The Independence of the Continuum Hypothesis in Isabelle/ZF

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

September 30, 2022

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

*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.

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* :: $i⇒i⇒i⇒i⇒o$ **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* :
$[\![\ d{\in}A\ ;\ \langle d,p\rangle{\in}r\ ;\ \langle d,g\rangle{\in}r\ ]\!] \Longrightarrow compat\_in(A,r,p,g)$
⟨*proof*⟩

lemma *refl_compat*:
$[\![\ refl(A,r)\ ;\ \langle p,q\rangle \in r \mid p{=}q \mid \langle q,p\rangle \in r\ ;\ p{\in}A\ ;\ q{\in}A]\!] \Longrightarrow compat\_in(A,r,p,q)$
⟨*proof*⟩

lemma *chain_compat*:
$refl(A,r) \Longrightarrow linear(A,r) \Longrightarrow\ (\forall\, p{\in}A.\forall\, q{\in}A.\ compat\_in(A,r,p,q))$
⟨*proof*⟩

lemma *subset_fun_image*: $f{:}N{\to}P \Longrightarrow f``N{\subseteq}P$
⟨*proof*⟩

lemma *refl_monot_domain*: $refl(B,r) \Longrightarrow A{\subseteq}B \Longrightarrow refl(A,r)$
⟨*proof*⟩

locale *forcing_notion* =
fixes P (‹ℙ›) **and** *leq* **and** *one* (‹**1**›)
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*:
$r{\in}\mathbb{P} \Longrightarrow r{\preceq}r$
⟨*proof*⟩

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

definition

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

lemma *dense_belowI* [*intro*!]:
assumes $\bigwedge q.\ q{\in}\mathbb{P} \Longrightarrow q{\preceq}p \Longrightarrow \exists\, d{\in}D.\ d{\in}\mathbb{P} \wedge d{\preceq}q$
shows *dense_below*(D,p)
⟨*proof*⟩

lemma *dense_below_cong*: $p{\in}\mathbb{P} \Longrightarrow D = D' \Longrightarrow$ *dense_below*(D,p) $\longleftrightarrow$ *dense_below*(D',p)
⟨*proof*⟩

lemma *dense_below_cong'*: $p{\in}\mathbb{P} \Longrightarrow [\![\bigwedge x.\ x{\in}\mathbb{P} \Longrightarrow Q(x) \longleftrightarrow Q'(x)]\!] \Longrightarrow$
dense_below($\{q{\in}\mathbb{P}.\ Q(q)\}$,$p$) $\longleftrightarrow$ *dense_below*($\{q{\in}\mathbb{P}.\ Q'(q)\}$,$p$)
⟨*proof*⟩

lemma *dense_below_mono*: $p{\in}\mathbb{P} \Longrightarrow D \subseteq D' \Longrightarrow$ *dense_below*(D,p) $\Longrightarrow$ *dense_below*(D',p)
⟨*proof*⟩

lemma *dense_below_under*:
assumes *dense_below*(D,p) $p{\in}\mathbb{P}$ $q{\in}\mathbb{P}$ $q{\preceq}p$
shows *dense_below*(D,q)
⟨*proof*⟩

lemma *ideal_dense_below*:
assumes $\bigwedge q.\ q{\in}\mathbb{P} \Longrightarrow q{\preceq}p \Longrightarrow q{\in}D$
shows *dense_below*(D,p)
⟨*proof*⟩

lemma *dense_below_dense_below*:
assumes *dense_below*($\{q{\in}\mathbb{P}.\ $ *dense_below*(D,q)$\}$,p) $p{\in}\mathbb{P}$
shows *dense_below*(D,p)
⟨*proof*⟩

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

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

lemma *filterD* : *filter*(G) $\Longrightarrow x \in G \Longrightarrow x \in \mathbb{P}$
⟨*proof*⟩

lemma *filter_subset_notion*[*dest*]: *filter*(G) $\Longrightarrow G \subseteq \mathbb{P}$
⟨*proof*⟩

lemma *filter_leqD* : *filter*(G) $\Longrightarrow x \in G \Longrightarrow y \in \mathbb{P} \Longrightarrow x{\preceq}y \Longrightarrow y \in G$
⟨*proof*⟩

lemma *filter_imp_compat*: *filter*(G) $\Longrightarrow p{\in}G \Longrightarrow q{\in}G \Longrightarrow$ *compat*(p,q)
⟨*proof*⟩

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

assumes *filter*(G) **and** $p \in G$ **and** $q \in G$
shows $\exists\, r \in G.\ r \preceq p \wedge r \preceq q$
⟨*proof*⟩

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*] : $p \in \mathbb{P} \Longrightarrow a \in A \Longrightarrow a \preceq p \Longrightarrow p \in upclosure(A)$
⟨*proof*⟩

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

lemma *upclosureD* [*dest*] :
$p \in upclosure(A) \Longrightarrow \exists\, a \in A.(a \preceq p) \wedge p \in \mathbb{P}$
⟨*proof*⟩

lemma *upclosure_increasing* :
assumes $A \subseteq \mathbb{P}$
shows *increasing*(*upclosure*(A))
⟨*proof*⟩

lemma *upclosure_in_P*: $A \subseteq \mathbb{P} \Longrightarrow upclosure(A) \subseteq \mathbb{P}$
⟨*proof*⟩

lemma *A_sub_upclosure*: $A \subseteq \mathbb{P} \Longrightarrow A \subseteq upclosure(A)$
⟨*proof*⟩

lemma *elem_upclosure*: $A \subseteq \mathbb{P} \Longrightarrow x \in A \Longrightarrow x \in upclosure(A)$
⟨*proof*⟩

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))
⟨*proof*⟩

lemma *aux_RS1*: $f \in N \to \mathbb{P} \Longrightarrow n \in N \Longrightarrow f`n \in upclosure(f\ ``N)$
⟨*proof*⟩

lemma *decr_succ_decr*:
assumes $f \in nat \to \mathbb{P}$ *preorder_on*($\mathbb{P}$,*leq*)
$\forall\, n \in nat.\ \langle f\ `\ succ(n), f\ `\ n\rangle \in leq$
$m \in nat$
shows $n \in nat \Longrightarrow n \le m \Longrightarrow \langle f\ `\ m, f\ `\ n\rangle \in leq$
⟨*proof*⟩

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

end — *forcing_notion*

2.2 Towards Rasiowa-Sikorski Lemma (RSL)

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

begin

definition
  D_generic :: $i{\Rightarrow}o$ **where**
  D_generic(G) ≡ *filter*(G) ∧ ($\forall\, n{\in}nat.(\mathcal{D}`n){\cap}G{\neq}0$)

The next lemma identifies a sufficient condition for obtaining RSL.

lemma *RS_sequence_imp_rasiowa_sikorski*:
  assumes
    $p{\in}\mathbb{P}$ f : $nat{\to}\mathbb{P}$ $f\ `\ 0 = p$
    $\bigwedge n.\ n{\in}nat \Longrightarrow f\ `\ succ(n){\preceq} f\ `\ n \wedge f\ `\ succ(n) \in \mathcal{D}\ `\ n$
  shows
    $\exists\, G.\ p{\in}G \wedge$ *D_generic*(G)
⟨*proof*⟩

end — *countable_generic*

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

consts *RS_seq* :: [i,i,i,i,i,i] ⇒ i
primrec
  RS_seq(0,P,*leq*,p,*enum*,$\mathcal{D}$) = p
  RS_seq(*succ*(n),P,*leq*,p,*enum*,$\mathcal{D}$) =
    enum`(μ $m$. ⟨*enum*`m, *RS_seq*(n,P,*leq*,p,*enum*,$\mathcal{D}$)⟩ ∈ *leq* ∧ *enum*`$m$ ∈ $\mathcal{D}$ ` n)

context *countable_generic*
begin

lemma *countable_RS_sequence_aux*:
  fixes p *enum*
  defines $f(n)$ ≡ *RS_seq*(n,$\mathbb{P}$,*leq*,p,*enum*,$\mathcal{D}$)
    and $Q(q,k,m)$ ≡ *enum*`$m{\preceq}$ $q$ ∧ *enum*`$m \in \mathcal{D}$ ` k

assumes $n{\in}nat$ $p{\in}\mathbb{P}$ $\mathbb{P} \subseteq range(enum)$ $enum{:}nat{\to}M$
$\bigwedge x\ k.\ x{\in}\mathbb{P} \Longrightarrow k{\in}nat \Longrightarrow\ \exists\, q{\in}\mathbb{P}.\ q{\preceq}\ x \wedge q \in \mathcal{D}\ `\ k$
shows
$f(succ(n)) \in \mathbb{P} \wedge f(succ(n)){\preceq}\ f(n) \wedge f(succ(n)) \in \mathcal{D}\ `\ n$
⟨*proof*⟩

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 $n{\in}nat$ $p{\in}\mathbb{P}$ $\mathbb{P} \subseteq range(enum)$ $enum{:}nat{\to}M$
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*⟩

lemma *RS_seq_type*:
assumes $n \in nat$ $p{\in}\mathbb{P}$ $\mathbb{P} \subseteq range(enum)$ $enum{:}nat{\to}M$
shows $RS\_seq(n,\mathbb{P},leq,p,enum,\mathcal{D}) \in \mathbb{P}$
⟨*proof*⟩

lemma *RS_seq_funtype*:
assumes $p{\in}\mathbb{P}$ $\mathbb{P} \subseteq range(enum)$ $enum{:}nat{\to}M$
shows $(\lambda n{\in}nat.\ RS\_seq(n,\mathbb{P},leq,p,enum,\mathcal{D}))\colon nat \to \mathbb{P}$
⟨*proof*⟩

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 κ 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$(I,J,w,x) \equiv$
$\{\ p \in Fn(\omega, I \times \omega, J)\ .\ (\exists\, n \in \omega.\ \langle\langle w,n\rangle, 1\rangle \in p \wedge \langle\langle x,n\rangle, 0\rangle \in p)\ \}$

lemma *dense_inj_dense*:
assumes $w \in I\ x \in I\ w \neq x\ p \in Fn(\omega, I \times \omega, J)\ 0 \in J\ 1 \in J$
shows $\exists\, d \in inj\_dense(I,J,w,x).\ \langle d\ , p\rangle \in Fnle(\omega, I \times \omega, J)$
⟨*proof*⟩

locale *add_reals* = *cohen_data nat _ 2*

3.1 Combinatorial results on Cohen posets

sublocale *cohen_data* $\subseteq$ *forcing_notion* $Fn(\kappa,I,J)$ $Fnle(\kappa,I,J)$ *0*
⟨*proof*⟩

context *cohen_data*
begin

lemma *compat_imp_Un_is_function*:
assumes $G \subseteq Fn(\kappa,\ I,\ J)$ $\bigwedge p\ q.\ p \in G \Longrightarrow q \in G \Longrightarrow compat(p,q)$
shows *function*$(\bigcup G)$
⟨*proof*⟩

lemma *Un_filter_is_function*: *filter*$(G) \Longrightarrow$ *function*$(\bigcup G)$
⟨*proof*⟩

end — *cohen_data*

locale *M_cohen* = *M_delta* +
assumes
countable_lepoll_assms2:
$M(A') \Longrightarrow M(A) \Longrightarrow M(b) \Longrightarrow M(f) \Longrightarrow separation(M,\ \lambda y.\ \exists\, x \in A'.\ y = \langle x,$
$\mu\ i.\ x \in if\_range\_F\_else\_F(\lambda a.\ \{p \in A\ .\ domain(p) = a\},\ b,\ f,\ i)\rangle)$
and
countable_lepoll_assms3:
$M(A) \Longrightarrow M(f) \Longrightarrow M(b) \Longrightarrow M(D) \Longrightarrow M(r') \Longrightarrow M(A') \Longrightarrow$
$separation(M,\ \lambda y.\ \exists\, x \in A'.\ y = \langle x,\ \mu\ i.\ x \in if\_range\_F\_else\_F(drSR\_Y(r',$
$D,\ A),\ b,\ f,\ i)\rangle)$

lemma (**in** *M_library*) *Fnle_rel_Aleph_rel1_closed*[*intro*,*simp*]:
$M(Fnle^M(\aleph_1{}^M,\ \aleph_1{}^M,\ \omega \rightarrow^M 2))$
⟨*proof*⟩

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

begin

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

end — *M_add_reals*

⟨*ML*⟩
context
  notes *Un_assoc*[*simp*] *Un_trasposition_aux2*[*simp*]
begin
⟨*ML*⟩
end

lemma (**in** *M_trivial*) *compat_in_abs*[*absolut*]:
  assumes
    $M(A)$ $M(r)$ $M(p)$ $M(q)$
  shows
    is_compat_in(M,A,r,p,q) $\longleftrightarrow$ *compat_in*(A,r,p,q)
  ⟨*proof*⟩

definition
  antichain :: $i \Rightarrow i \Rightarrow i \Rightarrow o$ **where**
  antichain(P,*leq*,A) $\equiv$ $A \subseteq P \wedge (\forall\, p \in A.\ \forall\, q \in A.\ p \neq q \longrightarrow \neg$*compat_in*($P$,*leq*,$p$,$q$))

⟨*ML*⟩

definition
  ccc :: $i \Rightarrow i \Rightarrow o$ **where**
  ccc(P,*leq*) $\equiv \forall\, A.$ *antichain*(P,*leq*,A) $\longrightarrow |A| \leq$ *nat*

abbreviation
  antichain_rel_abbr :: $[i \Rightarrow o,i,i,i] \Rightarrow o$ (‹*antichain*–′(_,_,_′)›) **where**
  $\text{antichain}^M(P,\text{leq},A) \equiv$ *antichain_rel*(M,P,*leq*,A)

abbreviation
  antichain_r_set :: $[i,i,i,i] \Rightarrow o$ (‹*antichain*–′(_,_,_′)›) **where**
  $\text{antichain}^M(P,\text{leq},A) \equiv$ *antichain_rel*(##M,P,*leq*,A)

context *M_trivial*
begin

lemma *antichain_abs* [*absolut*]:
  ⟦ $M(A)$; $M(P)$; $M(\text{leq})$ ⟧ $\Longrightarrow$ $\text{antichain}^M(P,\text{leq},A) \longleftrightarrow$ *antichain*(P,*leq*,A)
  ⟨*proof*⟩

end — *M_trivial*

⟨*ML*⟩

abbreviation
ccc_rel_abbr :: $[i \Rightarrow o,i,i] \Rightarrow o$ ($\langle ccc\text{–}'(\_,\_')\rangle$) **where**
$ccc\_rel\_abbr(M) \equiv ccc\_rel(M)$

abbreviation
ccc_r_set :: $[i,i,i] \Rightarrow o$ ($\langle ccc\text{–}'(\_,\_')\rangle$) **where**
$ccc\_r\_set(M) \equiv ccc\_rel(\#\#M)$

context *M_cardinals*
begin

lemma *def_ccc_rel*:
shows
$ccc^M(P,leq) \longleftrightarrow (\forall\, A[M].\ antichain^M(P,leq,A) \longrightarrow |A|^M \le \omega)$
⟨*proof*⟩

end — *M_cardinals*

context *M_FiniteFun*
begin

lemma *Fnle_nat_closed*[*intro*,*simp*]:
assumes $M(I)$ $M(J)$
shows $M(Fnle(\omega,I,J))$
⟨*proof*⟩

lemma *Fn_nat_closed*:
assumes $M(A)$ $M(B)$ **shows** $M(Fn(\omega,A,B))$
⟨*proof*⟩

end — *M_FiniteFun*

context *M_add_reals*
begin

lemma $lam\_replacement\_drSR\_Y$: $M(A) \Longrightarrow M(D) \Longrightarrow M(r') \Longrightarrow lam\_replacement(M, drSR\_Y(r',D,A))$
⟨*proof*⟩

lemma (**in** *M_trans*) *mem_F_bound3*:
fixes F A
defines $F \equiv dC\_F$
shows $x \in F(A,c) \Longrightarrow c \in (range(f) \cup \{domain(x).\ x \in A\})$
⟨*proof*⟩

lemma *ccc_rel_Fn_nat*:
assumes $M(I)$
shows $ccc^M(Fn(nat,I,2),\ Fnle(nat,I,2))$

⟨*proof*⟩

end — *M_add_reals*

end
theory *Edrel*
imports
Transitive_Models.ZF_Miscellanea
Transitive_Models.Recursion_Thms

begin

3.2 The well-founded relation *ed*

lemma *eclose_sing* : $x \in eclose(a) \Longrightarrow x \in eclose(\{a\})$
⟨*proof*⟩

lemma *ecloseE* :
assumes $x \in eclose(A)$
shows $x \in A \vee (\exists\ B \in A\ .\ x \in eclose(B))$
⟨*proof*⟩

lemma *eclose_singE* : $x \in eclose(\{a\}) \Longrightarrow x = a \vee x \in eclose(a)$
⟨*proof*⟩

lemma *in_eclose_sing* :
assumes $x \in eclose(\{a\})$ $a \in eclose(z)$
shows $x \in eclose(\{z\})$
⟨*proof*⟩

lemma *in_dom_in_eclose* :
assumes $x \in domain(z)$
shows $x \in eclose(z)$
⟨*proof*⟩

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

definition *ed* :: $[i,i] \Rightarrow o$ **where**
$ed(x,y) \equiv x \in domain(y)$

definition *edrel* :: $i \Rightarrow i$ **where**
$edrel(A) \equiv Rrel(ed,A)$

lemma *edI*[*intro*!]: $t{\in}domain(x) \Longrightarrow ed(t,x)$
⟨*proof*⟩

lemma *edD*[*dest*!]: $ed(t,x) \Longrightarrow t{\in}domain(x)$
⟨*proof*⟩

lemma *rank_ed*:

assumes $ed(y,x)$
shows $succ(rank(y)) \leq rank(x)$
⟨*proof*⟩

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

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

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

lemma *edrel_trans*: $Transset(A) \Longrightarrow y \in A \Longrightarrow x \in domain(y) \Longrightarrow \langle x,y \rangle \in edrel(A)$
⟨*proof*⟩

lemma *domain_trans*: $Transset(A) \Longrightarrow y \in A \Longrightarrow x \in domain(y) \Longrightarrow x \in A$
⟨*proof*⟩

lemma *relation_edrel* : $relation(edrel(A))$
⟨*proof*⟩

lemma *field_edrel* : $field(edrel(A)) \subseteq A$
⟨*proof*⟩

lemma *edrel_sub_memrel*: $edrel(A) \subseteq trancl(Memrel(eclose(A)))$
⟨*proof*⟩

lemma *wf_edrel* : $wf(edrel(A))$
⟨*proof*⟩

lemma *ed_induction*:
assumes $\bigwedge x.\ [\![\bigwedge y.\ \ ed(y,x) \Longrightarrow Q(y)\ ]\!] \Longrightarrow Q(x)$
shows $Q(a)$
⟨*proof*⟩

lemma *dom_under_edrel_eclose*: $edrel(eclose(\{x\}))\ \text{-``}\ \{x\} = domain(x)$
⟨*proof*⟩

lemma *ed_eclose* : $\langle y,z \rangle \in edrel(A) \Longrightarrow y \in eclose(z)$
⟨*proof*⟩

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

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

lemma *tr_edrel_subset* :
assumes $z \in domain(x)$
shows $tr\_down(edrel(eclose(\{x\})),z) \subseteq eclose(\{z\})$
⟨*proof*⟩

end

4 Well-founded relation on names

theory *FrecR*
imports
Transitive_Models.Discipline_Function
Edrel
begin

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

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

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

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

definition
$cond\_of :: i \Rightarrow i$ **where**
$cond\_of(x) \equiv snd(snd(snd((x))))$

lemma *components_simp*:
$ftype(\langle f,n1,n2,c\rangle) = f$
$name1(\langle f,n1,n2,c\rangle) = n1$
$name2(\langle f,n1,n2,c\rangle) = n2$
$cond\_of(\langle f,n1,n2,c\rangle) = c$
⟨*proof*⟩

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

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

lemma *components_in_eclose* :

$n1 \in ecloseN(\langle f,n1,n2,c\rangle)$
$n2 \in ecloseN(\langle f,n1,n2,c\rangle)$
⟨*proof*⟩

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

lemma *ecloseNI1* :
assumes $x \in eclose(n1) \vee x \in eclose(n2)$
shows $x \in ecloseN(\langle f,n1,n2,c\rangle)$
⟨*proof*⟩

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*⟩

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

definition
ftype_fm :: $[i,i] \Rightarrow i$ **where**
ftype_fm ≡ *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$(\#\#A,x,y) \longleftrightarrow sats(A,ftype\_fm(a,b), env)$
⟨*proof*⟩

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*]:
$[\![x \in nat; y \in nat; env \in list(A)]\!] \Longrightarrow$
$(A, env \models name1\_fm(x,y)) \longleftrightarrow is\_name1(\#\#A, nth(x,env), nth(y,env))$
⟨*proof*⟩

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*)
⟨*proof*⟩

definition
is_snd_snd :: (*i*⇒*o*)⇒*i*⇒*i*⇒*o* **where**
is_snd_snd(*M*,*x*,*t*) ≡ *is_hcomp*(*M*,*is_snd*(*M*),*is_snd*(*M*),*x*,*t*)

definition
snd_snd_fm :: [*i*,*i*]⇒*i* **where**
snd_snd_fm(*x*,*t*) ≡ *hcomp_fm*(*snd_fm*,*snd_fm*,*x*,*t*)

lemma *sats_snd2_fm* [*simp*]:
⟦ *x* ∈ *nat*; *y* ∈ *nat*;*env* ∈ *list*(*A*) ⟧ ⟹
(*A*, *env* $\models$ *snd_snd_fm*(*x*,*y*)) $\longleftrightarrow$ *is_snd_snd*(##*A*, *nth*(*x*,*env*), *nth*(*y*,*env*))
⟨*proof*⟩

definition
is_name2 :: (*i*⇒*o*)⇒*i*⇒*i*⇒*o* **where**
is_name2(*M*,*x*,*t3*) ≡ *is_hcomp*(*M*,*is_fst*(*M*),*is_snd_snd*(*M*),*x*,*t3*)

definition
name2_fm :: [*i*,*i*] ⇒ *i* **where**
name2_fm(*x*,*t3*) ≡ *hcomp_fm*(*fst_fm*,*snd_snd_fm*,*x*,*t3*)

lemma *sats_name2_fm* :
⟦ *x* ∈ *nat*; *y* ∈ *nat*;*env* ∈ *list*(*A*) ⟧
⟹ (*A* , *env* $\models$ *name2_fm*(*x*,*y*)) $\longleftrightarrow$ *is_name2*(##*A*, *nth*(*x*,*env*), *nth*(*y*,*env*))
⟨*proof*⟩

lemma *is_name2_iff_sats* [*iff_sats*]:
assumes
nth(*a*,*env*) = *x* *nth*(*b*,*env*) = *y* *a*∈*nat* *b*∈*nat* *env* ∈ *list*(*A*)
shows
is_name2(##*A*,*x*,*y*) $\longleftrightarrow$ *A*, *env* $\models$ *name2_fm*(*a*,*b*)
⟨*proof*⟩

definition
is_cond_of :: (*i*⇒*o*)⇒*i*⇒*i*⇒*o* **where**
is_cond_of(*M*,*x*,*t4*) ≡ *is_hcomp*(*M*,*is_snd*(*M*),*is_snd_snd*(*M*),*x*,*t4*)

definition
cond_of_fm :: [*i*,*i*] ⇒ *i* **where**
cond_of_fm(*x*,*t4*) ≡ *hcomp_fm*(*snd_fm*,*snd_snd_fm*,*x*,*t4*)

lemma *sats_cond_of_fm* :
⟦ *x* ∈ *nat*; *y* ∈ *nat*;*env* ∈ *list*(*A*) ⟧ ⟹
(*A*, *env* $\models$ *cond_of_fm*(*x*,*y*)) $\longleftrightarrow$ *is_cond_of*(##*A*, *nth*(*x*,*env*), *nth*(*y*,*env*))
⟨*proof*⟩

lemma *is_cond_of_iff_sats* [*iff_sats*]:
assumes
nth(*a*,*env*) = *x* *nth*(*b*,*env*) = *y* *a*∈*nat* *b*∈*nat* *env* ∈ *list*(*A*)
shows
is_cond_of(##*A*,*x*,*y*) ⟷ *A*, *env* ⊨ *cond_of_fm*(*a*,*b*)
⟨*proof*⟩

lemma *components_type*[*TC*] :
assumes *a*∈*nat* *b*∈*nat*
shows
ftype_fm(*a*,*b*)∈*formula*
name1_fm(*a*,*b*)∈*formula*
name2_fm(*a*,*b*)∈*formula*
cond_of_fm(*a*,*b*)∈*formula*
⟨*proof*⟩

lemmas *components_iff_sats* = *is_ftype_iff_sats* *is_name1_iff_sats* *is_name2_iff_sats*
is_cond_of_iff_sats

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

definition
is_eclose_n :: [*i*⇒*o*,[*i*⇒*o*,*i*,*i*]⇒*o*,*i*,*i*] ⇒ *o* **where**
is_eclose_n(*N*,*is_name*,*en*,*t*) ≡
∃ *n1*[*N*].∃ *s1*[*N*]. *is_name*(*N*,*t*,*n1*) ∧ *is_singleton*(*N*,*n1*,*s1*) ∧ *is_eclose*(*N*,*s1*,*en*)

definition
eclose_n1_fm :: [*i*,*i*] ⇒ *i* **where**
eclose_n1_fm(*m*,*t*) ≡ *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*] ⇒ *i* **where**
eclose_n2_fm(*m*,*t*) ≡ *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*⇒*o*,*i*,*i*] ⇒ *o* **where**
is_ecloseN(*N*,*t*,*en*) ≡ ∃ *en1*[*N*].∃ *en2*[*N*].
is_eclose_n(*N*,*is_name1*,*en1*,*t*) ∧ *is_eclose_n*(*N*,*is_name2*,*en2*,*t*)∧
union(*N*,*en1*,*en2*,*en*)

definition
ecloseN_fm :: [*i*,*i*] ⇒ *i* **where**
ecloseN_fm(*en*,*t*) ≡ *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*] :
⟦ *en* ∈ *nat* ; *t* ∈ *nat* ⟧ ⟹ *ecloseN_fm*(*en*,*t*) ∈ *formula*
⟨*proof*⟩

lemma *sats_ecloseN_fm* [*simp*]:
⟦ *en* ∈ *nat*; *t* ∈ *nat* ; *env* ∈ *list*(*A*) ⟧
⟹ (*A*, *env* ⊨ *ecloseN_fm*(*en*,*t*)) ⟷ *is_ecloseN*(##*A*,*nth*(*t*,*env*),*nth*(*en*,*env*))
⟨*proof*⟩

lemma *is_ecloseN_iff_sats* [*iff_sats*]:
⟦ *nth*(*en*, *env*) = *ena*; *nth*(*t*, *env*) = *ta*; *en* ∈ *nat*; *t* ∈ *nat* ; *env* ∈ *list*(*A*) ⟧
⟹ *is_ecloseN*(##*A*,*ta*,*ena*) ⟷ *A*, *env* ⊨ *ecloseN_fm*(*en*,*t*)
⟨*proof*⟩

definition
frecR :: *i* ⇒ *i* ⇒ *o* **where**
frecR(*x*,*y*) ≡
(*ftype*(*x*) = *1* ∧ *ftype*(*y*) = *0*
∧ (*name1*(*x*) ∈ *domain*(*name1*(*y*)) ∪ *domain*(*name2*(*y*)) ∧ (*name2*(*x*) = *name1*(*y*) ∨ *name2*(*x*) = *name2*(*y*))))
∨ (*ftype*(*x*) = *0* ∧ *ftype*(*y*) = *1* ∧ *name1*(*x*) = *name1*(*y*) ∧ *name2*(*x*) ∈ *domain*(*name2*(*y*)))

lemma *frecR_ftypeD* :
assumes *frecR*(*x*,*y*)
shows (*ftype*(*x*) = *0* ∧ *ftype*(*y*) = *1*) ∨ (*ftype*(*x*) = *1* ∧ *ftype*(*y*) = *0*)
⟨*proof*⟩

lemma *frecRI1*: *s* ∈ *domain*(*n1*) ∨ *s* ∈ *domain*(*n2*) ⟹ *frecR*(⟨*1*, *s*, *n1*, *q*⟩, ⟨*0*, *n1*, *n2*, *q'*⟩)
⟨*proof*⟩

lemma *frecRI1'*: *s* ∈ *domain*(*n1*) ∪ *domain*(*n2*) ⟹ *frecR*(⟨*1*, *s*, *n1*, *q*⟩, ⟨*0*, *n1*, *n2*, *q'*⟩)
⟨*proof*⟩

lemma *frecRI2*: *s* ∈ *domain*(*n1*) ∨ *s* ∈ *domain*(*n2*) ⟹ *frecR*(⟨*1*, *s*, *n2*, *q*⟩, ⟨*0*, *n1*, *n2*, *q'*⟩)
⟨*proof*⟩

lemma *frecRI2'*: *s* ∈ *domain*(*n1*) ∪ *domain*(*n2*) ⟹ *frecR*(⟨*1*, *s*, *n2*, *q*⟩, ⟨*0*, *n1*, *n2*, *q'*⟩)
⟨*proof*⟩

lemma *frecRI3*: ⟨*s*, *r*⟩ ∈ *n2* ⟹ *frecR*(⟨*0*, *n1*, *s*, *q*⟩, ⟨*1*, *n1*, *n2*, *q'*⟩)
⟨*proof*⟩

lemma *frecRI3'*: *s* ∈ *domain*(*n2*) ⟹ *frecR*(⟨*0*, *n1*, *s*, *q*⟩, ⟨*1*, *n1*, *n2*, *q'*⟩)

⟨*proof*⟩

lemma *frecR_D1* :
frecR(*x*,*y*) ⟹ *ftype*(*y*) = *0* ⟹ *ftype*(*x*) = *1* ∧
(*name1*(*x*) ∈ *domain*(*name1*(*y*)) ∪ *domain*(*name2*(*y*)) ∧ (*name2*(*x*) = *name1*(*y*)
∨ *name2*(*x*) = *name2*(*y*)))
⟨*proof*⟩

lemma *frecR_D2* :
frecR(*x*,*y*) ⟹ *ftype*(*y*) = *1* ⟹ *ftype*(*x*) = *0* ∧
ftype(*x*) = *0* ∧ *ftype*(*y*) = *1* ∧ *name1*(*x*) = *name1*(*y*) ∧ *name2*(*x*) ∈
domain(*name2*(*y*))
⟨*proof*⟩

lemma *frecR_DI* :
assumes *frecR*(⟨*a*,*b*,*c*,*d*⟩,⟨*ftype*(*y*),*name1*(*y*),*name2*(*y*),*cond_of*(*y*)⟩)
shows *frecR*(⟨*a*,*b*,*c*,*d*⟩,*y*)
⟨*proof*⟩

⟨*ML*⟩

schematic_goal *sats_frecR_fm_auto*:
assumes
i∈*nat* *j*∈*nat* *env*∈*list*(*A*)
shows
is_frecR(##*A*,*nth*(*i*,*env*),*nth*(*j*,*env*)) ⟷ *A*, *env* ⊨ *?fr_fm*(*i*,*j*)
⟨*proof*⟩

⟨*ML*⟩

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

lemma *eq_ftypep_not_frecrR*:
assumes *ftype*(*x*) = *ftype*(*y*)
shows ¬ *frecR*(*x*,*y*)
⟨*proof*⟩

definition
rank_names :: *i* ⇒ *i* **where**
rank_names(*x*) ≡ *max*(*rank*(*name1*(*x*)),*rank*(*name2*(*x*)))

lemma *rank_names_types* [*TC*]:
shows *Ord*(*rank_names*(*x*))
⟨*proof*⟩

definition
mtype_form :: *i* ⇒ *i* **where**
mtype_form(*x*) ≡ *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*$(x) \in 3$
⟨*proof*⟩

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

definition
Γ :: $i \Rightarrow i$ **where**
$\Gamma(x) = 3 ** rank\_names(x) ++ type\_form(x)$

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

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

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

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

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

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

lemma *core_induction_aux*:

fixes *A1 A2* :: *i*
assumes
Transset(A1)
$\bigwedge\tau\ \vartheta\ p.\ \ p \in A2 \Longrightarrow [\![\bigwedge q\ \sigma.\ [\![\ q{\in}A2\ ;\ \sigma{\in}domain(\vartheta)]\!] \Longrightarrow Q(0,\tau,\sigma,q)]\!] \Longrightarrow Q(1,\tau,\vartheta,p)$
$\bigwedge\tau\ \vartheta\ p.\ \ p \in A2 \Longrightarrow [\![\bigwedge q\ \sigma.\ [\![\ q{\in}A2\ ;\ \sigma{\in}domain(\tau) \cup domain(\vartheta)]\!] \Longrightarrow Q(1,\sigma,\tau,q) \wedge Q(1,\sigma,\vartheta,q)]\!] \Longrightarrow Q(0,\tau,\vartheta,p)$
shows $a{\in}2{\times}A1{\times}A1{\times}A2 \Longrightarrow Q(ftype(a),name1(a),name2(a),cond\_of(a))$
⟨*proof*⟩

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

lemma *frecrel_fst_snd*:
$frecrel(A) = \{z \in A{\times}A\ .$
$ftype(fst(z)) = 1 \wedge$
$ftype(snd(z)) = 0 \wedge name1(fst(z)) \in domain(name1(snd(z))) \cup domain(name2(snd(z))) \wedge$
$(name2(fst(z)) = name1(snd(z)) \vee name2(fst(z)) = name2(snd(z)))$
$\vee\ (ftype(fst(z)) = 0 \wedge$
$ftype(snd(z)) = 1 \wedge\ \ name1(fst(z)) = name1(snd(z)) \wedge name2(fst(z)) \in domain(name2(snd(z))))\}$
⟨*proof*⟩

end
theory *FrecR_Arities*
imports
FrecR
begin

context
notes *FOL_arities*[*simp*]
begin

⟨*ML*⟩
lemma *arity_fst_fm* [*arity*] :
$[\![x{\in}nat\ ;\ t{\in}nat]\!] \Longrightarrow arity(fst\_fm(x,t)) = succ(x) \cup succ(t)$
⟨*proof*⟩

⟨*ML*⟩
lemma *arity_snd_fm* [*arity*] :
$[\![x{\in}nat\ ;\ t{\in}nat]\!] \Longrightarrow arity(snd\_fm(x,t)) = succ(x) \cup succ(t)$
⟨*proof*⟩

lemma *arity_snd_snd_fm* [*arity*] :
$[\![x{\in}nat\ ;\ t{\in}nat]\!] \Longrightarrow arity(snd\_snd\_fm(x,t)) = succ(x) \cup succ(t)$
⟨*proof*⟩

lemma *arity_ftype_fm* [*arity*] :

$[\![x{\in}nat\ ;\ t{\in}nat]\!] \Longrightarrow arity(ftype\_fm(x,t)) = succ(x) \cup succ(t)$
⟨*proof*⟩

lemma *arity_name1_fm* [*arity*] :
$[\![x{\in}nat\ ;\ t{\in}nat]\!] \Longrightarrow arity(name1\_fm(x,t)) = succ(x) \cup succ(t)$
⟨*proof*⟩

lemma *arity_name2_fm* [*arity*] :
$[\![x{\in}nat\ ;\ t{\in}nat]\!] \Longrightarrow arity(name2\_fm(x,t)) = succ(x) \cup succ(t)$
⟨*proof*⟩

lemma *arity_cond_of_fm* [*arity*] :
$[\![x{\in}nat\ ;\ t{\in}nat]\!] \Longrightarrow arity(cond\_of\_fm(x,t)) = succ(x) \cup succ(t)$
⟨*proof*⟩

lemma *arity_eclose_n1_fm* [*arity*] :
$[\![x{\in}nat\ ;\ t{\in}nat]\!] \Longrightarrow arity(eclose\_n1\_fm(x,t)) = succ(x) \cup succ(t)$
⟨*proof*⟩

lemma *arity_eclose_n2_fm* [*arity*] :
$[\![x{\in}nat\ ;\ t{\in}nat]\!] \Longrightarrow arity(eclose\_n2\_fm(x,t)) = succ(x) \cup succ(t)$
⟨*proof*⟩

lemma *arity_ecloseN_fm* [*arity*] :
$[\![x{\in}nat\ ;\ t{\in}nat]\!] \Longrightarrow arity(ecloseN\_fm(x,t)) = succ(x) \cup succ(t)$
⟨*proof*⟩

lemma *arity_frecR_fm* [*arity*]:
$[\![a{\in}nat;b{\in}nat]\!] \Longrightarrow arity(frecR\_fm(a,b)) = succ(a) \cup succ(b)$
⟨*proof*⟩

end — *FOL_arities*

end

5 Concepts involved in instances of Replacement

theory *Fm_Definitions*
imports
Transitive_Models.Renaming_Auto
Transitive_Models.Aleph_Relative
FrecR_Arities
begin

no_notation *Aleph* (‹ℵ_› [*90*] *90*)

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

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

⟨*ML*⟩
definition *is_minimum′* **where**
is_minimum′(*M*,*R*,*X*,*u*) ≡ (*M*(*u*) ∧ *u* ∈ *X* ∧ (∀ *v*[*M*]. ∃ *a*[*M*]. (*v* ∈ *X* ⟶ *v* ≠ *u* ⟶ *a* ∈ *R*) ∧ *pair*(*M*, *u*, *v*, *a*))) ∧
(∃ *x*[*M*].
(*M*(*x*) ∧ *x* ∈ *X* ∧ (∀ *v*[*M*]. ∃ *a*[*M*]. (*v* ∈ *X* ⟶ *v* ≠ *x* ⟶ *a* ∈ *R*) ∧ *pair*(*M*, *x*, *v*, *a*))) ∧
(∀ *y*[*M*]. *M*(*y*) ∧ *y* ∈ *X* ∧ (∀ *v*[*M*]. ∃ *a*[*M*]. (*v* ∈ *X* ⟶ *v* ≠ *y* ⟶ *a* ∈ *R*) ∧ *pair*(*M*, *y*, *v*, *a*)) ⟶ *y* = *x*)) ∨
¬ (∃ *x*[*M*]. (*M*(*x*) ∧ *x* ∈ *X* ∧ (∀ *v*[*M*]. ∃ *a*[*M*]. (*v* ∈ *X* ⟶ *v* ≠ *x* ⟶ *a* ∈ *R*) ∧ *pair*(*M*, *x*, *v*, *a*))) ∧
(∀ *y*[*M*]. *M*(*y*) ∧ *y* ∈ *X* ∧ (∀ *v*[*M*]. ∃ *a*[*M*]. (*v* ∈ *X* ⟶ *v* ≠ *y* ⟶ *a* ∈ *R*) ∧ *pair*(*M*, *y*, *v*, *a*)) ⟶ *y* = *x*)) ∧
empty(*M*, *u*)

⟨*ML*⟩

lemma *is_lambda_iff_sats*[*iff_sats*]:
assumes *is_F_iff_sats*:
!!*a0 a1 a2*.
[|*a0*∈*Aa*; *a1*∈*Aa*; *a2*∈*Aa*|]
==> *is_F*(*a1*, *a0*) ⟷ *sats*(*Aa*, *is_F_fm*, *Cons*(*a0*,*Cons*(*a1*,*Cons*(*a2*,*env*))))
shows
nth(*A*, *env*) = *Ab* ⟹
nth(*r*, *env*) = *ra* ⟹
A ∈ *nat* ⟹
r ∈ *nat* ⟹
env ∈ *list*(*Aa*) ⟹
is_lambda(##*Aa*, *Ab*, *is_F*, *ra*) ⟷ *Aa*, *env* ⊨ *lambda_fm*(*is_F_fm*,*A*, *r*)
⟨*proof*⟩
lemma *sats_is_wfrec_fm′*:
assumes *MH_iff_sats*:

!!*a0 a1 a2 a3 a4*.
[|*a0*∈*A*; *a1*∈*A*; *a2*∈*A*; *a3*∈*A*; *a4*∈*A*|]
==> *MH*(*a2*, *a1*, *a0*) ⟷ *sats*(*A*, *p*, *Cons*(*a0*,*Cons*(*a1*,*Cons*(*a2*,*Cons*(*a3*,*Cons*(*a4*,*env*))))))
shows
[|*x* ∈ *nat*; *y* ∈ *nat*; *z* ∈ *nat*; *env* ∈ *list*(*A*); *0* ∈ *A*|]
==> *sats*(*A*, *is_wfrec_fm*(*p*,*x*,*y*,*z*), *env*) ⟷
is_wfrec(##*A*, *MH*, *nth*(*x*,*env*), *nth*(*y*,*env*), *nth*(*z*,*env*))
⟨*proof*⟩

lemma *is_wfrec_iff_sats'*[*iff_sats*]:
assumes *MH_iff_sats*:
!!*a0 a1 a2 a3 a4*.
[|*a0*∈*Aa*; *a1*∈*Aa*; *a2*∈*Aa*; *a3*∈*Aa*; *a4*∈*Aa*|]
==> *MH*(*a2*, *a1*, *a0*) ⟷ *sats*(*Aa*, *p*, *Cons*(*a0*,*Cons*(*a1*,*Cons*(*a2*,*Cons*(*a3*,*Cons*(*a4*,*env*))))))
nth(*x*, *env*) = *xx* *nth*(*y*, *env*) = *yy* *nth*(*z*, *env*) = *zz*
x ∈ *nat* *y* ∈ *nat* *z* ∈ *nat* *env* ∈ *list*(*Aa*) *0* ∈ *Aa*
shows
is_wfrec(##*Aa*, *MH*, *xx*, *yy*, *zz*) ⟷ *Aa*, *env* ⊨ *is_wfrec_fm*(*p*,*x*,*y*,*z*)
⟨*proof*⟩

lemma *is_wfrec_on_iff_sats*[*iff_sats*]:
assumes *MH_iff_sats*:
!!*a0 a1 a2 a3 a4*.
[|*a0*∈*Aa*; *a1*∈*Aa*; *a2*∈*Aa*; *a3*∈*Aa*; *a4*∈*Aa*|]
==> *MH*(*a2*, *a1*, *a0*) ⟷ *sats*(*Aa*, *p*, *Cons*(*a0*,*Cons*(*a1*,*Cons*(*a2*,*Cons*(*a3*,*Cons*(*a4*,*env*))))))
shows
nth(*x*, *env*) = *xx* ⟹
nth(*y*, *env*) = *yy* ⟹
nth(*z*, *env*) = *zz* ⟹
x ∈ *nat* ⟹
y ∈ *nat* ⟹
z ∈ *nat* ⟹
env ∈ *list*(*Aa*) ⟹
0 ∈ *Aa* ⟹ *is_wfrec_on*(##*Aa*, *MH*, *aa*,*xx*, *yy*, *zz*) ⟷ *Aa*, *env* ⊨ *is_wfrec_fm*(*p*,*x*,*y*,*z*)
⟨*proof*⟩

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* ⇒ *i* **where**
rcheck(*x*) ≡ *Memrel*(*eclose*({*x*}))^+

⟨*ML*⟩
definition
PHcheck :: [*i*⇒*o*,*i*,*i*,*i*,*i*] ⇒ *o* **where**
PHcheck(*M*,*o*,*f*,*y*,*p*) ≡ *M*(*p*) ∧ (∃*fy*[*M*]. *fun_apply*(*M*,*f*,*y*,*fy*) ∧ *pair*(*M*,*fy*,*o*,*p*))

⟨*ML*⟩
definition
is_Hcheck :: [*i*⇒*o*,*i*,*i*,*i*,*i*] ⇒ *o* **where**
is_Hcheck(*M*,*o*,*z*,*f*,*hc*) ≡ *is_Replace*(*M*,*z*,*PHcheck*(*M*,*o*,*f*),*hc*)

⟨*ML*⟩

lemma *arity_is_Hcheck_fm*:
assumes *m*∈*nat* *n*∈*nat* *p*∈*nat* *o*∈*nat*
shows *arity*(*is_Hcheck_fm*(*m*,*n*,*p*,*o*)) = *succ*(*o*) ∪ *succ*(*n*) ∪ *succ*(*p*) ∪ *succ*(*m*)
⟨*proof*⟩
definition
is_check :: [*i*⇒*o*,*i*,*i*,*i*] ⇒ *o* **where**
is_check(*M*,*o*,*x*,*z*) ≡ ∃ *rch*[*M*]. *is_rcheck*(*M*,*x*,*rch*) ∧
is_wfrec(*M*,*is_Hcheck*(*M*,*o*),*rch*,*x*,*z*)

— Finally, we internalize the formula.
definition
check_fm :: [*i*,*i*,*i*] ⇒ *i* **where**
check_fm(*o*,*x*,*z*) ≡ *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*∈*nat* ⟹ *o*∈*nat* ⟹ *z*∈*nat* ⟹ *check_fm*(*x*,*o*,*z*) ∈ *formula*
⟨*proof*⟩

lemma *sats_check_fm* :
assumes
o∈*nat* *x*∈*nat* *z*∈*nat* *env*∈*list*(*M*) 0∈*M*
shows
(*M* , *env* ⊨ *check_fm*(*o*,*x*,*z*)) ⟷ *is_check*(##*M*,*nth*(*o*,*env*),*nth*(*x*,*env*),*nth*(*z*,*env*))
⟨*proof*⟩

lemma *iff_sats_check_fm*[*iff_sats*] :
assumes
nth(*o*, *env*) = *oa* *nth*(*x*, *env*) = *xa* *nth*(*z*, *env*) = *za* *o* ∈ *nat* *x* ∈ *nat* *z* ∈ *nat* *env* ∈ *list*(*A*) 0 ∈ *A*
shows *is_check*(##*A*, *oa*,*xa*, *za*) ⟷ *A*, *env* ⊨ *check_fm*(*o*,*x*,*z*)
⟨*proof*⟩

lemma *arity_check_fm*[*arity*]:
assumes *m*∈*nat* *n*∈*nat* *o*∈*nat*
shows *arity*(*check_fm*(*m*,*n*,*o*)) = *succ*(*o*) ∪ *succ*(*n*) ∪ *succ*(*m*)
⟨*proof*⟩

notation *check_fm* (‹·_ ^{v} _ *is* _·›)

— 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$(A,l,q,p) \equiv \exists\, qp[A].\ (pair(A,q,p,qp) \wedge qp \in l)$

⟨*ML*⟩

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

5.1 Formulas used to prove some generic instances.

definition *ϱ_repl* :: $i \Rightarrow i$ **where**
ϱ_repl$(l) \equiv$ *rsum*$(\{\langle 0,\ 1\rangle, \langle 1,\ 0\rangle\},\ id(l),\ 2,\ 3,\ l)$

lemma *f_type* : $\{\langle 0,\ 1\rangle, \langle 1,\ 0\rangle\} \in 2 \to 3$
⟨*proof*⟩
hide_fact *Internalize.sum_type*

lemma *ren_type* :
assumes $l \in nat$
shows *ϱ_repl*$(l) : 2 +_\omega l \to 3 +_\omega l$
⟨*proof*⟩

definition *Lambda_in_M_fm* **where** [*simp*]:*Lambda_in_M_fm*$(\varphi,len) \equiv$
$\cdot(\cdot\exists\cdot$*pair_fm*$(1,\ 0,\ 2) \wedge$
ren(φ) ‘ $(2 +_\omega len)$ ‘ $(3 +_\omega len)$ ‘ *ϱ_repl*(len) $\cdot\cdot) \wedge \cdot 0 \in len +_\omega 2\cdot\cdot$

lemma *Lambda_in_M_fm_type*[*TC*]: $\varphi \in formula \Longrightarrow len \in nat \Longrightarrow$ *Lambda_in_M_fm*(φ,len) $\in formula$
⟨*proof*⟩

definition *ϱ_pair_repl* :: $i \Rightarrow i$ **where**
ϱ_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*$(\varphi,len) \equiv$
$\cdot(\cdot\exists\cdot$*pair_fm*$(1,\ 0,\ 2) \wedge$
ren$((\cdot\exists(\cdot\exists\cdot\cdot fst(2)\ is\ 0\cdot \wedge \cdot\cdot snd(2)\ is\ 1\cdot \wedge ren(\varphi)$ ‘ $(3 +_\omega len)$ ‘ $(4 +_\omega len)$ ‘ *ϱ_pair_repl*(len) $\cdots)\cdot))$ ‘ $(2 +_\omega len)$ ‘
$(3 +_\omega len)$ ‘
ϱ_repl(len) $\cdot\cdot) \wedge$
$\cdot 0 \in len +_\omega 2\cdot\cdot$

lemma *f_type′* : $\{\langle 0,0\ \rangle, \langle 1,\ 1\rangle, \langle 2,\ 3\rangle\} \in 3 \to 4$
⟨*proof*⟩

lemma *ren_type′* :
assumes $l \in nat$
shows *ϱ_pair_repl*$(l) : 3 +_\omega l \to 4 +_\omega l$
⟨*proof*⟩

lemma *LambdaPair_in_M_fm_type*[*TC*]: φ∈*formula* ⟹ *len*∈*nat* ⟹ *Lambda-Pair_in_M_fm*(φ,*len*) ∈*formula*
⟨*proof*⟩

5.2 The relation *frecrel*

definition
frecrelP :: [*i*⇒*o*,*i*] ⇒ *o* **where**
frecrelP(*M*,*xy*) ≡ (∃ *x*[*M*]. ∃ *y*[*M*]. *pair*(*M*,*x*,*y*,*xy*) ∧ *is_frecR*(*M*,*x*,*y*))

⟨*ML*⟩

definition
is_frecrel :: [*i*⇒*o*,*i*,*i*] ⇒ *o* **where**
is_frecrel(*M*,*A*,*r*) ≡ ∃ *A2*[*M*]. *cartprod*(*M*,*A*,*A*,*A2*) ∧ *is_Collect*(*M*,*A2*, *frecrelP*(*M*) ,*r*)

⟨*ML*⟩

definition
names_below :: *i* ⇒ *i* ⇒ *i* **where**
names_below(*P*,*x*) ≡ *2*×*ecloseN*(*x*)×*ecloseN*(*x*)×*P*

lemma *names_belowsD*:
assumes *x* ∈ *names_below*(*P*,*z*)
obtains *f n1 n2 p* **where**
x = ⟨*f*,*n1*,*n2*,*p*⟩ *f*∈*2 n1*∈*ecloseN*(*z*) *n2*∈*ecloseN*(*z*) *p*∈*P*
⟨*proof*⟩

⟨*ML*⟩

lemma *number2_iff* :
(*A*)(*c*) ⟹ *number2*(*A*,*c*) ⟷ (∃ *b*[*A*]. ∃ *a*[*A*]. *successor*(*A*, *b*, *c*) ∧ *successor*(*A*, *a*, *b*) ∧ *empty*(*A*, *a*))
⟨*proof*⟩
⟨*ML*⟩

definition
is_tuple :: [*i*⇒*o*,*i*,*i*,*i*,*i*,*i*] ⇒ *o* **where**
is_tuple(*M*,*z*,*t1*,*t2*,*p*,*t*) ≡ ∃ *t1t2p*[*M*]. ∃ *t2p*[*M*]. *pair*(*M*,*t2*,*p*,*t2p*) ∧ *pair*(*M*,*t1*,*t2p*,*t1t2p*) ∧
pair(*M*,*z*,*t1t2p*,*t*)

⟨*ML*⟩

5.3 Definition of Forces

5.3.1 Definition of *forces* for equality and membership

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

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

⟨*ML*⟩

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

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

⟨*ML*⟩
lemma *arity_eq_case_fm*[*arity*]:
assumes
n1∈*nat n2*∈*nat p*∈*nat P*∈*nat leq*∈*nat f*∈*nat*
shows
$arity(eq\_case\_fm(n1,n2,p,P,leq,f)) =$
$succ(n1) \cup succ(n2) \cup succ(p) \cup succ(P) \cup succ(leq) \cup succ(f)$
⟨*proof*⟩

⟨*ML*⟩
lemma *arity_mem_case_fm*[*arity*] :
assumes
n1∈*nat n2*∈*nat p*∈*nat P*∈*nat leq*∈*nat f*∈*nat*
shows
$arity(mem\_case\_fm(n1,n2,p,P,leq,f)) =$
$succ(n1) \cup succ(n2) \cup succ(p) \cup succ(P) \cup succ(leq) \cup succ(f)$
⟨*proof*⟩

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

⟨*ML*⟩

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

$is\_Hfrc\_at(M,P,leq,fnnc,f,b) \equiv$
$(empty(M,b) \wedge \neg\ is\_Hfrc(M,P,leq,fnnc,f))$
$\vee\ (number1(M,b) \wedge is\_Hfrc(M,P,leq,fnnc,f))$

⟨*ML*⟩

lemma *arity_Hfrc_fm*[*arity*] :
assumes
$P{\in}nat\ leq{\in}nat\ fnnc{\in}nat\ f{\in}nat$
shows
$arity(Hfrc\_fm(P,leq,fnnc,f)) = succ(P) \cup succ(leq) \cup succ(fnnc) \cup succ(f)$
⟨*proof*⟩

⟨*ML*⟩

5.3.2 The well-founded relation *forcerel*

definition
$forcerel :: i \Rightarrow i \Rightarrow i$ **where**
$forcerel(P,x) \equiv frecrel(names\_below(P,x))^{+}$

definition
$is\_forcerel :: [i{\Rightarrow}o,i,i,i] \Rightarrow o$ **where**
$is\_forcerel(M,P,x,z) \equiv \exists\, r[M].\ \exists\, nb[M].\ tran\_closure(M,r,z) \wedge$
$(is\_names\_below(M,P,x,nb) \wedge is\_frecrel(M,nb,r))$
⟨*ML*⟩

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**
$is\_frc\_at(M,P,leq,x,z) \equiv \exists\, r[M].\ is\_forcerel(M,P,x,r) \wedge$
$is\_wfrec(M,is\_Hfrc\_at(M,P,leq),r,x,z)$

definition
$frc\_at\_fm :: [i,i,i,i] \Rightarrow i$ **where**
$frc\_at\_fm(p,l,x,z) \equiv Exists(And(is\_forcerel\_fm(succ(p),succ(x),0),$
$is\_wfrec\_fm(Hfrc\_at\_fm(6{+_\omega}p,6{+_\omega}l,2,1,0),0,succ(x),succ(z))))$

lemma *frc_at_fm_type* [*TC*] :
$[\![p{\in}nat;l{\in}nat;x{\in}nat;z{\in}nat]\!] \Longrightarrow frc\_at\_fm(p,l,x,z){\in}formula$
⟨*proof*⟩

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*⟩

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*⟩

lemma *frc_at_fm_iff_sats*:
assumes $nth(i,env) = w$ $nth(j,env) = x$ $nth(k,env) = y$ $nth(l,env) = z$
$i \in nat$ $j \in nat$ $k \in nat$ $l{\in}nat$ $env \in list(A)$ $k{<}length(env)$ $l{<}length(env)$
shows $is\_frc\_at(\#\#A,\ w,\ x,\ y,z) \longleftrightarrow (A\ ,\ env \models frc\_at\_fm(i,j,k,l))$
⟨*proof*⟩

declare *frc_at_fm_iff_sats* [*iff_sats*]

definition
$forces\_eq' :: [i,i,i,i,i] \Rightarrow o$ **where**
$forces\_eq'(P,l,p,t1,t2) \equiv frc\_at(P,l,\langle 0,t1,t2,p\rangle) = 1$

definition
$forces\_mem' :: [i,i,i,i,i] \Rightarrow o$ **where**
$forces\_mem'(P,l,p,t1,t2) \equiv frc\_at(P,l,\langle 1,t1,t2,p\rangle) = 1$

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

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′ :: $[i⇒o,i,i,i,i,i] ⇒ o$ **where**
is_forces_neq′$(M,P,l,p,t1,t2) \equiv$
$\neg (\exists\, q[M].\ q∈P \wedge (\exists\, qp[M].\ pair(M,q,p,qp) \wedge qp∈l \wedge is\_forces\_eq'(M,P,l,q,t1,t2)))$

definition
is_forces_nmem′ :: $[i⇒o,i,i,i,i,i] ⇒ o$ **where**
is_forces_nmem′$(M,P,l,p,t1,t2) \equiv$
$\neg (\exists\, q[M].\ \exists\, qp[M].\ q∈P \wedge pair(M,q,p,qp) \wedge qp∈l \wedge is\_forces\_mem'(M,P,l,q,t1,t2))$

⟨*ML*⟩

context
notes *Un_assoc*[*simp*] *Un_trasposition_aux2*[*simp*]
begin
⟨*ML*⟩
end

5.5 Forcing for general formulas

definition
ren_forces_nand :: $i⇒i$ **where**
ren_forces_nand$(\varphi) \equiv Exists(And(Equal(0,1),iterates(\lambda p.\ incr\_bv(p)\text{`}1\ ,\ 2,\ \varphi)))$

lemma *ren_forces_nand_type*[*TC*] :
$\varphi∈formula \Longrightarrow ren\_forces\_nand(\varphi) \in formula$
⟨*proof*⟩

lemma *arity_ren_forces_nand* :
assumes $\varphi∈formula$
shows $arity(ren\_forces\_nand(\varphi)) \leq succ(arity(\varphi))$
⟨*proof*⟩

lemma *sats_ren_forces_nand*:
$[q,P,leq,o,p]$ @ $env \in list(M) \Longrightarrow \varphi∈formula \Longrightarrow$
$(M,\ [q,p,P,leq,o]$ @ $env \models ren\_forces\_nand(\varphi)) \longleftrightarrow (M,\ [q,P,leq,o]$ @ $env \models \varphi)$
⟨*proof*⟩

definition
ren_forces_forall :: $i⇒i$ **where**
ren_forces_forall$(\varphi) \equiv$
$Exists(Exists(Exists(Exists(Exists($
$And(Equal(0,6),And(Equal(1,7),And(Equal(2,8),And(Equal(3,9),$
$And(Equal(4,5),iterates(\lambda p.\ incr\_bv(p)\text{`}5\ ,\ 5,\ \varphi)))))))))))$

lemma *arity_ren_forces_all* :
assumes $\varphi∈formula$
shows $arity(ren\_forces\_forall(\varphi)) = 5 \cup arity(\varphi)$

⟨*proof*⟩

lemma *ren_forces_forall_type*[*TC*] :
  φ∈*formula* ⟹ *ren_forces_forall*(φ) ∈*formula*
  ⟨*proof*⟩

lemma *sats_ren_forces_forall* :
  [*x*,*P*,*leq*,*o*,*p*] @ *env* ∈ *list*(*M*) ⟹ φ∈*formula* ⟹
    (*M*, [*x*,*p*,*P*,*leq*,*o*] @ *env* ⊨ *ren_forces_forall*(φ)) ⟷ (*M*, [*p*,*P*,*leq*,*o*,*x*] @ *env*
⊨ φ)
  ⟨*proof*⟩

5.5.1 The primitive recursion

consts *forces′* :: *i*⇒*i*
primrec
  forces′(*Member*(*x*,*y*)) = *forces_mem_fm*(1,2,0,*x*+$_\omega$4,*y*+$_\omega$4)
  forces′(*Equal*(*x*,*y*)) = *forces_eq_fm*(1,2,0,*x*+$_\omega$4,*y*+$_\omega$4)
  forces′(*Nand*(*p*,*q*)) =
     Neg(*Exists*(*And*(*Member*(0,2),*And*(*is_leq_fm*(3,0,1),*And*(*ren_forces_nand*(*forces′*(*p*)),
                                  ren_forces_nand(*forces′*(*q*)))))))
  forces′(*Forall*(*p*)) = *Forall*(*ren_forces_forall*(*forces′*(*p*)))

definition
  forces :: *i*⇒*i* **where**
  forces(φ) ≡ *And*(*Member*(0,1),*forces′*(φ))

lemma *forces′_type* [*TC*]: φ∈*formula* ⟹ *forces′*(φ) ∈ *formula*
  ⟨*proof*⟩

lemma *forces_type*[*TC*] : φ∈*formula* ⟹ *forces*(φ) ∈ *formula*
  ⟨*proof*⟩

5.6 The arity of *forces*

lemma *arity_forces_at*:
  assumes *x* ∈ *nat* *y* ∈ *nat*
  shows *arity*(*forces*(*Member*(*x*, *y*))) = (*succ*(*x*) ∪ *succ*(*y*)) +$_\omega$ 4
    arity(*forces*(*Equal*(*x*, *y*))) = (*succ*(*x*) ∪ *succ*(*y*)) +$_\omega$ 4
  ⟨*proof*⟩

lemma *arity_forces′*:
  assumes φ∈*formula*
  shows *arity*(*forces′*(φ)) ≤ *arity*(φ) +$_\omega$ 4
  ⟨*proof*⟩

lemma *arity_forces* :
  assumes φ∈*formula*
  shows *arity*(*forces*(φ)) ≤ 4+$_\omega$*arity*(φ)

⟨*proof*⟩

lemma *arity_forces_le* :
assumes φ∈*formula* *n*∈*nat* *arity*(φ) ≤ *n*
shows *arity*(*forces*(φ)) ≤ *4*+$_\omega$*n*
⟨*proof*⟩

definition *rename_split_fm* **where**
rename_split_fm(φ) ≡ (·∃(·∃(·∃(·∃(·∃··*snd*(*9*) *is 0*· ∧ ··*fst*(*9*) *is 4*· ∧ ··1=*11*·
∧
··2=*12*· ∧ ··3=*13*· ∧ ··5=*7*· ∧
(λ*p*. *incr_bv*(*p*)‘*6*)^*8*(*forces*(φ)) ·····)·)·)·)·)·)

lemma *rename_split_fm_type*[*TC*]: φ∈*formula* ⟹ *rename_split_fm*(φ)∈*formula*
⟨*proof*⟩

schematic_goal *arity_rename_split_fm*: φ∈*formula* ⟹ *arity*(*rename_split_fm*(φ))
= *?m*
⟨*proof*⟩

lemma *arity_rename_split_fm_le*:
assumes φ∈*formula*
shows *arity*(*rename_split_fm*(φ)) ≤ *8* ∪ (*6* +$_\omega$ *arity*(φ))
⟨*proof*⟩

definition *body_ground_repl_fm* **where**
body_ground_repl_fm(φ) ≡ (·∃(·∃·*is_Vset_fm*(*2*, *0*) ∧ ··1 ∈ *0*· ∧ *rename_split_fm*(φ)
···)·)

lemma *body_ground_repl_fm_type*[*TC*]: φ∈*formula* ⟹ *body_ground_repl_fm*(φ)∈*formula*
⟨*proof*⟩

lemma *arity_body_ground_repl_fm_le*:
notes *le_trans*[*trans*]
assumes φ∈*formula*
shows *arity*(*body_ground_repl_fm*(φ)) ≤ *6* ∪ (*arity*(φ) +$_\omega$ *4*)
⟨*proof*⟩

definition *ground_repl_fm* **where**
ground_repl_fm(φ) ≡ *least_fm*(*body_ground_repl_fm*(φ), *1*)

lemma *ground_repl_fm_type*[*TC*]:
φ∈*formula* ⟹ *ground_repl_fm*(φ) ∈ *formula*
⟨*proof*⟩

lemma *arity_ground_repl_fm*:
assumes φ∈*formula*
shows *arity*(*ground_repl_fm*(φ)) ≤ *5* ∪ (*3* +$_\omega$ *arity*(φ))
⟨*proof*⟩

⟨*ML*⟩

definition *omap_wfrec_body* **where**
omap_wfrec_body(*A*,*r*) ≡ (·∃·*image_fm*(*2*, *0*, *1*) ∧ *pred_set_fm*(*9*+$_\omega$*A*, *3*,*9*+$_\omega$*r*, *0*) ··)

lemma *type_omap_wfrec_body_fm* :*A*∈*nat* ⟹ *r*∈*nat* ⟹ *omap_wfrec_body*(*A*,*r*)∈*formula*
⟨*proof*⟩

lemma *arity_omap_wfrec_aux* : *A*∈*nat* ⟹ *r*∈*nat* ⟹ *arity*(*omap_wfrec_body*(*A*,*r*)) = (*9*+$_\omega$*A*) ∪ (*9*+$_\omega$*r*)
⟨*proof*⟩

lemma *arity_omap_wfrec*: *A*∈*nat* ⟹ *r*∈*nat* ⟹
arity(*is_wfrec_fm*(*omap_wfrec_body*(*A*,*r*),*r*+$_\omega$*3*, *1*, *0*)) = (*4*+$_\omega$*A*) ∪ (*4*+$_\omega$*r*)
⟨*proof*⟩

lemma *arity_isordermap*: *A*∈*nat* ⟹ *r*∈*nat* ⟹*d*∈*nat*⟹
arity(*is_ordermap_fm*(*A*,*r*,*d*)) = *succ*(*d*) ∪ (*succ*(*A*) ∪ *succ*(*r*))
⟨*proof*⟩

lemma *arity_is_ordertype*: *A*∈*nat* ⟹ *r*∈*nat* ⟹*d*∈*nat*⟹
arity(*is_ordertype_fm*(*A*,*r*,*d*)) = *succ*(*d*) ∪ (*succ*(*A*) ∪ *succ*(*r*))
⟨*proof*⟩

lemma *arity_is_order_body*: *arity*(*is_order_body_fm*(*0*,*1*)) = *2*
⟨*proof*⟩

definition *H_order_pred* **where**
H_order_pred(*A*,*r*) ≡ λ*x f* . *f* ‘‘ *Order.pred*(*A*, *x*, *r*)

⟨*ML*⟩

definition *order_pred_wfrec_body* **where**
order_pred_wfrec_body(*M*,*A*,*r*,*z*,*x*) ≡ ∃ *y*[*M*].
pair(*M*, *x*, *y*, *z*) ∧
(∃ *f*[*M*].
(∀ *z*[*M*].
z ∈ *f* ⟷
(∃ *xa*[*M*].
∃ *y*[*M*].
∃ *xaa*[*M*].
∃ *sx*[*M*].
∃ *r_sx*[*M*].
∃ *f_r_sx*[*M*].
pair(*M*, *xa*, *y*, *z*) ∧
pair(*M*, *xa*, *x*, *xaa*) ∧
upair(*M*, *xa*, *xa*, *sx*) ∧

$pre\_image(M,\ r,\ sx,\ r\_sx) \wedge$
$restriction(M,\ f,\ r\_sx,\ f\_r\_sx) \wedge$
$xaa \in r \wedge (\exists\, a[M].\ image(M,\ f\_r\_sx,\ a,\ y) \wedge$
$pred\_set(M,\ A,\ xa,\ r,\ a)))) \wedge$
$(\exists\, a[M].\ image(M,\ f,\ a,\ y) \wedge pred\_set(M,\ A,\ x,\ r,\ a)))$

$\langle ML \rangle$

definition *ordtype_replacement_fm* **where** *ordtype_replacement_fm* $\equiv$ $(\cdot\exists\cdot$*is_order_body_fm*(*1*, *0*) $\wedge$ $\cdot\langle$*1,0*$\rangle$ *is 2* $\cdots)$
definition *wfrec_ordertype_fm* **where** *wfrec_ordertype_fm* $\equiv$ *order_pred_wfrec_body_fm*(*3*,*2*,*1*,*0*)
definition *replacement_is_aleph_fm* **where** *replacement_is_aleph_fm* $\equiv$ $\cdot\cdot$*0 is ordinal*$\cdot$ $\wedge$ $\cdot\aleph(0)$ *is 1*$\cdot\cdot$

definition
funspace_succ_rep_intf **where**
funspace_succ_rep_intf $\equiv \lambda p\ z\ n.\ \exists f\ b.\ p = <f,b> \ \&\ z = \{cons(<n,b>,\ f)\}$

$\langle ML \rangle$

definition *wfrec_Hfrc_at_fm* **where** *wfrec_Hfrc_at_fm* $\equiv$ $(\cdot\exists\cdot$*pair_fm*(*1*, *0*, *2*) $\wedge$ *is_wfrec_fm*(*Hfrc_at_fm*(*8*, *9*, *2*, *1*, *0*), *5*, *1*, *0*) $\cdot\cdot)$
definition *list_repl1_intf_fm* **where** *list_repl1_intf_fm* $\equiv$ $(\cdot\exists\cdot$*pair_fm*(*1*, *0*, *2*) $\wedge$ *is_wfrec_fm*(*iterates_MH_fm*(*list_functor_fm*(*13*, *1*, *0*), *10*, *2*, *1*, *0*), *3*, *1*, *0*) $\cdot\cdot)$
definition *list_repl2_intf_fm* **where** *list_repl2_intf_fm* $\equiv$ $\cdot\cdot$*0* $\in$ *4*$\cdot$ $\wedge$ *is_iterates_fm*(*list_functor_fm*(*13*, *1*, *0*), *3*, *0*, *1*) $\cdot$
definition *formula_repl2_intf_fm* **where** *formula_repl2_intf_fm* $\equiv$ $\cdot\cdot$*0* $\in$ *3*$\cdot$ $\wedge$ *is_iterates_fm*(*formula_functor_fm*(*1*, *0*), *2*, *0*, *1*) $\cdot$
definition *eclose_abs_fm* **where** *eclose_abs_fm* $\equiv$ $\cdot\cdot$*0* $\in$ *3*$\cdot$ $\wedge$ *is_iterates_fm*($\cdot\bigcup$ *1 is 0*$\cdot$, *2*, *0*, *1*) $\cdot$
definition *powapply_repl_fm* **where** *powapply_repl_fm* $\equiv$ *is_Powapply_fm*(*2*,*0*,*1*)
definition *wfrec_rank_fm* **where** *wfrec_rank_fm* $\equiv$ $(\cdot\exists\cdot$*pair_fm*(*1*, *0*, *2*) $\wedge$ *is_wfrec_fm*(*is_Hrank_fm*(*2*, *1*, *0*), *3*, *1*, *0*) $\cdot\cdot)$
definition *transrec_VFrom_fm* **where** *transrec_VFrom_fm* $\equiv$ $(\cdot\exists\cdot$*pair_fm*(*1*, *0*, *2*) $\wedge$ *is_wfrec_fm*(*is_HVfrom_fm*(*8*, *2*, *1*, *0*), *4*, *1*, *0*) $\cdot\cdot)$
definition *wfrec_Hcheck_fm* **where** *wfrec_Hcheck_fm* $\equiv$ $(\cdot\exists\cdot$*pair_fm*(*1*, *0*, *2*) $\wedge$ *is_wfrec_fm*(*is_Hcheck_fm*(*8*, *2*, *1*, *0*), *4*, *1*, *0*) $\cdot\cdot)$
definition *repl_PHcheck_fm* **where** *repl_PHcheck_fm* $\equiv$ *PHcheck_fm*(*2*,*3*,*0*,*1*)
definition *tl_repl_intf_fm* **where** *tl_repl_intf_fm* $\equiv$ $(\cdot\exists\cdot$*pair_fm*(*1*, *0*, *2*) $\wedge$ *is_wfrec_fm*(*iterates_MH_fm*(*tl_fm*(*1*,*0*), *9*, *2*, *1*, *0*), *3*, *1*, *0*) $\cdot\cdot)$
definition *formula_repl1_intf_fm* **where** *formula_repl1_intf_fm* $\equiv$ $(\cdot\exists\cdot$*pair_fm*(*1*, *0*, *2*) $\wedge$ *is_wfrec_fm*(*iterates_MH_fm*(*formula_functor_fm*(*1*,*0*), *9*, *2*, *1*, *0*), *3*, *1*, *0*) $\cdot\cdot)$
definition *eclose_closed_fm* **where** *eclose_closed_fm* $\equiv$ $(\cdot\exists\cdot$*pair_fm*(*1*, *0*, *2*) $\wedge$ *is_wfrec_fm*(*iterates_MH_fm*($\cdot\bigcup$ *1 is 0*$\cdot$, *9*, *2*, *1*, *0*), *3*, *1*, *0*) $\cdot\cdot)$

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*(φ))

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*)
⟨*proof*⟩

⟨*ML*⟩
notation *ZF_union_fm* (‹·*Union Ax*·›)

schematic_goal *ZF_power_auto*:
power_ax(##A) $\longleftrightarrow$ (A, [] $\models$ *?zfpow*)
⟨*proof*⟩

⟨*ML*⟩
notation *ZF_power_fm* (‹·*Powerset Ax*·›)

schematic_goal *ZF_pairing_auto*:
upair_ax(##A) $\longleftrightarrow$ (A, [] $\models$ *?zfpair*)
⟨*proof*⟩

⟨*ML*⟩
notation *ZF_pairing_fm* (‹·*Pairing*·›)

schematic_goal *ZF_foundation_auto*:
foundation_ax(##A) $\longleftrightarrow$ (A, [] $\models$ *?zffound*)
⟨*proof*⟩

⟨*ML*⟩
notation *ZF_foundation_fm* (‹·*Foundation*·›)

schematic_goal *ZF_extensionality_auto*:
extensionality(##A) $\longleftrightarrow$ (A, [] $\models$ *?zfext*)
⟨*proof*⟩

⟨*ML*⟩
notation *ZF_extensionality_fm* (‹·*Extensionality*·›)

schematic_goal *ZF_infinity_auto*:
  infinity_ax(##*A*) ⟷ (*A*, [] ⊨ (?φ(*i*,*j*,*h*)))
⟨*proof*⟩

⟨*ML*⟩
notation *ZF_infinity_fm* (‹·*Infinity*·›)

schematic_goal *ZF_choice_auto*:
  choice_ax(##*A*) ⟷ (*A*, [] ⊨ (?φ(*i*,*j*,*h*)))
⟨*proof*⟩

⟨*ML*⟩
notation *ZF_choice_fm* (‹·*AC*·›)

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 ≡ {·*Extensionality*·, ·*Foundation*·, ·*Pairing*·,
  ·*Union Ax*·, ·*Infinity*·, ·*Powerset Ax*·}

6.1 The Axiom of Separation, internalized

lemma *iterates_Forall_type* [*TC*]:
  ⟦ *n* ∈ *nat*; *p* ∈ *formula* ⟧ ⟹ *Forall*^*n*(*p*) ∈ *formula*
⟨*proof*⟩

lemma *last_init_eq* :
  assumes *l* ∈ *list*(*A*) *length*(*l*) = *succ*(*n*)
  shows ∃ *a*∈*A*. ∃ *l'*∈*list*(*A*). *l* = *l'*@[*a*]
⟨*proof*⟩

lemma *take_drop_eq* :
  assumes *l*∈*list*(*M*)
  shows ⋀ *n* . *n* < *succ*(*length*(*l*)) ⟹ *l* = *take*(*n*,*l*) @ *drop*(*n*,*l*)
  ⟨*proof*⟩

lemma *list_split* :
assumes *n* ≤ *succ*(*length*(*rest*)) *rest* ∈ *list*(*M*)
shows ∃ *re*∈*list*(*M*). ∃ *st*∈*list*(*M*). *rest* = *re* @ *st* ∧ *length*(*re*) = *pred*(*n*)
⟨*proof*⟩

lemma *sats_nForall*:
assumes
$\varphi \in formula$
shows
$n{\in}nat \Longrightarrow ms \in list(M) \Longrightarrow$
$(M,\ ms \models (Forall^{\frown}n(\varphi))) \longleftrightarrow$
$(\forall\, rest \in list(M).\ length(rest) = n \longrightarrow M,\ rest\ @\ ms \models \varphi)$
⟨*proof*⟩

definition
$sep\_body\_fm :: i \Rightarrow i$ **where**
$sep\_body\_fm(p) \equiv (\cdot\forall\,(\cdot\exists\,(\cdot\forall\cdot\cdot 0 \in 1\cdot \leftrightarrow \cdot\cdot 0 \in 2\cdot \wedge incr\_bv1^{\frown}2\ (p)\ \cdots)\cdot)\cdot)$

lemma *sep_body_fm_type* [*TC*]: $p \in formula \Longrightarrow sep\_body\_fm(p) \in formula$
⟨*proof*⟩

lemma *sats_sep_body_fm*:
assumes
$\varphi \in formula\ ms{\in}list(M)\ rest{\in}list(M)$
shows
$(M,\ rest\ @\ ms \models sep\_body\_fm(\varphi)) \longleftrightarrow$
$separation(\#\#M,\lambda x.\ M,\ [x]\ @\ rest\ @\ ms \models \varphi)$
⟨*proof*⟩

definition
$ZF\_separation\_fm :: i \Rightarrow i$ $(\langle\cdot Separation'(\_')\cdot\rangle)$ **where**
$ZF\_separation\_fm(p) \equiv Forall^{\frown}(pred(arity(p)))(sep\_body\_fm(p))$

lemma *ZF_separation_fm_type* [*TC*]: $p \in formula \Longrightarrow ZF\_separation\_fm(p) \in formula$
⟨*proof*⟩

lemma *sats_ZF_separation_fm_iff*:
assumes
$\varphi{\in}formula$
shows
$(M,\ [] \models \cdot Separation(\varphi)\cdot)$
$\longleftrightarrow$
$(\forall\, env{\in}list(M).\ arity(\varphi) \le 1 +_\omega length(env) \longrightarrow$
$separation(\#\#M,\lambda x.\ M,\ [x]\ @\ env \models \varphi))$
⟨*proof*⟩

6.2 The Axiom of Replacement, internalized

schematic_goal *sats_univalent_fm_auto*:
assumes

Q_iff_sats:$\bigwedge x\ y\ z.\ x \in A \Longrightarrow y \in A \Longrightarrow z{\in}A \Longrightarrow$
$Q(x,z) \longleftrightarrow (A,Cons(z,Cons(y,Cons(x,env))) \models Q1\_fm)$

$\bigwedge x\ y\ z.\ x \in A \Longrightarrow y \in A \Longrightarrow z \in A \Longrightarrow$
$Q(x,y) \longleftrightarrow (A, Cons(z, Cons(y, Cons(x, env))) \models Q2\_fm)$

and

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

shows

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

⟨*proof*⟩

⟨*ML*⟩

lemma *univalent_fm_type* [*TC*]: $q1 \in formula \Longrightarrow q2 \in formula \Longrightarrow i \in nat \Longrightarrow$
$univalent\_fm(q2, q1, i) \in formula$
⟨*proof*⟩

lemma *sats_univalent_fm* :

assumes

Q_iff_sats: $\bigwedge x\ y\ z.\ x \in A \Longrightarrow y \in A \Longrightarrow z \in A \Longrightarrow$
$Q(x,z) \longleftrightarrow (A, Cons(z, Cons(y, Cons(x, env))) \models Q1\_fm)$
$\bigwedge x\ y\ z.\ x \in A \Longrightarrow y \in A \Longrightarrow z \in A \Longrightarrow$
$Q(x,y) \longleftrightarrow (A, Cons(z, Cons(y, Cons(x, env))) \models Q2\_fm)$

and

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

shows

$(A, env \models univalent\_fm(Q1\_fm, Q2\_fm, i)) \longleftrightarrow univalent(\#\#A, B, Q)$
⟨*proof*⟩

definition

$swap\_vars :: i \Rightarrow i$ **where**
$swap\_vars(\varphi) \equiv$
$Exists(Exists(And(Equal(0,3), And(Equal(1,2), iterates(\lambda p.\ incr\_bv(p)\ `2\ ,\ 2, \varphi)))))$

lemma *swap_vars_type*[*TC*] :
$\varphi \in formula \Longrightarrow swap\_vars(\varphi) \in formula$
⟨*proof*⟩

lemma *sats_swap_vars* :
$[x,y]\ @\ env \in list(M) \Longrightarrow \varphi \in formula \Longrightarrow$
$(M, [x,y]\ @\ env \models swap\_vars(\varphi)) \longleftrightarrow M, [y,x]\ @\ env \models \varphi$
⟨*proof*⟩

definition

$univalent\_Q1 :: i \Rightarrow i$ **where**
$univalent\_Q1(\varphi) \equiv incr\_bv1(swap\_vars(\varphi))$

definition

$univalent\_Q2 :: i \Rightarrow i$ **where**
$univalent\_Q2(\varphi) \equiv incr\_bv(swap\_vars(\varphi))\ `0$

lemma *univalent_Qs_type* [*TC*]:
assumes φ∈*formula*
shows *univalent_Q1*(φ) ∈ *formula univalent_Q2*(φ) ∈ *formula*
⟨*proof*⟩

lemma *sats_univalent_fm_assm*:
assumes
$x \in A$ $y \in A$ $z \in A$ *env*∈ *list*(A) $\varphi \in$ *formula*
shows
(A, ([*x*,*z*] @ *env*) ⊨ φ) ⟷ (A, *Cons*(*z*,*Cons*(*y*,*Cons*(*x*,*env*))) ⊨ (*univalent_Q1*(φ)))
(A, ([*x*,*y*] @ *env*) ⊨ φ) ⟷ (A, *Cons*(*z*,*Cons*(*y*,*Cons*(*x*,*env*))) ⊨ (*univalent_Q2*(φ)))
⟨*proof*⟩

definition
rep_body_fm :: $i \Rightarrow i$ **where**
rep_body_fm(*p*) ≡ *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* ∈ *formula* ⟹ *rep_body_fm*(*p*) ∈ *formula*
⟨*proof*⟩

lemmas *ZF_replacement_simps* = *formula_add_params1*[*of* φ 2 _ M [_,_]]
sats_incr_bv_iff[*of* _ _ M _ []] — simplifies iterates of λx. *incr_bv*(*x*) ‘ 0
sats_incr_bv_iff[*of* _ _ M _ [_,_]]— simplifies λx. *incr_bv*(*x*) ‘ 2
sats_incr_bv1_iff[*of* _ _ M] *sats_swap_vars* **for** φ M

lemma *sats_rep_body_fm*:
assumes
$\varphi \in$ *formula* *ms*∈*list*(M) *rest*∈*list*(M)
shows
(M, *rest* @ *ms* ⊨ *rep_body_fm*(φ)) ⟷
strong_replacement(##M,λx *y*. M, [*x*,*y*] @ *rest* @ *ms* ⊨ φ)
⟨*proof*⟩

definition
ZF_replacement_fm :: $i \Rightarrow i$ (‹·*Replacement*′(_′)·›) **where**
ZF_replacement_fm(*p*) ≡ *Forall*^(*pred*(*pred*(*arity*(*p*))))(*rep_body_fm*(*p*))

lemma *ZF_replacement_fm_type* [*TC*]: *p* ∈ *formula* ⟹ *ZF_replacement_fm*(*p*)
∈ *formula*
⟨*proof*⟩

lemma *sats_ZF_replacement_fm_iff*:
assumes
φ∈*formula*
shows
(M, [] ⊨ ·*Replacement*(φ)·) ⟷ ($\forall$ *env*. *replacement_assm*(M,*env*,φ))

⟨*proof*⟩

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*
⟨*proof*⟩

lemma *ZF_schemes_subset_formula* [*TC*]: *ZF_schemes* ⊆ *formula*
⟨*proof*⟩

lemma *ZF_fin_subset_formula* [*TC*]: *ZF_fin* ⊆ *formula*
⟨*proof*⟩

definition
ZF :: *i* **where**
ZF ≡ *ZF_schemes* ∪ *ZF_fin*

lemma *ZF_subset_formula* [*TC*]: *ZF* ⊆ *formula*
⟨*proof*⟩

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*
⟨*proof*⟩

Satisfaction of a set of sentences

definition
satT :: [*i*,*i*] ⇒ *o* (_ ⊨ _ [*36*,*36*] *60*) **where**
A ⊨ Φ ≡ ∀φ∈Φ. (*A*,[] ⊨ φ)

lemma *satTI* [*intro*!]:
assumes ⋀φ. φ∈Φ ⟹ *A*,[] ⊨ φ

shows $A \models \Phi$
⟨*proof*⟩

lemma *satTD* [*dest*] :$A \models \Phi \Longrightarrow \varphi{\in}\Phi \Longrightarrow A,[] \models \varphi$
⟨*proof*⟩

lemma *satT_mono*: $A \models \Phi \Longrightarrow \Psi \subseteq \Phi \Longrightarrow A \models \Psi$
⟨*proof*⟩

lemma *satT_Un_iff*: $M \models \Phi \cup \Psi \longleftrightarrow M \models \Phi \wedge M \models \Psi$ ⟨*proof*⟩

lemma *sats_ZFC_iff_sats_ZF_AC*:
$(N \models ZFC) \longleftrightarrow (N \models ZF) \wedge (N, [] \models \cdot AC\cdot)$
⟨*proof*⟩

lemma *satT_ZF_imp_satT_Z*: $M \models ZF \Longrightarrow M \models \cdot Z\cdot$
⟨*proof*⟩

lemma *satT_ZFC_imp_satT_ZC*: $M \models ZFC \Longrightarrow M \models ZC$
⟨*proof*⟩

lemma *satT_Z_ZF_replacement_imp_satT_ZF*: $N \models \cdot Z\cdot \Longrightarrow N \models \{\cdot Replacement(x)\cdot$
$. x \in formula\} \Longrightarrow N \models ZF$
⟨*proof*⟩

lemma *satT_ZC_ZF_replacement_imp_satT_ZFC*: $N \models ZC \Longrightarrow N \models \{\cdot Replacement(x)\cdot$
$. x \in formula\} \Longrightarrow N \models ZFC$
⟨*proof*⟩

lemma *ground_repl_fm_sub_ZF*: $\{\cdot Replacement(ground\_repl\_fm(\varphi))\cdot . \varphi \in for-$
$mula\} \subseteq ZF$
⟨*proof*⟩

lemma *ZF_replacement_fms_sub_ZFC*: $\{\cdot Replacement(\varphi)\cdot . \varphi \in formula\} \subseteq$
ZFC
⟨*proof*⟩

lemma *ground_repl_fm_sub_ZFC*: $\{\cdot Replacement(ground\_repl\_fm(\varphi))\cdot . \varphi \in$
$formula\} \subseteq ZFC$
⟨*proof*⟩

lemma *ZF_replacement_ground_repl_fm_type*: $\{\cdot Replacement(ground\_repl\_fm(\varphi))\cdot$
$. \varphi \in formula\} \subseteq formula$
⟨*proof*⟩

end

7 Interface between set models and Constructibility

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

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

theory *Interface*
  imports
    Fm_Definitions
    Transitive_Models.Cardinal_AC_Relative
begin

locale *M_Z_basic* =
  fixes *M*
  assumes
    upair_ax: *upair_ax*(##*M*) **and**
    Union_ax: *Union_ax*(##*M*) **and**
    power_ax: *power_ax*(##*M*) **and**
    extensionality:*extensionality*(##*M*) **and**
    foundation_ax: *foundation_ax*(##*M*) **and**
    infinity_ax: *infinity_ax*(##*M*) **and**
    separation_ax: $\varphi \in$ *formula* $\Longrightarrow$ *env* $\in$ *list*(*M*) $\Longrightarrow$
                arity(φ) $\leq$ 1 $+_\omega$ *length*(*env*) $\Longrightarrow$
                separation(##*M*,λx. (*M*, [*x*] @ *env* $\models$ 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* ≡

{ *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* ⊆ *formula*
⟨*proof*⟩

declare (**in** *M_ZF1*) *replacement_instances1_defs*[*simp*]

locale *M_ZF1_trans* = *M_ZF1* + *M_Z_trans*

context *M_Z_trans*
begin

lemmas *transitivity* = *Transset_intf*[*OF trans_M*]

7.1 Interface with *M_trivial*

lemma *zero_in_M*: *0* ∈ *M*
⟨*proof*⟩

lemma *separation_in_ctm* :
assumes
φ ∈ *formula env*∈*list*(*M*)
arity(*φ*) ≤ *1* +$_\omega$ *length*(*env*) **and**
satsQ: ⋀*x*. *x*∈*M* ⟹ (*M*, [*x*]@*env* ⊨ *φ*) ⟷ *Q*(*x*)
shows
separation(##*M*,*Q*)
⟨*proof*⟩

end — *M_Z_trans*

locale *M_ZC_basic* = *M_Z_basic* + *M_AC* ##*M*

locale *M_ZFC1* = *M_ZF1* + *M_ZC_basic*

locale *M_ZFC1_trans* = *M_ZF1_trans* + *M_ZFC1*

sublocale *M_Z_trans* ⊆ *M_trans* ##*M*
⟨*proof*⟩

sublocale *M_Z_trans* $\subseteq$ *M_trivial* ##*M*
⟨*proof*⟩

7.2 Interface with *M_basic*

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

⟨*ML*⟩

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

⟨*ML*⟩

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

⟨*ML*⟩

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

⟨*ML*⟩

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

⟨*ML*⟩

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

⟨*ML*⟩

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

⟨*ML*⟩

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

⟨*ML*⟩

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

$pair(N,x,a,xa) \wedge xa \in r \wedge pair(N,x,b,xb) \wedge xb \in r \wedge$
$(\exists\, fx[N].\ \exists\, gx[N].\ fun\_apply(N,f,x,fx) \wedge fun\_apply(N,g,x,gx) \wedge$
$fx \neq gx)$

⟨*ML*⟩

context *M_Z_trans*
begin

lemma *inter_sep_intf* :
assumes $A{\in}M$
shows $separation(\#\#M,\lambda x\ .\ \forall\, y{\in}M\ .\ y{\in}A \longrightarrow x{\in}y)$
⟨*proof*⟩

lemma *diff_sep_intf* :
assumes $B{\in}M$
shows $separation(\#\#M,\lambda x\ .\ x{\notin}B)$
⟨*proof*⟩

lemma *cartprod_sep_intf* :
assumes $A{\in}M$ **and** $B{\in}M$
shows $separation(\#\#M,\lambda z.\ \exists\, x{\in}M.\ x{\in}A \wedge (\exists\, y{\in}M.\ y{\in}B \wedge pair(\#\#M,x,y,z)))$
⟨*proof*⟩

lemma *image_sep_intf* :
assumes $A{\in}M$ **and** $B{\in}M$
shows $separation(\#\#M,\ \lambda y.\ \exists\, p{\in}M.\ p{\in}B \wedge (\exists\, x{\in}M.\ x{\in}A \wedge pair(\#\#M,x,y,p)))$
⟨*proof*⟩

lemma *converse_sep_intf* :
assumes $R{\in}M$
shows $separation(\#\#M,\lambda z.\ \exists\, p{\in}M.\ p{\in}R \wedge (\exists\, x{\in}M.\exists\, y{\in}M.\ pair(\#\#M,x,y,p) \wedge pair(\#\#M,y,x,z)))$
⟨*proof*⟩

lemma *restrict_sep_intf* :
assumes $A{\in}M$
shows $separation(\#\#M,\lambda z.\ \exists\, x{\in}M.\ x{\in}A \wedge (\exists\, y{\in}M.\ pair(\#\#M,x,y,z)))$
⟨*proof*⟩

lemma *comp_sep_intf* :
assumes $R{\in}M$ **and** $S{\in}M$
shows $separation(\#\#M,\lambda xz.\ \exists\, x{\in}M.\ \exists\, y{\in}M.\ \exists\, z{\in}M.\ \exists\, xy{\in}M.\ \exists\, yz{\in}M.$
$pair(\#\#M,x,z,xz) \wedge pair(\#\#M,x,y,xy) \wedge pair(\#\#M,y,z,yz) \wedge xy{\in}S \wedge yz{\in}R)$
⟨*proof*⟩

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*))
⟨*proof*⟩

lemma *memrel_sep_intf*:
separation(*##M*, $\lambda z.\ \exists\, x{\in}M.\ \exists\, y{\in}M.$ *pair*(*##M*,*x*,*y*,*z*) $\wedge\ x \in y$)
⟨*proof*⟩

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*)
$\wedge$
$fx \neq gx$))
⟨*proof*⟩

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* $\subseteq$ *M_basic_no_repl* *##M*
⟨*proof*⟩

lemma *Replace_eq_Collect*:
assumes $\bigwedge x\ y\ y'.\ x{\in}A \Longrightarrow P(x,y) \Longrightarrow P(x,y') \Longrightarrow y{=}y'$ $\{y\ .\ x \in A,\ P(x,\ y)\} \subseteq B$
shows $\{y\ .\ x \in A,\ P(x,\ y)\} = \{y{\in}B\ .\ \exists\, x{\in}A.\ P(x,y)\}$
⟨*proof*⟩

context *M_Z_trans*
begin

lemma *Pow_inter_M_closed*: **assumes** $A \in M$ **shows** $Pow(A) \cap M \in M$
⟨*proof*⟩

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

end — *M_Z_trans*

context *M_basic_no_repl*
begin

lemma *Replace_funspace_succ_rep_intf_sub*:
assumes

$M(A)$ $M(n)$
shows
$\{z\ .\ p \in A,\ funspace\_succ\_rep\_intf\_rel(M,p,z,n)\}$
$\subseteq Pow^M(Pow^M(\bigcup domain(A) \cup (\{n\} \times range(A)) \cup (\bigcup(\{n\} \times range(A)))))$
⟨*proof*⟩

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
$z = z'$
⟨*proof*⟩

lemma *Replace_funspace_succ_rep_intf_eq*:
assumes
$M(A)$ $M(n)$
shows
$\{z\ .\ p \in A,\ funspace\_succ\_rep\_intf\_rel(M,p,z,n)\} =$
$\{z \in Pow^M(Pow^M(\bigcup domain(A) \cup (\{n\} \times range(A)) \cup (\bigcup(\{n\} \times range(A)))))$
.
$\exists\, p{\in}A.\ funspace\_succ\_rep\_intf\_rel(M,p,z,n)\}$
⟨*proof*⟩

end — *M_basic_no_repl*

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

⟨*ML*⟩

context *M_Z_trans*
begin

lemma *separation_fsri*:
$(\#\#M)(A) \Longrightarrow (\#\#M)(B) \Longrightarrow separation(\#\#M,\ is\_fsri(\#\#M,A,B))$
⟨*proof*⟩

lemma *separation_funspace_succ_rep_intf_rel*:
$(\#\#M)(A) \Longrightarrow (\#\#M)(B) \Longrightarrow separation(\#\#M,\ \lambda z.\ \exists\, p{\in}A.\ funspace\_succ\_rep\_intf\_rel(\#\#M,p,z,B))$
⟨*proof*⟩

lemma *Replace_funspace_succ_rep_intf_in_M*:
assumes
$A \in M$ $n \in M$
shows
$\{z\ .\ p \in A,\ funspace\_succ\_rep\_intf\_rel(\#\#M,p,z,n)\} \in M$
⟨*proof*⟩

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

end — *M_Z_trans*

sublocale *M_Z_trans* $\subseteq$ *M_basic* ##M
⟨*proof*⟩

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*))
⟨*proof*⟩

lemma *wftrancl_separation_intf*:
assumes $r{\in}M$ **and** $Z{\in}M$
shows *separation* (##M, *wellfounded_trancl*(##M,*Z*,*r*))
⟨*proof*⟩

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, $\lambda x.\ x{\in}nat$)
⟨*proof*⟩

lemma *nat_subset_I*: $\exists I{\in}M.\ nat \subseteq I$
⟨*proof*⟩

lemma *nat_in_M*: $nat \in M$
⟨*proof*⟩

end — *M_ZF1_trans*

sublocale *M_ZF1_trans* $\subseteq$ *M_trancl* ##M
⟨*proof*⟩

7.4 Interface with *M_eclose*

lemma *repl_sats*:

assumes
sat:$\bigwedge x\ z.\ x{\in}M \Longrightarrow z{\in}M \Longrightarrow (M,\ Cons(x,Cons(z,env)) \models \varphi) \longleftrightarrow P(x,z)$
shows
strong_replacement$(\#\#M,\lambda x\ z.\ (M,\ Cons(x,Cons(z,env)) \models \varphi)) \longleftrightarrow$
strong_replacement$(\#\#M,P)$
⟨*proof*⟩

⟨*ML*⟩

context *M_ZF1_trans*
begin

This lemma obtains *iterates_replacement* for predicates without parameters.

lemma *iterates_repl_intf* :
assumes
$v{\in}M$ **and**
isfm:*is_F_fm* ∈ *formula* **and**
arty:*arity*(*is_F_fm*)=*2* **and**
satsf: $\bigwedge a\ b\ env'.\ [\![\ a{\in}M\ ;\ b{\in}M\ ;\ env'{\in}list(M)\ ]\!]$
$\Longrightarrow is\_F(a,b) \longleftrightarrow (M,\ [b,a]@env' \models\ is\_F\_fm)$
and *is_F_fm_replacement*:
$\bigwedge env.\ (\cdot\exists\cdot\cdot\langle 1,0\rangle$ *is* $2\cdot \wedge$ *is_wfrec_fm*(*iterates_MH_fm*(*is_F_fm*,9,2,1,0),3,1,0)
$\cdot\cdot) \in formula \Longrightarrow env \in list(M) \Longrightarrow$
$arity((\cdot\exists\cdot\cdot\langle 1,0\rangle$ *is* $2\cdot \wedge$ *is_wfrec_fm*(*iterates_MH_fm*(*is_F_fm*,9,2,1,0),3,1,0)
$\cdot\cdot)) \le 2 +_\omega length(env) \Longrightarrow$
strong_replacement$(\#\#M,\lambda x\ y.$
$M, [x,y] @ env \models (\cdot\exists\cdot\cdot\langle 1,0\rangle$ *is* $2\cdot \wedge$ *is_wfrec_fm*(*iterates_MH_fm*(*is_F_fm*,9,2,1,0),3,1,0)
$\cdot\cdot))$
shows
iterates_replacement$(\#\#M,is\_F,v)$
⟨*proof*⟩

lemma *eclose_repl1_intf*:
assumes $A{\in}M$
shows *iterates_replacement*$(\#\#M,\ big\_union(\#\#M),\ A)$
⟨*proof*⟩

lemma *eclose_repl2_intf*:
assumes $A{\in}M$
shows *strong_replacement*$(\#\#M,\lambda n\ y.\ n{\in}nat \wedge is\_iterates(\#\#M,\ big\_union(\#\#M),$
$A,\ n,\ y))$
⟨*proof*⟩

end — *M_ZF1_trans*

sublocale *M_ZF1_trans* ⊆ *M_eclose* $\#\#M$
⟨*proof*⟩

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))
⟨*proof*⟩

⟨*ML*⟩

declare *is_Hrank_fm_def*[*fm_definitions add*]

context *M_ZF1_trans*
begin

lemma *wfrec_rank* :
assumes *X∈M*
shows *wfrec_replacement(##M,is_Hrank(##M),rrank(X))*
⟨*proof*⟩

lemma *trans_repl_HVFrom* :
assumes *A∈M i∈M*
shows *transrec_replacement(##M,is_HVfrom(##M,A),i)*
⟨*proof*⟩

end — *M_ZF1_trans*

7.5 Interface for proving Collects and Replace in M.

context *M_ZF1_trans*
begin

lemma *Collect_in_M* :
assumes
φ ∈ formula env∈list(M)
arity(φ) ≤ 1 +ω length(env) A∈M **and**
satsQ: ⋀*x. x∈M ⟹ (M, [x]@env ⊨ φ) ⟷ Q(x)*
shows
{y∈A . Q(y)}∈M
⟨*proof*⟩
lemma *separation_in_M* :
assumes
φ ∈ formula env∈list(M)
arity(φ) ≤ 1 +ω length(env) A∈M **and**
satsQ: ⋀*x. x∈A ⟹ (M, [x]@env ⊨ φ) ⟷ Q(x)*
shows
{y∈A . Q(y)} ∈ M
⟨*proof*⟩

end — *M_ZF1_trans*

context *M_Z_trans*
begin

lemma *strong_replacement_in_ctm*:
  assumes
    f_fm: $\varphi \in formula$ **and**
    f_ar: $arity(\varphi) \leq 2 +_\omega length(env)$ **and**
    fsats: $\bigwedge x\ y.\ x{\in}M \Longrightarrow y{\in}M \Longrightarrow (M,[x,y]@env \models \varphi) \longleftrightarrow y = f(x)$ **and**
    fclosed: $\bigwedge x.\ x{\in}M \Longrightarrow f(x) \in M$ **and**
    phi_replacement:$replacement\_assm(M,env,\varphi)$ **and**
    $env{\in}list(M)$
  shows $strong\_replacement(\#\#M,\ \lambda x\ y\ .\ y = f(x))$
  ⟨*proof*⟩

lemma *strong_replacement_rel_in_ctm* :
  assumes
    f_fm: $\varphi \in formula$ **and**
    f_ar: $arity(\varphi) \leq 2 +_\omega length(env)$ **and**
    fsats: $\bigwedge x\ y.\ x{\in}M \Longrightarrow y{\in}M \Longrightarrow (M,[x,y]@env \models \varphi) \longleftrightarrow f(x,y)$ **and**
    phi_replacement:$replacement\_assm(M,env,\varphi)$ **and**
    $env{\in}list(M)$
  shows $strong\_replacement(\#\#M,\ f)$
  ⟨*proof*⟩

lemma *Replace_in_M* :
  assumes
    f_fm: $\varphi \in formula$ **and**
    f_ar: $arity(\varphi) \leq 2 +_\omega length(env)$ **and**
    fsats: $\bigwedge x\ y.\ x{\in}A \Longrightarrow y{\in}M \Longrightarrow (M,[x,y]@env \models \varphi) \longleftrightarrow y = f(x)$ **and**
    fclosed: $\bigwedge x.\ x{\in}A \Longrightarrow f(x) \in M$ **and**
    $A{\in}M\ env{\in}list(M)$ **and**
    phi'_replacement:$replacement\_assm(M,env@[A],\ \cdot\varphi \wedge \cdot 0 \in length(env) +_\omega 2\cdot\cdot$
)
  shows $\{f(x)\ .\ x{\in}A\}{\in}M$
⟨*proof*⟩

lemma *Replace_relativized_in_M* :
  assumes
    f_fm: $\varphi \in formula$ **and**
    f_ar: $arity(\varphi) \leq 2 +_\omega length(env)$ **and**
    fsats: $\bigwedge x\ y.\ x{\in}A \Longrightarrow y{\in}M \Longrightarrow (M,[x,y]@env \models \varphi) \longleftrightarrow is\_f(x,y)$ **and**
    fabs: $\bigwedge x\ y.\ x{\in}A \Longrightarrow y{\in}M \Longrightarrow is\_f(x,y) \longleftrightarrow y = f(x)$ **and**
    fclosed: $\bigwedge x.\ x{\in}A \Longrightarrow f(x) \in M$ **and**
    $A{\in}M\ env{\in}list(M)$ **and**
    phi'_replacement:$replacement\_assm(M,env@[A],\ \cdot\varphi \wedge \cdot 0 \in length(env) +_\omega 2\cdot\cdot$
)
  shows $\{f(x)\ .\ x{\in}A\}{\in}M$

⟨*proof*⟩

lemma *ren_action* :
assumes
env∈*list*(*M*) *x*∈*M y*∈*M z*∈*M*
shows ∀ *i* . *i* < *2*+$_\omega$*length*(*env*) ⟶
nth(*i*,[*x*,*z*]@*env*) = *nth*(*ϱ_repl*(*length*(*env*))‘*i*,[*z*,*x*,*y*]@*env*)
⟨*proof*⟩

lemma *Lambda_in_M* :
assumes
f_fm: *φ* ∈ *formula* **and**
f_ar: *arity*(*φ*)≤ *2* +$_\omega$ *length*(*env*) **and**
fsats: ⋀*x y*. *x*∈*A* ⟹ *y*∈*M* ⟹ (*M*,[*x*,*y*]@*env* ⊨ *φ*) ⟷ *is_f*(*x*,*y*) **and**
fabs: ⋀*x y*. *x*∈*A* ⟹ *y*∈*M* ⟹ *is_f*(*x*,*y*) ⟷ *y* = *f*(*x*) **and**
fclosed: ⋀*x*. *x*∈*A* ⟹ *f*(*x*) ∈ *M* **and**
A∈*M env*∈*list*(*M*) **and**
phi′_replacement2: *replacement_assm*(*M*,*env*@[*A*],*Lambda_in_M_fm*(*φ*,*length*(*env*)))
shows (*λx*∈*A* . *f*(*x*)) ∈*M*
⟨*proof*⟩

lemma *ren_action′* :
assumes
env∈*list*(*M*) *x*∈*M y*∈*M z*∈*M u*∈*M*
shows ∀ *i* . *i* < *3*+$_\omega$*length*(*env*) ⟶
nth(*i*,[*x*,*z*,*u*]@*env*) = *nth*(*ϱ_pair_repl*(*length*(*env*))‘*i*,[*x*,*z*,*y*,*u*]@*env*)
⟨*proof*⟩

lemma *LambdaPair_in_M* :
assumes
f_fm: *φ* ∈ *formula* **and**
f_ar: *arity*(*φ*)≤ *3* +$_\omega$ *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**
A∈*M env*∈*list*(*M*) **and**
phi′_replacement3: *replacement_assm*(*M*,*env*@[*A*],*LambdaPair_in_M_fm*(*φ*,*length*(*env*)))
shows (*λx*∈*A* . *f*(*fst*(*x*),*snd*(*x*))) ∈*M*
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *lam_replacement2_in_ctm* :
assumes
f_fm: *φ* ∈ *formula* **and**
f_ar: *arity*(*φ*)≤ *3* +$_\omega$ *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* , λ*x* . *f*(*fst*(*x*),*snd*(*x*)))
⟨*proof*⟩

⟨*ML*⟩

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*, λ*r*. *M*, [*f*(*r*), *a*, *b*, *c*, *d*, *g*(*r*)] ⊨ χ)
⟨*proof*⟩

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*, λ*r*. *M*, [*f*(*r*), *a*, *b*, *c*, *d*, *g*(*r*), *h*(*r*)] ⊨ χ)
⟨*proof*⟩

lemma *separation_sat_after_function*:
assumes [*a*, *b*, *c*, *d*, τ]∈*list*(*M*) **and** χ∈*formula* **and** *arity*(χ) ≤ 7
and
f_fm: *f_fm* ∈ *formula* **and**
f_ar: *arity*(*f_fm*) ≤ 7 **and**

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

definition $separation\_assm\_fm :: [i,i,i] \Rightarrow i$
where
$separation\_assm\_fm(A,x,f\_fm) \equiv (\cdot\exists\ (\cdot\exists\ \cdot\cdot 0 \in A +_\omega 2\cdot \wedge \cdot\cdot\langle 0,1\rangle\ is\ x +_\omega 2\ \cdot \wedge f\_fm\ \cdots)\cdot)$

lemma *separation_assm_fm_type*[*TC*]:
$A \in \omega \Longrightarrow y \in \omega \Longrightarrow f\_fm \in formula \Longrightarrow separation\_assm\_fm(A,\ y,f\_fm) \in formula$
⟨*proof*⟩

lemma $arity\_separation\_assm\_fm : A \in \omega \Longrightarrow x \in \omega \Longrightarrow f\_fm \in formula \Longrightarrow arity(separation\_assm\_fm(A,\ x,\ f\_fm)) = succ(A) \cup succ(x) \cup pred(pred(arity(f\_fm)))$
⟨*proof*⟩

definition *separation_assm_bin_fm* **where**
$separation\_assm\_bin\_fm(A,y,f\_fm) \equiv$
$(\cdot\exists(\cdot\exists(\cdot\exists(\cdot(\cdot\cdot 3 \in A +_\omega 4\cdot \wedge\ \cdot\langle 3,2\rangle\ is\ y +_\omega 4\cdot\cdot\ ) \wedge \cdot f\_fm \wedge \cdot \cdot fst(3)\ is\ 0\ \cdot \wedge \cdot snd(3)\ is\ 1\cdots\ )\cdot)\cdot)\cdot)\cdot)$

lemma *separation_assm_bin_fm_type*[*TC*]:
$A \in \omega \Longrightarrow y \in \omega \Longrightarrow f\_fm \in formula \Longrightarrow separation\_assm\_bin\_fm(A,\ y,f\_fm) \in formula$
⟨*proof*⟩

lemma $arity\_separation\_assm\_bin\_fm : A \in \omega \Longrightarrow x \in \omega \Longrightarrow f\_fm \in formula \Longrightarrow arity(separation\_assm\_bin\_fm(A,\ x,\ f\_fm)) = succ(A) \cup succ(x) \cup (pred\hat{}4(arity(f\_fm)))$
⟨*proof*⟩

context *M_Z_trans*
begin

lemma *separation_assm_sats* :
assumes
f_fm: $\varphi \in formula$ **and**
f_ar: $arity(\varphi) = 2$ **and**
fsats: $\bigwedge env\ x\ y.\ env{\in}list(M) \Longrightarrow x{\in}M \Longrightarrow y{\in}M \Longrightarrow (M,[x,y]@env \models \varphi) \longleftrightarrow$

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

lemma *separation_assm_bin_sats* :
assumes
f_fm: $\varphi \in$ *formula* **and**
f_ar: *arity*(φ) = 3 **and**
fsats: $\bigwedge$*env x z y. env*∈*list*(*M*) ⟹ *x*∈*M* ⟹ *z*∈*M* ⟹ *y*∈*M* ⟹ (*M*,[*x*,*z*,*y*]@*env* ⊨ φ) ⟷ *is_f*(*x*,*z*,*y*) **and**
fabs: $\bigwedge x\ z\ y.\ x{\in}M \Longrightarrow\ z{\in}M \Longrightarrow y{\in}M \Longrightarrow$ *is_f*(*x*,*z*,*y*) $\longleftrightarrow y = f(x,z)$ **and**
fclosed: $\bigwedge x\ z\ .\ x{\in}M \Longrightarrow\ z{\in}M \Longrightarrow f(x,z) \in M$ **and**
$A{\in}M$
shows *separation*(##*M*, $\lambda y.\ \exists\, x \in M\ .\ x{\in}A \wedge y = \langle x, f(fst(x),snd(x))\rangle$)
⟨*proof*⟩

lemma *separation_Union*: $A{\in}M \Longrightarrow$
separation(##*M*, $\lambda y.\ \exists\, x \in M\ .\ x{\in}A \wedge y = \langle x,$ *Union*(*x*)$\rangle$)
⟨*proof*⟩

lemma *lam_replacement_Union*: *lam_replacement*(##*M*, *Union*)
⟨*proof*⟩

lemma *separation_fst*: $A{\in}M \Longrightarrow$
separation(##*M*, $\lambda y.\ \exists\, x \in M\ .\ x{\in}A \wedge y = \langle x,$ *fst*(*x*)$\rangle$)
⟨*proof*⟩

lemma *lam_replacement_fst*: *lam_replacement*(##*M*, *fst*)
⟨*proof*⟩

lemma *separation_snd*: $A{\in}M \Longrightarrow$
separation(##*M*, $\lambda y.\ \exists\, x \in M\ .\ x{\in}A \wedge y = \langle x,$ *snd*(*x*)$\rangle$)
⟨*proof*⟩

lemma *lam_replacement_snd*: *lam_replacement*(##*M*, *snd*)
⟨*proof*⟩

Binary lambda-replacements

lemma *separation_Image*: $A{\in}M \Longrightarrow$
separation(##*M*, $\lambda y.\ \exists\, x{\in}M.\ x \in A \wedge y = \langle x,$ *fst*(*x*) ‘‘ *snd*(*x*)$\rangle$)
⟨*proof*⟩

lemma *lam_replacement_Image*: *lam_replacement*(##*M*, $\lambda x\ .$ *fst*(*x*) ‘‘ *snd*(*x*))
⟨*proof*⟩

lemma *separation_middle_del*: $A{\in}M \Longrightarrow$

separation(##*M*, λ*y*. ∃ *x*∈*M*. *x* ∈ *A* ∧ *y* = ⟨*x*, *middle_del*(*fst*(*x*), *snd*(*x*))⟩)
⟨*proof*⟩

lemma *lam_replacement_middle_del*: *lam_replacement*(##*M*, λ*r* . *middle_del*(*fst*(*r*),*snd*(*r*)))
⟨*proof*⟩

lemma *separation_prodRepl*: *A*∈*M* ⟹
separation(##*M*, λ*y*. ∃ *x*∈*M*. *x* ∈ *A* ∧ *y* = ⟨*x*, *prodRepl*(*fst*(*x*), *snd*(*x*))⟩)
⟨*proof*⟩

lemma *lam_replacement_prodRepl*: *lam_replacement*(##*M*, λ*r* . *prodRepl*(*fst*(*r*),*snd*(*r*)))
⟨*proof*⟩

end — *M_Z_trans*

context *M_trivial*
begin

lemma *first_closed*:
M(*B*) ⟹ *M*(*r*) ⟹ *first*(*u*,*r*,*B*) ⟹ *M*(*u*)
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

end — *M_trivial*

context *M_Z_trans*
begin

lemma (**in** *M_basic*) *is_minimum_equivalence* :
M(*R*) ⟹ *M*(*X*) ⟹ *M*(*u*) ⟹ *is_minimum*(*M*,*R*,*X*,*u*) ⟷ *is_minimum*′(*M*,*R*,*X*,*u*)
⟨*proof*⟩

lemma *separation_minimum*: *A*∈*M* ⟹
separation(##*M*, λ*y*. ∃ *x*∈*M*. *x* ∈ *A* ∧ *y* = ⟨*x*, *minimum*(*fst*(*x*), *snd*(*x*))⟩)
⟨*proof*⟩

lemma *lam_replacement_minimum*: *lam_replacement*(##*M*, λ*x* . *minimum*(*fst*(*x*),*snd*(*x*)))
⟨*proof*⟩

end — *M_Z_trans*

end

7.6 More Instances of Separation

theory *Separation_Instances*
imports
Interface
begin

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

definition *radd_body* :: $[i,i,i] \Rightarrow o$ **where**
$radd\_body(R,S) \equiv \lambda z.\ (\exists\, x\ y.\ z = \langle Inl(x),\ Inr(y)\rangle) \vee$
$(\exists\, x'\ x.\ z = \langle Inl(x'),\ Inl(x)\rangle \wedge \langle x',\ x\rangle \in R) \vee$
$(\exists\, y'\ y.\ z = \langle Inr(y'),\ Inr(y)\rangle \wedge \langle y',\ y\rangle \in S)$

⟨*ML*⟩

definition *rmult_body* :: $[i,i,i] \Rightarrow o$ **where**
$rmult\_body(b,d) \equiv \lambda z.\ \exists\, x'\ y'\ x\ y.\ z = \langle\langle x',\ y'\rangle,\ x,\ y\rangle \wedge (\langle x',\ x\rangle \in b \vee$
$x' = x \wedge \langle y',\ y\rangle \in d)$

⟨*ML*⟩

lemma (**in** *M_replacement*) *separation_well_ord_iso*:
$(M)(f) \Longrightarrow (M)(r) \Longrightarrow (M)(A) \Longrightarrow separation$
$(M,\ \lambda x.\ x \in A \longrightarrow (\exists\, y[M].\ \exists\, p[M].\ is\_apply(M,\ f,\ x,\ y) \wedge pair(M,\ y,\ x,\ p)$
$\wedge\ p \in r))$
⟨*proof*⟩

definition *is_obase_body* :: $[i \Rightarrow o,i,i,i] \Rightarrow o$ **where**
$is\_obase\_body(N,A,r,x) \equiv x \in A \longrightarrow$
$\neg\ (\exists\, y[N].$
$\exists\, g[N].$
$ordinal(N,\ y) \wedge$
$(\exists\, my[N].$
$\exists\, pxr[N].$
$membership(N,\ y,\ my) \wedge$
$pred\_set(N,\ A,\ x,\ r,\ pxr) \wedge$
$order\_isomorphism(N,\ pxr,\ r,\ y,\ my,\ g)))$

⟨*ML*⟩

definition *is_obase_equals* :: $[i \Rightarrow o,i,i,i] \Rightarrow o$ **where**
$is\_obase\_equals(N,A,r,a) \equiv \exists\, x[N].$
$\exists\, g[N].$
$\exists\, mx[N].$
$\exists\, par[N].$
$ordinal(N,\ x) \wedge$
$membership(N,\ x,\ mx) \wedge$

$pred\_set(N,\ A,\ a,\ r,\ par) \wedge order\_isomorphism(N,\ par,\ r,\ x,\ mx,\ g)$

⟨*ML*⟩

context *M_ZF1_trans*
begin

lemma *radd_body_abs*:
assumes $(\#\#M)(R)$ $(\#\#M)(S)$ $(\#\#M)(x)$
shows $is\_radd\_body(\#\#M,R,S,x) \longleftrightarrow radd\_body(R,S,x)$
⟨*proof*⟩

lemma *separation_radd_body*:
$(\#\#M)(R) \Longrightarrow (\#\#M)(S) \Longrightarrow separation$
$(\#\#M,\ \lambda z.\ (\exists\, x\ y.\ z = \langle Inl(x),\ Inr(y)\rangle) \vee$
$(\exists\, x'\ x.\ z = \langle Inl(x'),\ Inl(x)\rangle \wedge \langle x',\ x\rangle \in R) \vee$
$(\exists\, y'\ y.\ z = \langle Inr(y'),\ Inr(y)\rangle \wedge \langle y',\ y\rangle \in S))$
⟨*proof*⟩

lemma *rmult_body_abs*:
assumes $(\#\#M)(b)$ $(\#\#M)(d)$ $(\#\#M)(x)$
shows $is\_rmult\_body(\#\#M,b,d,x) \longleftrightarrow rmult\_body(b,d,x)$
⟨*proof*⟩

lemma *separation_rmult_body*:
$(\#\#M)(b) \Longrightarrow (\#\#M)(d) \Longrightarrow separation$
$(\#\#M,\ \lambda z.\ \exists\, x'\ y'\ x\ y.\ z = \langle\langle x',\ y'\rangle,\ x,\ y\rangle \wedge (\langle x',\ x\rangle \in b \vee x' = x \wedge \langle y',\ y\rangle \in d))$
⟨*proof*⟩

lemma *separation_is_obase*:
$(\#\#M)(f) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(A) \Longrightarrow separation$
$(\#\#M,\ \lambda x.\ x \in A \longrightarrow$
$\neg\ (\exists\, y[\#\#M].$
$\exists\, g[\#\#M].$
$ordinal(\#\#M,\ y) \wedge$
$(\exists\, my[\#\#M].$
$\exists\, pxr[\#\#M].$
$membership(\#\#M,\ y,\ my) \wedge$
$pred\_set(\#\#M,\ A,\ x,\ r,\ pxr) \wedge$
$order\_isomorphism(\#\#M,\ pxr,\ r,\ y,\ my,\ g))))$
⟨*proof*⟩

lemma *separation_obase_equals*:
$(\#\#M)(f) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(A) \Longrightarrow separation$
$(\#\#M,\ \lambda a.\ \exists\, x[\#\#M].$
$\exists\, g[\#\#M].$

$\exists\, mx[\#\#M].$
$\exists\, par[\#\#M].$
$ordinal(\#\#M,\ x) \wedge$
$membership(\#\#M,\ x,\ mx) \wedge$
$pred\_set(\#\#M,\ A,\ a,\ r,\ par) \wedge order\_isomorphism(\#\#M,$
$par,\ r,\ x,\ mx,\ g))$
$\langle proof \rangle$

lemma *separation_PiP_rel*:
$(\#\#M)(A) \Longrightarrow separation(\#\#M,\ PiP\_rel(\#\#M,A))$
$\langle proof \rangle$

lemma *separation_injP_rel*:
$(\#\#M)(A) \Longrightarrow separation(\#\#M,\ injP\_rel(\#\#M,A))$
$\langle proof \rangle$

lemma *separation_surjP_rel*:
$(\#\#M)(A) \Longrightarrow (\#\#M)(B) \Longrightarrow separation(\#\#M,\ surjP\_rel(\#\#M,A,B))$
$\langle proof \rangle$

lemma *separation_is_function*:
$separation(\#\#M,\ is\_function(\#\#M))$
$\langle proof \rangle$

end — *M_ZF1_trans*

definition *fstsnd_in_sndsnd* :: $[i] \Rightarrow o$ **where**
$fstsnd\_in\_sndsnd \equiv \lambda x.\ fst(snd(x)) \in snd(snd(x))$
$\langle ML \rangle$

definition *sndfst_eq_fstsnd* :: $[i] \Rightarrow o$ **where**
$sndfst\_eq\_fstsnd \equiv \lambda x.\ snd(fst(x)) = fst(snd(x))$
$\langle ML \rangle$

context *M_ZF1_trans*
begin

lemma *fstsnd_in_sndsnd_abs*:
assumes $(\#\#M)(x)$
shows $is\_fstsnd\_in\_sndsnd(\#\#M,x) \longleftrightarrow fstsnd\_in\_sndsnd(x)$
$\langle proof \rangle$

lemma *separation_fstsnd_in_sndsnd*:
$separation(\#\#M,\ \lambda x.\ fst(snd(x)) \in snd(snd(x)))$
$\langle proof \rangle$

lemma *sndfst_eq_fstsnd_abs*:

assumes $(\#\#M)(x)$
shows *is_sndfst_eq_fstsnd*$(\#\#M,x) \longleftrightarrow$ *sndfst_eq_fstsnd*(x)
⟨*proof*⟩

lemma *separation_sndfst_eq_fstsnd*:
separation$(\#\#M,\ \lambda x.\ snd(fst(x)) = fst(snd(x)))$
⟨*proof*⟩

end — *M_ZF1_trans*

end

8 More Instances of Replacement

theory *Replacement_Instances*
imports
Separation_Instances
Transitive_Models.Pointed_DC_Relative
begin

lemma *composition_fm_type*[*TC*]: $a0 \in \omega \Longrightarrow a1 \in \omega \Longrightarrow a2 \in \omega \Longrightarrow$
composition_fm$(a0,a1,a2) \in formula$
⟨*proof*⟩

⟨*ML*⟩

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

⟨*ML*⟩

definition *HAleph_wfrec_repl_body* **where**
HAleph_wfrec_repl_body$(N,mesa,x,z) \equiv \exists\, y[N].$
$pair(N,\ x,\ y,\ z) \wedge$
$(\exists\, g[N].$
$(\forall\, u[N].$
$u \in g \longleftrightarrow$
$(\exists\, a[N].$
$\exists\, y[N].$
$\exists\, ax[N].$
$\exists\, sx[N].$
$\exists\, r\_sx[N].$
$\exists f\_r\_sx[N].$
$pair(N,\ a,\ y,\ u) \wedge$
$pair(N,\ a,\ x,\ ax) \wedge$
$upair(N,\ a,\ a,\ sx) \wedge$
$pre\_image(N,\ mesa,\ sx,\ r\_sx) \wedge$
$restriction(N,\ g,\ r\_sx,\ f\_r\_sx) \wedge ax \in mesa \wedge is\_HAleph(N,\ a,\ f\_r\_sx,\ y)))$

$\wedge$
$is\_HAleph(N,\ x,\ g,\ y))$

$\langle ML \rangle$

definition *dcwit_repl_body* **where**
dcwit_repl_body$(N,mesa,A,a,s,R) \equiv \lambda x\ z.\ \exists\, y[N].\ pair(N,\ x,\ y,\ z) \wedge$
is_wfrec
$(N,\ \lambda n\ f.\ is\_nat\_case$
$(N,\ a,$
$\lambda m\ bmfm.$
$\exists\, fm[N].$
$\exists\, cp[N].$
$is\_apply(N,\ f,\ m,\ fm) \wedge$
$is\_Collect(N,\ A,\ \lambda x.\ \exists\, fmx[N].\ (N(x)$
$\wedge\ fmx \in R) \wedge pair(N,\ fm,\ x,\ fmx),\ cp) \wedge$
$is\_apply(N,\ s,\ cp,\ bmfm),$
$n),$
$mesa,\ x,\ y)$

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

$\langle ML \rangle$

definition *dcwit_aux_fm* **where**
dcwit_aux_fm$(A,s,R) \equiv (\cdot\exists \cdot\cdot 4\text{`}2\ is\ 0\cdot \wedge$
$(\cdot\exists \cdot Collect\_fm$
$(succ(succ(succ(succ(succ(succ(succ(succ(succ(succ(A)))))))))),$
$(\cdot\exists \cdot\cdot 0 \in$
$succ(succ(succ(succ(succ(succ(succ(succ(succ(succ(succ(R)))))))))))$
$\cdot \wedge$
$pair\_fm(3,\ 1,\ 0)\ \cdot\cdot),$
$0) \wedge$
$\cdot\ succ(succ(succ(succ(succ(succ(succ(succ(succ(s)))))))))\text{`}0\ is$
$2\cdots)\cdot\cdot)$

$\langle ML \rangle$

lemma *dcwit_aux_fm_type*$[TC]$: $A \in \omega \Longrightarrow s \in \omega \Longrightarrow R \in \omega \Longrightarrow dcwit\_aux\_fm(A,s,R)$
$\in formula$
$\langle proof \rangle$

definition *is_nat_case_dcwit_aux_fm* **where**
is_nat_case_dcwit_aux_fm$(A,a,s,R) \equiv is\_nat\_case\_fm$
$(succ(succ(succ(succ(succ(succ(a)))))),dcwit\_aux\_fm(A,s,R),$
$2,\ 0)$

lemma *is_nat_case_dcwit_aux_fm_type*[*TC*]: $A \in \omega \Longrightarrow a \in \omega \Longrightarrow s \in \omega \Longrightarrow R \in \omega \Longrightarrow$ *is_nat_case_dcwit_aux_fm*(A,a,s,R) $\in$ *formula*
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

lemma *arity_dcwit_repl_body*: *arity*(*dcwit_repl_body_fm*(6,5,4,3,2,0,1)) = 7
⟨*proof*⟩

definition *fst2_snd2*
where *fst2_snd2*(x) ≡ ⟨*fst*(*fst*(x)), *snd*(*snd*(x))⟩

⟨*ML*⟩

lemma (**in** *M_trivial*) *fst2_snd2_abs*:
assumes $M(x)$ $M(res)$
shows *is_fst2_snd2*(M, x, res) ⟷ res = *fst2_snd2*(x)
⟨*proof*⟩

⟨*ML*⟩

definition *sndfst_fst2_snd2*
where *sndfst_fst2_snd2*(x) ≡ ⟨*snd*(*fst*(x)), *fst*(*fst*(x)), *snd*(*snd*(x))⟩

⟨*ML*⟩

definition *order_eq_map* **where**
order_eq_map(M,A,r,a,z) ≡ ∃ x[M]. ∃ g[M]. ∃ mx[M]. ∃ 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)

⟨*ML*⟩

definition *banach_body_iterates* **where**
banach_body_iterates(M,X,Y,f,g,W,n,x,z) ≡
∃ y[M].
pair(M, x, y, z) ∧
(∃ fa[M].
(∀ z[M].
$z \in fa$ ⟷
(∃ xa[M].
∃ y[M].
∃ xaa[M].
∃ sx[M].
∃ r_sx[M].

$\exists\, f\_r\_sx[M].\ \exists\, sn[M].\ \exists\, msn[M].\ successor(M,n,sn)$
$\wedge$
$membership(M,sn,msn)\ \wedge$
$pair(M,\ xa,\ y,\ z)\ \wedge$
$pair(M,\ xa,\ x,\ xaa)\ \wedge$
$upair(M,\ xa,\ xa,\ sx)\ \wedge$
$pre\_image(M,\ msn,\ sx,\ r\_sx)\ \wedge$
$restriction(M,\ fa,\ r\_sx,\ f\_r\_sx)\ \wedge$
$xaa \in msn\ \wedge$
$(empty(M,\ xa) \longrightarrow y = W)\ \wedge$
$(\forall\, m[M].$
$successor(M,\ m,\ xa) \longrightarrow$
$(\exists\, gm[M].$
$is\_apply(M,\ f\_r\_sx,\ m,\ gm)\ \wedge$
$is\_banach\_functor(M,\ X,\ Y,\ f,\ g,\ gm,\ y)))\ \wedge$
$(is\_quasinat(M,\ xa) \vee empty(M,\ y))))\ \wedge$
$(empty(M,\ x) \longrightarrow y = W)\ \wedge$
$(\forall\, m[M].$
$successor(M,\ m,\ x) \longrightarrow$
$(\exists\, gm[M].\ is\_apply(M,\ fa,\ m,\ gm)\ \wedge\ is\_banach\_functor(M,$
$X,\ Y,\ f,\ g,\ gm,\ y)))\ \wedge$
$(is\_quasinat(M,\ x) \vee empty(M,\ y)))$

$\langle ML \rangle$

definition *banach_is_iterates_body* **where**
$banach\_is\_iterates\_body(M,X,Y,f,g,W,n,y) \equiv \exists\, om[M].\ omega(M,om)\ \wedge\ n \in om\ \wedge$
$(\exists\, sn[M].$
$\exists\, msn[M].$
$successor(M,\ n,\ sn)\ \wedge$
$membership(M,\ sn,\ msn)\ \wedge$
$(\exists\, fa[M].$
$(\forall\, z[M].$
$z \in fa \longleftrightarrow$
$(\exists\, x[M].$
$\exists\, y[M].$
$\exists\, xa[M].$
$\exists\, sx[M].$
$\exists\, r\_sx[M].$
$\exists\, f\_r\_sx[M].$
$pair(M,\ x,\ y,\ z)\ \wedge$
$pair(M,\ x,\ n,\ xa)\ \wedge$
$upair(M,\ x,\ x,\ sx)\ \wedge$
$pre\_image(M,\ msn,\ sx,\ r\_sx)\ \wedge$
$restriction(M,\ fa,\ r\_sx,\ f\_r\_sx)\ \wedge$
$xa \in msn\ \wedge$
$(empty(M,\ x) \longrightarrow y = W)\ \wedge$
$(\forall\, m[M].$

$$successor(M,\ m,\ x) \longrightarrow$$
$$(\exists\, gm[M].$$
$$fun\_apply(M,\ f\_r\_sx,\ m,\ gm) \wedge$$
$$is\_banach\_functor(M,\ X,\ Y,\ f,\ g,\ gm,\ y))) \wedge$$
$$(is\_quasinat(M,\ x) \vee empty(M,\ y)))) \wedge$$
$$(empty(M,\ n) \longrightarrow y = W) \wedge$$
$$(\forall\, m[M].$$
$$successor(M,\ m,\ n) \longrightarrow$$
$$(\exists\, gm[M].\ fun\_apply(M,\ fa,\ m,\ gm) \wedge is\_banach\_functor(M,$$
$$X,\ Y,\ f,\ g,\ gm,\ y))) \wedge$$
$$(is\_quasinat(M,\ n) \vee empty(M,\ y))))$$

⟨*ML*⟩

definition *trans_apply_image* **where**
trans_apply_image(*f*) ≡ λ*a g*. *f* ‘ (*g* ‘‘ *a*)

⟨*ML*⟩

schematic_goal *arity_is_recfun_fm*[*arity*]:
p ∈ *formula* ⟹ *a* ∈ ω ⟹ *z* ∈ ω ⟹ *r* ∈ ω ⟹ *arity*(*is_recfun_fm*(*p*, *a*, *z* ,*r*)) = *?ar*
⟨*proof*⟩

schematic_goal *arity_is_wfrec_fm*[*arity*]:
p ∈ *formula* ⟹ *a* ∈ ω ⟹ *z* ∈ ω ⟹ *r* ∈ ω ⟹ *arity*(*is_wfrec_fm*(*p*, *a*, *z* ,*r*)) = *?ar*
⟨*proof*⟩
schematic_goal *arity_is_transrec_fm*[*arity*]:
p ∈ *formula* ⟹ *a* ∈ ω ⟹ *z* ∈ ω ⟹ *arity*(*is_transrec_fm*(*p*, *a*, *z*)) = *?ar*
⟨*proof*⟩

⟨*ML*⟩

definition *transrec_apply_image_body* **where**
transrec_apply_image_body(*M*,*f*,*mesa*,*x*,*z*) ≡ ∃ *y*[*M*]. *pair*(*M*, *x*, *y*, *z*) ∧
(∃ *fa*[*M*].
(∀ *z*[*M*].
z ∈ *fa* ⟷
(∃ *xa*[*M*].
∃ *y*[*M*].
∃ *xaa*[*M*].
∃ *sx*[*M*].
∃ *r_sx*[*M*].
∃ *f_r_sx*[*M*].

$$pair(M,\ xa,\ y,\ z)\ \wedge$$
$$pair(M,\ xa,\ x,\ xaa)\ \wedge$$
$$upair(M,\ xa,\ xa,\ sx)\ \wedge$$
$$pre\_image(M,\ mesa,\ sx,\ r\_sx)\ \wedge$$
$$restriction(M,\ fa,\ r\_sx,\ f\_r\_sx)\ \wedge$$
$$xaa \in mesa \wedge is\_trans\_apply\_image(M,\ f,\ xa,\ f\_r\_sx,\ y)))\ \wedge$$
$$is\_trans\_apply\_image(M,\ f,\ x,\ fa,\ y))$$

⟨*ML*⟩

definition *is_trans_apply_image_body* **where**
is_trans_apply_image_body(*M*,*f*,*β*,*a*,*w*) ≡ ∃ *z*[*M*]. *pair*(*M*,*a*,*z*,*w*) ∧ *a*∈*β* ∧ (∃ *sa*[*M*].
∃ *esa*[*M*].
∃ *mesa*[*M*].
upair(*M*, *a*, *a*, *sa*) ∧
is_eclose(*M*, *sa*, *esa*) ∧
membership(*M*, *esa*, *mesa*) ∧
(∃ *fa*[*M*].
(∀ *z*[*M*].
z ∈ *fa* ⟷
(∃ *x*[*M*].
∃ *y*[*M*].
∃ *xa*[*M*].
∃ *sx*[*M*].
∃ *r_sx*[*M*].
∃ *f_r_sx*[*M*].
pair(*M*, *x*, *y*, *z*) ∧
pair(*M*, *x*, *a*, *xa*) ∧
upair(*M*, *x*, *x*, *sx*) ∧
pre_image(*M*, *mesa*, *sx*, *r_sx*) ∧
restriction(*M*, *fa*, *r_sx*, *f_r_sx*) ∧
xa ∈ *mesa* ∧ *is_trans_apply_image*(*M*, *f*, *x*, *f_r_sx*, *y*))) ∧
is_trans_apply_image(*M*, *f*, *a*, *fa*, *z*)))

⟨*ML*⟩

definition *replacement_is_omega_funspace_fm* **where** *replacement_is_omega_funspace_fm* ≡ *omega_funspace_fm*(2,0,1)
definition *wfrec_Aleph_fm* **where** *wfrec_Aleph_fm* ≡ *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*
  ⟨*proof*⟩

locale *M_ZF_ground_notCH* = *M_ZF_ground* +
  assumes
    ZF_ground_notCH_replacements:
    replacement_assm(*M*,*env*,*rec_constr_abs_fm*)
    replacement_assm(*M*,*env*,*rec_constr_fm*)

definition *instances_ground_notCH_fms* **where** *instances_ground_notCH_fms*
≡
  { *rec_constr_abs_fm*,
    rec_constr_fm }

lemma *instances_ground_notCH_fms_type*[*TC*]: *instances_ground_notCH_fms*
⊆ *formula*
  ⟨*proof*⟩

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

locale *M_ZF_ground_notCH_trans* = *M_ZF_ground_trans* + *M_ZF_ground_notCH*

locale *M_ZF_ground_CH* = *M_ZF_ground_notCH* +
  assumes
    dcwit_replacement: *replacement_assm*(*M*,*env*,*dc_abs_fm*)

declare (**in** *M_ZF_ground_CH*) *dc_abs_fm_def* [*simp*]

locale *M_ZF_ground_CH_trans* = *M_ZF_ground_notCH_trans* + *M_ZF_ground_CH*

locale *M_ctm1* = *M_ZF1_trans* + *M_ZF_ground_trans* +
  fixes *enum*
  assumes *M_countable*: *enum*∈*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*))
  ⟨*proof*⟩

lemma *dcwit_repl*:
  (##*M*)(*sa*) ⟹
       (##*M*)(*esa*) ⟹
           (##*M*)(*mesa*) ⟹ (##*M*)(*A*) ⟹ (##*M*)(*a*) ⟹ (##*M*)(*s*) ⟹
(##*M*)(*R*) ⟹
       strong_replacement
            ((##*M*), λ*x z*. ∃ *y*[(##*M*)]. *pair*((##*M*), *x*, *y*, *z*) ∧
                        is_wfrec
                         ((##*M*), λ*n f*. *is_nat_case*
                                   ((##*M*), *a*,
                                    λ*m bmfm*.
                                       ∃ *fm*[(##*M*)].
                                          ∃ *cp*[(##*M*)].
                                             is_apply((##*M*), *f*, *m*, *fm*) ∧
                                           is_Collect((##*M*), *A*, λ*x*. ∃ *fmx*[(##*M*)].
((##*M*)(*x*) ∧ *fmx* ∈ *R*) ∧ *pair*((##*M*), *fm*, *x*, *fmx*), *cp*) ∧
                                                   is_apply((##*M*), *s*, *cp*, *bmfm*),
                                    n),
                          mesa, *x*, *y*))
  ⟨*proof*⟩

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*$(\#\#M,\ A,\ a,\ g,\ R))$
⟨*proof*⟩

end — *M_ZF1_trans*

sublocale *M_ZF1_trans* $\subseteq$ *M_replacement* $\#\#M$
⟨*proof*⟩

context *M_ZF1_trans*
begin

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

lemma *strong_replacement_Powapply_rel*:
$f{\in}M \Longrightarrow$ *strong_replacement*$(\#\#M,\ \lambda x\ y.\ y =$ *Powapply*$^{\#\#M}(f,x))$
⟨*proof*⟩

end — *M_ZF1_trans*

sublocale *M_ZF1_trans* $\subseteq$ *M_Vfrom* $\#\#M$
⟨*proof*⟩

sublocale *M_ZF1_trans* $\subseteq$ *M_Perm* $\#\#M$
⟨*proof*⟩

sublocale *M_ZF1_trans* $\subseteq$ *M_pre_seqspace* $\#\#M$
⟨*proof*⟩

context *M_ZF1_trans*
begin

lemma *separation_inj_rel*: $A{\in}M \Longrightarrow$
separation$(\#\#M,\ \lambda y.\ \exists\, x{\in}M.\ x \in A \wedge y = \langle x,$ *inj_rel*$(\#\#M,$*fst*$(x),$ *snd*$(x))\rangle)$
⟨*proof*⟩

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

⟨*proof*⟩

end — *M_ZF1_trans*

lemma (**in** *M_basic*) *rel2_trans_apply*:
M(*f*) ⟹ *relation2*(*M*,*is_trans_apply_image*(*M*,*f*),*trans_apply_image*(*f*))
⟨*proof*⟩

lemma (**in** *M_basic*) *apply_image_closed*:
shows *M*(*f*) ⟹ ∀ *x*[*M*]. ∀ *g*[*M*]. *M*(*trans_apply_image*(*f*, *x*, *g*))
⟨*proof*⟩

context *M_ZF_ground_notCH_trans*
begin

lemma *replacement_transrec_apply_image_body* :
(##*M*)(*f*) ⟹ (##*M*)(*mesa*) ⟹ *strong_replacement*(##*M*,*transrec_apply_image_body*(##*M*,*f*,*mesa*))
⟨*proof*⟩

lemma *transrec_replacement_apply_image*:
assumes (##*M*)(*f*) (##*M*)(α)
shows *transrec_replacement*(##*M*, *is_trans_apply_image*(##*M*, *f*), α)
⟨*proof*⟩

lemma *rec_trans_apply_image_abs*:
assumes (##*M*)(*f*) (##*M*)(*x*) (##*M*)(*y*) *Ord*(*x*)
shows *is_transrec*(##*M*,*is_trans_apply_image*(##*M*, *f*),*x*,*y*) ⟷ *y* = *transrec*(*x*,*trans_apply_image*(*f*))
⟨*proof*⟩

lemma *replacement_is_trans_apply_image*:
(##*M*)(*f*) ⟹ (##*M*)(β) ⟹ *strong_replacement*(##*M*, λ *x z* .
∃ *y*[##*M*]. *pair*(##*M*,*x*,*y*,*z*) ∧ *x*∈β ∧ (*is_transrec*(##*M*,*is_trans_apply_image*(##*M*, *f*),*x*,*y*)))
⟨*proof*⟩

lemma *trans_apply_abs*:
(##*M*)(*f*) ⟹ (##*M*)(β) ⟹ *Ord*(β) ⟹ (##*M*)(*x*) ⟹ (##*M*)(*z*) ⟹
(*x*∈β ∧ *z* = ⟨*x*, *transrec*(*x*, λ*a g*. *f* ‘ (*g* ‘‘ *a*)) ⟩) ⟷
(∃ *y*[##*M*]. *pair*(##*M*,*x*,*y*,*z*) ∧ *x*∈β ∧ (*is_transrec*(##*M*,*is_trans_apply_image*(##*M*, *f*),*x*,*y*)))
⟨*proof*⟩

lemma *replacement_trans_apply_image*:
(##*M*)(*f*) ⟹ (##*M*)(β) ⟹ *Ord*(β) ⟹
strong_replacement(##*M*, λ*x y*. *x*∈β ∧ *y* = ⟨*x*, *transrec*(*x*, λ*a g*. *f* ‘ (*g* ‘‘ *a*))))
⟨*proof*⟩

end — *M_ZF_ground_notCH_trans*

definition *ifrFb_body* **where**
ifrFb_body(*M*,*b*,*f*,*x*,*i*) ≡ *x* ∈
(*if b* = *0 then if i* ∈ *range*(*f*) *then*
if M(*converse*(*f*) ‘ *i*) *then converse*(*f*) ‘ *i else 0 else 0 else if M*(*i*) *then i else 0*)

⟨*ML*⟩

definition *ifrangeF_body* :: [*i*⇒*o*,*i*,*i*,*i*,*i*] ⇒ *o* **where**
ifrangeF_body(*M*,*A*,*b*,*f*) ≡ λ*y*. ∃ *x*∈*A*. *y* = ⟨*x*,μ *i*. *ifrFb_body*(*M*,*b*,*f*,*x*,*i*)⟩

⟨*ML*⟩

lemma (**in** *M_Z_trans*) *separation_is_ifrangeF_body*:
(##*M*)(*A*) ⟹ (##*M*)(*r*) ⟹ (##*M*)(*s*) ⟹ *separation*(##*M*, *is_ifrangeF_body*(##*M*,*A*,*r*,*s*))
⟨*proof*⟩

lemma (**in** *M_basic*) *is_ifrFb_body_closed*: *M*(*r*) ⟹ *M*(*s*) ⟹ *is_ifrFb_body*(*M*,
r, *s*, *x*, *i*) ⟹ *M*(*i*)
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *ifrangeF_body_abs*:
assumes (##*M*)(*A*) (##*M*)(*r*) (##*M*)(*s*) (##*M*)(*x*)
shows *is_ifrangeF_body*(##*M*,*A*,*r*,*s*,*x*) ⟷ *ifrangeF_body*(##*M*,*A*,*r*,*s*,*x*)
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *separation_ifrangeF_body*:
(##*M*)(*A*) ⟹ (##*M*)(*b*) ⟹ (##*M*)(*f*) ⟹ *separation*
(##*M*, λ*y*. ∃ *x*∈*A*. *y* = ⟨*x*, μ *i*. *x* ∈ *if_range_F_else_F*(λ*x*. *if* (##*M*)(*x*)
then x else 0, *b*, *f*, *i*)⟩)
⟨*proof*⟩

definition *ifrFb_body2* **where**
ifrFb_body2(*M*,*G*,*b*,*f*,*x*,*i*) ≡ *x* ∈
(*if b* = *0 then if i* ∈ *range*(*f*) *then*
if M(*converse*(*f*) ‘ *i*) *then G*‘(*converse*(*f*) ‘ *i*) *else 0 else 0 else if M*(*i*) *then G*‘*i*
else 0)

⟨*ML*⟩

definition *ifrangeF_body2* :: [*i*⇒*o*,*i*,*i*,*i*,*i*,*i*] ⇒ *o* **where**
ifrangeF_body2(*M*,*A*,*G*,*b*,*f*) ≡ λ*y*. ∃ *x*∈*A*. *y* = ⟨*x*,μ *i*. *ifrFb_body2*(*M*,*G*,*b*,*f*,*x*,*i*)⟩

⟨*ML*⟩

lemma (**in** *M_Z_trans*) *separation_is_ifrangeF_body2*:
(##*M*)(*A*) ⟹ (##*M*)(*G*) ⟹ (##*M*)(*r*) ⟹ (##*M*)(*s*) ⟹ *separation*(##*M*, *is_ifrangeF_body2*(##*M*,*A*,*G*,*r*,*s*))
⟨*proof*⟩

lemma (**in** *M_basic*) *is_ifrFb_body2_closed*: *M*(*G*) ⟹ *M*(*r*) ⟹ *M*(*s*) ⟹ *is_ifrFb_body2*(*M*, *G*, *r*, *s*, *x*, *i*) ⟹ *M*(*i*)
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *ifrangeF_body2_abs*:
assumes (##*M*)(*A*) (##*M*)(*G*) (##*M*)(*r*) (##*M*)(*s*) (##*M*)(*x*)
shows *is_ifrangeF_body2*(##*M*,*A*,*G*,*r*,*s*,*x*) ⟷ *ifrangeF_body2*(##*M*,*A*,*G*,*r*,*s*,*x*)
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *separation_ifrangeF_body2*:
(##*M*)(*A*) ⟹ (##*M*)(*G*) ⟹ (##*M*)(*b*) ⟹ (##*M*)(*f*) ⟹
separation
(##*M*,
λ*y*. ∃ *x*∈*A*.
y =
⟨*x*, μ *i*. *x* ∈
if_range_F_else_F(λ*a*. *if* (##*M*)(*a*) *then* *G* ‘ *a* *else* 0, *b*, *f*, *i*)⟩)
⟨*proof*⟩

definition *ifrFb_body3* **where**
ifrFb_body3(*M*,*G*,*b*,*f*,*x*,*i*) ≡ *x* ∈
(*if* *b* = 0 *then* *if* *i* ∈ *range*(*f*) *then*
if *M*(*converse*(*f*) ‘ *i*) *then* *G*-‘‘{*converse*(*f*) ‘ *i*} *else* 0 *else* 0 *else* *if* *M*(*i*) *then* *G*-‘‘{*i*} *else* 0)

⟨*ML*⟩

definition *ifrangeF_body3* :: [*i*⇒*o*,*i*,*i*,*i*,*i*,*i*] ⇒ *o* **where**
ifrangeF_body3(*M*,*A*,*G*,*b*,*f*) ≡ λ*y*. ∃ *x*∈*A*. *y* = ⟨*x*,μ *i*. *ifrFb_body3*(*M*,*G*,*b*,*f*,*x*,*i*)⟩

⟨*ML*⟩

lemma (**in** *M_Z_trans*) *separation_is_ifrangeF_body3*:
(##*M*)(*A*) ⟹ (##*M*)(*G*) ⟹ (##*M*)(*r*) ⟹ (##*M*)(*s*) ⟹ *separation*(##*M*, *is_ifrangeF_body3*(##*M*,*A*,*G*,*r*,*s*))
⟨*proof*⟩

lemma (**in** *M_basic*) *is_ifrFb_body3_closed*: *M*(*G*) ⟹ *M*(*r*) ⟹ *M*(*s*) ⟹ *is_ifrFb_body3*(*M*, *G*, *r*, *s*, *x*, *i*) ⟹ *M*(*i*)
⟨*proof*⟩

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

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

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* $G`i$)

⟨*ML*⟩

definition *ifrangeF_body4* :: $[i{\Rightarrow}o,i,i,i,i,i] \Rightarrow o$ **where**
ifrangeF_body4$(M,A,G,b,f) \equiv \lambda y.\ \exists\, x{\in}A.\ y = \langle x,\mu\ i.$ *ifrFb_body4*$(G,b,f,x,i)\rangle$

⟨*ML*⟩

lemma (**in** *M_Z_trans*) *separation_is_ifrangeF_body4*:
$(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(s) \Longrightarrow$ *separation*$(\#\#M,$
is_ifrangeF_body4$(\#\#M,A,G,r,s))$
⟨*proof*⟩

lemma (**in** *M_basic*) *is_ifrFb_body4_closed*: $M(G) \Longrightarrow M(r) \Longrightarrow M(s) \Longrightarrow$
is_ifrFb_body4$(M,\ G,\ r,\ s,\ x,\ i) \Longrightarrow M(i)$
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *ifrangeF_body4_abs*:
assumes $(\#\#M)(A)$ $(\#\#M)(G)$ $(\#\#M)(r)$ $(\#\#M)(s)$ $(\#\#M)(x)$
shows *is_ifrangeF_body4*$(\#\#M,A,G,r,s,x) \longleftrightarrow$ *ifrangeF_body4*$(\#\#M,A,G,r,s,x)$
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *separation_ifrangeF_body4*:
$(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(b) \Longrightarrow (\#\#M)(f) \Longrightarrow$
separation$(\#\#M,\ \lambda y.\ \exists\, x{\in}A.\ y = \langle x,\ \mu\ i.\ x \in$ *if_range_F_else_F*$((`)(G),$
$b,\ f,\ i)\rangle)$
⟨*proof*⟩

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*})

⟨*ML*⟩

definition *ifrangeF_body5* :: [*i*⇒*o*,*i*,*i*,*i*,*i*,*i*] ⇒ *o* **where**
ifrangeF_body5(*M*,*A*,*G*,*b*,*f*) ≡ λ*y*. ∃ *x*∈*A*. *y* = ⟨*x*,*μ i*. *ifrFb_body5*(*G*,*b*,*f*,*x*,*i*)⟩

⟨*ML*⟩

lemma (**in** *M_Z_trans*) *separation_is_ifrangeF_body5*:
(##*M*)(*A*) ⟹ (##*M*)(*G*) ⟹ (##*M*)(*r*) ⟹ (##*M*)(*s*) ⟹ *separation*(##*M*, *is_ifrangeF_body5*(##*M*,*A*,*G*,*r*,*s*))
⟨*proof*⟩

lemma (**in** *M_basic*) *is_ifrFb_body5_closed*: *M*(*G*) ⟹ *M*(*r*) ⟹ *M*(*s*) ⟹ *is_ifrFb_body5*(*M*, *G*, *r*, *s*, *x*, *i*) ⟹ *M*(*i*)
⟨*proof*⟩

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*) ⟷ *ifrangeF_body5*(##*M*,*A*,*G*,*r*,*s*,*x*)
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *separation_ifrangeF_body5*:
(##*M*)(*A*) ⟹ (##*M*)(*G*) ⟹ (##*M*)(*b*) ⟹ (##*M*)(*f*) ⟹
separation(##*M*, λ*y*. ∃ *x*∈*A*. *y* = ⟨*x*, *μ i*. *x* ∈ *if_range_F_else_F*(λ*x*. {*xa* ∈ *G* . *x* ∈ *xa*}, *b*, *f*, *i*)⟩)
⟨*proof*⟩

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*})

⟨*ML*⟩

definition *ifrangeF_body6* :: [*i*⇒*o*,*i*,*i*,*i*,*i*,*i*] ⇒ *o* **where**
ifrangeF_body6(*M*,*A*,*G*,*b*,*f*) ≡ λ*y*. ∃ *x*∈*A*. *y* = ⟨*x*,*μ i*. *ifrFb_body6*(*G*,*b*,*f*,*x*,*i*)⟩

⟨*ML*⟩

lemma (**in** *M_Z_trans*) *separation_is_ifrangeF_body6*:

$(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(s) \Longrightarrow separation(\#\#M,$
$is\_ifrangeF\_body6(\#\#M,A,G,r,s))$
⟨*proof*⟩

lemma (**in** *M_basic*) *ifrFb_body6_closed*: $M(G) \Longrightarrow M(r) \Longrightarrow M(s) \Longrightarrow ifrFb\_body6(G,$
$r,\ s,\ x,\ i) \longleftrightarrow\ M(i) \wedge ifrFb\_body6(G,\ r,\ s,\ x,\ i)$
⟨*proof*⟩

lemma (**in** *M_basic*) *is_ifrFb_body6_closed*: $M(G) \Longrightarrow M(r) \Longrightarrow M(s) \Longrightarrow$
$is\_ifrFb\_body6(M,\ G,\ r,\ s,\ x,\ i) \Longrightarrow M(i)$
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *ifrangeF_body6_abs*:
assumes $(\#\#M)(A)\ (\#\#M)(G)\ (\#\#M)(r)\ (\#\#M)(s)\ (\#\#M)(x)$
shows $is\_ifrangeF\_body6(\#\#M,A,G,r,s,x) \longleftrightarrow ifrangeF\_body6(\#\#M,A,G,r,s,x)$
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *separation_ifrangeF_body6*:
$(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(b) \Longrightarrow (\#\#M)(f) \Longrightarrow$
$separation(\#\#M,$
$\lambda y.\ \exists\, x{\in}A.\ y = \langle x,\ \mu\ i.\ x \in if\_range\_F\_else\_F(\lambda a.\ \{p \in G\ .\ domain(p) =$
$a\},\ b,\ f,\ i)\rangle)$
⟨*proof*⟩

definition *ifrFb_body7* **where**
$ifrFb\_body7(B,D,A,b,f,x,i) \equiv x \in$
$(if\ b = 0\ then\ if\ i \in range(f)\ then$
$\{d \in D\ .\ \exists\, r{\in}A.\ restrict(r,\ B) = converse(f)\ `\ i \wedge d = domain(r)\}\ else\ 0$
$else\ \{d \in D\ .\ \exists\, r{\in}A.\ restrict(r,\ B) = i \wedge d = domain(r)\})$

⟨*ML*⟩

definition *ifrangeF_body7* :: $[i{\Rightarrow}o,i,i,i,i,i,i,i] \Rightarrow o$ **where**
$ifrangeF\_body7(M,A,B,D,G,b,f) \equiv \lambda y.\ \exists\, x{\in}A.\ y = \langle x,\mu\ i.\ ifrFb\_body7(B,D,G,b,f,x,i)\rangle$

⟨*ML*⟩

lemma (**in** *M_Z_trans*) *separation_is_ifrangeF_body7*:
$(\#\#M)(A) \Longrightarrow (\#\#M)(B) \Longrightarrow (\#\#M)(D) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(r)$
$\Longrightarrow (\#\#M)(s) \Longrightarrow separation(\#\#M,\ is\_ifrangeF\_body7(\#\#M,A,B,D,G,r,s))$
⟨*proof*⟩

lemma (**in** *M_basic*) *ifrFb_body7_closed*: $M(B) \Longrightarrow M(D) \Longrightarrow M(G) \Longrightarrow M(r)$
$\Longrightarrow M(s) \Longrightarrow$
$ifrFb\_body7(B,D,G,\ r,\ s,\ x,\ i) \longleftrightarrow\ M(i) \wedge ifrFb\_body7(B,D,G,\ r,\ s,\ x,\ i)$

⟨*proof*⟩

lemma (**in** *M_basic*) *is_ifrFb_body7_closed*: $M(B) \Longrightarrow M(D) \Longrightarrow M(G) \Longrightarrow M(r) \Longrightarrow M(s) \Longrightarrow$
is_ifrFb_body7$(M,\ B,D,G,\ r,\ s,\ x,\ i) \Longrightarrow M(i)$
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *ifrangeF_body7_abs*:
assumes $(\#\#M)(A)$ $(\#\#M)(B)$ $(\#\#M)(D)$ $(\#\#M)(G)$ $(\#\#M)(r)$ $(\#\#M)(s)$ $(\#\#M)(x)$
shows *is_ifrangeF_body7*$(\#\#M,A,B,D,G,r,s,x) \longleftrightarrow$ *ifrangeF_body7*$(\#\#M,A,B,D,G,r,s,x)$
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *separation_ifrangeF_body7*:
$(\#\#M)(A) \Longrightarrow (\#\#M)(B) \Longrightarrow (\#\#M)(D) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(b) \Longrightarrow (\#\#M)(f) \Longrightarrow$
separation$(\#\#M,$
$\lambda y.\ \exists\, x{\in}A.\ y = \langle x,\ \mu\ i.\ x \in$ *if_range_F_else_F*$(drSR\_Y(B,\ D,\ G),\ b,\ f,\ i)\rangle)$
⟨*proof*⟩

definition *omfunspace* :: $[i,i] \Rightarrow o$ **where**
omfunspace$(B) \equiv \lambda z.\ \exists\, x.\ \exists\, n{\in}\omega.\ z{\in}x \wedge x = n{\to}B$
⟨*ML*⟩

context *M_pre_seqspace*
begin

⟨*ML*⟩
⟨*proof*⟩

end — *M_pre_seqspace*

context *M_ZF1_trans*
begin

lemma *separation_omfunspace*:
assumes $(\#\#M)(B)$
shows *separation*$(\#\#M,\ \lambda z.\ \exists\, x[\#\#M].\ \exists\, n[\#\#M].\ n \in \omega \wedge z \in x \wedge x = n \to^M B)$
⟨*proof*⟩

end — *M_ZF1_trans*

sublocale *M_ZF1_trans* $\subseteq$ *M_seqspace* $\#\#M$
⟨*proof*⟩

definition *cdltgamma* :: $[i,i] \Rightarrow o$ **where**
cdltgamma$(\gamma) \equiv \lambda Z\ .\ |Z| < \gamma$
⟨*ML*⟩

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

context *M_Perm*
begin

⟨*ML*⟩
  ⟨*proof*⟩

⟨*ML*⟩
  ⟨*proof*⟩

lemma *is_cdeqgamma_iff_split*: $M(Z) \Longrightarrow cdeqgamma\_rel(M,\ Z) \longleftrightarrow (\lambda\langle x,y\rangle.\ |x|^M = y)(Z)$
  ⟨*proof*⟩

end

context *M_ZF1_trans*
begin

lemma *separation_cdltgamma*:
  assumes $(\#\#M)(\gamma)$
  shows $separation(\#\#M,\ \lambda Z\ .\ cardinal\_rel(\#\#M,Z) < \gamma)$
  ⟨*proof*⟩

lemma *separation_cdeqgamma*:
  shows $separation(\#\#M,\ \lambda Z.\ (\lambda\langle x,y\rangle\ .\ cardinal\_rel(\#\#M,x) = y)(Z))$
  ⟨*proof*⟩

end — *M_ZF1_trans*

end

9 Further instances of axiom-schemes

theory *ZF_Trans_Interpretations*
  imports
    Internal_ZFC_Axioms
    Replacement_Instances

begin

locale *M_ZF2* = *M_ZF1* +
  assumes
    replacement_ax2:
    $replacement\_assm(M,env,ordtype\_replacement\_fm)$

replacement_assm(*M*,*env*,*wfrec_ordertype_fm*)
replacement_assm(*M*,*env*,*wfrec_Aleph_fm*)
replacement_assm(*M*,*env*,*omap_replacement_fm*)

definition *instances2_fms* **where** *instances2_fms* ≡
{ *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*:
$B{\in}M \Longrightarrow$ *strong_replacement*$(\#\#M,$ *HAleph_wfrec_repl_body*$(\#\#M,B))$
⟨*proof*⟩

lemma *HAleph_wfrec_repl*:
$(\#\#M)(sa) \Longrightarrow$
$(\#\#M)(esa) \Longrightarrow$
$(\#\#M)(mesa) \Longrightarrow$
strong_replacement
$(\#\#M,$
$\lambda x\ z.\ \exists\, y[\#\#M].$
$pair(\#\#M,\ x,\ y,\ z)\ \wedge$
$(\exists\, f[\#\#M].$
$(\forall\, z[\#\#M].$
$z \in f \longleftrightarrow$
$(\exists\, xa[\#\#M].$
$\exists\, y[\#\#M].$
$\exists\, xaa[\#\#M].$
$\exists\, sx[\#\#M].$
$\exists\, r\_sx[\#\#M].$
$\exists\, f\_r\_sx[\#\#M].$
$pair(\#\#M,\ xa,\ y,\ z)\ \wedge$
$pair(\#\#M,\ xa,\ x,\ xaa)\ \wedge$
$upair(\#\#M,\ xa,\ xa,\ sx)\ \wedge$
$pre\_image(\#\#M,\ mesa,\ sx,\ r\_sx)\ \wedge$
$restriction(\#\#M, f, r\_sx, f\_r\_sx) \wedge xaa \in mesa \wedge is\_HAleph(\#\#M, xa, f\_r\_sx, y))) \wedge$
$is\_HAleph(\#\#M,\ x,\ f,\ y)))$
⟨*proof*⟩

lemma *replacement_is_order_eq_map*:
$A{\in}M \Longrightarrow r{\in}M \Longrightarrow$ *strong_replacement*$(\#\#M,$ *order_eq_map*$(\#\#M,A,r))$
⟨*proof*⟩

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* :$A{\in}nat \Longrightarrow r{\in}nat \Longrightarrow omap\_wfrec\_body(A,r){\in}formula$
⟨*proof*⟩

lemma *arity_aux* : $A{\in}nat \Longrightarrow r{\in}nat \Longrightarrow arity(omap\_wfrec\_body(A,r)) = (9{+}_\omega A) \cup (9{+}_\omega r)$
⟨*proof*⟩

lemma *arity_omap_wfrec*: $A{\in}nat \Longrightarrow r{\in}nat \Longrightarrow$
$arity(is\_wfrec\_fm(omap\_wfrec\_body(A,r),succ(succ(succ(r))),\ 1,\ 0)) =$

$(4+_\omega A) \cup (4+_\omega r)$
⟨*proof*⟩

lemma *arity_isordermap*: $A{\in}nat \Longrightarrow r{\in}nat \Longrightarrow d{\in}nat \Longrightarrow$
$arity(is\_ordermap\_fm(A,r,d)) = succ(d) \cup (succ(A) \cup succ(r))$
⟨*proof*⟩

lemma *arity_is_ordertype*: $A{\in}nat \Longrightarrow r{\in}nat \Longrightarrow d{\in}nat \Longrightarrow$
$arity(is\_ordertype\_fm(A,r,d)) = succ(d) \cup (succ(A) \cup succ(r))$
⟨*proof*⟩

lemma *arity_is_order_body*: $arity(is\_order\_body\_fm(1,0)) = 2$
⟨*proof*⟩

lemma (**in** *M_ZF2_trans*) *replacement_is_order_body*:
strong_replacement(##*M*, $\lambda x\ z\ .\ \exists\, y[\#\#M].$ *is_order_body*(##*M*,*x*,*y*) $\wedge$ $z = \langle x,y\rangle$)
⟨*proof*⟩

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

⟨*ML*⟩

lemma (**in** *M_basic*) *H_order_pred_abs* :
$M(A) \Longrightarrow M(r) \Longrightarrow M(x) \Longrightarrow M(f) \Longrightarrow M(z) \Longrightarrow$
is_H_order_pred(*M*,*A*,*r*,*x*,*f*,*z*) $\longleftrightarrow$ $z =$ *H_order_pred*(*A*,*r*,*x*,*f*)
⟨*proof*⟩

⟨*ML*⟩

lemma (**in** *M_ZF2_trans*) *wfrec_replacement_order_pred*:
$A{\in}M \Longrightarrow r{\in}M \Longrightarrow$ *wfrec_replacement*(##*M*, $\lambda x\ g\ z.$ *is_H_order_pred*(##*M*,*A*,*r*,*x*,*g*,*z*), *r*)
⟨*proof*⟩

lemma (**in** *M_ZF2_trans*) *wfrec_replacement_order_pred'*:
$A{\in}M \Longrightarrow r{\in}M \Longrightarrow$ *wfrec_replacement*(##*M*, $\lambda x\ g\ z.\ z =$ *H_order_pred*(*A*,*r*,*x*,*g*), *r*)
⟨*proof*⟩

sublocale *M_ZF2_trans* $\subseteq$ *M_pre_cardinal_arith* ##*M*
⟨*proof*⟩

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*))

⟨*ML*⟩

lemma (**in** *M_ZF2_trans*) *separation_well_ord*: *separation*(*##M*, λ*x*. *is_well_ord*(*##M*,*fst*(*x*), *snd*(*x*)))
⟨*proof*⟩

sublocale *M_ZF2_trans* ⊆ *M_pre_aleph* *##M*
⟨*proof*⟩

⟨*ML*⟩
lemma *arity_is_HAleph_fm*: *arity*(*is_HAleph_fm*(*2*, *1*, *0*)) = *3*
⟨*proof*⟩

lemma *arity_is_Aleph*[*arity*]: *arity*(*is_Aleph_fm*(*0*, *1*)) = *2*
⟨*proof*⟩

definition *bex_Aleph_rel* :: [*i*⇒*o*,*i*,*i*] ⇒ *o* **where**
bex_Aleph_rel(*M*,*x*) ≡ λ*y*. ∃ *z*∈*x*. *y* = ℵ$_z{}^M$

⟨*ML*⟩

schematic_goal *sats_is_bex_Aleph_fm_auto*:
a ∈ *nat* ⟹ *c* ∈ *nat* ⟹ *env* ∈ *list*(*A*) ⟹
a < *length*(*env*) ⟹ *c* < *length*(*env*) ⟹ *0* ∈ *A* ⟹
is_bex_Aleph(*##A*, *nth*(*a*, *env*), *nth*(*c*, *env*)) ⟷ *A*, *env* ⊨ *?fm*(*a*, *c*)
⟨*proof*⟩

⟨*ML*⟩

lemma *is_bex_Aleph_fm_type* [*TC*]:
x ∈ ω ⟹ *z* ∈ ω ⟹ *is_bex_Aleph_fm*(*x*, *z*) ∈ *formula*
⟨*proof*⟩

lemma *sats_is_bex_Aleph_fm*:
x ∈ ω ⟹
z ∈ ω ⟹ *x* < *length*(*env*) ⟹ *z* < *length*(*env*) ⟹
env ∈ *list*(*Aa*) ⟹
0 ∈ *Aa* ⟹
(*Aa*, *env* ⊨ *is_bex_Aleph_fm*(*x*, *z*)) ⟷
is_bex_Aleph(*##Aa*,*nth*(*x*, *env*), *nth*(*z*, *env*))
⟨*proof*⟩

lemma *is_bex_Aleph_iff_sats* [*iff_sats*]:
nth(*x*, *env*) = *xa* ⟹
nth(*z*, *env*) = *za* ⟹
x ∈ ω ⟹
z ∈ ω ⟹ *x* < *length*(*env*) ⟹ *z* < *length*(*env*) ⟹

$env \in list(Aa) \Longrightarrow$
$0 \in Aa \Longrightarrow$
$is\_bex\_Aleph(\#\#Aa,\ xa,\ za) \longleftrightarrow$
$Aa,\ env \models is\_bex\_Aleph\_fm(x,\ z)$
⟨*proof*⟩

⟨*ML*⟩

lemma (**in** *M_ZF1_trans*) *separation_is_bex_Aleph*:
assumes $(\#\#M)(A)$
shows $separation(\#\#M, is\_bex\_Aleph(\#\#M,\ A))$
⟨*proof*⟩

lemma (**in** *M_pre_aleph*) *bex_Aleph_rel_abs*:
assumes $Ord(u)\ M(u)\ M(v)$
shows $is\_bex\_Aleph(M,\ u,\ v) \longleftrightarrow bex\_Aleph\_rel(M,u,v)$
⟨*proof*⟩

lemma (**in** *M_ZF2_trans*) *separation_bex_Aleph_rel*:
$Ord(x) \Longrightarrow (\#\#M)(x) \Longrightarrow separation(\#\#M,\ bex\_Aleph\_rel(\#\#M,x))$
⟨*proof*⟩

sublocale $M\_ZF2\_trans \subseteq M\_aleph\ \#\#M$
⟨*proof*⟩

sublocale $M\_ZF1\_trans \subseteq M\_FiniteFun\ \#\#M$
⟨*proof*⟩

sublocale $M\_ZFC2\_trans \subseteq M\_cardinal\_AC\ \#\#M$
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *separation_cardinal_rel_lesspoll_rel*:
$(\#\#M)(\kappa) \Longrightarrow separation(\#\#M,\ \lambda x.\ x \prec^M \kappa)$
⟨*proof*⟩

sublocale $M\_ZFC2\_trans \subseteq M\_library\ \#\#M$
⟨*proof*⟩

locale $M\_ZF3 = M\_ZF2 +$
assumes
ground_replacements3:
$ground\_replacement\_assm(M,env,ordtype\_replacement\_fm)$
$ground\_replacement\_assm(M,env,wfrec\_ordertype\_fm)$
$ground\_replacement\_assm(M,env,eclose\_abs\_fm)$
$ground\_replacement\_assm(M,env,wfrec\_rank\_fm)$
$ground\_replacement\_assm(M,env,transrec\_VFrom\_fm)$
$ground\_replacement\_assm(M,env,eclose\_closed\_fm)$

ground_replacement_assm(M,*env*,*wfrec_Aleph_fm*)
ground_replacement_assm(M,*env*,*omap_replacement_fm*)

definition *instances3_fms* **where** *instances3_fms* ≡
{ *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 ≡ *instances1_fms* ∪ *instances_ground_fms*

definition *overhead_notCH* **where**
overhead_notCH ≡ *overhead* ∪ *instances2_fms* ∪
instances3_fms ∪ *instances_ground_notCH_fms*

definition *overhead_CH* **where**
overhead_CH ≡ *overhead_notCH* ∪ { *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* ⊆ *formula*
⟨*proof*⟩

lemma *overhead_type*: *overhead* ⊆ *formula*
⟨*proof*⟩

lemma *overhead_notCH_type*: *overhead_notCH* ⊆ *formula*
⟨*proof*⟩

lemma *overhead_CH_type*: *overhead_CH* ⊆ *formula*
⟨*proof*⟩

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*⟩

lemma *M_satT_imp_M_ZFC1*:
shows (*M* ⊨ *ZFC*) ⟶ *M_ZFC1*(*M*)
⟨*proof*⟩

lemma *M_satT_instances1_imp_M_ZF1*:
assumes (*M* ⊨ ·*Z*· ∪ {·*Replacement*(*p*)· . *p* ∈ *instances1_fms* })
shows *M_ZF1*(*M*)
⟨*proof*⟩

theorem *M_satT_imp_M_ZF_ground_trans*:
assumes *Transset*(*M*) *M* ⊨ ·*Z*· ∪ {·*Replacement*(*p*)· . *p* ∈ *overhead*}
shows *M_ZF_ground_trans*(*M*)
⟨*proof*⟩

theorem *M_satT_imp_M_ZF_ground_notCH_trans*:
assumes
Transset(*M*)
M ⊨ ·*Z*· ∪ {·*Replacement*(*p*)· . *p* ∈ *overhead_notCH*}
shows *M_ZF_ground_notCH_trans*(*M*)
⟨*proof*⟩

theorem *M_satT_imp_M_ZF_ground_CH_trans*:
assumes
Transset(*M*)
M ⊨ ·*Z*· ∪ {·*Replacement*(*p*)· . *p* ∈ *overhead_CH* }
shows *M_ZF_ground_CH_trans*(*M*)
⟨*proof*⟩

lemma (**in** *M_Z_basic*) *M_satT_Zermelo_fms*: *M* ⊨ ·*Z*·
⟨*proof*⟩

lemma (**in** *M_ZFC1*) *M_satT_ZC*: *M* ⊨ *ZC*
⟨*proof*⟩

locale *M_ZF* = *M_Z_basic* +
assumes
replacement_ax:*replacement_assm*(*M*,*env*,*φ*)

sublocale *M_ZF* ⊆ *M_ZF3*
⟨*proof*⟩

lemma *M_satT_imp_M_ZF*: *M* ⊨ *ZF* ⟹ *M_ZF*(*M*)
⟨*proof*⟩

lemma (**in** *M_ZF*) *M_satT_ZF*: $M \models ZF$
⟨*proof*⟩

lemma *M_ZF_iff_M_satT*: $M\_ZF(M) \longleftrightarrow (M \models ZF)$
⟨*proof*⟩

locale $M\_ZFC = M\_ZF + M\_ZC\_basic$

sublocale $M\_ZFC \subseteq M\_ZFC3$
⟨*proof*⟩

lemma *M_ZFC_iff_M_satT*:
notes *iff_trans*[*trans*]
shows $M\_ZFC(M) \longleftrightarrow (M \models ZFC)$
⟨*proof*⟩

lemma *M_satT_imp_M_ZF3*: $(M \models ZF) \longrightarrow M\_ZF3(M)$
⟨*proof*⟩

lemma *M_satT_imp_M_ZFC3*:
shows $(M \models ZFC) \longrightarrow M\_ZFC3(M)$
⟨*proof*⟩

lemma *M_satT_overhead_imp_M_ZF3*:
$(M \models ZC \cup \{\cdot Replacement(p) \cdot . p \in overhead\_notCH\}) \longrightarrow M\_ZFC3(M)$
⟨*proof*⟩

end

10 Transitive set models of ZF

This theory defines locales for countable transitive models of ZF, and on top of that, one that includes a forcing notion. Weakened versions of both locales are included, that only assume finitely many replacement instances.

theory *Forcing_Data*
imports
Forcing_Notions
Cohen_Posets_Relative
ZF_Trans_Interpretations
begin

no_notation *Aleph* (‹ℵ_› [*90*] *90*)

10.1 A forcing locale and generic filters

Ideally, countability should be separated from the assumption of this locale. The fact is that our present proofs of the "definition of forces" (and many

consequences) and of the lemma for "forcing a value" of function unnecessarily depend on the countability of the ground model.

locale *forcing_data1* = *forcing_notion* + *M_ctm1* +
  assumes *P_in_M*: $\mathbb{P} \in M$
    and *leq_in_M*: $leq \in M$

locale *forcing_data2* = *forcing_data1* + *M_ctm2_AC*

locale *forcing_data3* = *forcing_data2* + *M_ctm3_AC*

context *forcing_data1*
begin

lemma *P_sub_M* : $\mathbb{P} \subseteq M$
  ⟨*proof*⟩

definition
  M_generic :: $i \Rightarrow o$ **where**
  $M\_generic(G) \equiv filter(G) \wedge (\forall D{\in}M.\ D{\subseteq}\mathbb{P} \wedge dense(D) \longrightarrow D \cap G \neq 0)$

declare *iff_trans* [*trans*]

lemma *M_generic_imp_filter*[*dest*]: $M\_generic(G) \Longrightarrow filter(G)$
  ⟨*proof*⟩

lemma *generic_filter_existence*:
  $p{\in}\mathbb{P} \Longrightarrow \exists\, G.\ p{\in}G \wedge M\_generic(G)$
⟨*proof*⟩

lemma *one_in_M*: $\mathbf{1} \in M$
  ⟨*proof*⟩

declare *P_in_M* [*simp*,*intro*]
declare *one_in_M* [*simp*,*intro*]
declare *leq_in_M* [*simp*,*intro*]
declare *one_in_P* [*intro*]

end — *forcing_data1*

locale *G_generic1* = *forcing_data1* +
  fixes *G* :: *i*
  assumes *generic* : $M\_generic(G)$
begin

lemma *G_nonempty*: $G \neq 0$
  ⟨*proof*⟩

lemma *M_genericD* [*dest*]: $x{\in}G \Longrightarrow x{\in}\mathbb{P}$
  ⟨*proof*⟩

lemma *M_generic_leqD* [*dest*]: $p{\in}G \Longrightarrow q{\in}\mathbb{P} \Longrightarrow p{\preceq}q \Longrightarrow q{\in}G$
⟨*proof*⟩

lemma *M_generic_compatD* [*dest*]: $p{\in}G \Longrightarrow r{\in}G \Longrightarrow \exists\, q{\in}G.\ q{\preceq}p \wedge q{\preceq}r$
⟨*proof*⟩

lemma *M_generic_denseD* [*dest*]: $dense(D) \Longrightarrow D{\subseteq}\mathbb{P} \Longrightarrow D{\in}M \Longrightarrow \exists\, q{\in}G.\ q{\in}D$
⟨*proof*⟩

lemma *G_subset_P*: $G{\subseteq}\mathbb{P}$
⟨*proof*⟩

lemma *one_in_G* : $\mathbf{1} \in G$
⟨*proof*⟩

lemma *G_subset_M*: $G \subseteq M$
⟨*proof*⟩

end — *G_generic1*

locale *G_generic1_AC* = *G_generic1* + *M_ctm1_AC*

end

11 The definition of *forces*

theory *Forces_Definition*
imports
Forcing_Data
begin

This is the core of our development.

11.1 The relation *frecrel*

lemma *names_belowsD*:
assumes $x \in names\_below(P,z)$
obtains *f n1 n2 p* **where**
$x = \langle f,n1,n2,p\rangle$ *f∈2 n1∈ecloseN*(z) *n2∈ecloseN*(z) *p∈P*
⟨*proof*⟩

context *forcing_data1*
begin

lemma *ftype_abs*:
$[\![x{\in}M;\ y{\in}M\ ]\!] \Longrightarrow is\_ftype(\#\#M,x,y) \longleftrightarrow y = ftype(x)$
⟨*proof*⟩

lemma *name1_abs*:
$[\![x{\in}M;\ y{\in}M\ ]\!] \Longrightarrow is\_name1(\#\#M,x,y) \longleftrightarrow y = name1(x)$
⟨*proof*⟩

lemma *snd_snd_abs*:
$[\![x{\in}M;\ y{\in}M\ ]\!] \Longrightarrow is\_snd\_snd(\#\#M,x,y) \longleftrightarrow y = snd(snd(x))$
⟨*proof*⟩

lemma *name2_abs*:
$[\![x{\in}M;\ y{\in}M\ ]\!] \Longrightarrow is\_name2(\#\#M,x,y) \longleftrightarrow y = name2(x)$
⟨*proof*⟩

lemma *cond_of_abs*:
$[\![x{\in}M;\ y{\in}M\ ]\!] \Longrightarrow is\_cond\_of(\#\#M,x,y) \longleftrightarrow y = cond\_of(x)$
⟨*proof*⟩

lemma *tuple_abs*:
$[\![z{\in}M;t1{\in}M;t2{\in}M;p{\in}M;t{\in}M]\!] \Longrightarrow$
$is\_tuple(\#\#M,z,t1,t2,p,t) \longleftrightarrow t = \langle z,t1,t2,p\rangle$
⟨*proof*⟩

lemmas *components_abs = ftype_abs name1_abs name2_abs cond_of_abs tuple_abs*

lemma *comp_in_M*:
$p \preceq q \Longrightarrow p{\in}M$
$p \preceq q \Longrightarrow q{\in}M$
⟨*proof*⟩

lemma *eq_case_abs* [*simp*]:
assumes *t1*∈*M t2*∈*M p*∈*M f*∈*M*
shows $is\_eq\_case(\#\#M,t1,t2,p,\mathbb{P},leq,f) \longleftrightarrow eq\_case(t1,t2,p,\mathbb{P},leq,f)$
⟨*proof*⟩

lemma *mem_case_abs* [*simp*]:
assumes *t1*∈*M t2*∈*M p*∈*M f*∈*M*
shows $is\_mem\_case(\#\#M,t1,t2,p,\mathbb{P},leq,f) \longleftrightarrow mem\_case(t1,t2,p,\mathbb{P},leq,f)$
⟨*proof*⟩

lemma *Hfrc_abs*:
$[\![fnnc{\in}M;\ f{\in}M]\!] \Longrightarrow$
$is\_Hfrc(\#\#M,\mathbb{P},leq,fnnc,f) \longleftrightarrow Hfrc(\mathbb{P},leq,fnnc,f)$
⟨*proof*⟩

lemma *Hfrc_at_abs*:
$[\![fnnc{\in}M;\ f{\in}M\ ;\ z{\in}M]\!] \Longrightarrow$

$is\_Hfrc\_at(\#\#M,\mathbb{P},leq,fnnc,f,z) \longleftrightarrow\ z = bool\_of\_o(Hfrc(\mathbb{P},leq,fnnc,f))$
⟨*proof*⟩

lemma *components_closed* :
$x{\in}M \Longrightarrow (\#\#M)(ftype(x))$
$x{\in}M \Longrightarrow (\#\#M)(name1(x))$
$x{\in}M \Longrightarrow (\#\#M)(name2(x))$
$x{\in}M \Longrightarrow (\#\#M)(cond\_of(x))$
⟨*proof*⟩

lemma *ecloseN_closed*:
$(\#\#M)(A) \Longrightarrow (\#\#M)(ecloseN(A))$
$(\#\#M)(A) \Longrightarrow (\#\#M)(eclose\_n(name1,A))$
$(\#\#M)(A) \Longrightarrow (\#\#M)(eclose\_n(name2,A))$
⟨*proof*⟩

lemma *eclose_n_abs* :
assumes $x{\in}M\ ec{\in}M$
shows $is\_eclose\_n(\#\#M,is\_name1,ec,x) \longleftrightarrow ec = eclose\_n(name1,x)$
$is\_eclose\_n(\#\#M,is\_name2,ec,x) \longleftrightarrow ec = eclose\_n(name2,x)$
⟨*proof*⟩

lemma *ecloseN_abs* :
$[\![x{\in}M;ec{\in}M]\!] \Longrightarrow is\_ecloseN(\#\#M,x,ec) \longleftrightarrow ec = ecloseN(x)$
⟨*proof*⟩

lemma *frecR_abs* :
$x{\in}M \Longrightarrow y{\in}M \Longrightarrow frecR(x,y) \longleftrightarrow is\_frecR(\#\#M,x,y)$
⟨*proof*⟩

lemma *frecrelP_abs* :
$z{\in}M \Longrightarrow frecrelP(\#\#M,z) \longleftrightarrow (\exists\, x\ y.\ z = \langle x,y\rangle \wedge frecR(x,y))$
⟨*proof*⟩

lemma *frecrel_abs*:
assumes $A{\in}M\ r{\in}M$
shows $is\_frecrel(\#\#M,A,r) \longleftrightarrow\ r = frecrel(A)$
⟨*proof*⟩

lemma *frecrel_closed*:
assumes $x{\in}M$
shows $frecrel(x){\in}M$
⟨*proof*⟩

lemma *field_frecrel* : $field(frecrel(names\_below(\mathbb{P},x))) \subseteq names\_below(\mathbb{P},x)$
⟨*proof*⟩

lemma *forcerelD* : $uv \in forcerel(\mathbb{P},x) \Longrightarrow uv{\in}\ names\_below(\mathbb{P},x) \times names\_below(\mathbb{P},x)$

⟨*proof*⟩

lemma *wf_forcerel* :
wf(*forcerel*(ℙ,*x*))
⟨*proof*⟩

lemma *restrict_trancl_forcerel*:
assumes *frecR*(*w*,*y*)
shows *restrict*(*f*,*frecrel*(*names_below*(ℙ,*x*))-‘‘{*y*})‘*w*
= *restrict*(*f*,*forcerel*(ℙ,*x*)-‘‘{*y*})‘*w*
⟨*proof*⟩

lemma *names_belowI* :
assumes *frecR*(⟨*ft*,*n1*,*n2*,*p*⟩,⟨*a*,*b*,*c*,*d*⟩) *p*∈ℙ
shows ⟨*ft*,*n1*,*n2*,*p*⟩ ∈ *names_below*(ℙ,⟨*a*,*b*,*c*,*d*⟩) (**is** *?x* ∈ *names_below*(_,*?y*))
⟨*proof*⟩

lemma *names_below_tr* :
assumes *x*∈ *names_below*(ℙ,*y*) *y*∈ *names_below*(ℙ,*z*)
shows *x*∈ *names_below*(ℙ,*z*)
⟨*proof*⟩

lemma *arg_into_names_below2* :
assumes ⟨*x*,*y*⟩ ∈ *frecrel*(*names_below*(ℙ,*z*))
shows *x* ∈ *names_below*(ℙ,*y*)
⟨*proof*⟩

lemma *arg_into_names_below* :
assumes ⟨*x*,*y*⟩ ∈ *frecrel*(*names_below*(ℙ,*z*))
shows *x* ∈ *names_below*(ℙ,*x*)
⟨*proof*⟩

lemma *forcerel_arg_into_names_below* :
assumes ⟨*x*,*y*⟩ ∈ *forcerel*(ℙ,*z*)
shows *x* ∈ *names_below*(ℙ,*x*)
⟨*proof*⟩

lemma *names_below_mono* :
assumes ⟨*x*,*y*⟩ ∈ *frecrel*(*names_below*(ℙ,*z*))
shows *names_below*(ℙ,*x*) ⊆ *names_below*(ℙ,*y*)
⟨*proof*⟩

lemma *frecrel_mono* :
assumes ⟨*x*,*y*⟩ ∈ *frecrel*(*names_below*(ℙ,*z*))
shows *frecrel*(*names_below*(ℙ,*x*)) ⊆ *frecrel*(*names_below*(ℙ,*y*))
⟨*proof*⟩

lemma *forcerel_mono2* :
assumes ⟨*x*,*y*⟩ ∈ *frecrel*(*names_below*(ℙ,*z*))

shows $forcerel(\mathbb{P},x) \subseteq forcerel(\mathbb{P},y)$
⟨*proof*⟩

lemma *forcerel_mono_aux* :
assumes $\langle x,y\rangle \in frecrel(names\_below(\mathbb{P},\ w))^{\frown}+$
shows $forcerel(\mathbb{P},x) \subseteq forcerel(\mathbb{P},y)$
⟨*proof*⟩

lemma *forcerel_mono* :
assumes $\langle x,y\rangle \in forcerel(\mathbb{P},z)$
shows $forcerel(\mathbb{P},x) \subseteq forcerel(\mathbb{P},y)$
⟨*proof*⟩

lemma *forcerel_eq_aux*: $x \in names\_below(\mathbb{P},\ w) \Longrightarrow \langle x,y\rangle \in forcerel(\mathbb{P},z) \Longrightarrow$
$(y \in names\_below(\mathbb{P},\ w) \longrightarrow \langle x,y\rangle \in forcerel(\mathbb{P},w))$
⟨*proof*⟩

lemma *forcerel_eq* :
assumes $\langle z,x\rangle \in forcerel(\mathbb{P},x)$
shows $forcerel(\mathbb{P},z) = forcerel(\mathbb{P},x) \cap names\_below(\mathbb{P},z) \times names\_below(\mathbb{P},z)$
⟨*proof*⟩

lemma *forcerel_below_aux* :
assumes $\langle z,x\rangle \in forcerel(\mathbb{P},x)$ $\langle u,z\rangle \in forcerel(\mathbb{P},x)$
shows $u \in names\_below(\mathbb{P},z)$
⟨*proof*⟩

lemma *forcerel_below* :
assumes $\langle z,x\rangle \in forcerel(\mathbb{P},x)$
shows $forcerel(\mathbb{P},x)$ -`` $\{z\} \subseteq names\_below(\mathbb{P},z)$
⟨*proof*⟩

lemma *relation_forcerel* :
shows $relation(forcerel(\mathbb{P},z))$ $trans(forcerel(\mathbb{P},z))$
⟨*proof*⟩

lemma *Hfrc_restrict_trancl*: $bool\_of\_o(Hfrc(\mathbb{P},\ leq,\ y,\ restrict(f,frecrel(names\_below(\mathbb{P},x))\text{-``}\{y\})))$
$= bool\_of\_o(Hfrc(\mathbb{P},\ leq,\ y,\ restrict(f,(frecrel(names\_below(\mathbb{P},x)))^{\frown}+)\text{-``}\{y\})))$
⟨*proof*⟩

lemma *frc_at_trancl*: $frc\_at(\mathbb{P},leq,z) = wfrec(forcerel(\mathbb{P},z),z,\lambda x\, f.\ bool\_of\_o(Hfrc(\mathbb{P},leq,x,f)))$
⟨*proof*⟩

lemma *forcerelI1* :
assumes $n1 \in domain(b) \vee n1 \in domain(c)$ $p \in \mathbb{P}$ $d \in \mathbb{P}$
shows $\langle\langle 1,\ n1,\ b,\ p\rangle,\ \langle 0,b,c,d\rangle\rangle \in forcerel(\mathbb{P},\langle 0,b,c,d\rangle)$
⟨*proof*⟩

lemma *forcerelI2* :
assumes *n1* ∈ *domain*(*b*) ∨ *n1* ∈ *domain*(*c*) *p*∈ℙ *d*∈ℙ
shows ⟨⟨*1*, *n1*, *c*, *p*⟩, ⟨*0*,*b*,*c*,*d*⟩⟩∈ *forcerel*(ℙ,⟨*0*,*b*,*c*,*d*⟩)
⟨*proof*⟩

lemma *forcerelI3* :
assumes ⟨*n2*, *r*⟩ ∈ *c* *p*∈ℙ *d*∈ℙ *r* ∈ ℙ
shows ⟨⟨*0*, *b*, *n2*, *p*⟩,⟨*1*, *b*, *c*, *d*⟩⟩ ∈ *forcerel*(ℙ,⟨*1*,*b*,*c*,*d*⟩)
⟨*proof*⟩

lemmas *forcerelI* = *forcerelI1*[*THEN vimage_singleton_iff*[*THEN iffD2*]]
forcerelI2[*THEN vimage_singleton_iff*[*THEN iffD2*]]
forcerelI3[*THEN vimage_singleton_iff*[*THEN iffD2*]]

lemma *aux_def_frc_at*:
assumes *z* ∈ *forcerel*(ℙ,*x*) -‘‘ {*x*}
shows *wfrec*(*forcerel*(ℙ,*x*), *z*, *H*) = *wfrec*(*forcerel*(ℙ,*z*), *z*, *H*)
⟨*proof*⟩

11.2 Recursive expression of *frc_at*

lemma *def_frc_at* :
assumes *p*∈ℙ
shows
frc_at(ℙ,*leq*,⟨*ft*,*n1*,*n2*,*p*⟩) =
bool_of_o(*p* ∈ℙ ∧
(*ft* = *0* ∧ (∀ *s*. *s*∈*domain*(*n1*) ∪ *domain*(*n2*) ⟶
(∀ *q*. *q*∈ℙ ∧ *q* ≼ *p* ⟶ (*frc_at*(ℙ,*leq*,⟨*1*,*s*,*n1*,*q*⟩) =*1* ⟷ *frc_at*(ℙ,*leq*,⟨*1*,*s*,*n2*,*q*⟩)
=*1*)))
∨ *ft* = *1* ∧ (∀ *v*∈ℙ. *v* ≼ *p* ⟶
(∃ *q*. ∃ *s*. ∃ *r*. *r*∈ℙ ∧ *q*∈ℙ ∧ *q* ≼ *v* ∧ ⟨*s*,*r*⟩ ∈ *n2* ∧ *q* ≼ *r* ∧ *frc_at*(ℙ,*leq*,⟨*0*,*n1*,*s*,*q*⟩)
= *1*))))
⟨*proof*⟩

11.3 Absoluteness of *frc_at*

lemma *forcerel_in_M* :
assumes *x*∈*M*
shows *forcerel*(ℙ,*x*)∈*M*
⟨*proof*⟩

lemma *relation2_Hfrc_at_abs*:
relation2(##*M*,*is_Hfrc_at*(##*M*,ℙ,*leq*),λ*x f*. *bool_of_o*(*Hfrc*(ℙ,*leq*,*x*,*f*)))
⟨*proof*⟩

lemma *Hfrc_at_closed* :
∀ *x*∈*M*. ∀ *g*∈*M*. *function*(*g*) ⟶ *bool_of_o*(*Hfrc*(ℙ,*leq*,*x*,*g*))∈*M*
⟨*proof*⟩

lemma *wfrec_Hfrc_at* :

assumes $X{\in}M$
shows *wfrec_replacement*(*##M*,*is_Hfrc_at*(*##M*,ℙ,*leq*),*forcerel*(ℙ,*X*))
⟨*proof*⟩

lemma *names_below_abs* :
⟦$Q{\in}M$;$x{\in}M$;$nb{\in}M$⟧ ⟹ *is_names_below*(*##M*,*Q*,*x*,*nb*) ⟷ $nb = names\_below(Q,x)$
⟨*proof*⟩

lemma *names_below_closed*:
⟦$Q{\in}M$;$x{\in}M$⟧ ⟹ $names\_below(Q,x) \in M$
⟨*proof*⟩

lemma *names_below_productE* :
assumes $Q \in M$ $x \in M$
⋀*A1 A2 A3 A4*. *A1* ∈ *M* ⟹ *A2* ∈ *M* ⟹ *A3* ∈ *M* ⟹ *A4* ∈ *M* ⟹ *R*(*A1* × *A2* × *A3* × *A4*)
shows *R*(*names_below*(*Q*,*x*))
⟨*proof*⟩

lemma *forcerel_abs* :
⟦$x{\in}M$;$z{\in}M$⟧ ⟹ *is_forcerel*(*##M*,ℙ,*x*,*z*) ⟷ $z = forcerel(\mathbb{P},x)$
⟨*proof*⟩

lemma *frc_at_abs*:
assumes *fnnc*∈*M* *z*∈*M*
shows *is_frc_at*(*##M*,ℙ,*leq*,*fnnc*,*z*) ⟷ $z = frc\_at(\mathbb{P},leq,fnnc)$
⟨*proof*⟩

lemma *forces_eq′_abs* :
⟦$p{\in}M$; $t1{\in}M$; $t2{\in}M$⟧ ⟹ *is_forces_eq′*(*##M*,ℙ,*leq*,*p*,*t1*,*t2*) ⟷ *forces_eq′*(ℙ,*leq*,*p*,*t1*,*t2*)
⟨*proof*⟩

lemma *forces_mem′_abs* :
⟦$p{\in}M$; $t1{\in}M$; $t2{\in}M$⟧ ⟹ *is_forces_mem′*(*##M*,ℙ,*leq*,*p*,*t1*,*t2*) ⟷ *forces_mem′*(ℙ,*leq*,*p*,*t1*,*t2*)
⟨*proof*⟩

lemma *forces_neq′_abs* :
assumes $p{\in}M$ $t1{\in}M$ $t2{\in}M$
shows *is_forces_neq′*(*##M*,ℙ,*leq*,*p*,*t1*,*t2*) ⟷ *forces_neq′*(ℙ,*leq*,*p*,*t1*,*t2*)
⟨*proof*⟩

lemma *forces_nmem′_abs* :
assumes $p{\in}M$ $t1{\in}M$ $t2{\in}M$
shows *is_forces_nmem′*(*##M*,ℙ,*leq*,*p*,*t1*,*t2*) ⟷ *forces_nmem′*(ℙ,*leq*,*p*,*t1*,*t2*)
⟨*proof*⟩

lemma *leq_abs*:
⟦ $l{\in}M$; $q{\in}M$; $p{\in}M$ ⟧ ⟹ *is_leq*(*##M*,*l*,*q*,*p*) ⟷ $\langle q,p\rangle{\in}l$
⟨*proof*⟩

11.4 Forcing for atomic formulas in context

definition
$forces\_eq :: [i,i,i] \Rightarrow o$ ($\langle \_\ forces_a\ '(\_ = \_')\rangle$ $[36,1,1]$ 60) **where**
$forces\_eq \equiv forces\_eq'(\mathbb{P},leq)$

definition
$forces\_mem :: [i,i,i] \Rightarrow o$ ($\langle \_\ forces_a\ '(\_ \in \_')\rangle$ $[36,1,1]$ 60) **where**
$forces\_mem \equiv forces\_mem'(\mathbb{P},leq)$

abbreviation *is_forces_eq*
where $is\_forces\_eq \equiv is\_forces\_eq'(\#\#M,\mathbb{P},leq)$

abbreviation
$is\_forces\_mem :: [i,i,i] \Rightarrow o$ **where**
$is\_forces\_mem \equiv is\_forces\_mem'(\#\#M,\mathbb{P},leq)$

lemma *def_forces_eq*: $p{\in}\mathbb{P} \Longrightarrow p\ forces_a\ (t1 = t2) \longleftrightarrow$
$(\forall\, s{\in}domain(t1) \cup domain(t2).\ \forall\, q.\ q{\in}\mathbb{P} \wedge q \preceq p \longrightarrow$
$(q\ forces_a\ (s \in t1) \longleftrightarrow q\ forces_a\ (s \in t2)))$
⟨*proof*⟩

lemma *def_forces_mem*: $p{\in}\mathbb{P} \Longrightarrow p\ forces_a\ (t1 \in t2) \longleftrightarrow$
$(\forall\, v{\in}\mathbb{P}.\ v \preceq p \longrightarrow$
$(\exists\, q.\ \exists\, s.\ \exists\, r.\ r{\in}\mathbb{P} \wedge q{\in}\mathbb{P} \wedge q \preceq v \wedge \langle s,r\rangle \in t2 \wedge q \preceq r \wedge q\ forces_a\ (t1 = s)))$
⟨*proof*⟩

lemma *forces_eq_abs* :
$[\![p{\in}M\ ;\ t1{\in}M\ ;\ t2{\in}M]\!] \Longrightarrow is\_forces\_eq(p,t1,t2) \longleftrightarrow p\ forces_a\ (t1 = t2)$
⟨*proof*⟩

lemma *forces_mem_abs* :
$[\![p{\in}M\ ;\ t1{\in}M\ ;\ t2{\in}M]\!] \Longrightarrow is\_forces\_mem(p,t1,t2) \longleftrightarrow p\ forces_a\ (t1 \in t2)$
⟨*proof*⟩

definition
$forces\_neq :: [i,i,i] \Rightarrow o$ ($\langle \_\ forces_a\ '(\_ \neq \_')\rangle$ $[36,1,1]$ 60) **where**
$p\ forces_a\ (t1 \neq t2) \equiv \neg\ (\exists\, q{\in}\mathbb{P}.\ q{\preceq}p \wedge q\ forces_a\ (t1 = t2))$

definition
$forces\_nmem :: [i,i,i] \Rightarrow o$ ($\langle \_\ forces_a\ '(\_ \notin \_')\rangle$ $[36,1,1]$ 60) **where**
$p\ forces_a\ (t1 \notin t2) \equiv \neg\ (\exists\, q{\in}\mathbb{P}.\ q{\preceq}p \wedge q\ forces_a\ (t1 \in t2))$

lemma *forces_neq* :
$p\ forces_a\ (t1 \neq t2) \longleftrightarrow forces\_neq'(\mathbb{P},leq,p,t1,t2)$
⟨*proof*⟩

lemma *forces_nmem* :

p *forces*$_a$ $(t1 \notin t2) \longleftrightarrow$ *forces_nmem'*$(\mathbb{P},leq,p,t1,t2)$
⟨*proof*⟩

abbreviation *Forces* :: $[i, i, i] \Rightarrow o$ $(\_ \Vdash \_ \_ \ [36,36,36] \ 60)$ **where**
$p \Vdash \varphi \ env \quad \equiv \quad M, ([p,\mathbb{P},leq,\mathbf{1}] \ @ \ env) \models forces(\varphi)$

lemma *sats_forces_Member* :
assumes $x{\in}nat \ y{\in}nat \ env{\in}list(M)$
$nth(x,env){=}xx \ nth(y,env){=}yy \ q{\in}M$
shows $q \Vdash \cdot x \in y \cdot \ env \longleftrightarrow q \in \mathbb{P} \wedge is\_forces\_mem(q, xx, yy)$
⟨*proof*⟩

lemma *sats_forces_Equal* :
assumes $a{\in}nat \ b{\in}nat \ env{\in}list(M) \ nth(a,env){=}x \ nth(b,env){=}y \ q{\in}M$
shows $q \Vdash \cdot a = b \cdot \ env \longleftrightarrow q \in \mathbb{P} \wedge is\_forces\_eq(q, x, y)$
⟨*proof*⟩

lemma *sats_forces_Nand* :
assumes $\varphi{\in}formula \ \psi{\in}formula \ env{\in}list(M) \ p{\in}M$
shows $p \Vdash \cdot\neg(\varphi \wedge \psi)\cdot \ env \longleftrightarrow$
$p{\in}\mathbb{P} \wedge \neg(\exists\, q{\in}M.\ q{\in}\mathbb{P} \wedge is\_leq(\#\#M,leq,q,p) \wedge (q \Vdash \varphi \ env) \wedge (q \Vdash \psi \ env))$
⟨*proof*⟩

lemma *sats_forces_Neg* :
assumes $\varphi{\in}formula \ env{\in}list(M) \ p{\in}M$
shows $p \Vdash \cdot\neg\varphi\cdot \ env \longleftrightarrow$
$(p{\in}\mathbb{P} \wedge \neg(\exists\, q{\in}M.\ q{\in}\mathbb{P} \wedge is\_leq(\#\#M,leq,q,p) \wedge (q \Vdash \varphi \ env)))$
⟨*proof*⟩

lemma *sats_forces_Forall* :
assumes $\varphi{\in}formula \ env{\in}list(M) \ p{\in}M$
shows $p \Vdash (\cdot\forall \varphi\cdot) \ env \longleftrightarrow p \in \mathbb{P} \wedge (\forall\, x{\in}M.\ p \Vdash \varphi \ ([x] \ @ \ env))$
⟨*proof*⟩

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**

$Hv(G,x,f) \equiv \{\ z\ .\ y{\in}\ domain(x),\ (\exists\, p{\in}G.\ \langle y,p\rangle \in x) \wedge z{=}f`y\}$

The funcion *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**
$val(G,\tau) \equiv wfrec(edrel(eclose(\{\tau\})),\ \tau\ ,Hv(G))$

definition
$GenExt :: [i,i]{\Rightarrow}i \quad (\_[\_]\ [71,1])$
where $M[G] \equiv \{val(G,\tau).\ \tau \in M\}$

lemma *map_val_in_MG*:
assumes
$env{\in}list(M)$
shows
$map(val(G),env){\in}list(M[G])$
⟨*proof*⟩

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 $\sigma{\in}M\ p{\in}M$
⟨*proof*⟩

definition
$Hcheck :: [i,i] \Rightarrow i$ **where**
$Hcheck(z,f)\ \equiv \{\ \langle f`y,\mathbf{1}\rangle\ .\ y \in z\}$

definition
$check :: i \Rightarrow i$ **where**
$check(x) \equiv transrec(x\ ,\ Hcheck)$

lemma *checkD*:
$check(x) =\ \ wfrec(Memrel(eclose(\{x\})),\ x,\ Hcheck)$
⟨*proof*⟩

lemma *Hcheck_trancl*:$Hcheck(y,\ restrict(f,Memrel(eclose(\{x\}))\text{-``}\{y\}))$
$= Hcheck(y,\ restrict(f,(Memrel(eclose(\{x\}))^{+})\text{-``}\{y\}))$
⟨*proof*⟩

lemma *check_trancl*: $check(x) = wfrec(rcheck(x),\ x,\ Hcheck)$
⟨*proof*⟩

lemma *rcheck_in_M* : $x \in M \Longrightarrow rcheck(x) \in M$

⟨*proof*⟩

lemma *rcheck_subset_M* : $x \in M \Longrightarrow \mathit{field}(\mathit{rcheck}(x)) \subseteq \mathit{eclose}(\{x\})$
⟨*proof*⟩

lemma *aux_def_check*: $x \in y \Longrightarrow$
$\mathit{wfrec}(\mathit{Memrel}(\mathit{eclose}(\{y\})),\ x,\ \mathit{Hcheck}) =$
$\mathit{wfrec}(\mathit{Memrel}(\mathit{eclose}(\{x\})),\ x,\ \mathit{Hcheck})$
⟨*proof*⟩

lemma *def_check* : $\mathit{check}(y) = \{\ \langle \mathit{check}(w), \mathbf{1}\rangle\ .\ w \in y\}$
⟨*proof*⟩

lemma *def_checkS* :
fixes n
assumes $n \in \mathit{nat}$
shows $\mathit{check}(\mathit{succ}(n)) = \mathit{check}(n) \cup \{\langle \mathit{check}(n), \mathbf{1}\rangle\}$
⟨*proof*⟩

lemma *field_Memrel2* :
assumes $x \in M$
shows $\mathit{field}(\mathit{Memrel}(\mathit{eclose}(\{x\}))) \subseteq M$
⟨*proof*⟩

lemma *aux_def_val*:
assumes $z \in \mathit{domain}(x)$
shows $\mathit{wfrec}(\mathit{edrel}(\mathit{eclose}(\{x\})), z, \mathit{Hv}(G)) = \mathit{wfrec}(\mathit{edrel}(\mathit{eclose}(\{z\})), z, \mathit{Hv}(G))$
⟨*proof*⟩

The next lemma provides the usual recursive expresion for the definition of *val*.

lemma *def_val*: $\mathit{val}(G,x) = \{z\ .\ t{\in}\mathit{domain}(x)\ ,\ (\exists\, p{\in}G\ .\ \langle t,p\rangle{\in}x) \wedge z{=}\mathit{val}(G,t)\}$
⟨*proof*⟩

lemma *val_mono* : $x{\subseteq}y \Longrightarrow \mathit{val}(G,x) \subseteq \mathit{val}(G,y)$
⟨*proof*⟩

Check-names are the canonical names for elements of the ground model. Here we show that this is the case.

lemma *val_check* : $\mathbf{1} \in G \Longrightarrow\ \mathbf{1} \in \mathbb{P} \Longrightarrow \mathit{val}(G,\mathit{check}(y))\ = y$
⟨*proof*⟩

lemma *val_of_name* :
$\mathit{val}(G,\{x{\in}A{\times}\mathbb{P}.\ Q(x)\}) = \{z\ .\ t{\in}A\ ,\ (\exists\, p{\in}\mathbb{P}\ .\ \ Q(\langle t,p\rangle) \wedge p \in G) \wedge z{=}\mathit{val}(G,t)\}$
⟨*proof*⟩

lemma *val_of_name_alt* :
$\mathit{val}(G,\{x{\in}A{\times}\mathbb{P}.\ Q(x)\}) = \{z\ .\ t{\in}A\ ,\ (\exists\, p{\in}\mathbb{P}{\cap}G\ .\ \ Q(\langle t,p\rangle)) \wedge z{=}\mathit{val}(G,t)\ \}$
⟨*proof*⟩

lemma *val_only_names*: $val(F,\tau) = val(F,\{x{\in}\tau.\ \exists\, t{\in}domain(\tau).\ \exists\, p{\in}F.\ x{=}\langle t,p\rangle\})$
(**is** _ = $val(F,?name)$)
⟨*proof*⟩

lemma *val_only_pairs*: $val(F,\tau) = val(F,\{x{\in}\tau.\ \exists\, t\ p.\ x{=}\langle t,p\rangle\})$
⟨*proof*⟩

lemma *val_subset_domain_times_range*: $val(F,\tau) \subseteq val(F,domain(\tau){\times}range(\tau))$
⟨*proof*⟩

lemma *val_of_elem*: $\langle\vartheta,p\rangle \in \pi \Longrightarrow p{\in}G \Longrightarrow val(G,\vartheta) \in val(G,\pi)$
⟨*proof*⟩

lemma *elem_of_val*: $x{\in}val(G,\pi) \Longrightarrow \exists\,\vartheta{\in}domain(\pi).\ val(G,\vartheta) = x$
⟨*proof*⟩

lemma *elem_of_val_pair*: $x{\in}val(G,\pi) \Longrightarrow \exists\,\vartheta.\ \exists\, p{\in}G.\ \ \langle\vartheta,p\rangle{\in}\pi \wedge val(G,\vartheta) = x$
⟨*proof*⟩

lemma *elem_of_val_pair'*:
assumes $\pi{\in}M$ $x{\in}val(G,\pi)$
shows $\exists\,\vartheta{\in}M.\ \exists\, p{\in}G.\ \ \langle\vartheta,p\rangle{\in}\pi \wedge val(G,\vartheta) = x$
⟨*proof*⟩

lemma *GenExtD*: $x \in M[G] \Longrightarrow \exists\,\tau{\in}M.\ x = val(G,\tau)$
⟨*proof*⟩

lemma *GenExtI*: $x \in M \Longrightarrow val(G,x) \in M[G]$
⟨*proof*⟩

lemma *Transset_MG* : $Transset(M[G])$
⟨*proof*⟩

lemmas *transitivity_MG* = *Transset_intf*[*OF Transset_MG*]

This lemma can be proved before having *check_in_M*. At some point Miguel naïvely thought that the *check_in_M* could be proved using this argument.

lemma *check_nat_M* :
assumes $n \in nat$
shows $check(n) \in M$
⟨*proof*⟩

lemma *def_PHcheck*:
assumes
$z{\in}M$ $f{\in}M$
shows
$Hcheck(z,f) = Replace(z,PHcheck(\#\#M,\mathbf{1},f))$
⟨*proof*⟩

lemma *wfrec_Hcheck* :
assumes $X{\in}M$
shows *wfrec_replacement*(##M,*is_Hcheck*(##M,**1**),*rcheck*(X))
⟨*proof*⟩

lemma *Hcheck_closed'* : $f{\in}M \Longrightarrow z{\in}M \Longrightarrow \{f\ `\ x\ .\ x \in z\} \in M$
⟨*proof*⟩

lemma *repl_PHcheck* :
assumes $f{\in}M$
shows *lam_replacement*(##M,$\lambda x.$ *Hcheck*(x,f))
⟨*proof*⟩

lemma *univ_PHcheck* : $[\![\ z{\in}M\ ;\ f{\in}M\ ]\!] \Longrightarrow$ *univalent*(##M,z,*PHcheck*(##M,**1**,f))
⟨*proof*⟩

lemma *PHcheck_closed* : $[\![z{\in}M\ ;\ f{\in}M\ ;\ x{\in}z;\ PHcheck(\#\#M,\mathbf{1},f,x,y)\ ]\!] \Longrightarrow$ (##M)(y)
⟨*proof*⟩

lemma *relation2_Hcheck* : *relation2*(##M,*is_Hcheck*(##M,**1**),*Hcheck*)
⟨*proof*⟩

lemma *Hcheck_closed* : $\forall\, y{\in}M.\ \forall\, g{\in}M.\ \ Hcheck(y,g){\in}M$
⟨*proof*⟩

lemma *wf_rcheck* : $x{\in}M \Longrightarrow$ *wf*(*rcheck*(x))
⟨*proof*⟩

lemma *trans_rcheck* : $x{\in}M \Longrightarrow$ *trans*(*rcheck*(x))
⟨*proof*⟩

lemma *relation_rcheck* : $x{\in}M \Longrightarrow$ *relation*(*rcheck*(x))
⟨*proof*⟩

lemma *check_in_M* : $x{\in}M \Longrightarrow$ *check*(x) $\in M$
⟨*proof*⟩

lemma *rcheck_abs*[*Rel*] : $[\![\ x{\in}M\ ;\ r{\in}M\ ]\!] \Longrightarrow$ *is_rcheck*(##M,x,r) $\longleftrightarrow$ r = *rcheck*(x)
⟨*proof*⟩

lemma *check_abs*[*Rel*] :
assumes $x{\in}M$ $z{\in}M$
shows *is_check*(##M,**1**,x,z) $\longleftrightarrow$ z = *check*(x)
⟨*proof*⟩

lemma *check_lam_replacement*: *lam_replacement*(*##M*,*check*)
⟨*proof*⟩

lemma *check_replacement*: {*check*(*x*). *x*∈ℙ} ∈ *M*
⟨*proof*⟩

lemma *M_subset_MG* : **1** ∈ *G* ⟹ *M* ⊆ *M*[*G*]
⟨*proof*⟩

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*
⟨*proof*⟩

lemma *zero_in_MG* : *0* ∈ *M*[*G*]
⟨*proof*⟩

declare *check_in_M* [*simp*,*intro*]

end — *forcing_data1*

context *G_generic1*
begin

lemma *val_G_dot* : *val*(*G*,*G_dot*) = *G*
⟨*proof*⟩

lemma *G_in_Gen_Ext* : *G* ∈ *M*[*G*]
⟨*proof*⟩

lemmas *generic_simps* = *val_check*[*OF one_in_G one_in_P*]
M_subset_MG[*OF one_in_G*, *THEN subsetD*]
GenExtI P_in_M

lemmas *generic_dests* = *M_genericD M_generic_compatD*

bundle *G_generic1_lemmas* = *generic_simps*[*simp*] *generic_dests*[*dest*]

end — *G_generic1*

sublocale *G_generic1* ⊆ *ext*: *M_trans* *##M*[*G*]
⟨*proof*⟩

end

13 The Forcing Theorems

theory *Forcing_Theorems*
  imports
    Cohen_Posets_Relative
    Forces_Definition
    Names

begin

context *forcing_data1*
begin

13.1 The forcing relation in context

lemma *separation_forces* :
  assumes
    fty: $\varphi \in formula$ **and**
    far: $arity(\varphi) \le length(env)$ **and**
    envty: $env \in list(M)$
  shows
    $separation(\#\#M, \lambda p.\ (p \Vdash \varphi\ env))$
  ⟨*proof*⟩

lemma *Collect_forces* :
  assumes
    $\varphi \in formula$ **and**
    $arity(\varphi) \le length(env)$ **and**
    $env \in list(M)$
  shows
    $\{p \in \mathbb{P}\ .\ p \Vdash \varphi\ env\} \in M$
  ⟨*proof*⟩

lemma *forces_mem_iff_dense_below*: $p \in \mathbb{P} \Longrightarrow p\ forces_a\ (t1 \in t2) \longleftrightarrow dense\_below(\{q \in \mathbb{P}.\ \exists s.\ \exists r.\ r \in \mathbb{P} \wedge \langle s,r\rangle \in t2 \wedge q \preceq r \wedge q\ forces_a\ (t1 = s)\}, p)$
  ⟨*proof*⟩

13.2 Kunen 2013, Lemma IV.2.37(a)

lemma *strengthening_eq*:
  assumes $p \in \mathbb{P}$ $r \in \mathbb{P}$ $r \preceq p$ $p\ forces_a\ (t1 = t2)$
  shows $r\ forces_a\ (t1 = t2)$
  ⟨*proof*⟩

13.3 Kunen 2013, Lemma IV.2.37(a)

lemma *strengthening_mem*:
  assumes $p \in \mathbb{P}$ $r \in \mathbb{P}$ $r \preceq p$ $p\ forces_a\ (t1 \in t2)$
  shows $r\ forces_a\ (t1 \in t2)$

⟨*proof*⟩

13.4 Kunen 2013, Lemma IV.2.37(b)

lemma *density_mem*:
assumes $p \in \mathbb{P}$
shows $p\ forces_a\ (t1 \in t2) \longleftrightarrow dense\_below(\{q \in \mathbb{P}.\ q\ forces_a\ (t1 \in t2)\}, p)$
⟨*proof*⟩

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*⟩

lemma *density_eq*:
assumes $p \in \mathbb{P}$
shows $p\ forces_a\ (t1 = t2) \longleftrightarrow dense\_below(\{q \in \mathbb{P}.\ q\ forces_a\ (t1 = t2)\}, p)$
⟨*proof*⟩

13.5 Kunen 2013, Lemma IV.2.38

lemma *not_forces_neq*:
assumes $p \in \mathbb{P}$
shows $p\ forces_a\ (t1 = t2) \longleftrightarrow \neg\ (\exists\, q \in \mathbb{P}.\ q \preceq p \wedge q\ forces_a\ (t1 \neq t2))$
⟨*proof*⟩

lemma *not_forces_nmem*:
assumes $p \in \mathbb{P}$
shows $p\ forces_a\ (t1 \in t2) \longleftrightarrow \neg\ (\exists\, q \in \mathbb{P}.\ q \preceq p \wedge q\ forces_a\ (t1 \notin t2))$
⟨*proof*⟩

13.6 The relation of forcing and atomic formulas

lemma *Forces_Equal*:
assumes
$p \in \mathbb{P}\ t1 \in M\ t2 \in M\ env \in list(M)\ nth(n, env) = t1\ nth(m, env) = t2\ n \in nat\ m \in nat$
shows
$(p \Vdash Equal(n, m)\ env) \longleftrightarrow p\ forces_a\ (t1 = t2)$
⟨*proof*⟩

lemma *Forces_Member*:
assumes
$p \in \mathbb{P}\ t1 \in M\ t2 \in M\ env \in list(M)\ nth(n, env) = t1\ nth(m, env) = t2\ n \in nat\ m \in nat$
shows

$(p \Vdash Member(n,m)\ env) \longleftrightarrow p\ forces_a\ (t1 \in t2)$
⟨*proof*⟩

lemma *Forces_Neg*:
assumes
$p \in \mathbb{P}\ env \in list(M)\ \varphi \in formula$
shows
$(p \Vdash Neg(\varphi)\ env) \longleftrightarrow \neg(\exists\, q \in M.\ q \in \mathbb{P} \wedge q \preceq p \wedge (q \Vdash \varphi\ env))$
⟨*proof*⟩

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
$(p \Vdash Nand(\varphi,\psi)\ env) \longleftrightarrow \neg(\exists\, q \in M.\ q \in \mathbb{P} \wedge q \preceq p \wedge (q \Vdash \varphi\ env) \wedge (q \Vdash \psi\ env))$
⟨*proof*⟩

lemma *Forces_And_aux*:
assumes
$p \in \mathbb{P}\ env \in list(M)\ \varphi \in formula\ \psi \in formula$
shows
$p \Vdash And(\varphi,\psi)\ env \longleftrightarrow$
$(\forall\, q \in M.\ q \in \mathbb{P} \wedge q \preceq p \longrightarrow (\exists\, r \in M.\ r \in \mathbb{P} \wedge r \preceq q \wedge (r \Vdash \varphi\ env) \wedge (r \Vdash \psi\ env)))$
⟨*proof*⟩

lemma *Forces_And_iff_dense_below*:
assumes
$p \in \mathbb{P}\ env \in list(M)\ \varphi \in formula\ \psi \in formula$
shows
$(p \Vdash And(\varphi,\psi)\ env) \longleftrightarrow dense\_below(\{r \in \mathbb{P}.\ (r \Vdash \varphi\ env) \wedge (r \Vdash \psi\ env)\ \},p)$
⟨*proof*⟩

lemma *Forces_Forall*:
assumes
$p \in \mathbb{P}\ env \in list(M)\ \varphi \in formula$
shows
$(p \Vdash Forall(\varphi)\ env) \longleftrightarrow (\forall\, x \in M.\ (p \Vdash \varphi\ ([x]\ @\ env)))$
⟨*proof*⟩

bundle *some_rules* = *elem_of_val_pair* [*dest*]

context
includes *some_rules*
begin

lemma *elem_of_valI*: $\exists\, \vartheta.\ \exists\, p \in \mathbb{P}.\ p \in G \wedge \langle \vartheta,p \rangle \in \pi \wedge val(G,\vartheta) = x \Longrightarrow x \in val(G,\pi)$

⟨*proof*⟩

lemma *GenExt_iff*: $x{\in}M[G] \longleftrightarrow (\exists\,\tau{\in}M.\ x = val(G,\tau))$
⟨*proof*⟩
end

end
context *G_generic1*

begin

13.8 Kunen 2013, Lemma IV.2.29

lemma *generic_inter_dense_below*:
assumes $D{\in}M$ *dense_below*(D,p) $p{\in}G$
shows $D \cap G \neq 0$
⟨*proof*⟩

13.9 Auxiliary results for Lemma IV.2.40(a)

lemma (**in** *forcing_data1*) *IV240a_mem_Collect*:
assumes
$\pi{\in}M$ $\tau{\in}M$
shows
$\{q{\in}\mathbb{P}.\ \exists\,\sigma.\ \exists\,r.\ r{\in}\mathbb{P} \wedge \langle\sigma,r\rangle \in \tau \wedge q{\preceq}r \wedge q\ forces_a\ (\pi = \sigma)\}{\in}M$
⟨*proof*⟩

lemma *IV240a_mem*:
assumes
$p{\in}G$ $\pi{\in}M$ $\tau{\in}M$ $p\ forces_a\ (\pi \in \tau)$
$\bigwedge q\ \sigma.\ q{\in}\mathbb{P} \Longrightarrow q{\in}G \Longrightarrow \sigma{\in}domain(\tau) \Longrightarrow q\ forces_a\ (\pi = \sigma) \Longrightarrow$
$val(G,\pi) = val(G,\sigma)$
shows
$val(G,\pi){\in}val(G,\tau)$
⟨*proof*⟩

lemma *refl_forces_eq*:$p{\in}\mathbb{P} \Longrightarrow p\ forces_a\ (x = x)$
⟨*proof*⟩

lemma *forces_memI*: $\langle\sigma,r\rangle{\in}\tau \Longrightarrow p{\in}\mathbb{P} \Longrightarrow r{\in}\mathbb{P} \Longrightarrow p{\preceq}r \Longrightarrow p\ forces_a\ (\sigma \in \tau)$
⟨*proof*⟩

lemma *IV240a_eq_1st_incl*:
includes *some_rules*
assumes
$p{\in}G\ p\ forces_a\ (\tau = \vartheta)$
and

IH:$\bigwedge q\ \sigma.\ q{\in}\mathbb{P} \Longrightarrow q{\in}G \Longrightarrow \sigma{\in}domain(\tau) \cup domain(\vartheta) \Longrightarrow$
$(q\ forces_a\ (\sigma \in \tau) \longrightarrow val(G,\sigma) \in val(G,\tau)) \wedge$
$(q\ forces_a\ (\sigma \in \vartheta) \longrightarrow val(G,\sigma) \in val(G,\vartheta))$

shows
$val(G,\tau) \subseteq val(G,\vartheta)$
⟨*proof*⟩

lemma *IV240a_eq_2nd_incl*:
includes *some_rules*
assumes
$p{\in}G\ p\ forces_a\ (\tau = \vartheta)$
and
IH:$\bigwedge q\ \sigma.\ q{\in}\mathbb{P} \Longrightarrow q{\in}G \Longrightarrow \sigma{\in}domain(\tau) \cup domain(\vartheta) \Longrightarrow$
$(q\ forces_a\ (\sigma \in \tau) \longrightarrow val(G,\sigma) \in val(G,\tau)) \wedge$
$(q\ forces_a\ (\sigma \in \vartheta) \longrightarrow val(G,\sigma) \in val(G,\vartheta))$
shows
$val(G,\vartheta) \subseteq val(G,\tau)$
⟨*proof*⟩

lemma *IV240a_eq*:
includes *some_rules*
assumes
$p{\in}G\ p\ forces_a\ (\tau = \vartheta)$
and
IH:$\bigwedge q\ \sigma.\ q{\in}\mathbb{P} \Longrightarrow q{\in}G \Longrightarrow \sigma{\in}domain(\tau) \cup domain(\vartheta) \Longrightarrow$
$(q\ forces_a\ (\sigma \in \tau) \longrightarrow val(G,\sigma) \in val(G,\tau)) \wedge$
$(q\ forces_a\ (\sigma \in \vartheta) \longrightarrow val(G,\sigma) \in val(G,\vartheta))$
shows
$val(G,\tau) = val(G,\vartheta)$
⟨*proof*⟩

13.10 Induction on names

lemma (**in** *forcing_data1*) *core_induction*:
assumes
$\bigwedge \tau\ \vartheta\ p.\ p \in \mathbb{P} \Longrightarrow [\![\bigwedge q\ \sigma.\ [\![q{\in}\mathbb{P}\ ;\ \sigma{\in}domain(\vartheta)]\!] \Longrightarrow Q(0,\tau,\sigma,q)]\!] \Longrightarrow Q(1,\tau,\vartheta,p)$
$\bigwedge \tau\ \vartheta\ p.\ p \in \mathbb{P} \Longrightarrow [\![\bigwedge q\ \sigma.\ [\![q{\in}\mathbb{P}\ ;\ \sigma{\in}domain(\tau) \cup domain(\vartheta)]\!] \Longrightarrow Q(1,\sigma,\tau,q) \wedge Q(1,\sigma,\vartheta,q)]\!] \Longrightarrow Q(0,\tau,\vartheta,p)$
$ft \in 2\ p \in \mathbb{P}$
shows
$Q(ft,\tau,\vartheta,p)$
⟨*proof*⟩

lemma (**in** *forcing_data1*) *forces_induction_with_conds*:
assumes

$\bigwedge\tau\ \vartheta\ p.\ p \in \mathbb{P} \Longrightarrow [\![\bigwedge q\ \sigma.\ [\![q{\in}\mathbb{P}\ ;\ \sigma{\in}domain(\vartheta)]\!] \Longrightarrow Q(q,\tau,\sigma)]\!] \Longrightarrow R(p,\tau,\vartheta)$
$\bigwedge\tau\ \vartheta\ p.\ p \in \mathbb{P} \Longrightarrow [\![\bigwedge q\ \sigma.\ [\![q{\in}\mathbb{P}\ ;\ \sigma{\in}domain(\tau) \cup domain(\vartheta)]\!] \Longrightarrow R(q,\sigma,\tau) \wedge R(q,\sigma,\vartheta)]\!] \Longrightarrow Q(p,\tau,\vartheta)$
$p \in \mathbb{P}$
shows
$Q(p,\tau,\vartheta) \wedge R(p,\tau,\vartheta)$
⟨*proof*⟩

lemma (**in** *forcing_data1*) *forces_induction*:
assumes
$\bigwedge\tau\ \vartheta.\ [\![\bigwedge\sigma.\ \sigma{\in}domain(\vartheta) \Longrightarrow Q(\tau,\sigma)]\!] \Longrightarrow R(\tau,\vartheta)$
$\bigwedge\tau\ \vartheta.\ [\![\bigwedge\sigma.\ \sigma{\in}domain(\tau) \cup domain(\vartheta) \Longrightarrow R(\sigma,\tau) \wedge R(\sigma,\vartheta)]\!] \Longrightarrow Q(\tau,\vartheta)$
shows
$Q(\tau,\vartheta) \wedge R(\tau,\vartheta)$
⟨*proof*⟩

13.11 Lemma IV.2.40(a), in full

lemma *IV240a*:
shows
$(\tau{\in}M \longrightarrow \vartheta{\in}M \longrightarrow (\forall\, p{\in}G.\ p\ forces_a\ (\tau = \vartheta) \longrightarrow val(G,\tau) = val(G,\vartheta))) \wedge$
$(\tau{\in}M \longrightarrow \vartheta{\in}M \longrightarrow (\forall\, p{\in}G.\ p\ forces_a\ (\tau \in \vartheta) \longrightarrow val(G,\tau) \in val(G,\vartheta)))$
(**is** *?Q*$(\tau,\vartheta) \wedge$ *?R*(τ,ϑ))
⟨*proof*⟩

13.12 Lemma IV.2.40(b)

lemma *IV240b_mem*:
includes *some_rules*
assumes
$val(G,\pi){\in}val(G,\tau)\ \pi{\in}M\ \tau{\in}M$
and
IH:$\bigwedge\sigma.\ \sigma{\in}domain(\tau) \Longrightarrow val(G,\pi) = val(G,\sigma) \Longrightarrow$
$\exists\, p{\in}G.\ p\ forces_a\ (\pi = \sigma)$
shows
$\exists\, p{\in}G.\ p\ forces_a\ (\pi \in \tau)$
⟨*proof*⟩

end — *G_generic1*

context *forcing_data1*
begin

lemma *Collect_forces_eq_in_M*:
assumes $\tau \in M\ \vartheta \in M$
shows $\{p{\in}\mathbb{P}.\ p\ forces_a\ (\tau = \vartheta)\} \in M$
⟨*proof*⟩

lemma *IV240b_eq_Collects*:
assumes $\tau \in M\ \vartheta \in M$

shows $\{p{\in}\mathbb{P}.\ \exists\sigma{\in}domain(\tau)\cup domain(\vartheta).\ p\ forces_a\ (\sigma\in\tau)\wedge p\ forces_a\ (\sigma\notin\vartheta)\}{\in}M$ **and**
$\{p{\in}\mathbb{P}.\ \exists\sigma{\in}domain(\tau)\cup domain(\vartheta).\ p\ forces_a\ (\sigma\notin\tau)\wedge p\ forces_a\ (\sigma\in\vartheta)\}{\in}M$
⟨*proof*⟩

end — *forcing_data1*

context *G_generic1*
begin

lemma *IV240b_eq*:
includes *some_rules*
assumes
$val(G,\tau)=val(G,\vartheta)\ \tau{\in}M\ \vartheta{\in}M$
and
$IH{:}\bigwedge\sigma.\ \sigma{\in}domain(\tau){\cup}domain(\vartheta)\Longrightarrow$
$(val(G,\sigma){\in}val(G,\tau)\longrightarrow(\exists\,q{\in}G.\ q\ forces_a\ (\sigma\in\tau)))\wedge$
$(val(G,\sigma){\in}val(G,\vartheta)\longrightarrow(\exists\,q{\in}G.\ q\ forces_a\ (\sigma\in\vartheta)))$

shows
$\exists\,p{\in}G.\ p\ forces_a\ (\tau=\vartheta)$
⟨*proof*⟩

lemma *IV240b*:
$(\tau{\in}M{\longrightarrow}\vartheta{\in}M{\longrightarrow}val(G,\tau)=val(G,\vartheta)\longrightarrow(\exists\,p{\in}G.\ p\ forces_a\ (\tau=\vartheta)))\wedge$
$(\tau{\in}M{\longrightarrow}\vartheta{\in}M{\longrightarrow}val(G,\tau)\in val(G,\vartheta)\longrightarrow(\exists\,p{\in}G.\ p\ forces_a\ (\tau\in\vartheta)))$
(**is** $?Q(\tau,\vartheta)\wedge ?R(\tau,\vartheta)$)
⟨*proof*⟩

lemma *truth_lemma_mem*:
assumes
$env{\in}list(M)$
$n{\in}nat\ m{\in}nat\ n{<}length(env)\ m{<}length(env)$
shows
$(\exists\,p{\in}G.\ p\Vdash Member(n,m)\ env)\longleftrightarrow M[G],\ map(val(G),env)\models Member(n,m)$
⟨*proof*⟩

lemma *truth_lemma_eq*:
assumes
$env{\in}list(M)$
$n{\in}nat\ m{\in}nat\ n{<}length(env)\ m{<}length(env)$
shows
$(\exists\,p{\in}G.\ p\Vdash Equal(n,m)\ env)\longleftrightarrow M[G],\ map(val(G),env)\models Equal(n,m)$
⟨*proof*⟩

end — *G_generic1*

lemma *arities_at_aux*:
assumes
$n \in nat\ m \in nat\ env \in list(M)\ succ(n) \cup succ(m) \le length(env)$
shows
$n < length(env)\ m < length(env)$
⟨*proof*⟩

13.13 The Strenghtening Lemma

context *forcing_data1*
begin

lemma *strengthening_lemma*:
assumes
$p{\in}\mathbb{P}\ \varphi{\in}formula\ r{\in}\mathbb{P}\ r{\preceq}p$
$env{\in}list(M)\ arity(\varphi){\le}length(env)$
shows
$p \Vdash \varphi\ env \Longrightarrow r \Vdash \varphi\ env$
⟨*proof*⟩

13.14 The Density Lemma

lemma *arity_Nand_le*:
assumes $\varphi \in formula\ \psi \in formula\ arity(Nand(\varphi, \psi)) \le length(env)\ env{\in}list(A)$
shows $arity(\varphi) \le length(env)\ arity(\psi) \le length(env)$
⟨*proof*⟩

lemma *dense_below_imp_forces*:
assumes
$p{\in}\mathbb{P}\ \varphi{\in}formula$
$env{\in}list(M)\ arity(\varphi){\le}length(env)$
shows
$dense\_below(\{q{\in}\mathbb{P}.\ (q \Vdash \varphi\ env)\},p) \Longrightarrow (p \Vdash \varphi\ env)$
⟨*proof*⟩

lemma *density_lemma*:
assumes
$p{\in}\mathbb{P}\ \varphi{\in}formula\ env{\in}list(M)\ arity(\varphi){\le}length(env)$
shows
$p \Vdash \varphi\ env \quad \longleftrightarrow \quad dense\_below(\{q{\in}\mathbb{P}.\ (q \Vdash \varphi\ env)\},p)$
⟨*proof*⟩

13.15 The Truth Lemma

lemma *Forces_And*:
assumes
$p{\in}\mathbb{P}\ env \in list(M)\ \varphi{\in}formula\ \psi{\in}formula$
$arity(\varphi) \le length(env)\ arity(\psi) \le length(env)$
shows
$p \Vdash And(\varphi,\psi)\ env \quad \longleftrightarrow \quad (p \Vdash \varphi\ env) \wedge (p \Vdash \psi\ env)$

⟨*proof*⟩

lemma *Forces_Nand_alt*:
assumes
p∈ℙ *env* ∈ *list*(*M*) *φ*∈*formula ψ*∈*formula*
arity(*φ*) ≤ *length*(*env*) *arity*(*ψ*) ≤ *length*(*env*)
shows
(*p* ⊩ *Nand*(*φ*,*ψ*) *env*) ⟷ (*p* ⊩ *Neg*(*And*(*φ*,*ψ*)) *env*)
⟨*proof*⟩

end

context *G_generic1*
begin

lemma *truth_lemma_Neg*:
assumes
φ∈*formula env*∈*list*(*M*) *arity*(*φ*)≤*length*(*env*) **and**
IH: (∃ *p*∈*G*. *p* ⊩ *φ env*) ⟷ *M*[*G*], *map*(*val*(*G*),*env*) ⊨ *φ*
shows
(∃ *p*∈*G*. *p* ⊩ *Neg*(*φ*) *env*) ⟷ *M*[*G*], *map*(*val*(*G*),*env*) ⊨ *Neg*(*φ*)
⟨*proof*⟩

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*)
⟨*proof*⟩

end

definition
ren_truth_lemma :: *i*⇒*i* **where**
ren_truth_lemma(*φ*) ≡
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(λ*p*. *incr_bv*(*p*)‘5 , 6, *φ*))))))))))

lemma *ren_truth_lemma_type*[*TC*] :
φ∈*formula* ⟹ *ren_truth_lemma*(*φ*) ∈*formula*
⟨*proof*⟩

lemma *arity_ren_truth* :
assumes *φ*∈*formula*

shows $arity(ren\_truth\_lemma(\varphi)) \leq 6 \cup succ(arity(\varphi))$
⟨*proof*⟩

lemma *sats_ren_truth_lemma*:
$[q,b,d,a1,a2,a3]$ @ $env \in list(M) \Longrightarrow \varphi \in formula \Longrightarrow$
$(M, [q,b,d,a1,a2,a3]$ @ $env \models ren\_truth\_lemma(\varphi)\ ) \longleftrightarrow$
$(M, [q,a1,a2,a3,b]$ @ $env \models \varphi)$
⟨*proof*⟩

context *forcing_data1*
begin

lemma *truth_lemma′* :
assumes
$\varphi \in formula\ env \in list(M)\ arity(\varphi) \leq succ(length(env))$
shows
$separation(\#\#M, \lambda d.\ \exists\, b \in M.\ \forall\, q \in \mathbb{P}.\ q \preceq d \longrightarrow \neg(q \Vdash \varphi\ ([b]@env)))$
⟨*proof*⟩

end

context *G_generic1*
begin

lemma *truth_lemma*:
assumes
$\varphi \in formula$
$env \in list(M)\ arity(\varphi) \leq length(env)$
shows
$(\exists\, p \in G.\ p \Vdash \varphi\ env) \quad \longleftrightarrow \quad M[G],\ map(val(G), env) \models \varphi$
⟨*proof*⟩

end

context *forcing_data1*
begin

13.16 The "Definition of forcing"

lemma *definition_of_forcing*:
assumes
$p \in \mathbb{P}\ \varphi \in formula\ env \in list(M)\ arity(\varphi) \leq length(env)$
shows
$(p \Vdash \varphi\ env) \longleftrightarrow$
$(\forall\, G.\ M\_generic(G) \wedge p \in G \longrightarrow M[G],\ map(val(G), env) \models \varphi)$
⟨*proof*⟩

lemmas *definability* = *forces_type*

end — *forcing_data1*

end

14 Ordinals in generic extensions

theory *Ordinals_In_MG*
  imports
    Forcing_Theorems
begin

context *G_generic1*
begin

lemma *rank_val*: *rank*(*val*(*G*,*x*)) ≤ *rank*(*x*) (**is** *?Q*(*x*))
⟨*proof*⟩

lemma *Ord_MG_iff*:
  assumes *Ord*(α)
  shows α ∈ *M* ⟷ α ∈ *M*[*G*]
⟨*proof*⟩

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*
  ⟨*proof*⟩

lemma *nth_concat2* : *env*∈ *list*(*A*) ⟹ *nth*(*length*(*env*),*env* @ [*p*,*t*]) = *p*
  ⟨*proof*⟩

lemma *nth_concat3* : *env*∈ *list*(*A*) ⟹ *u* = *nth*(*succ*(*length*(*env*)), *env* @ [*pi*, *u*])
  ⟨*proof*⟩

definition

sep_var :: $i \Rightarrow i$ **where**
sep_var(*n*) $\equiv \{\langle 0,1\rangle,\langle 1,3\rangle,\langle 2,4\rangle,\langle 3,5\rangle,\langle 4,0\rangle,\langle 5+_\omega n,6\rangle,\langle 6+_\omega n,2\rangle\}$

definition
sep_env :: $i \Rightarrow i$ **where**
sep_env(*n*) $\equiv \lambda\ i \in (5+_\omega n)\text{-}5\ .\ i+_\omega 2$

definition *weak* :: $[i,\ i] \Rightarrow i$ **where**
weak(*n*,*m*) $\equiv \{i+_\omega m\ .\ i \in n\}$

lemma *weakD* :
assumes $n \in nat\ k \in nat\ x \in weak(n,k)$
shows $\exists\ i \in n\ .\ x = i+_\omega k$
⟨*proof*⟩

lemma *weak_equal* :
assumes $n \in nat\ m \in nat$
shows $weak(n,m) = (m+_\omega n)$ - m
⟨*proof*⟩

lemma *weak_zero*:
shows $weak(0,n) = 0$
⟨*proof*⟩

lemma *weakening_diff* :
assumes $n \in nat$
shows $weak(n,7)$ - $weak(n,5) \subseteq \{5+_\omega n,\ 6+_\omega n\}$
⟨*proof*⟩

lemma *in_add_del* :
assumes $x \in j+_\omega n\ n \in nat\ j \in nat$
shows $x < j \vee x \in weak(n,j)$
⟨*proof*⟩

lemma *sep_env_action*:
assumes
$[t,p,u,P,leq,o,pi] \in list(M)$
$env \in list(M)$
shows $\forall\ i\ .\ i \in weak(length(env),5) \longrightarrow$
$nth(sep\_env(length(env))\text{`}i,[t,p,u,P,leq,o,pi]@env) = nth(i,[p,P,leq,o,t]\ @\ env\ @\ [pi,u])$
⟨*proof*⟩

lemma *sep_env_type* :
assumes $n \in nat$
shows $sep\_env(n) : (5+_\omega n)\text{-}5 \to (7+_\omega n)\text{-}7$
⟨*proof*⟩

lemma *sep_var_fin_type* :
assumes $n \in nat$
shows $sep\_var(n) : 7+_\omega n \ -||> \ 7+_\omega n$
⟨*proof*⟩

lemma *sep_var_domain* :
assumes $n \in nat$
shows $domain(sep\_var(n)) = \ 7+_\omega n \ - \ weak(n,5)$
⟨*proof*⟩

lemma *sep_var_type* :
assumes $n \in nat$
shows $sep\_var(n) : (7+_\omega n)\text{-}weak(n,5) \to 7+_\omega n$
⟨*proof*⟩

lemma *sep_var_action* :
assumes
$[t,p,u,P,leq,o,pi] \in list(M)$
$env \in list(M)$
shows $\forall \ i \ . \ i \in (7+_\omega length(env)) \ - \ weak(length(env),5) \longrightarrow$
$nth(sep\_var(length(env))\text{`}i,[t,p,u,P,leq,o,pi]@env) \ = \ nth(i,[p,P,leq,o,t] \ @ \ env \ @ \ [pi,u])$
⟨*proof*⟩

definition
$rensep :: i \Rightarrow i$ **where**
$rensep(n) \equiv union\_fun(sep\_var(n),sep\_env(n),7+_\omega n\text{-}weak(n,5),weak(n,5))$

lemma *rensep_aux* :
assumes $n{\in}nat$
shows $(7+_\omega n\text{-}weak(n,5)) \cup weak(n,5) = 7+_\omega n \ 7+_\omega n \cup (\ 7 \ +_\omega \ n \ - \ 7) = 7+_\omega n$
⟨*proof*⟩

lemma *rensep_type* :
assumes $n{\in}nat$
shows $rensep(n) \in 7+_\omega n \to 7+_\omega n$
⟨*proof*⟩

lemma *rensep_action* :
assumes $[t,p,u,P,leq,o,pi] \ @ \ env \in list(M)$
shows $\forall \ i \ . \ i < 7+_\omega length(env) \longrightarrow nth(rensep(length(env))\text{`}i,[t,p,u,P,leq,o,pi]@env) = nth(i,[p,P,leq,o,t] \ @ \ env \ @ \ [pi,u])$
⟨*proof*⟩

definition $sep\_ren :: [i,i] \Rightarrow i$ **where**
$sep\_ren(n,\varphi) \equiv ren(\varphi)\text{`}(7+_\omega n)\text{`}(7+_\omega n)\text{`}rensep(n)$

lemma *arity_rensep*: **assumes** $\varphi{\in}formula \ env \in list(M)$
$arity(\varphi) \le 7+_\omega length(env)$

shows *arity*(*sep_ren*(*length*(*env*),φ)) $\leq$ 7+$_\omega$*length*(*env*)
⟨*proof*⟩

lemma *type_rensep* [*TC*]:
assumes φ∈*formula* *env*∈*list*(*M*)
shows *sep_ren*(*length*(*env*),φ) ∈ *formula*
⟨*proof*⟩

lemma *sepren_action*:
assumes *arity*(φ) $\leq$ 7 +$_\omega$ *length*(*env*)
[*t*,*p*,*u*,*P*,*leq*,*o*,*pi*] ∈ *list*(*M*)
env∈*list*(*M*)
φ∈*formula*
shows *sats*(*M*, *sep_ren*(*length*(*env*),φ),[*t*,*p*,*u*,*P*,*leq*,*o*,*pi*] @ *env*) ⟷ *sats*(*M*, φ,[*p*,*P*,*leq*,*o*,*t*] @ *env* @ [*pi*,*u*])
⟨*proof*⟩

end

16 The Axiom of Separation in $M[G]$

theory *Separation_Axiom*
imports *Forcing_Theorems Separation_Rename*
begin

context *G_generic1*
begin

lemma *map_val* :
assumes *env*∈*list*(*M*[*G*])
shows ∃ *nenv*∈*list*(*M*). *env* = *map*(*val*(*G*),*nenv*)
⟨*proof*⟩

lemma *Collect_sats_in_MG* :
assumes
A∈*M*[*G*]
φ ∈ *formula* *env*∈*list*(*M*[*G*]) *arity*(φ) $\leq$ 1 +$_\omega$ *length*(*env*)
shows
{*x* ∈ *A* . (*M*[*G*], [*x*] @ *env* ⊨ φ)} ∈ *M*[*G*]
⟨*proof*⟩

theorem *separation_in_MG*:
assumes
φ∈*formula* **and** *arity*(φ) $\leq$ 1 +$_\omega$ *length*(*env*) **and** *env*∈*list*(*M*[*G*])
shows
separation(##*M*[*G*],λ*x*. (*M*[*G*], [*x*] @ *env* ⊨ φ))
⟨*proof*⟩

end — *G_generic1*

end

17 The Axiom of Pairing in $M[G]$

theory *Pairing_Axiom*
  imports
    Names
begin

context *G_generic1*
begin

lemma *val_Upair* :
  $\mathbf{1} \in G \Longrightarrow val(G,\{\langle\tau,\mathbf{1}\rangle,\langle\varrho,\mathbf{1}\rangle\}) = \{val(G,\tau),val(G,\varrho)\}$
  ⟨*proof*⟩

lemma *pairing_in_MG* : *upair_ax*(##$M[G]$)
⟨*proof*⟩

end — *G_generic1*

end

18 The Axiom of Unions in $M[G]$

theory *Union_Axiom*
  imports *Names*
begin

definition *Union_name_body* :: $[i,i,i,i] \Rightarrow o$ **where**
  $Union\_name\_body(P,leq,\tau,x) \equiv \exists\ \sigma{\in}domain(\tau)\ .\ \exists\ q{\in}P\ .\ \exists\ r{\in}P\ .$
    $\langle\sigma,q\rangle \in \tau \wedge \langle fst(x),r\rangle \in \sigma \wedge \langle snd(x),r\rangle \in leq \wedge \langle snd(x),q\rangle \in leq$

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*⟩

lemma *Union_MG_Eq* :
  assumes $a \in M[G]$ **and** $a = val(G,\tau)$ **and** *filter*(G) **and** $\tau \in M$
  shows $\bigcup\ a = val(G,Union\_name(\mathbb{P},leq,\tau))$

⟨*proof*⟩

lemma *union_in_MG* :
  assumes *filter*(*G*)
  shows *Union_ax*(*##M*[*G*])
  ⟨*proof*⟩

theorem *Union_MG* : *M_generic*(*G*) ⟹ *Union_ax*(*##M*[*G*])
  ⟨*proof*⟩

end — *forcing_data1*

end

19 The Powerset Axiom in $M[G]$

theory *Powerset_Axiom*
  imports
    Separation_Axiom Pairing_Axiom Union_Axiom
begin

⟨*ML*⟩

context *G_generic1*
begin

lemma *sats_fst_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** *?ϑ* ∈ *M*)
⟨*proof*⟩

declare *nat_into_M*[*rule del*, *simplified setclass_iff*, *intro*]
lemmas *ssimps* = *domain_closed cartprod_closed cons_closed Pow_rel_closed*
declare *ssimps* [*simp del*, *simplified setclass_iff*, *simp*, *intro*]

— We keep *Pow*(*a*) ∩ *M*[*G*] to be consistent with Kunen.
lemma *Pow_inter_MG*:
  assumes *a*∈*M*[*G*]
  shows *Pow*(*a*) ∩ *M*[*G*] ∈ *M*[*G*]
⟨*proof*⟩

end — *G_generic1*

sublocale *G_generic1* ⊆ *ext*: *M_trivial ##M*[*G*]
  ⟨*proof*⟩

context *G_generic1* **begin**

theorem *power_in_MG* : *power_ax*(##(*M*[*G*]))
⟨*proof*⟩

end — *G_generic1*

end

20 The Axiom of Extensionality in $M[G]$

theory *Extensionality_Axiom*
imports
Names
begin

context *forcing_data1*
begin

lemma *extensionality_in_MG* : *extensionality*(##(*M*[*G*]))
⟨*proof*⟩

end — *forcing_data1*

end

21 The Axiom of Foundation in $M[G]$

theory *Foundation_Axiom*
imports
Names
begin

context *forcing_data1*
begin

lemma *foundation_in_MG* : *foundation_ax*(##(*M*[*G*]))
⟨*proof*⟩

lemma *foundation_ax*(##(*M*[*G*]))
⟨*proof*⟩

end — *forcing_data1*

end

22 The Axiom of Replacement in $M[G]$

theory *Replacement_Axiom*
imports
Separation_Axiom
begin

context *forcing_data1*
begin

bundle *sharp_simps1* = *snd_abs*[*simp*] *fst_abs*[*simp*] *fst_closed*[*simp del*, *simplified*, *simp*]
snd_closed[*simp del*, *simplified*, *simp*] *M_inhabited*[*simplified*, *simp*]
pair_in_M_iff[*simp del*, *simplified*, *simp*]

lemma *sats_body_ground_repl_fm*:
includes *sharp_simps1*
assumes
$\exists\, t\ p.\ x{=}\langle t{,}p\rangle\ [x{,}\alpha{,}m{,}\mathbb{P}{,}leq{,}\mathbf{1}]\ @\ nenv \in list(M)$
$\varphi \in formula$
shows
$(\exists\, \tau{\in}M.\ \exists\, V{\in}M.\ is\_Vset(\lambda a.\ (\#\#M)(a){,}\alpha{,}V)\ \wedge\ \tau\ \in\ V\ \wedge\ (snd(x)\ \Vdash\ \varphi\ ([fst(x){,}\tau]@nenv)))$
$\longleftrightarrow\ M,\ [\alpha,\ x,\ m,\ \mathbb{P},\ leq,\ \mathbf{1}]\ @\ nenv \models body\_ground\_repl\_fm(\varphi)$
⟨*proof*⟩

end — *forcing_data1*

context *G_generic1*
begin

lemma *Replace_sats_in_MG*:
assumes
$c{\in}M[G]\ \ env \in list(M[G])$
$\varphi \in formula\ \ arity(\varphi) \le 2 +_\omega length(env)$
$univalent(\#\#M[G],\ c,\ \lambda x\ v.\ (M[G]\ ,\ [x{,}v]@env \models \varphi)\ )$
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*⟩

theorem *strong_replacement_in_MG*:
assumes
$\varphi{\in}formula$ **and** $arity(\varphi) \le 2 +_\omega length(env)\ \ env \in list(M[G])$
and
ground_replacement:
$\bigwedge nenv.\ ground\_replacement\_assm(M{,}[\mathbb{P}{,}leq{,}\mathbf{1}]\ @\ nenv,\ \varphi)$

shows
strong_replacement(##$M[G]$,$\lambda x\ v.\ M[G]$,$[x,v]$ @ *env* $\models \varphi$)
⟨*proof*⟩

lemma *replacement_assm_MG*:
assumes
ground_replacement:
$\bigwedge$*nenv*. *ground_replacement_assm*(M,[$\mathbb{P}$,*leq*,$\mathbf{1}$] @ *nenv*, φ)
shows
replacement_assm($M[G]$,*env*,φ)
⟨*proof*⟩

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*##$M[G]$
⟨*proof*⟩

lemma *infinity_in_MG* : *infinity_ax*(##$M[G]$)
⟨*proof*⟩

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(τ,ϱ,*on*) ≡ *Upair*(⟨τ,*on*⟩,⟨ϱ,*on*⟩)

definition
opair_name :: $i \Rightarrow i \Rightarrow i \Rightarrow i$ **where**
$opair\_name(\tau,\varrho,on) \equiv upair\_name(upair\_name(\tau,\tau,on),upair\_name(\tau,\varrho,on),on)$

definition
induced_surj :: $i \Rightarrow i \Rightarrow i \Rightarrow i$ **where**
$induced\_surj(f,a,e) \equiv f\text{-}``(range(f)\text{-}a)\times\{e\} \cup restrict(f,f\text{-}``a)$

lemma *domain_induced_surj*: $domain(induced\_surj(f,a,e)) = domain(f)$
⟨*proof*⟩

lemma *range_restrict_vimage*:
assumes *function*(f)
shows $range(restrict(f,f\text{-}``a)) \subseteq a$
⟨*proof*⟩

lemma *induced_surj_type*:
assumes *function*(f)
shows
$induced\_surj(f,a,e)$: $domain(f) \to \{e\} \cup a$
and
$x \in f\text{-}``a \Longrightarrow induced\_surj(f,a,e)`x = f`x$
⟨*proof*⟩

lemma *induced_surj_is_surj* :
assumes
$e \in a$ *function*(f) $domain(f) = \alpha$ $\bigwedge y.\ y \in a \Longrightarrow \exists\, x{\in}\alpha.\ f\ `\ x = y$
shows $induced\_surj(f,a,e) \in surj(\alpha,a)$
⟨*proof*⟩

lemma (**in** *M_ZF1_trans*) *upair_name_closed* :
$[\![\ x{\in}M;\ y{\in}M\ ;\ o{\in}M\ ]\!] \Longrightarrow upair\_name(x,y,o){\in}M$
⟨*proof*⟩

context *G_generic1*
begin

lemma *val_upair_name* : $val(G,upair\_name(\tau,\varrho,\mathbf{1})) = \{val(G,\tau),val(G,\varrho)\}$
⟨*proof*⟩

lemma *val_opair_name* : $val(G,opair\_name(\tau,\varrho,\mathbf{1})) = \langle val(G,\tau),val(G,\varrho)\rangle$
⟨*proof*⟩

lemma *val_RepFun_one*: $val(G,\{\langle f(x),\mathbf{1}\rangle\ .\ x{\in}a\}) = \{val(G,f(x))\ .\ x{\in}a\}$
⟨*proof*⟩

end— *G_generic1*

24.1 $M[G]$ is a transitive model of ZF

sublocale *G_generic1* ⊆ *ext*:*M_Z_trans* *M*[*G*]
⟨*proof*⟩

lemma (**in** *M_replacement*) *upair_name_lam_replacement* :
M(*z*) ⟹ *lam_replacement*(*M*,λ*x* . *upair_name*(*fst*(*x*),*snd*(*x*),*z*))
⟨*proof*⟩

lemma (**in** *forcing_data1*) *repl_opname_check* :
assumes *A*∈*M* *f*∈*M*
shows {*opair_name*(*check*(*x*),*f*‘*x*,**1**). *x*∈*A*}∈*M*
⟨*proof*⟩

theorem (**in** *G_generic1*) *choice_in_MG*:
assumes *choice_ax*(#⁺#*M*)
shows *choice_ax*(#⁺#*M*[*G*])
⟨*proof*⟩

sublocale *G_generic1_AC* ⊆ *ext*:*M_ZC_basic* *M*[*G*]
⟨*proof*⟩

end

25 Separative notions and proper extensions

theory *Proper_Extension*
imports
Names

begin

The key ingredient to obtain a proper extension is to have a *separative preorder*:

locale *separative_notion* = *forcing_notion* +
assumes *separative*: *p*∈ℙ ⟹ ∃ *q*∈ℙ. ∃ *r*∈ℙ. *q* ≼ *p* ∧ *r* ≼ *p* ∧ *q* ⊥ *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*(ℙ - *G*)
⟨*proof*⟩

end — *separative_notion*

locale *ctm_separative* = *forcing_data1* + *separative_notion*
begin

context
  fixes G
  assumes *generic*: $M\_generic(G)$
begin

interpretation *G_generic1* $\mathbb{P}$ *leq* **1** M *enum* G
  ⟨*proof*⟩

lemma *generic_not_in_M*:
  shows $G \notin M$
⟨*proof*⟩

theorem *proper_extension*: $M \neq M[G]$
  ⟨*proof*⟩
end
end — *ctm_separative*

end

26 A poset of succession

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 ω -||> *2* or equivalently as the set of partial functions *Fn*(ω, ω, *2*), i.e. the set of partial functions bounded by ω.

The order relation of the poset is that of being less defined as functions (cf. *Fnlerel*($A^{<\omega}$)), so it could be surprising that we have not used *Fn*(ω, ω, *2*) for the set. The only reason why we keep this alternative definition is because we can prove $A^{<\omega} \in M$ (and therefore *Fnlerel*($A^{<\omega}$) $\in M$) using only one instance of separation.

definition *seq_upd* :: $i \Rightarrow i \Rightarrow i$ **where**
  seq_upd(f,a) $\equiv \lambda\ j \in succ(domain(f))$. *if* $j < domain(f)$ *then* $f`j$ *else* a

lemma *seq_upd_succ_type* :
  assumes $n \in nat$ $f \in n \to A$ $a \in A$
  shows *seq_upd*(f,a)$\in succ(n) \to A$
⟨*proof*⟩

lemma *seq_upd_type* :
  assumes $f \in A^{<\omega}$ $a \in A$

shows $seq\_upd(f,a) \in A^{<\omega}$
⟨*proof*⟩

lemma *seq_upd_apply_domain* [*simp*]:
assumes $f{:}n{\to}A$ $n{\in}nat$
shows $seq\_upd(f,a)`n = a$
⟨*proof*⟩

lemma *zero_in_seqspace* :
shows $0 \in A^{<\omega}$
⟨*proof*⟩

definition
$seqlerel :: i \Rightarrow i$ **where**
$seqlerel(A) \equiv Fnlerel(A^{<\omega})$

definition
$seqle :: i$ **where**
$seqle \equiv seqlerel(2)$

lemma *seqleI*[*intro*!]:
$\langle f,g\rangle \in 2^{<\omega}{\times}2^{<\omega} \Longrightarrow g \subseteq f \Longrightarrow \langle f,g\rangle \in seqle$
⟨*proof*⟩

lemma *seqleD*[*dest*!]:
$z \in seqle \Longrightarrow \exists x\ y.\ \langle x,y\rangle \in 2^{<\omega}{\times}2^{<\omega} \wedge y \subseteq x \wedge z = \langle x,y\rangle$
⟨*proof*⟩

lemma *upd_leI* :
assumes $f{\in}2^{<\omega}$ $a{\in}2$
shows $\langle seq\_upd(f,a),f\rangle{\in}seqle$ (**is** $\langle ?f,\_\rangle{\in}\_$)
⟨*proof*⟩

lemma *preorder_on_seqle*: $preorder\_on(2^{<\omega},seqle)$
⟨*proof*⟩

lemma *zero_seqle_max*: $x{\in}2^{<\omega} \Longrightarrow \langle x,0\rangle \in seqle$
⟨*proof*⟩

interpretation *sp*:*forcing_notion* $2^{<\omega}$ *seqle* 0
⟨*proof*⟩

notation *sp.Leq* (**infixl** $\preceq s$ 50)
notation *sp.Incompatible* (**infixl** $\perp s$ 50)

lemma *seqspace_separative*:
assumes $f{\in}2^{<\omega}$
shows $seq\_upd(f,0)\ \perp s\ seq\_upd(f,1)$ (**is** $?f\ \perp s\ ?g$)
⟨*proof*⟩

definition *seqleR_fm* :: $i \Rightarrow i$ **where**
  $seqleR\_fm(fg) \equiv Exists(Exists(And(pair\_fm(0,1,fg+_\omega 2),subset\_fm(1,0))))$

lemma *type_seqleR_fm* : $fg \in nat \Longrightarrow seqleR\_fm(fg) \in formula$
  ⟨*proof*⟩

⟨*ML*⟩

lemma (**in** *M_ctm1*) *seqleR_fm_sats* :
  assumes $fg{\in}nat$ $env{\in}list(M)$
 shows $(M, env \models seqleR\_fm(fg)) \longleftrightarrow (\exists f[\#\#M].\ \exists g[\#\#M].\ pair(\#\#M,f,g,nth(fg,env)) \wedge f \supseteq g)$
  ⟨*proof*⟩

context *M_ctm1*
begin

lemma *seqle_in_M*: $seqle \in M$
  ⟨*proof*⟩

26.1 Cohen extension is proper

interpretation *ctm_separative* $2^{<\omega}$ *seqle 0*
⟨*proof*⟩

lemma *cohen_extension_is_proper*: $\exists\, G.\ M\_generic(G) \wedge M \neq M[G]$
  ⟨*proof*⟩

end — *M_ctm1*

end

27 The existence of generic extensions

theory *Forcing_Main*
  imports
    Ordinals_In_MG
    Choice_Axiom
    Succession_Poset

begin

27.1 The generic extension is countable

lemma (**in** *forcing_data1*) *surj_nat_MG* : $\exists f.\ f \in surj(\omega,M[G])$
⟨*proof*⟩

lemma (**in** *G_generic1*) *MG_eqpoll_nat*: $M[G] \approx \omega$

⟨*proof*⟩

27.2 Extensions of ctms of fragments of *ZFC*

context *G_generic1*
begin

lemma *sats_ground_repl_fm_imp_sats_ZF_replacement_fm*:
assumes
$\varphi \in formula$ M, $[] \models \cdot Replacement(ground\_repl\_fm(\varphi))\cdot$
shows
$M[G]$, $[] \models \cdot Replacement(\varphi)\cdot$
⟨*proof*⟩

lemma *satT_ground_repl_fm_imp_satT_ZF_replacement_fm*:
assumes
$\Phi \subseteq formula$ $M \models \{ \cdot Replacement(ground\_repl\_fm(\varphi))\cdot \,.\, \varphi \in \Phi\}$
shows
$M[G] \models \{ \cdot Replacement(\varphi)\cdot \,.\, \varphi \in \Phi\}$
⟨*proof*⟩

end — *G_generic1*

theorem *extensions_of_ctms*:
assumes
$M \approx \omega$ $Transset(M)$
$M \models \cdot Z\cdot \cup \{\cdot Replacement(p)\cdot \,.\, p \in overhead\}$
$\Phi \subseteq formula$ $M \models \{ \cdot Replacement(ground\_repl\_fm(\varphi))\cdot \,.\, \varphi \in \Phi\}$
shows
$\exists N.$
$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge M{\neq}N \wedge$
$(\forall \alpha.\ Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N)) \wedge$
$((M, []\models \cdot AC\cdot) \longrightarrow N, [] \models \cdot AC\cdot) \wedge N \models \cdot Z\cdot \cup \{ \cdot Replacement(\varphi)\cdot \,.\, \varphi \in \Phi\}$
⟨*proof*⟩

lemma *ZF_replacement_overhead_sub_ZF*: $\{\cdot Replacement(p)\cdot \,.\, p \in overhead\} \subseteq ZF$
⟨*proof*⟩

theorem *extensions_of_ctms_ZF*:
assumes
$M \approx \omega$ $Transset(M)$ $M \models ZF$
shows
$\exists N.$
$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZF \wedge M{\neq}N \wedge$
$(\forall \alpha.\ Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N)) \wedge$
$((M, []\models \cdot AC\cdot) \longrightarrow N \models ZFC)$
⟨*proof*⟩

end

28 Preservation of cardinals in generic extensions

theory *Cardinal_Preservation*
  imports
    Forcing_Main
begin

context *forcing_data1*

begin

lemma *antichain_abs'* [*absolut*]:
  ⟦ *A*∈*M* ⟧ ⟹ *antichain*^{M}(ℙ,*leq*,*A*) ⟷ *antichain*(ℙ,*leq*,*A*)
  ⟨*proof*⟩

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*⟩

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*]
  ⟨*proof*⟩

lemma (**in** *forcing_data1*) *forces_neq_apply_imp_incompatible*:
  assumes
    p ⊩ ·*0*‘*1 is 2*· [*f*,*a*,*b*^{v}]
    q ⊩ ·*0*‘*1 is 2*· [*f*,*a*,*b′*^{v}]
    b ≠ *b′*
   — More general version: taking general names b^v and b'^v, satisfying $p \Vdash \cdot\neg\cdot 0 = 1\cdot\cdot\ [b^v,\ b'^v]$ and $q \Vdash \cdot\neg\cdot 0 = 1\cdot\cdot\ [b^v,\ b'^v]$.
    and
    types:*f*∈*M a*∈*M b*∈*M b′*∈*M p*∈ℙ *q*∈ℙ
  shows

$p \perp q$
⟨*proof*⟩
include *G_generic1_lemmas*
⟨*proof*⟩

context *M_ctm2_AC*
begin

— Simplifying simp rules (because of the occurrence of *setclass*)
lemmas *sharp_simps* = *Card_rel_Union Card_rel_cardinal_rel Collect_abs Cons_abs Cons_in_M_iff Diff_closed Equal_abs Equal_in_M_iff Finite_abs Forall_abs Forall_in_M_iff Inl_abs Inl_in_M_iff Inr_abs Inr_in_M_iff Int_closed Inter_abs Inter_closed M_nat Member_abs Member_in_M_iff Memrel_closed Nand_abs Nand_in_M_iff Nil_abs Nil_in_M Ord_cardinal_rel Pow_rel_closed Un_closed Union_abs Union_closed and_abs and_closed apply_abs apply_closed bij_rel_closed bijection_abs bool_of_o_abs bool_of_o_closed cadd_rel_0 cadd_rel_closed cardinal_rel_0_iff_0 cardinal_rel_closed cardinal_rel_idem cartprod_abs cartprod_closed cmult_rel_0 cmult_rel_1 cmult_rel_closed comp_closed composition_abs cons_abs cons_closed converse_abs converse_closed csquare_lam_closed csquare_rel_closed depth_closed domain_abs domain_closed eclose_abs eclose_closed empty_abs field_abs field_closed finite_funspace_closed finite_ordinal_abs fst_closed function_abs function_space_rel_closed hd_abs image_abs image_closed inj_rel_closed injection_abs inter_abs irreflexive_abs is_eclose_n_abs is_funspace_abs iterates_closed length_closed lepoll_rel_refl limit_ordinal_abs linear_rel_abs mem_bij_abs mem_eclose_abs mem_inj_abs membership_abs minimum_closed nat_case_abs nat_case_closed nonempty not_abs not_closed number1_abs number2_abs number3_abs omega_abs or_abs or_closed order_isomorphism_abs ordermap_closed ordertype_closed ordinal_abs pair_abs pair_in_M_iff powerset_abs pred_closed pred_set_abs quasilist_abs quasinat_abs radd_closed rall_abs range_abs range_closed relation_abs restrict_closed restriction_abs rex_abs rmult_closed rtrancl_abs rtrancl_closed rvimage_closed separation_closed setdiff_abs singleton_abs singleton_in_M_iff snd_closed strong_replacement_closed subset_abs succ_in_M_iff successor_abs successor_ordinal_abs sum_abs sum_closed surj_rel_closed surjection_abs tl_abs trancl_abs trancl_closed transitive_rel_abs transitive_set_abs typed_function_abs union_abs upair_abs upair_in_M_iff vimage_abs vimage_closed well_ord_abs nth_closed Aleph_rel_closed csucc_rel_closed Card_rel_Aleph_rel*

declare *sharp_simps*[*simp del*, *simplified setclass_iff*, *simp*]

lemmas *sharp_intros* = *nat_into_M Aleph_rel_closed Card_rel_Aleph_rel*

declare *sharp_intros*[*rule del*, *simplified setclass_iff*, *intro*]

end — *M_ctm2_AC*

context *G_generic3_AC* **begin**

context
includes *G_generic1_lemmas*
begin

lemmas *mg_sharp_simps = ext.Card_rel_Union ext.Card_rel_cardinal_rel*
ext.Collect_abs ext.Cons_abs ext.Cons_in_M_iff ext.Diff_closed
ext.Equal_abs ext.Equal_in_M_iff ext.Finite_abs ext.Forall_abs
ext.Forall_in_M_iff ext.Inl_abs ext.Inl_in_M_iff ext.Inr_abs
ext.Inr_in_M_iff ext.Int_closed ext.Inter_abs ext.Inter_closed
ext.M_nat ext.Member_abs ext.Member_in_M_iff ext.Memrel_closed
ext.Nand_abs ext.Nand_in_M_iff ext.Nil_abs ext.Nil_in_M
ext.Ord_cardinal_rel ext.Pow_rel_closed ext.Un_closed
ext.Union_abs ext.Union_closed ext.and_abs ext.and_closed
ext.apply_abs ext.apply_closed ext.bij_rel_closed
ext.bijection_abs ext.bool_of_o_abs ext.bool_of_o_closed
ext.cadd_rel_0 ext.cadd_rel_closed ext.cardinal_rel_0_iff_0
ext.cardinal_rel_closed ext.cardinal_rel_idem ext.cartprod_abs
ext.cartprod_closed ext.cmult_rel_0 ext.cmult_rel_1
ext.cmult_rel_closed ext.comp_closed ext.composition_abs
ext.cons_abs ext.cons_closed ext.converse_abs ext.converse_closed
ext.csquare_lam_closed ext.csquare_rel_closed ext.depth_closed
ext.domain_abs ext.domain_closed ext.eclose_abs ext.eclose_closed
ext.empty_abs ext.field_abs ext.field_closed
ext.finite_funspace_closed ext.finite_ordinal_abs
ext.fst_closed ext.function_abs ext.function_space_rel_closed
ext.hd_abs ext.image_abs ext.image_closed ext.inj_rel_closed
ext.injection_abs ext.inter_abs ext.irreflexive_abs
ext.is_eclose_n_abs ext.is_funspace_abs
ext.iterates_closed ext.length_closed
ext.lepoll_rel_refl ext.limit_ordinal_abs ext.linear_rel_abs
ext.mem_bij_abs ext.mem_eclose_abs
ext.mem_inj_abs ext.membership_abs
ext.nat_case_abs ext.nat_case_closed
ext.nonempty ext.not_abs ext.not_closed
ext.number1_abs ext.number2_abs ext.number3_abs ext.omega_abs
ext.or_abs ext.or_closed ext.order_isomorphism_abs
ext.ordermap_closed ext.ordertype_closed ext.ordinal_abs
ext.pair_abs ext.pair_in_M_iff ext.powerset_abs ext.pred_closed
ext.pred_set_abs ext.quasilist_abs ext.quasinat_abs
ext.radd_closed ext.rall_abs ext.range_abs ext.range_closed
ext.relation_abs ext.restrict_closed ext.restriction_abs
ext.rex_abs ext.rmult_closed ext.rtrancl_abs ext.rtrancl_closed
ext.rvimage_closed ext.separation_closed ext.setdiff_abs
ext.singleton_abs ext.singleton_in_M_iff ext.snd_closed

ext.strong_replacement_closed ext.subset_abs ext.succ_in_M_iff
ext.successor_abs ext.successor_ordinal_abs ext.sum_abs
ext.sum_closed ext.surj_rel_closed ext.surjection_abs ext.tl_abs
ext.trancl_abs ext.trancl_closed ext.transitive_rel_abs
ext.transitive_set_abs ext.typed_function_abs ext.union_abs
ext.upair_abs ext.upair_in_M_iff ext.vimage_abs ext.vimage_closed
ext.well_ord_abs ext.nth_closed ext.Aleph_rel_closed
ext.csucc_rel_closed ext.Card_rel_Aleph_rel

— The following was motivated by the fact that *ext.apply_closed* did not simplify appropriately.
declare *mg_sharp_simps*[*simp del*, *simplified setclass_iff*, *simp*]

lemmas *mg_sharp_intros* = *ext.nat_into_M ext.Aleph_rel_closed*
ext.Card_rel_Aleph_rel

declare *mg_sharp_intros*[*rule del*, *simplified setclass_iff*, *intro*]

— Kunen IV.2.31
lemma *forces_below_filter*:
assumes $M[G]$, *map*(*val*(G),*env*) $\models \varphi\ p \in G$
arity(φ) $\leq$ *length*(*env*) $\varphi \in$ *formula env* $\in$ *list*(M)
shows $\exists\, q{\in}G.\ q \preceq p \wedge q \Vdash \varphi\ env$
⟨*proof*⟩

28.1 Preservation by ccc forcing notions

lemma *ccc_fun_closed_lemma_aux*:
assumes *f_dot*$\in M$ $p{\in}M$ $a{\in}M$ $b{\in}M$
shows $\{q \in \mathbb{P}\ .\ q \preceq p \wedge (M, [q, \mathbb{P}, leq, \mathbf{1}, f\_dot, a^v, b^v] \models forces(\cdot 0\text{`}1 \text{ is } 2\cdot))\} \in M$
⟨*proof*⟩

lemma *ccc_fun_closed_lemma_aux2*:
assumes $B{\in}M$ *f_dot*$\in M$ $p{\in}M$ $a{\in}M$
shows $(\#\#M)(\lambda b{\in}B.\ \{q \in \mathbb{P}\ .\ q \preceq p \wedge (M, [q, \mathbb{P}, leq, \mathbf{1}, f\_dot, a^v, b^v] \models forces(\cdot 0\text{`}1 \text{ is } 2\cdot))\})$
⟨*proof*⟩

lemma *ccc_fun_closed_lemma*:
assumes $A{\in}M$ $B{\in}M$ *f_dot*$\in M$ $p{\in}M$
shows $(\lambda a{\in}A.\ \{b{\in}B.\ \exists\, q{\in}\mathbb{P}.\ q \preceq p \wedge (q \Vdash \cdot 0\text{`}1 \text{ is } 2\cdot\ [f\_dot, a^v, b^v])\}) \in M$
⟨*proof*⟩
lemma *ccc_fun_approximation_lemma*:
notes *le_trans*[*trans*]
assumes $ccc^M(\mathbb{P}, leq)$ $A{\in}M$ $B{\in}M$ $f{\in}M[G]$ $f : A \to B$
shows
$\exists\, F{\in}M.\ F : A \to Pow^M(B) \wedge (\forall\, a{\in}A.\ f\text{`}a \in F\text{`}a \wedge |F\text{`}a|^M \leq \omega)$
⟨*proof*⟩

end — G_generic1_lemmas bundle

end — *G_generic3_AC*

end

29 Model of the negation of the Continuum Hypothesis

theory *Not_CH*
imports
Cardinal_Preservation
begin

We are taking advantage that the poset of finite functions is absolute, and thus we work with the unrelativized *Fn*. But it would have been more appropriate to do the following using the relative *Fn_rel*. As it turns out, the present theory was developed prior to having *Fn* relativized!

We also note that $Fn(\omega,\ \kappa \times \omega,\ 2)$ is separative, i.e. each $X \in Fn(\omega,\ \kappa \times \omega,\ 2)$ has two incompatible extensions; therefore we may recover part of our previous theorem *extensions_of_ctms_ZF*. But that result also included the possibility of not having AC in the ground model, which would not be sensible in a context where the cardinality of the continuum is under discussion. It is also the case that *extensions_of_ctms_ZF* was historically our first formalized result (with a different proof) that showed the forcing machinery had all of its elements in place.

abbreviation
$Add\_subs :: i \Rightarrow i$ **where**
$Add\_subs(\kappa) \equiv Fn(\omega,\kappa\times\omega,2)$

abbreviation
$Add\_le :: i \Rightarrow i$ **where**
$Add\_le(\kappa) \equiv Fnle(\omega,\kappa \times \omega,2)$

lemma (**in** *M_aleph*) *Aleph_rel2_closed*[*intro*,*simp*]: $M({\aleph_2}^M)$
⟨*proof*⟩

locale *M_master* = *M_cohen* + *M_library* +
assumes
UN_lepoll_assumptions:
$M(A) \Longrightarrow M(b) \Longrightarrow M(f) \Longrightarrow M(A') \Longrightarrow separation(M,\ \lambda y.\ \exists x{\in}A'.\ y = \langle x,\ \mu\ i.\ x{\in}if\_range\_F\_else\_F((`)(A),\ b,\ f,\ i)\rangle)$

29.1 Non-absolute concepts between extensions

sublocale *M_master* ⊆ *M_Pi_replacement*

⟨*proof*⟩

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* ⊆ *M_N_Perm*
⟨*proof*⟩

context *M_master_sub*
begin

lemma *cardinal_rel_le_cardinal_rel*: $M(X) \Longrightarrow |X|^N \leq |X|^M$
⟨*proof*⟩

lemma *Aleph_rel_sub_closed*: $Ord(\alpha) \Longrightarrow M(\alpha) \Longrightarrow N({\aleph_\alpha}^M)$
⟨*proof*⟩

lemma *Card_rel_imp_Card_rel*: $Card^N(\kappa) \Longrightarrow M(\kappa) \Longrightarrow Card^M(\kappa)$
⟨*proof*⟩

lemma *csucc_rel_le_csucc_rel*:
assumes $Ord(\kappa)$ $M(\kappa)$
shows $(\kappa^+)^M \leq (\kappa^+)^N$
⟨*proof*⟩

lemma *Aleph_rel_le_Aleph_rel*: $Ord(\alpha) \Longrightarrow M(\alpha) \Longrightarrow {\aleph_\alpha}^M \leq {\aleph_\alpha}^N$
⟨*proof*⟩

end — *M_master_sub*

lemmas (**in** *M_ZF2_trans*) *sep_instances* =
separation_ifrangeF_body separation_ifrangeF_body2 separation_ifrangeF_body3
separation_ifrangeF_body4 separation_ifrangeF_body5 separation_ifrangeF_body6
separation_ifrangeF_body7 separation_cardinal_rel_lesspoll_rel
separation_is_dcwit_body separation_cdltgamma separation_cdeqgamma

lemmas (**in** *M_ZF2_trans*) *repl_instances* = *lam_replacement_inj_rel*

sublocale *M_ZFC2_ground_notCH_trans* ⊆ *M_master* ##*M*
⟨*proof*⟩

sublocale *M_ZFC2_trans* ⊆ *M_Pi_replacement* ##*M*
⟨*proof*⟩

29.2 Cohen forcing is ccc

context *M_ctm2_AC*

begin

lemma *ccc_Add_subs_Aleph_2*: $ccc^M(Add\_subs(\aleph_2{}^M), Add\_le(\aleph_2{}^M))$
⟨*proof*⟩

end — *M_ctm2_AC*

sublocale *G_generic3_AC* ⊆ *M_master_sub* ##*M* ##(*M*[*G*])
⟨*proof*⟩

lemma (**in** *M_trans*) *mem_F_bound4*:
fixes *F A*
defines $F \equiv (`)$
shows $x{\in}F(A,c) \Longrightarrow c \in (range(f) \cup domain(A))$
⟨*proof*⟩

lemma (**in** *M_trans*) *mem_F_bound5*:
fixes *F A*
defines $F \equiv \lambda\_.\ x.\ A`x$
shows $x{\in}F(A,c) \Longrightarrow c \in (range(f) \cup domain(A))$
⟨*proof*⟩

sublocale *M_ctm2_AC* ⊆ *M_replacement_lepoll* ##*M* (`)
⟨*proof*⟩

context *G_generic3_AC* **begin**

context
includes *G_generic1_lemmas*
begin

lemma *G_in_MG*: $G \in M[G]$
⟨*proof*⟩

lemma *ccc_preserves_Aleph_succ*:
assumes $ccc^M(\mathbb{P}, leq)$ $Ord(z)$ $z \in M$
shows $Card^{M[G]}(\aleph_{succ(z)}{}^M)$
⟨*proof*⟩

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*(ω, $\aleph_2^{\#\#M} \times \omega$, *2*) *Fnle*($\omega$, $\aleph_2^{\#\#M} \times \omega$, *2*) *0*

sublocale *add_generic3* $\subseteq$ *cohen_data* ω $\aleph_2^M \times \omega$ *2* ⟨*proof*⟩

context *add_generic3*
begin

notation *Leq* (**infixl** $\preceq$ *50*)
notation *Incompatible* (**infixl** $\perp$ *50*)

lemma *Add_subs_preserves_Aleph_succ*: $Ord(z) \Longrightarrow z{\in}M \Longrightarrow Card^{M[G]}(\aleph_{succ(z)}{}^M)$
⟨*proof*⟩

lemma *Aleph_rel_nats_MG_eq_Aleph_rel_nats_M*:
includes *G_generic1_lemmas*
assumes $z \in \omega$
shows $\aleph_z{}^{M[G]} = \aleph_z{}^M$
⟨*proof*⟩

abbreviation
$f\_G :: i$ (‹f_G›) **where**
$f_G \equiv \bigcup G$

abbreviation
dom_dense :: $i \Rightarrow i$ **where**
$dom\_dense(x) \equiv \{p \in Add\ .\ x \in domain(p)\ \}$

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*]: $x \in \aleph_2^M \times \omega \Longrightarrow dom\_dense(x) \in M$
⟨*proof*⟩

lemma *domain_f_G*: **assumes** $x \in \aleph_2^M$ $y \in \omega$
shows $\langle x, y\rangle \in domain(f_G)$
⟨*proof*⟩

lemma *f_G_funtype*:
includes *G_generic1_lemmas*
shows $f_G : \aleph_2^M \times \omega \to 2$
⟨*proof*⟩

lemma *inj_dense_closed*[*intro*,*simp*]:
  $w \in \aleph_2^M \Longrightarrow x \in \aleph_2^M \Longrightarrow inj\_dense(\aleph_2^M,2,w,x) \in M$
  ⟨*proof*⟩

lemma *Aleph_rel2_new_reals*:
  assumes $w \in \aleph_2^M$ $x \in \aleph_2^M$ $w \neq x$
  shows $(\lambda n{\in}\omega.\ f_G\ `\ \langle w,\ n\rangle) \neq (\lambda n{\in}\omega.\ f_G\ `\ \langle x,\ n\rangle)$
⟨*proof*⟩

definition
  $h\_G :: i\ (‹h_G›)$ **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 $h_G \in M[G]$
  ⟨*proof*⟩

lemma *h_G_inj_Aleph_rel2_reals*: $h_G \in inj^{M[G]}(\aleph_2^M,\ \omega \rightarrow^{M[G]} 2)$
  ⟨*proof*⟩

lemma *Aleph2_extension_le_continuum_rel*:
  includes *G_generic1_lemmas*
  shows $\aleph_2^{M[G]} \leq 2^{\uparrow\aleph_0{}^{M[G]},M[G]}$
⟨*proof*⟩

lemma *Aleph_rel_lt_continuum_rel*: $\aleph_1^{M[G]} < 2^{\uparrow\aleph_0{}^{M[G]},M[G]}$
  ⟨*proof*⟩

corollary *not_CH*: $\aleph_1^{M[G]} \neq 2^{\uparrow\aleph_0{}^{M[G]},M[G]}$
  ⟨*proof*⟩

end — *add_generic3*

29.3 Models of fragments of $ZFC + \neg CH$

definition
  ContHyp :: *o* **where**
  $ContHyp \equiv \aleph_1 = 2^{\uparrow\aleph_0}$

⟨*ML*⟩
notation *ContHyp_rel* (‹*CH*–›)
⟨*ML*⟩

context *M_ZF_library*
begin

⟨*ML*⟩

⟨*proof*⟩

end — *M_ZF_library*

⟨*ML*⟩

notation *is_ContHyp_fm* (‹·*CH*·›)

theorem *ctm_of_not_CH*:
assumes
$M \approx \omega$ *Transset*(*M*) $M \models ZC \cup \{\cdot Replacement(p)\cdot . p \in overhead\_notCH\}$
$\Phi \subseteq formula$ $M \models \{ \cdot Replacement(ground\_repl\_fm(\varphi))\cdot . \varphi \in \Phi\}$
shows
$\exists N.$
$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZC \cup \{\cdot\neg\cdot CH\cdot\cdot\} \cup \{ \cdot Replacement(\varphi)\cdot . \varphi \in \Phi\} \wedge$
$(\forall \alpha.\ Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$
⟨*proof*⟩

lemma *ZF_replacement_overhead_sub_ZFC*: $\{\cdot Replacement(p)\cdot . p \in overhead\} \subseteq ZFC$
⟨*proof*⟩

lemma *ZF_replacement_overhead_notCH_sub_ZFC*: $\{\cdot Replacement(p)\cdot . p \in overhead\_notCH\} \subseteq ZFC$
⟨*proof*⟩

lemma *ZF_replacement_overhead_CH_sub_ZFC*: $\{\cdot Replacement(p)\cdot . p \in overhead\_CH\} \subseteq ZFC$
⟨*proof*⟩

corollary *ctm_ZFC_imp_ctm_not_CH*:
assumes
$M \approx \omega$ *Transset*(*M*) $M \models ZFC$
shows
$\exists N.$
$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZFC \cup \{\cdot\neg\cdot CH\cdot\cdot\} \wedge$
$(\forall \alpha.\ Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$
⟨*proof*⟩

end

30 Preservation results for κ-closed forcing notions

theory *Kappa_Closed_Notions*
imports
Not_CH
begin

definition
lerel :: $i \Rightarrow i$ **where**
$lerel(\alpha) \equiv Memrel(\alpha) \cup id(\alpha)$

lemma *lerelI*[*intro*!]: $x \leq y \Longrightarrow y \in \alpha \Longrightarrow Ord(\alpha) \Longrightarrow \langle x,y\rangle \in lerel(\alpha)$
⟨*proof*⟩

lemma *lerelD*[*dest*]: $\langle x,y\rangle \in lerel(\alpha) \Longrightarrow Ord(\alpha) \Longrightarrow x \leq y$
⟨*proof*⟩

definition
mono_seqspace :: $[i,i,i] \Rightarrow i$ ($\langle\_\ {}_{<}\!\!\rightarrow\ '(\_,\_')\rangle$ [61] 60) **where**
$\alpha\ {}_{<}\!\!\rightarrow\ (P,leq) \equiv mono\_map(\alpha,Memrel(\alpha),P,leq)$

⟨*ML*⟩

context *M_ZF_library*
begin

⟨*ML*⟩
⟨*proof*⟩

end — *M_ZF_library*

abbreviation
mono_seqspace_r ($\langle\_\ {}_{<}\!\!\rightarrow\!\!-\ '(\_,\_')\rangle$ [61] 60) **where**
$\alpha\ {}_{<}\!\!\rightarrow^{M}\ (P,leq) \equiv mono\_seqspace\_rel(M,\alpha,P,leq)$

abbreviation
mono_seqspace_r_set ($\langle\_\ {}_{<}\!\!\rightarrow\!\!-\ '(\_,\_')\rangle$ [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*: $A \rightarrow P$ $\bigwedge x\ y.\ x \in A \Longrightarrow y \in A \Longrightarrow x < y \Longrightarrow \langle f`x,\ f`y\rangle \in leq\ Ord(A)$
shows *f*: $A\ {}_{<}\!\!\rightarrow\ (P,leq)$
⟨*proof*⟩

lemma (**in** *M_ZF_library*) *mono_seqspace_rel_char*:
assumes $M(A)$ $M(P)$ $M(leq)$
shows $A\ {}_{<}\!\!\rightarrow^{M}\ (P,leq) = \{f \in A\ {}_{<}\!\!\rightarrow\ (P,leq).\ M(f)\}$
⟨*proof*⟩

lemma (**in** *M_ZF_library*) *mono_seqspace_relI*[*intro*!]:
assumes *f*: $A \rightarrow^{M} P$ $\bigwedge x\ y.\ x \in A \Longrightarrow y \in A \Longrightarrow x < y \Longrightarrow \langle f`x,\ f`y\rangle \in leq$
$Ord(A)$ $M(A)$ $M(P)$ $M(leq)$
shows *f*: $A\ {}_{<}\!\!\rightarrow^{M}\ (P,leq)$
⟨*proof*⟩

lemma *mono_seqspace_is_fun*[*dest*]:
includes *mono_map_rules*
shows *j*: $A\ {}_{<}\!\!\to (P,leq) \Longrightarrow j\colon A\to P$
⟨*proof*⟩

lemma *mono_map_lt_le_is_mono*[*dest*]:
includes *mono_map_rules*
assumes *j*: $A\ {}_{<}\!\!\to (P,leq)$ $a{\in}A$ $c{\in}A$ $a{\le}c$ $Ord(A)$ $refl(P,leq)$
shows $\langle j`a,j`c\rangle \in leq$
⟨*proof*⟩

lemma (**in** *M_ZF_library*) *mem_mono_seqspace_abs*[*absolut*]:
assumes $M(f)$ $M(A)$ $M(P)$ $M(leq)$
shows $f{:}A\ {}_{<}\!\!\to^{M} (P,leq) \longleftrightarrow f\colon A\ {}_{<}\!\!\to (P,leq)$
⟨*proof*⟩

definition
$mono\_map\_lt\_le :: [i,i] \Rightarrow i$ (**infixr** $\langle_{<}\!\!\to_{\le}\rangle$ *60*) **where**
$\alpha\ {}_{<}\!\!\to_{\le} \beta \equiv \alpha\ {}_{<}\!\!\to (\beta,lerel(\beta))$

lemma *mono_map_lt_leI*[*intro*!]:
includes *mono_map_rules*
assumes $f\colon A\to B$ $\bigwedge x\ y.\ x{\in}A \Longrightarrow y{\in}A \Longrightarrow x{<}y \Longrightarrow f`x \le f`y$ $Ord(A)$ $Ord(B)$
shows $f\colon A\ {}_{<}\!\!\to_{\le} B$
⟨*proof*⟩
definition
$kappa\_closed :: [i,i,i] \Rightarrow o$ ($\langle\_\text{-}closed'(\_,\_')\rangle$) **where**
$\kappa\text{-}closed(P,leq) \equiv \forall \delta.\ \delta{<}\kappa \longrightarrow (\forall f{\in}\delta\ {}_{<}\!\!\to (P,converse(leq)).\ \exists q{\in}P.\ \forall \alpha{\in}\delta.$
$\langle q,f`\alpha\rangle{\in}leq)$

⟨*ML*⟩

abbreviation
$kappa\_closed\_r$ ($\langle\_\text{-}closed{-}'(\_,\_')\rangle$ [*61*] *60*) **where**
$\kappa\text{-}closed^{M}(P,leq) \equiv kappa\_closed\_rel(M,\kappa,P,leq)$

abbreviation
$kappa\_closed\_r\_set$ ($\langle\_\text{-}closed{-}'(\_,\_')\rangle$ [*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 $p \Vdash \cdot 0{:}1\to 2\cdot\ [f\_dot,\ A^v,\ B^v]$ $a \in A$
$q \preceq p$ $q \in \mathbb{P}$ $p{\in}\mathbb{P}$ $f\_dot \in M$ $A{\in}M$ $B{\in}M$
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*⟩
include *G_generic1_lemmas*
⟨*proof*⟩

locale *M_master_CH* = *M_master* + *M_library_DC*

sublocale *M_ZFC2_ground_CH_trans* ⊆ *M_master_CH* *##M*
⟨*proof*⟩

context *G_generic3_AC_CH* **begin**

context
includes *G_generic1_lemmas*
begin

lemma *separation_check_snd_aux*:
assumes *f_dot*∈*M* *τ*∈*M* *χ*∈*formula* *arity*(*χ*) ≤ 7
shows *separation*(*##M*, *λr*. *M*, [*fst*(*r*), ℙ, *leq*, **1**, *f_dot*, *τ*, *snd*(*r*)^v] ⊨ *χ*)
⟨*proof*⟩

lemma *separation_check_fst_snd_aux* :
assumes *f_dot*∈*M* *r*∈*M* *χ*∈*formula* *arity*(*χ*) ≤ 7
shows *separation*(*##M*, *λp*. *M*, [*r*, ℙ, *leq*, **1**, *f_dot*, *fst*(*p*)^v, *snd*(*p*)^v] ⊨ *χ*)
⟨*proof*⟩

lemma *separation_leq_and_forces_apply_aux*:
assumes *f_dot*∈*M* *B*∈*M*
shows ∀ *n*∈*M*. *separation*(*##M*, *λx*. *snd*(*x*) ≼ *fst*(*x*) ∧
(∃ *b*∈*B*. *M*, [*snd*(*x*), ℙ, *leq*, **1**, *f_dot*, (⋃(*n*))^v, *b*^{v}] ⊨ *forces*(⋅0‘1 *is* 2⋅)))
⟨*proof*⟩

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, *b*^{v}] ⊨ *forces*(⋅0‘1 *is* 2⋅)))
⟨*proof*⟩

lemma *separation_closed_leq_and_forces_eq_check_aux* :
assumes *A*∈*M* *r*∈*G* *τ* ∈ *M*
shows (*##M*)({*q*∈ℙ. ∃ *h*∈*A*. *q* ≼ *r* ∧ *q* ⊩ ⋅0 = 1⋅ [*τ*, *h*^{v}]})
⟨*proof*⟩

lemma *separation_closed_forces_apply_aux*:
assumes *B*∈*M* *f_dot*∈*M* *r*∈*M*
shows (*##M*)({⟨*n*,*b*⟩ ∈ *ω* × *B*. *r* ⊩ ⋅0‘1 *is* 2⋅ [*f_dot*, *n*^{v}, *b*^{v}]})
⟨*proof*⟩
lemma *kunen_IV_6_9_function_space_rel_eq*:
assumes ⋀*p* *τ*. *p* ⊩ ⋅0:1→2⋅ [*τ*, *A*^{v}, *B*^{v}] ⟹ *p*∈ℙ ⟹ *τ* ∈ *M* ⟹
∃ *q*∈ℙ. ∃ *h*∈*A* →^{M} *B*. *q* ≼ *p* ∧ *q* ⊩ ⋅0 = 1⋅ [*τ*, *h*^{v}] *A*∈*M* *B*∈*M*
shows
A →^{M} *B* = *A* →$^{M[G]}$ *B*
⟨*proof*⟩

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*) $f : \omega \to B$ $f{\in}M[G]$ $B{\in}M$
  shows $f{\in}M$
⟨*proof*⟩

declare *mono_seqspace_rel_closed*[*rule del*]
  — Mysteriously breaks the end of the next proof

lemma *succ_omega_closed_imp_no_new_reals*:
  assumes *succ*(ω)-*closed*^{M}($\mathbb{P}$,*leq*)
  shows $\omega \to^M 2 = \omega \to^{M[G]} 2$
⟨*proof*⟩

lemma *succ_omega_closed_imp_Aleph_1_preserved*:
  assumes *succ*(ω)-*closed*^{M}($\mathbb{P}$,*leq*)
  shows $\aleph_1{}^M = \aleph_1{}^{M[G]}$
⟨*proof*⟩

end — bundle G_generic1_lemmas

end — *G_generic3_AC*

end

31 Forcing extension satisfying the Continuum Hypothesis

theory *CH*
  imports
    Kappa_Closed_Notions
    Cohen_Posets_Relative
begin

context *M_ctm2_AC*
begin

declare *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 ≡ *Fn*^{M}($\aleph_1{}^M$, $\aleph_1{}^M$, $\omega \to^M 2$)

abbreviation
  Colleq :: *i* **where**
  Colleq ≡ *Fnle*^{M}($\aleph_1{}^M$, $\aleph_1{}^M$, $\omega \to^M 2$)

lemma *Coll_in_M*[*intro*,*simp*]: $Coll \in M$ ⟨*proof*⟩

lemma *Colleq_refl* : *refl*(*Coll*,*Colleq*)
⟨*proof*⟩

31.1 Collapse forcing is sufficiently closed

lemma *succ_omega_closed_Coll*: *succ*(ω)-*closed*^{M}(*Coll*,*Colleq*)
⟨*proof*⟩

end — *M_ctm2_AC*

locale *collapse_CH* = *G_generic3_AC_CH* $Fn^M(\aleph_1{}^{\#\#M}, \aleph_1{}^M, \omega \to^M 2)$ $Fnle^M(\aleph_1{}^{\#\#M}, \aleph_1{}^M, \omega \to^M 2)$ *0*

sublocale *collapse_CH* ⊆ *forcing_notion Coll Colleq 0*
⟨*proof*⟩

context *collapse_CH*
begin

notation *Leq* (**infixl** ≼ *50*)
notation *Incompatible* (**infixl** ⊥ *50*)

abbreviation
f_G :: *i* (‹f_G›) **where**
$f_G \equiv \bigcup G$

lemma *f_G_in_MG*[*simp*]:
shows $f_G \in M[G]$
⟨*proof*⟩

abbreviation
dom_dense :: *i*⇒*i* **where**
dom_dense(*x*) ≡ { *p*∈*Coll* . *x* ∈ *domain*(*p*) }

lemma *dom_dense_closed*[*intro*,*simp*]: *x*∈*M* ⟹ *dom_dense*(*x*) ∈ *M*
⟨*proof*⟩

lemma *domain_f_G*: **assumes** $x \in \aleph_1{}^M$
shows *x* ∈ *domain*(f_G)
⟨*proof*⟩

lemma *Un_filter_is_function*:
assumes *filter*(*G*)
shows *function*(⋃ *G*)
⟨*proof*⟩

lemma *f_G_funtype*:
shows $f_G : \aleph_1^M \to \omega \to^{M[G]} 2$
⟨*proof*⟩

abbreviation
surj_dense :: $i \Rightarrow i$ **where**
$surj\_dense(x) \equiv \{ p \in Coll . x \in range(p) \}$

lemma *surj_dense_closed*[*intro*,*simp*]:
$x \in \omega \to^M 2 \Longrightarrow surj\_dense(x) \in M$
⟨*proof*⟩

lemma *reals_sub_image_f_G*:
assumes $x \in \omega \to^M 2$
shows $\exists \alpha \in \aleph_1^M. f_G \text{ ` } \alpha = x$
⟨*proof*⟩

lemma *f_G_surj_Aleph_rel1_reals*: $f_G \in surj^{M[G]}(\aleph_1^M, \omega \to^{M[G]} 2)$
⟨*proof*⟩

lemma *continuum_rel_le_Aleph1_extension*:
includes *G_generic1_lemmas*
shows $2^{\uparrow \aleph_0^{M[G]}, M[G]} \leq \aleph_1^{M[G]}$
⟨*proof*⟩

theorem *CH*: $\aleph_1^{M[G]} = 2^{\uparrow \aleph_0^{M[G]}, M[G]}$
⟨*proof*⟩

end — *collapse_CH*

31.2 Models of fragments of $ZFC + CH$

theorem *ctm_of_CH*:
assumes
$M \approx \omega$ *Transset*(M)
$M \models ZC \cup \{\cdot Replacement(p) \cdot . p \in overhead\_CH\}$
$\Phi \subseteq formula$ $M \models \{ \cdot Replacement(ground\_repl\_fm(\varphi)) \cdot . \varphi \in \Phi\}$
shows
$\exists N.$
$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZC \cup \{\cdot CH \cdot\} \cup \{ \cdot Replacement(\varphi) \cdot . \varphi \in \Phi\} \wedge$
$(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$
⟨*proof*⟩

corollary *ctm_ZFC_imp_ctm_CH*:
assumes
$M \approx \omega$ *Transset*(M) $M \models ZFC$
shows
$\exists N.$

$M \subseteq N \wedge N \approx \omega \wedge \mathit{Transset}(N) \wedge N \models \mathit{ZFC} \cup \{\cdot \mathit{CH} \cdot\} \wedge$
$(\forall\, \alpha.\ \mathit{Ord}(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$
⟨*proof*⟩

end

32 From M to $\mathcal{V}$

theory *Absolute_Versions*
imports
CH
ZF.Cardinal_AC
begin

hide_const (**open**) *Order.pred*

32.1 Locales of a class M hold in $\mathcal{V}$

interpretation *V*: *M_trivial* $\mathcal{V}$
⟨*proof*⟩

lemmas *bad_simps* = *V.nonempty V.Forall_in_M_iff V.Inl_in_M_iff V.Inr_in_M_iff V.succ_in_M_iff V.singleton_in_M_iff V.Equal_in_M_iff V.Member_in_M_iff V.Nand_in_M_iff*
V.Cons_in_M_iff V.pair_in_M_iff V.upair_in_M_iff

lemmas *bad_M_trivial_simps*[*simp del*] = *V.Forall_in_M_iff V.Equal_in_M_iff*
V.nonempty

lemmas *bad_M_trivial_rules*[*rule del*] = *V.pair_in_MI V.singleton_in_MI V.pair_in_MD V.nat_into_M*
V.depth_closed V.length_closed V.nat_case_closed V.separation_closed
V.Un_closed V.strong_replacement_closed V.nonempty

interpretation *V*:*M_basic* $\mathcal{V}$
⟨*proof*⟩

interpretation *V*:*M_eclose* $\mathcal{V}$
⟨*proof*⟩

lemmas *bad_M_basic_rules*[*simp del*, *rule del*] =
V.cartprod_closed V.finite_funspace_closed V.converse_closed
V.list_case'_closed V.pred_closed

interpretation *V*:*M_cardinal_arith* $\mathcal{V}$
⟨*proof*⟩

lemmas *bad_M_cardinals_rules*[*simp del*, *rule del*] =
V.iterates_closed V.M_nat V.trancl_closed V.rvimage_closed

interpretation V:*M_cardinal_arith_jump* $\mathcal{V}$
⟨*proof*⟩

lemma *choice_ax_Universe*: *choice_ax*($\mathcal{V}$)
⟨*proof*⟩

interpretation V:*M_master* $\mathcal{V}$
⟨*proof*⟩

named_theorems *V_simps*

— To work systematically, ASCII versions of "_absolute" theorems as those below are preferable.
lemma *eqpoll_rel_absolute*[*V_simps*]: $x \approx^{\mathcal{V}} y \longleftrightarrow x \approx y$
⟨*proof*⟩

lemma *cardinal_rel_absolute*[*V_simps*]: $|x|^{\mathcal{V}} = |x|$
⟨*proof*⟩

lemma *Card_rel_absolute*[*V_simps*]:*Card*$^{\mathcal{V}}(a) \longleftrightarrow$ *Card*(a)
⟨*proof*⟩

lemma *csucc_rel_absolute*[*V_simps*]:$(a^+)^{\mathcal{V}} = a^+$
⟨*proof*⟩

lemma *function_space_rel_absolute*[*V_simps*]:$x \to^{\mathcal{V}} y = x \to y$
⟨*proof*⟩

lemma *cexp_rel_absolute*[*V_simps*]:$x^{\uparrow y,\mathcal{V}} = x^{\uparrow y}$
⟨*proof*⟩

lemma *HAleph_rel_absolute*[*V_simps*]:*HAleph_rel*($\mathcal{V}$,a,b) = *HAleph*(a,b)
⟨*proof*⟩

lemma *Aleph_rel_absolute*[*V_simps*]: *Ord*(x) $\Longrightarrow \aleph_x{}^{\mathcal{V}} = \aleph_x$
⟨*proof*⟩

Example of absolute lemmas obtained from the relative versions. Note the *only* declarations

lemma *Ord_cardinal_idem'*: *Ord*(A) $\Longrightarrow ||A|| = |A|$
⟨*proof*⟩

lemma *Aleph_succ'*: *Ord*(α) $\Longrightarrow \aleph_{succ(\alpha)} = \aleph_\alpha{}^+$
⟨*proof*⟩

These two results are new, first obtained in relative form (not ported).

lemma *csucc_cardinal*:
assumes *Ord*(κ) **shows** $|\kappa|^+ = \kappa^+$

⟨*proof*⟩

lemma *csucc_le_mono*:
assumes $\kappa \leq \nu$ **shows** $\kappa^+ \leq \nu^+$
⟨*proof*⟩

Example of transferring results from a transitive model to $\mathcal{V}$

lemma (**in** *M_Perm*) *eqpoll_rel_transfer_absolute*:
assumes $M(A)$ $M(B)$ $A \approx^M B$
shows $A \approx B$
⟨*proof*⟩

The "relationalized" *CH* with respect to $\mathcal{V}$ corresponds to the real *CH*.

lemma *is_ContHyp_iff_CH*: *is_ContHyp*($\mathcal{V}$) $\longleftrightarrow$ *ContHyp*
⟨*proof*⟩

end

33 Main definitions of the development

theory *Definitions_Main*
imports
Absolute_Versions
begin

This theory gathers the main definitions of the `Transitive_Models` session and the present one.

It might be considered as the bare minimum reading requisite to trust that our development indeed formalizes the theory of forcing. This should be mathematically clear since this is the only known method for obtaining proper extensions of ctms while preserving the ordinals.

The main theorem of this session and all of its relevant definitions appear in Section 33.4. The reader trusting all the libraries on which our development is based, might jump directly to Section 33.3, which treats relative cardinal arithmetic as implemented in `Transitive_Models`. But in case one wants to dive deeper, the following sections treat some basic concepts of the ZF logic (Section 33.1) and in the ZF-Constructible library (Section 33.2) on which our definitions are built.

declare [[*show_question_marks*=*false*]]

33.1 ZF

For the basic logic ZF we restrict ourselves to just a few concepts.

thm *bij_def*[*unfolded inj_def surj_def*]

$bij(A,\ B) \equiv$
$\{f \in A \to B\ .\ \forall\, w{\in}A.\ \forall\, x{\in}A.\ f\ `\ w = f\ `\ x \longrightarrow w = x\} \cap$
$\{f \in A \to B\ .\ \forall\, y{\in}B.\ \exists\, x{\in}A.\ f\ `\ x = y\}$

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*

$i < j \equiv i \in j \wedge Ord(j)$
$i \leq j \longleftrightarrow i < j \vee i = j \wedge Ord(j)$

With the concepts of empty set and successor in place,

lemma *empty_def'*: $\forall\, x.\ x \notin 0$ ⟨*proof*⟩
lemma *succ_def'*: $succ(i) = i \cup \{i\}$ ⟨*proof*⟩

we can define the set of natural numbers ω. In the sources, it is defined as a fixpoint, but here we just write its characterization as the first limit ordinal.

thm *Limit_nat*[*unfolded Limit_def*] *nat_le_Limit*[*unfolded Limit_def*]

$Ord(\omega) \wedge 0 < \omega \wedge (\forall\, y.\ y < \omega \longrightarrow succ(y) < \omega)$
$Ord(i) \wedge 0 < i \wedge (\forall\, y.\ y < i \longrightarrow succ(y) < i) \Longrightarrow \omega \leq i$

Then, addition and predecessor on ω are inductively characterized as follows:

thm *add_0_right add_succ_right pred_0 pred_succ_eq*

$m +_\omega succ(n) = succ(m +_\omega n)$
$m \in \omega \Longrightarrow m +_\omega 0 = m$
$pred(0) = 0$
$pred(succ(y)) = y$

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*

$[] \in list(A)$
$[\![a \in A;\ l \in list(A)]\!] \Longrightarrow Cons(a,\ l) \in list(A)$
$[\![x \in list(A);\ P([]);\ \bigwedge a\ l.\ [\![a \in A;\ l \in list(A);\ P(l)]\!] \Longrightarrow P(Cons(a,\ l))]\!]$
$\Longrightarrow P(x)$

Length, concatenation, and nth element of lists are recursively characterized as follows.

thm *length.simps app.simps nth_0 nth_Cons*

$length([]) = 0$
$length(Cons(a,\ l)) = succ(length(l))$
$[] \ @\ ys = ys$
$Cons(a,\ l) \ @\ ys = Cons(a,\ l \ @\ ys)$
$nth(0,\ Cons(a,\ l)) = a$
$n \in \omega \Longrightarrow nth(succ(n),\ Cons(a,\ l)) = nth(n,\ l)$

We have the usual Haskell-like notation for iterated applications of *Cons*:

lemma *Cons_app*: $[a,b,c] = Cons(a,Cons(b,Cons(c,[])))$ ⟨*proof*⟩

Relative quantifiers restrict the range of the bound variable to a class M of type $i \Rightarrow o$; that is, a truth-valued function with set arguments.

lemma $\forall\, x[M].\ P(x) \equiv \forall\, x.\ M(x) \longrightarrow P(x)$
$\exists\, x[M].\ P(x) \equiv \exists\, x.\ M(x) \wedge P(x)$
⟨*proof*⟩

Finally, a set can be viewed ("cast") as a class using the following function of type $i \Rightarrow i \Rightarrow o$.

thm *setclass_iff*

$(\#\#A)(x) \longleftrightarrow x \in A$

33.2 Relative concepts

A list of relative concepts (mostly from the ZF-Constructible library) follows next.

thm *big_union_def*

$$big\_union(M,\ A,\ z) \equiv \forall\, x[M].\ x \in z \longleftrightarrow (\exists\, y[M].\ y \in A \wedge x \in y)$$

thm *upair_def*

$$upair(M,\ a,\ b,\ z) \equiv a \in z \wedge b \in z \wedge (\forall\, x[M].\ x \in z \longrightarrow x = a \vee x = b)$$

thm *pair_def*

$$pair(M,\ a,\ b,\ z) \equiv \\ \exists\, x[M].\ upair(M,\ a,\ a,\ x) \wedge (\exists\, y[M].\ upair(M,\ a,\ b,\ y) \wedge upair(M,\ x,\ y,\ z))$$

thm *successor_def*[*unfolded is_cons_def union_def*]

$$successor(M,\ a,\ z) \equiv \\ \exists\, x[M].\ upair(M,\ a,\ a,\ x) \wedge (\forall\, xa[M].\ xa \in z \longleftrightarrow xa \in x \vee xa \in a)$$

thm *empty_def*

$$empty(M,\ z) \equiv \forall\, x[M].\ x \notin z$$

thm *transitive_set_def*[*unfolded subset_def*]

$$transitive\_set(M,\ a) \equiv \forall\, x[M].\ x \in a \longrightarrow (\forall\, xa[M].\ xa \in x \longrightarrow xa \in a)$$

thm *ordinal_def*

$$ordinal(M,\ a) \equiv \\ transitive\_set(M,\ a) \wedge (\forall\, x[M].\ x \in a \longrightarrow transitive\_set(M,\ x))$$

thm *image_def*

$$image(M,\ r,\ A,\ z) \equiv \\ \forall\, y[M].\ y \in z \longleftrightarrow (\exists\, w[M].\ w \in r \wedge (\exists\, x[M].\ x \in A \wedge pair(M,\ x,\ y,\ w)))$$

thm *fun_apply_def*

$$is\_apply(M,\ f,\ x,\ y) \equiv$$
$$\exists\, xs[M].$$
$$\quad \exists\, fxs[M].\ upair(M,\ x,\ x,\ xs) \wedge image(M,\ f,\ xs,\ fxs) \wedge big\_union(M,\ fxs,\ y)$$

thm *is_function_def*

$$is\_function(M,\ r) \equiv$$
$$\forall\, x[M].$$
$$\quad \forall\, y[M].$$
$$\quad\quad \forall\, y'[M].$$
$$\quad\quad\quad \forall\, p[M].$$
$$\quad\quad\quad\quad \forall\, p'[M].$$
$$\quad\quad\quad\quad\quad pair(M,\ x,\ y,\ p) \longrightarrow$$
$$\quad\quad\quad\quad\quad pair(M,\ x,\ y',\ p') \longrightarrow p \in r \longrightarrow p' \in r \longrightarrow y = y'$$

thm *is_relation_def*

$$is\_relation(M,\ r) \equiv \forall\, z[M].\ z \in r \longrightarrow (\exists\, x[M].\ \exists\, y[M].\ pair(M,\ x,\ y,\ z))$$

thm *is_domain_def*

$$is\_domain(M,\ r,\ z) \equiv$$
$$\forall\, x[M].\ x \in z \longleftrightarrow (\exists\, w[M].\ w \in r \wedge (\exists\, y[M].\ pair(M,\ x,\ y,\ w)))$$

thm *typed_function_def*

$$typed\_function(M,\ A,\ B,\ r) \equiv$$
$$is\_function(M,\ r) \wedge$$
$$is\_relation(M,\ r) \wedge$$
$$is\_domain(M,\ r,\ A) \wedge$$
$$(\forall\, u[M].\ u \in r \longrightarrow (\forall\, x[M].\ \forall\, y[M].\ pair(M,\ x,\ y,\ u) \longrightarrow y \in B))$$

thm *is_function_space_def*[*unfolded is_funspace_def*]
function_space_rel_def surjection_def

$$is\_function\_space(M,\ A,\ B,\ fs) \equiv$$
$$M(fs) \wedge (\forall\, f[M].\ f \in fs \longleftrightarrow typed\_function(M,\ A,\ B,\ f))$$
$$A \to^M B \equiv THE\ d.\ is\_function\_space(M,\ A,\ B,\ d)$$
$$surjection(M,\ A,\ B,\ f) \equiv$$
$$typed\_function(M,\ A,\ B,\ f) \wedge$$
$$(\forall\, y[M].\ y \in B \longrightarrow (\exists\, x[M].\ x \in A \wedge is\_apply(M,\ f,\ x,\ y)))$$

Relative version of the *ZFC* axioms

thm *extensionality_def*

$$extensionality(M) \equiv \forall\, x[M].\ \forall\, y[M].\ (\forall\, z[M].\ z \in x \longleftrightarrow z \in y) \longrightarrow x = y$$

thm *foundation_ax_def*

$$foundation\_ax(M) \equiv \forall\, x[M].\ (\exists\, y[M].\ y \in x) \longrightarrow (\exists\, y[M].\ y \in x \wedge \neg\ (\exists\, z[M].\ z \in x \wedge z \in y))$$

thm *upair_ax_def*

$$upair\_ax(M) \equiv \forall\, x[M].\ \forall\, y[M].\ \exists\, z[M].\ upair(M,\ x,\ y,\ z)$$

thm *Union_ax_def*

$$Union\_ax(M) \equiv \forall\, x[M].\ \exists\, z[M].\ big\_union(M,\ x,\ z)$$

thm *power_ax_def*[*unfolded powerset_def subset_def*]

$$power\_ax(M) \equiv \forall\, x[M].\ \exists\, z[M].\ \forall\, xa[M].\ xa \in z \longleftrightarrow (\forall\, xb[M].\ xb \in xa \longrightarrow xb \in x)$$

thm *infinity_ax_def*

$$infinity\_ax(M) \equiv \exists\, I[M].\ (\exists\, z[M].\ empty(M,\ z) \wedge z \in I) \wedge (\forall\, y[M].\ y \in I \longrightarrow (\exists\, sy[M].\ successor(M,\ y,\ sy) \wedge sy \in I))$$

thm *choice_ax_def*

$$choice\_ax(M) \equiv \forall\, x[M].\ \exists\, a[M].\ \exists\, f[M].\ ordinal(M,\ a) \wedge surjection(M,\ a,\ x,\ f)$$

thm *separation_def*

$$separation(M,\ P) \equiv \forall\, z[M].\ \exists\, y[M].\ \forall\, x[M].\ x \in y \longleftrightarrow x \in z \wedge P(x)$$

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*

$strong\_replacement(M,\ P) \equiv$
$\forall\, A[M].$
$\quad univalent(M,\ A,\ P) \longrightarrow (\exists\, Y[M].\ \forall\, b[M].\ b \in Y \longleftrightarrow (\exists\, x[M].\ x \in A \wedge P(x,\ b)))$

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*

$[\![x \in \omega;\ y \in \omega]\!] \Longrightarrow \cdot x \in y\cdot \in formula$
$[\![x \in \omega;\ y \in \omega]\!] \Longrightarrow \cdot x = y\cdot \in formula$
$[\![p \in formula;\ q \in formula]\!] \Longrightarrow \cdot\neg(p \wedge q)\cdot \in formula$
$p \in formula \Longrightarrow (\cdot\forall\, p\cdot) \in formula$
$[\![x \in formula;\ \bigwedge x\ y.\ [\![x \in \omega;\ y \in \omega]\!] \Longrightarrow P(\cdot x \in y\cdot);$
$\bigwedge x\ y.\ [\![x \in \omega;\ y \in \omega]\!] \Longrightarrow P(\cdot x = y\cdot);$
$\bigwedge p\ q.\ [\![p \in formula;\ P(p);\ q \in formula;\ P(q)]\!] \Longrightarrow P(\cdot\neg(p \wedge q)\cdot);$
$\bigwedge p.\ [\![p \in formula;\ P(p)]\!] \Longrightarrow P((\cdot\forall\, p\cdot))]\!]$
$\Longrightarrow P(x)$

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*

$[\![nth(i,\ env) = x;\ nth(j,\ env) = y;\ env \in list(A)]\!]$
$\Longrightarrow x \in y \longleftrightarrow A,\ env \models \cdot i \in j\cdot$
$[\![nth(i,\ env) = x;\ nth(j,\ env) = y;\ env \in list(A)]\!]$
$\Longrightarrow x = y \longleftrightarrow A,\ env \models \cdot i = j\cdot$
$env \in list(A) \Longrightarrow (A,\ env \models \cdot\neg(p \wedge q)\cdot) \longleftrightarrow \neg\ ((A,\ env \models p) \wedge (A,\ env \models q))$
$env \in list(A) \Longrightarrow (A,\ env \models (\cdot\forall\, p\cdot)) \longleftrightarrow (\forall\, x{\in}A.\ A,\ Cons(x,\ env) \models p)$

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

$Union\_ax(\#\#A) \longleftrightarrow A,\ [] \models \cdot Union\ Ax\cdot$
$power\_ax(\#\#A) \longleftrightarrow A,\ [] \models \cdot Powerset\ Ax\cdot$
$upair\_ax(\#\#A) \longleftrightarrow A,\ [] \models \cdot Pairing\cdot$
$foundation\_ax(\#\#A) \longleftrightarrow A,\ [] \models \cdot Foundation\cdot$
$extensionality(\#\#A) \longleftrightarrow A,\ [] \models \cdot Extensionality\cdot$
$infinity\_ax(\#\#A) \longleftrightarrow A,\ [] \models \cdot Infinity\cdot$
$\varphi \in formula \Longrightarrow$
$(M,\ [] \models \cdot Separation(\varphi)\cdot) \longleftrightarrow$
$(\forall\, env{\in}list(M).$
$\quad arity(\varphi) \le 1 +_\omega length(env) \longrightarrow separation(\#\#M,\ \lambda x.\ M,\ [x]\ @\ env \models \varphi))$
$\varphi \in formula \Longrightarrow$
$(M,\ [] \models \cdot Replacement(\varphi)\cdot) \longleftrightarrow (\forall\, env.\ replacement\_assm(M,\ env,\ \varphi))$
$choice\_ax(\#\#A) \longleftrightarrow A,\ [] \models \cdot AC\cdot$

Above, we use the following:

thm *replacement_assm_def*

$replacement\_assm(M,\ env,\ \varphi) \equiv$
$\varphi \in formula \longrightarrow$
$env \in list(M) \longrightarrow$
$arity(\varphi) \le 2 +_\omega length(env) \longrightarrow$
$strong\_replacement(\#\#M,\ \lambda x\ y.\ M,\ [x,\ y]\ @\ env \models \varphi)$

Finally, the axiom sets are defined as follows.

thm *ZF_fin_def ZF_schemes_def Zermelo_fms_def ZC_def ZF_def ZFC_def*

$ZF\_fin \equiv$
$\{\cdot Extensionality\cdot, \cdot Foundation\cdot, \cdot Pairing\cdot, \cdot Union\ Ax\cdot, \cdot Infinity\cdot,$
$\cdot Powerset\ Ax\cdot\}$
$ZF\_schemes \equiv$
$\{\cdot Separation(p)\cdot\ .\ p \in formula\} \cup \{\cdot Replacement(p)\cdot\ .\ p \in formula\}$
$\cdot Z\cdot \equiv ZF\_fin \cup \{\cdot Separation(p)\cdot\ .\ p \in formula\}$
$ZC \equiv \cdot Z\cdot \cup \{\cdot AC\cdot\}$
$ZF \equiv ZF\_schemes \cup ZF\_fin$
$ZFC \equiv ZF \cup \{\cdot AC\cdot\}$

33.3 Relativization of infinitary arithmetic

In order to state the defining property of the relative equipotence relation, we work under the assumptions of the locale *M_cardinals*. They comprise a finite set of instances of Separation and Replacement to prove closure properties of the transitive class M.

lemma (**in** *M_cardinals*) *eqpoll_def'*:
assumes $M(A)$ $M(B)$ **shows** $A \approx^M B \longleftrightarrow (\exists f[M].\ f \in bij(A,B))$
⟨*proof*⟩

Below, μ denotes the minimum operator on the ordinals.

lemma *cardinalities_defs*:
fixes $M{::}i \Rightarrow o$
shows
$|A|^M \equiv \mu\ i.\ M(i) \wedge i \approx^M A$
$Card^M(\alpha) \equiv \alpha = |\alpha|^M$
$\kappa^{\uparrow\nu,M} \equiv |\nu \rightarrow^M \kappa|^M$
$(\kappa^+)^M \equiv \mu\ x.\ M(x) \wedge Card^M(x) \wedge \kappa < x$
⟨*proof*⟩

context *M_aleph*
begin

Analogous to the previous Lemma *eqpoll_def'*, we are now under the assumptions of the locale *M_aleph*. The axiom instances included are sufficient to state and prove the defining properties of the relativized *Aleph* function (in particular, the required ability to perform transfinite recursions).

thm *Aleph_rel_zero Aleph_rel_succ Aleph_rel_limit*

$\aleph_0{}^M = \omega$
$[\![Ord(\alpha);\ M(\alpha)]\!] \Longrightarrow \aleph_{succ(\alpha)}{}^M = (\aleph_\alpha{}^{M+})^M$
$[\![Limit(\alpha);\ M(\alpha)]\!] \Longrightarrow \aleph_\alpha{}^M = (\bigcup j{\in}\alpha.\ \aleph_j{}^M)$

end — *M_aleph*

lemma *ContHyp_rel_def'*:
fixes $N{::}i{\Rightarrow}o$
shows
$CH^N \equiv \aleph_1{}^N = 2^{\uparrow\aleph_0{}^N,N}$
⟨*proof*⟩

Under appropriate hypotheses (this time, from the locale *M_ZF_library*), CH^M is equivalent to its fully relational version *is_ContHyp*. As a sanity check, we see that if the transitive class is indeed $\mathcal{V}$, we recover the original CH.

thm *M_ZF_library.is_ContHyp_iff is_ContHyp_iff_CH*[*unfolded ContHyp_def*]

$M\_ZF\_library(M) \Longrightarrow is\_ContHyp(M) \longleftrightarrow CH^M$
$is\_ContHyp(\mathcal{V}) \longleftrightarrow \aleph_1 = 2^{\uparrow\aleph_0}$

In turn, the fully relational version evaluated on a nonempty transitive A is equivalent to the satisfaction of the first-order formula $\cdot CH\cdot$.

thm *is_ContHyp_iff_sats*

$[\![env \in list(A);\ 0 \in A]\!] \Longrightarrow is\_ContHyp(\#\#A) \longleftrightarrow A,\ env \models \cdot CH\cdot$

33.4 Forcing

Our first milestone was to obtain a proper extension using forcing. Its original proof didn't required the previous developments involving the relativization of material on cardinal arithmetic. Now it is derived from a stronger result, namely *extensions_of_ctms* below.

thm *extensions_of_ctms_ZF*

$[\![M \approx \omega;\ Transset(M);\ M \models ZF]\!]$
$\Longrightarrow \exists N.\ M \subseteq N \wedge$
$N \approx \omega \wedge$
$Transset(N) \wedge$
$N \models ZF \wedge$
$M \neq N \wedge (\forall \alpha.\ Ord(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N) \wedge ((M, [] \models \cdot AC\cdot) \longrightarrow N \models ZFC)$

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*

$[\![M \approx \omega;\ \mathit{Transset}(M);\ M \models \mathit{ZFC}]\!]$
$\Longrightarrow \exists N.\ M \subseteq N \wedge$
$\quad N \approx \omega \wedge$
$\quad \mathit{Transset}(N) \wedge N \models \mathit{ZFC} \cup \{\cdot\neg\cdot \mathit{CH}\cdot\cdot\} \wedge (\forall \alpha.\ \mathit{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N)$

thm *ctm_ZFC_imp_ctm_CH*

$[\![M \approx \omega;\ \mathit{Transset}(M);\ M \models \mathit{ZFC}]\!]$
$\Longrightarrow \exists N.\ M \subseteq N \wedge$
$\quad N \approx \omega \wedge$
$\quad \mathit{Transset}(N) \wedge N \models \mathit{ZFC} \cup \{\cdot \mathit{CH}\cdot\} \wedge (\forall \alpha.\ \mathit{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N)$

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*

$\mathit{instances1\_fms} \equiv$
$\{\mathit{eclose\_closed\_fm},\ \mathit{eclose\_abs\_fm},\ \mathit{wfrec\_rank\_fm},\ \mathit{transrec\_VFrom\_fm}\}$

thm *overhead_def overhead_notCH_def*

$\mathit{overhead} \equiv \mathit{instances1\_fms} \cup \mathit{instances\_ground\_fms}$
$\mathit{overhead\_notCH} \equiv$
$\mathit{overhead} \cup \mathit{instances2\_fms} \cup \mathit{instances3\_fms} \cup \mathit{instances\_ground\_notCH\_fms}$
$\mathit{overhead\_CH} \equiv \mathit{overhead\_notCH} \cup \{\mathit{dc\_abs\_fm}\}$

One further instance is needed to force CH, with a total count of 32 instances:

thm *overhead_CH_def*

$\mathit{overhead\_CH} \equiv \mathit{overhead\_notCH} \cup \{\mathit{dc\_abs\_fm}\}$

thm *extensions_of_ctms*

$[\![M \approx \omega;\ \mathit{Transset}(M);\ M \models \cdot Z\cdot \cup \{\cdot \mathit{Replacement}(p)\cdot \ .\ p \in \mathit{overhead}\};$
$\Phi \subseteq \mathit{formula};\ M \models \{\cdot \mathit{Replacement}(\mathit{ground\_repl\_fm}(\varphi))\cdot \ .\ \varphi \in \Phi\}]\!]$
$\Longrightarrow \exists N.\ M \subseteq N \wedge$
$N \approx \omega \wedge$
$\mathit{Transset}(N) \wedge$
$M \neq N \wedge$
$(\forall \alpha.\ \mathit{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N) \wedge$
$((M,\ [] \models \cdot AC\cdot) \longrightarrow N,\ [] \models \cdot AC\cdot) \wedge$
$N \models \cdot Z\cdot \cup \{\cdot \mathit{Replacement}(\varphi)\cdot \ .\ \varphi \in \Phi\}$

thm *ctm_of_not_CH*

$[\![M \approx \omega;\ \mathit{Transset}(M);\ M \models ZC \cup \{\cdot \mathit{Replacement}(p)\cdot \ .\ p \in \mathit{overhead\_notCH}\};$
$\Phi \subseteq \mathit{formula};\ M \models \{\cdot \mathit{Replacement}(\mathit{ground\_repl\_fm}(\varphi))\cdot \ .\ \varphi \in \Phi\}]\!]$
$\Longrightarrow \exists N.\ M \subseteq N \wedge$
$N \approx \omega \wedge$
$\mathit{Transset}(N) \wedge$
$N \models ZC \cup \{\cdot \neg \cdot CH \cdot\cdot\} \cup \{\cdot \mathit{Replacement}(\varphi)\cdot \ .\ \varphi \in \Phi\} \wedge$
$(\forall \alpha.\ \mathit{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N)$

thm *ctm_of_CH*

$[\![M \approx \omega;\ \mathit{Transset}(M);\ M \models ZC \cup \{\cdot \mathit{Replacement}(p)\cdot \ .\ p \in \mathit{overhead\_CH}\};$
$\Phi \subseteq \mathit{formula};\ M \models \{\cdot \mathit{Replacement}(\mathit{ground\_repl\_fm}(\varphi))\cdot \ .\ \varphi \in \Phi\}]\!]$
$\Longrightarrow \exists N.\ M \subseteq N \wedge$
$N \approx \omega \wedge$
$\mathit{Transset}(N) \wedge$
$N \models ZC \cup \{\cdot CH\cdot\} \cup \{\cdot \mathit{Replacement}(\varphi)\cdot \ .\ \varphi \in \Phi\} \wedge$
$(\forall \alpha.\ \mathit{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N)$

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*

$[\![\varphi \in \mathit{formula};\ M,\ [] \models \cdot \mathit{Replacement}(\mathit{ground\_repl\_fm}(\varphi))\cdot]\!]$
$\Longrightarrow M[G],\ [] \models \cdot \mathit{Replacement}(\varphi)\cdot$

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 t_1 t_2
  assumes
    ts_in_nat[*simp*]: $t_1 \in \omega$ $t_2 \in \omega$
    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:
    $\bigwedge P$. *strong_replacement*(*M*,*P*)
    $\bigwedge F$. *lam_replacement*(*M*,*F*)
    $\bigwedge Q\ S$. *iterates_replacement*(*M*,*Q*,*S*)
    $\bigwedge Q\ S$. *wfrec_replacement*(*M*,*Q*,*S*)
    $\bigwedge Q\ S$. *transrec_replacement*(*M*,*Q*,*S*)
    and
    sorried_separations: $\bigwedge Q$. *separation*(*M*,*Q*)
  shows
    M_master(*M*)
  ⟨*proof*⟩
no_notation *mem* (**infixl** ‹∈› *50*)
no_notation *conj* (**infixr** ‹∧› *35*)
no_notation *disj* (**infixr** ‹∨› *30*)
no_notation *iff* (**infixr** ‹⟷› *25*)
no_notation *imp* (**infixr** ‹⟶› *25*)
no_notation *not* (‹¬ _› [*40*] *40*)

no_notation *All* (‹'(∀_')›)
no_notation *Ex* (‹'(∃_')›)

no_notation *Member* (‹·_ ∈/ _·›)
no_notation *Equal* (‹·_ =/ _·›)
no_notation *Nand* (‹·¬'(_ ∧/ _')·›)
no_notation *And* (‹·_ ∧/ _·›)
no_notation *Or* (‹·_ ∨/ _·›)
no_notation *Iff* (‹·_ ↔/ _·›)
no_notation *Implies* (‹·_ →/ _·›)
no_notation *Neg* (‹·¬_·›)
no_notation *Forall* (‹'(·∀(/_)·')›)
no_notation *Exists* (‹'(·∃(/_)·')›)

notation *Member* (**infixl** ‹∈› *50*)
notation *Equal* (**infixl** ‹≡› *50*)
notation *Nand* (‹¬'(_ ∧/ _')›)
notation *And* (**infixr** ‹∧› *35*)
notation *Or* (**infixr** ‹∨› *30*)
notation *Iff* (**infixr** ‹⟷› *25*)
notation *Implies* (**infixr** ‹⟶› *25*)
notation *Neg* (‹¬ _› [*40*] *40*)
notation *Forall* (‹'(∀_')›)
notation *Exists* (‹'(∃_')›)

lemma *forces*(t_1∈t_2) = (*0* ∈ *1* ∧ *forces_mem_fm*(*1*, *2*, *0*, t_1+$_\omega$*4*, t_2+$_\omega$*4*))
⟨*proof*⟩

definition *forces_0_mem_1* **where** *forces_0_mem_1*≡*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*,*ω*) ⟹ *mem*(*φ*,*formula*) ⟹ *incr_bv*(*Neg*(*φ*))‘*n* = *?x*
⟨*proof*⟩

schematic_goal *incr_bv_Exists* [*incr_bv_new_simps*]:

$mem(n,\omega) \Longrightarrow mem(\varphi,formula) \Longrightarrow incr\_bv(Exists(\varphi))\,`n = ?x$
$\langle proof \rangle$

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).