

$$\begin{aligned} bij(A, B) &\equiv \\ &\{f \in A \rightarrow B . \forall w \in A. \forall x \in A. f \text{ ' } w = f \text{ ' } x \longrightarrow w = x\} \cap \\ &\{f \in A \rightarrow B . \forall y \in B. \exists x \in A. f \text{ ' } x = y\} \end{aligned}$$

thm *eqpoll_def*

$$A \approx B \equiv \exists f. f \in bij(A, B)$$

thm *Transset_def*

$$Transset(i) \equiv \forall x \in i. x \subseteq i$$

thm *Ord_def*

$$Ord(i) \equiv Transset(i) \wedge (\forall x \in i. Transset(x))$$

thm *lt_def le_iff*

$$\begin{aligned} i < j &\equiv i \in j \wedge Ord(j) \\ i \leq j &\longleftrightarrow i < j \vee i = j \wedge Ord(j) \end{aligned}$$

With the concepts of empty set and successor in place,

lemma *empty_def'*: $\forall x. x \notin 0$ **by** *simp*

lemma *succ_def'*: $succ(i) = i \cup \{i\}$ **by** *blast*

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* $[unfolded\ Limit_def]$

$$\begin{aligned} Ord(\omega) \wedge 0 < \omega \wedge (\forall y. y < \omega \longrightarrow succ(y) < \omega) \\ Ord(i) \wedge 0 < i \wedge (\forall y. y < i \longrightarrow succ(y) < i) &\Longrightarrow \omega \leq i \end{aligned}$$

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*

$$\begin{aligned} [] &\in \text{list}(A) \\ \llbracket a \in A; l \in \text{list}(A) \rrbracket &\implies \text{Cons}(a, l) \in \text{list}(A) \\ \llbracket x \in \text{list}(A); P([]); \bigwedge a l. \llbracket a \in A; l \in \text{list}(A); P(l) \rrbracket \rrbracket &\implies P(\text{Cons}(a, l)) \\ &\implies P(x) \end{aligned}$$

Length, concatenation, and n th element of lists are recursively characterized as follows.

thm *length.simps app.simps nth_0 nth_Cons*

$$\begin{aligned} \text{length}([]) &= 0 \\ \text{length}(\text{Cons}(a, l)) &= \text{succ}(\text{length}(l)) \\ [] @ ys &= ys \\ \text{Cons}(a, l) @ ys &= \text{Cons}(a, l @ ys) \\ \text{nth}(0, \text{Cons}(a, l)) &= a \\ n \in \omega &\implies \text{nth}(\text{succ}(n), \text{Cons}(a, l)) = \text{nth}(n, l) \end{aligned}$$

We have the usual Haskell-like notation for iterated applications of Cons :

lemma *Cons_app*: $[a,b,c] = \text{Cons}(a, \text{Cons}(b, \text{Cons}(c, [])))$..

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 $\forall x[M]. P(x) \equiv \forall x. M(x) \longrightarrow P(x)$

$\exists x[M]. P(x) \equiv \exists x. M(x) \wedge P(x)$

unfolding *rall_def rex_def* .

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*

$is_function_space(M, A, B, fs) \equiv$
 $M(fs) \wedge (\forall f[M]. f \in fs \longleftrightarrow typed_function(M, A, B, f))$
 $A \rightarrow^M B \equiv \text{THE } d. is_function_space(M, A, B, d)$
 $surjection(M, A, B, f) \equiv$
 $typed_function(M, A, B, f) \wedge$
 $(\forall y[M]. y \in B \longrightarrow (\exists x[M]. x \in A \wedge is_apply(M, f, x, y)))$

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*

$$\text{foundation_ax}(M) \equiv \forall x[M]. (\exists y[M]. y \in x) \longrightarrow (\exists y[M]. y \in x \wedge \neg (\exists z[M]. z \in x \wedge z \in y))$$

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** $A \approx^M B \iff (\exists f[M]. f \in bij(A,B))$
using *assms unfolding eqpoll_rel_def* **by** *auto*

Below, μ denotes the minimum operator on the ordinals.

lemma *cardinalities_defs*:
fixes $M::i \Rightarrow o$
shows
 $|A|^M \equiv \mu i. M(i) \wedge i \approx^M A$
 $Card^M(\alpha) \equiv \alpha = |\alpha|^M$
 $\kappa^{\uparrow \nu, M} \equiv |\nu \rightarrow^M \kappa|^M$
 $(\kappa^+)^M \equiv \mu x. M(x) \wedge Card^M(x) \wedge \kappa < x$
unfolding *cardinal_rel_def cexp_rel_def*
csucc_rel_def Card_rel_def .

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

$$CH^N \equiv \aleph_I^N = \mathcal{P}^{\uparrow \aleph_0^{N,N}}$$

unfolding $ContHyp_rel_def$.

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*]

$$M_ZF_library(M) \Longrightarrow is_ContHyp(M) \longleftrightarrow CH^M$$

$$is_ContHyp(\mathcal{V}) \longleftrightarrow \aleph_I = \mathcal{P}^{\uparrow \aleph_0}$$

In turn, the fully relational version evaluated on a nonempty transitive A is equivalent to the satisfaction of the first-order formula $\cdot CH$.

thm $is_ContHyp_iff_sats$

$$\llbracket env \in list(A); 0 \in A \rrbracket \Longrightarrow is_ContHyp(\#\#A) \longleftrightarrow A, env \models \cdot CH$$

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 $extensions_of_ctms_ZF$

$$\llbracket M \approx \omega; Transset(M); M \models ZF \rrbracket$$

$$\Longrightarrow \exists N. M \subseteq N \wedge$$

$$N \approx \omega \wedge$$

$$Transset(N) \wedge$$

$$N \models ZF \wedge$$

$$M \neq N \wedge (\forall \alpha. Ord(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N) \wedge ((M, [] \models \cdot AC) \longrightarrow N \models ZFC)$$

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 $ctm_ZFC_imp_ctm_not_CH$

$$\begin{aligned} & \llbracket M \approx \omega; \text{Transset}(M); M \models \text{ZFC} \rrbracket \\ \implies & \exists N. M \subseteq N \wedge \\ & N \approx \omega \wedge \\ & \text{Transset}(N) \wedge N \models \text{ZFC} \cup \{\neg \cdot CH \cdot\} \wedge (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \\ & \alpha \in N) \end{aligned}$$

thm *ctm_ZFC_imp_ctm_CH*

$$\begin{aligned} & \llbracket M \approx \omega; \text{Transset}(M); M \models \text{ZFC} \rrbracket \\ \implies & \exists N. M \subseteq N \wedge \\ & N \approx \omega \wedge \\ & \text{Transset}(N) \wedge N \models \text{ZFC} \cup \{CH\} \wedge (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \\ & \in N) \end{aligned}$$

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:
       $\bigwedge P. \text{strong\_replacement}(M, P)$ 
       $\bigwedge F. \text{lam\_replacement}(M, F)$ 
       $\bigwedge Q S. \text{iterates\_replacement}(M, Q, S)$ 
       $\bigwedge Q S. \text{wfrec\_replacement}(M, Q, S)$ 
       $\bigwedge Q S. \text{transrec\_replacement}(M, Q, S)$ 
  and
    sorried_separations:  $\bigwedge Q. \text{separation}(M, Q)$ 
  shows
    M_master(M)
  apply unfold_locales
    apply
      (simp_all add:
        sorried_replacements(1-2)
        sorried_separations
        power_infty)
  oops

```

— NOTE: Only for pretty-printing purposes, overrides previous fundamental notations

no_notation *mem* (**infixl** $\langle \in \rangle$ 50)
no_notation *conj* (**infixr** $\langle \wedge \rangle$ 35)
no_notation *disj* (**infixr** $\langle \vee \rangle$ 30)
no_notation *iff* (**infixr** $\langle \longleftrightarrow \rangle$ 25)
no_notation *imp* (**infixr** $\langle \longrightarrow \rangle$ 25)
no_notation *not* ($\langle \neg _ \rangle$ [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* ($\langle \neg' (_ \wedge / _) \rangle$)
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* ($\langle '(\forall (/ _) \cdot) \rangle$)
no_notation *Exists* ($\langle '(\exists (/ _) \cdot) \rangle$)

notation *Member* (**infixl** $\langle \in \rangle$ 50)
notation *Equal* (**infixl** $\langle \equiv \rangle$ 50)
notation *Nand* ($\langle \neg' (_ \wedge / _) \rangle$)
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*($t_1 \in t_2$) = ($0 \in 1 \wedge \text{forces_mem_fm}(1, 2, 0, t_1 + \omega 4, t_2 + \omega 4)$)
unfolding *forces_def* **by** *simp*

definition *forces_0_mem_1* **where** *forces_0_mem_1* \equiv *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*:

$mem(n,\omega) \implies mem(\varphi,formula) \implies incr_bv(Neg(\varphi))'n = ?x$

unfolding *Neg_def* **by** *simp*

schematic_goal *incr_bv_Exists* [*incr_bv_new_simps*]:

$mem(n,\omega) \implies mem(\varphi,formula) \implies incr_bv(Exists(\varphi))'n = ?x$

unfolding *Exists_def* **by** (*simp add: incr_bv_Neg*)

— The two renamings involved in the definition of *forces* depend on the recursive function *incr_bv*. Here we have an apparently exponential bottleneck, since all the propositional connectives (even *Neg*) duplicate the appearances of *incr_bv*. Not even the double negation of an atomic formula can be managed by the system (in version 2021-1).

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).