Transitive Models of Fragments of ZF

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

March 4, 2022

Abstract

We extend the ZF-Constructibility library by relativizing theories of the Isabelle/ZF and Delta System Lemma sessions to a transitive class. We also relativize Paulson's work on Aleph and our former treatment of the Axiom of Dependent Choices. This work is a prerequisite to our formalization of the independence of the Continuum Hypothesis.

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

Relativization of concepts is a key tool to obtain results in forcing, as it is explained in [1, Sect. 3] and elsewhere.

In this session, we cast some theories in relative form, in a way that they now refer to a fixed class M as the universe of discourse. Whenever it was possible, we tried to minimize the changes to the structure and proof scripts of the absolute concepts. For this reason, some comments of the original text as well as outdated **apply** commands appear profusely in the following theories.

A repeated pattern that appears is that the relativized result can be proved *mutatis mutandis*, with remaining proof obligations that the objects constructed actually belong to the model M. Another aspect was that the management of higher order constructs always posed some extra problems, already noted by Paulson [2, Sect. 7.3].

In the theory `Lambda_Replacement` we introduce a new locale assuming two instances of separation and twelve instances of "lambda replacements" (i.e., replacements using definable functions of the form $\lambda xy.y = \langle x, f(x)\rangle$) that allow for having some form of compositionality of further instances of separations and replacements.

We proceed to enumerate the theories that were "ported" to relative form, briefly commenting on each of them. Below, we refer to the original theories as the *source* and, correspondingly, call *target* the relativized version. We omit the `.thy` suffixes.

1. From *ZF*:
   (a) `Univ`. Here we decided to relativize only the term `Vfrom` that constructs the cumulative hierarchy up to some ordinal length and starting from an arbitrary set.
   (b) `Cardinal`. There are two targets for this source, `Least` and `Cardinal_Relative`. Both require some fair amount of preparation, trying to take advantage of absolute concepts. It is not straightforward to compare source and targets in a line-by-line fashion at this point.
   (c) `CardinalArith`. The hardest part was to formalize the cardinal successor function. We also disregarded the part treating finite cardinals since it is an absolute concept. Apart from that, the relative version closely parallels the source.
   (d) `Cardinal_AC`. After some boilerplate, porting was rather straightforward, excepting cardinal arithmetic involving the higher-order union operator.

2. From *ZF-Constructible*:

   (a) `Normal`. The target here is `Aleph_Relative` since that is the only concept that we ported. Instead of porting all the machinery of normal functions (since it involved higher-order variables), we particularized the results for the Aleph function. We also used an alternative definition of the latter that worked better with our relativization discipline.

3. From *Delta_System_Lemma*:

   (a) `ZF_Library`. The target includes a big section of auxiliary lemmas and commands that aid the relativization. We needed to make explicit the witnesses (mainly functions) in some of the existential results proved in the source, since only in that way we would be able to show that they belonged to the model.

   (b) `Cardinal_Library`. Porting was relatively straightforward; most of the extra work laid in adjusting locale assumptions to obtain an appropriate context to state and prove the theorems.

   (c) `Delta_System`. Same comments as in the case of `Cardinal_Library` apply here.

4. From *Forcing*:

   (a) `Pointed_DC`. This case was similar to `Cardinal_AC` above, although a bit of care was needed to handle the recursive construction. Also, a fraction of the theory `AC` from *ZF* was ported here as it was a prerequisite. A complete relativization of `AC` would be desirable but still missing.

2 Auxiliary results on arithmetic

theory *Nat_Miscellanea*
  imports
    Delta_System_Lemma.ZF_Library
begin

notation *add* (**infixl** ‹+$_\omega$› *65*)
notation *diff* (**infixl** ‹-$_\omega$› *65*)

Most of these results will get used at some point for the calculation of arities.

lemmas *nat_succI* = *Ord_succ_mem_iff* [*THEN iffD2*,*OF nat_into_Ord*]

lemma *nat_succD* : $m \in nat \Longrightarrow succ(n) \in succ(m) \Longrightarrow n \in m$
  ⟨*proof*⟩

lemmas *zero_in_succ* = *ltD* [*OF nat_0_le*]

lemma *in_n_in_nat* : $m \in nat \Longrightarrow n \in m \Longrightarrow n \in nat$
⟨*proof*⟩

lemma *ltI_neg* : $x \in nat \Longrightarrow j \leq x \Longrightarrow j \neq x \Longrightarrow j < x$
⟨*proof*⟩

lemma *succ_pred_eq* : $m \in nat \Longrightarrow m \neq 0 \Longrightarrow succ(pred(m)) = m$
⟨*proof*⟩

lemma *succ_ltI* : $succ(j) < n \Longrightarrow j < n$
⟨*proof*⟩

lemmas *succ_leD* = *succ_leE*[*OF leI*]

lemma *succpred_leI* : $n \in nat \Longrightarrow n \leq succ(pred(n))$
⟨*proof*⟩

lemma *succpred_n0* : $succ(n) \in p \Longrightarrow p \neq 0$
⟨*proof*⟩

lemmas *natEin* = *natE* [*OF lt_nat_in_nat*]

lemmas *Un_least_lt_iffn* = *Un_least_lt_iff* [*OF nat_into_Ord nat_into_Ord*]

lemma *pred_type* : $m \in nat \Longrightarrow n \leq m \Longrightarrow n \in nat$
⟨*proof*⟩

lemma *pred_le* : $m \in nat \Longrightarrow n \leq succ(m) \Longrightarrow pred(n) \leq m$
⟨*proof*⟩

lemma *pred_le2* : $n \in nat \Longrightarrow m \in nat \Longrightarrow pred(n) \leq m \Longrightarrow n \leq succ(m)$
⟨*proof*⟩

lemma *Un_leD1* : $Ord(i) \Longrightarrow Ord(j) \Longrightarrow Ord(k) \Longrightarrow i \cup j \leq k \Longrightarrow i \leq k$
⟨*proof*⟩

lemma *Un_leD2* : $Ord(i) \Longrightarrow Ord(j) \Longrightarrow Ord(k) \Longrightarrow i \cup j \leq k \Longrightarrow j \leq k$
⟨*proof*⟩

lemma *gt1* : $n \in nat \Longrightarrow i \in n \Longrightarrow i \neq 0 \Longrightarrow i \neq 1 \Longrightarrow 1 < i$
⟨*proof*⟩

lemma *pred_mono* : $m \in nat \Longrightarrow n \leq m \Longrightarrow pred(n) \leq pred(m)$
⟨*proof*⟩

lemma *succ_mono* : $m \in nat \Longrightarrow n \leq m \Longrightarrow succ(n) \leq succ(m)$

⟨*proof*⟩

lemma *union_abs1* :
⟦ $i \leq j$ ⟧ $\Longrightarrow$ $i \cup j = j$
⟨*proof*⟩

lemma *union_abs2* :
⟦ $i \leq j$ ⟧ $\Longrightarrow$ $j \cup i = j$
⟨*proof*⟩

lemma *ord_un_max* : $Ord(i) \Longrightarrow Ord(j) \Longrightarrow i \cup j = max(i,j)$
⟨*proof*⟩

lemma *ord_max_ty* : $Ord(i) \Longrightarrow Ord(j) \Longrightarrow Ord(max(i,j))$
⟨*proof*⟩

lemmas *ord_simp_union* = *ord_un_max ord_max_ty max_def*

lemma *le_succ* : $x \in nat \Longrightarrow x \leq succ(x)$ ⟨*proof*⟩

lemma *le_pred* : $x \in nat \Longrightarrow pred(x) \leq x$
⟨*proof*⟩

lemma *not_le_anti_sym* : $x \in nat \Longrightarrow y \in nat \Longrightarrow \neg\, x \leq y \Longrightarrow \neg y \leq x \Longrightarrow y = x$
⟨*proof*⟩

lemma *Un_le_compat* : $o \leq p \Longrightarrow q \leq r \Longrightarrow Ord(o) \Longrightarrow Ord(p) \Longrightarrow Ord(q) \Longrightarrow Ord(r) \Longrightarrow o \cup q \leq p \cup r$
⟨*proof*⟩

lemma *Un_le* : $p \leq r \Longrightarrow q \leq r \Longrightarrow$
$Ord(p) \Longrightarrow Ord(q) \Longrightarrow Ord(r) \Longrightarrow$
$p \cup q \leq r$
⟨*proof*⟩

lemma *Un_leI3* : $o \leq r \Longrightarrow p \leq r \Longrightarrow q \leq r \Longrightarrow$
$Ord(o) \Longrightarrow Ord(p) \Longrightarrow Ord(q) \Longrightarrow Ord(r) \Longrightarrow$
$o \cup p \cup q \leq r$
⟨*proof*⟩

lemma *diff_mono* :
assumes $m \in nat$ $n \in nat$ $p \in nat$ $m < n$ $p \leq m$
shows $m\#\text{-}p < n\#\text{-}p$
⟨*proof*⟩

lemma *pred_Un*:
$x \in nat \Longrightarrow y \in nat \Longrightarrow pred(succ(x) \cup y) = x \cup pred(y)$
$x \in nat \Longrightarrow y \in nat \Longrightarrow pred(x \cup succ(y)) = pred(x) \cup y$
⟨*proof*⟩

lemma *le_natI* : $j \leq n \Longrightarrow n \in nat \Longrightarrow j \in nat$
⟨*proof*⟩

lemma *le_natE* : $n \in nat \Longrightarrow j < n \Longrightarrow j \in n$
⟨*proof*⟩

lemma *leD* : **assumes** $n \in nat$ $j \leq n$
shows $j < n \mid j = n$
⟨*proof*⟩

lemma *pred_nat_eq* :
assumes $n \in nat$
shows $pred(n) = \bigcup n$
⟨*proof*⟩

2.1 Some results in ordinal arithmetic

The following results are auxiliary to the proof of wellfoundedness of the relation *frecR*

lemma *max_cong* :
assumes $x \leq y$ $Ord(y)$ $Ord(z)$
shows $max(x,y) \leq max(y,z)$
⟨*proof*⟩

lemma *max_commutes* :
assumes $Ord(x)$ $Ord(y)$
shows $max(x,y) = max(y,x)$
⟨*proof*⟩

lemma *max_cong2* :
assumes $x \leq y$ $Ord(y)$ $Ord(z)$ $Ord(x)$
shows $max(x,z) \leq max(y,z)$
⟨*proof*⟩

lemma *max_D1* :
assumes $x = y$ $w < z$ $Ord(x)$ $Ord(w)$ $Ord(z)$ $max(x,w) = max(y,z)$
shows $z \leq y$
⟨*proof*⟩

lemma *max_D2* :
assumes $w = y \vee w = z$ $x < y$ $Ord(x)$ $Ord(w)$ $Ord(y)$ $Ord(z)$ $max(x,w) = max(y,z)$
shows $x < w$
⟨*proof*⟩

lemma *oadd_lt_mono2* :
assumes $Ord(n)$ $Ord(\alpha)$ $Ord(\beta)$ $\alpha < \beta$ $x < n$ $y < n$ $0 < n$
shows $n ** \alpha + x < n ** \beta + y$

⟨*proof*⟩
end

3 Various results missing from ZF.

theory *ZF_Miscellanea*
imports
ZF
Nat_Miscellanea
begin

lemma *function_subset*:
function(*f*) ⟹ *g*⊆*f* ⟹ *function*(*g*)
⟨*proof*⟩

lemma *converse_refl* : *refl*(*A*,*r*) ⟹ *refl*(*A*,*converse*(*r*))
⟨*proof*⟩

lemma *Ord_lt_subset* : *Ord*(*b*) ⟹ *a*<*b* ⟹ *a*⊆*b*
⟨*proof*⟩

lemma *funcI* : *f* ∈ *A* → *B* ⟹ *a* ∈ *A* ⟹ *b*= *f* ‘ *a* ⟹ ⟨*a*, *b*⟩ ∈ *f*
⟨*proof*⟩

lemma *vimage_fun_sing*:
assumes *f*∈*A*→*B* *b*∈*B*
shows {*a*∈*A* . *f*‘*a*=*b*} = *f*-‘‘{*b*}
⟨*proof*⟩

lemma *image_fun_subset*: *S*∈*A*→*B* ⟹ *C*⊆*A*⟹ {*S* ‘ *x* . *x*∈ *C*} = *S*‘‘*C*
⟨*proof*⟩

lemma *subset_Diff_Un*: *X* ⊆ *A* ⟹ *A* = (*A* - *X*) ∪ *X* ⟨*proof*⟩

lemma *Diff_bij*:
assumes ∀ *A*∈*F*. *X* ⊆ *A* **shows** (λ*A*∈*F*. *A*-*X*) ∈ *bij*(*F*, {*A*-*X*. *A*∈*F*})
⟨*proof*⟩

lemma *function_space_nonempty*:
assumes *b*∈*B*
shows (λ*x*∈*A*. *b*) : *A* → *B*
⟨*proof*⟩

lemma *vimage_lam*: (λ*x*∈*A*. *f*(*x*)) -‘‘ *B* = { *x*∈*A* . *f*(*x*) ∈ *B* }
⟨*proof*⟩

lemma *range_fun_subset_codomain*:
assumes *h*:*B* → *C*
shows *range*(*h*) ⊆ *C*

⟨*proof*⟩

lemma *Pi_rangeD*:
assumes $f \in Pi(A,B)$ $b \in range(f)$
shows $\exists\, a \in A.\ f`a = b$
⟨*proof*⟩

lemma *Pi_range_eq*: $f \in Pi(A,B) \Longrightarrow range(f) = \{f\ `\ x\ .\ x \in A\}$
⟨*proof*⟩

lemma *Pi_vimage_subset* : $f \in Pi(A,B) \Longrightarrow f\text{-}``C \subseteq A$
⟨*proof*⟩

definition
$minimum :: i \Rightarrow i \Rightarrow i$ **where**
$minimum(r,B) \equiv THE\ b.\ first(b,B,r)$

lemma *minimum_in*: $[\![\ well\_ord(A,r);\ B \subseteq A;\ B \neq 0\ ]\!] \Longrightarrow minimum(r,B) \in B$
⟨*proof*⟩

lemma *well_ord_surj_imp_inj_inverse*:
assumes $well\_ord(A,r)$ $h \in surj(A,B)$
shows $(\lambda b \in B.\ minimum(r,\ \{a \in A.\ h`a = b\})) \in inj(B,A)$
⟨*proof*⟩

lemma *well_ord_surj_imp_lepoll*:
assumes $well\_ord(A,r)$ $h \in surj(A,B)$
shows $B \lesssim A$
⟨*proof*⟩
lemma *surj_imp_well_ord*:
assumes $well\_ord(A,r)$ $h \in surj(A,B)$
shows $\exists\, s.\ well\_ord(B,s)$
⟨*proof*⟩

lemma *Pow_sing* : $Pow(\{a\}) = \{0,\{a\}\}$
⟨*proof*⟩

lemma *Pow_cons*:
shows $Pow(cons(a,A)) = Pow(A) \cup \{\{a\} \cup X\ .\ X{:}\ Pow(A)\}$
⟨*proof*⟩

lemma *app_nm* :
assumes $n \in nat$ $m \in nat$ $f \in n \rightarrow m$ $x \in nat$
shows $f`x \in nat$
⟨*proof*⟩

lemma *Upair_eq_cons*: $Upair(a,b) = \{a,b\}$
⟨*proof*⟩

lemma *converse_apply_eq* : *converse*(*f*) ‘ *x* = ⋃(*f* -‘‘ {*x*})
⟨*proof*⟩

lemmas *app_fun* = *apply_iff*[*THEN iffD1*]

lemma *Finite_imp_lesspoll_nat*:
assumes *Finite*(*A*)
shows *A* ≺ *nat*
⟨*proof*⟩

end

4 Renaming of variables in internalized formulas

theory *Renaming*
imports
ZF_Miscellanea
ZF-Constructible.Formula
begin

4.1 Renaming of free variables

definition
union_fun :: [*i*,*i*,*i*,*i*] ⇒ *i* **where**
union_fun(*f*,*g*,*m*,*p*) ≡ λ*j* ∈ *m* ∪ *p* . *if j*∈*m then f*‘*j else g*‘*j*

lemma *union_fun_type*:
assumes *f* ∈ *m* → *n*
g ∈ *p* → *q*
shows *union_fun*(*f*,*g*,*m*,*p*) ∈ *m* ∪ *p* → *n* ∪ *q*
⟨*proof*⟩

lemma *union_fun_action* :
assumes
env ∈ *list*(*M*)
env′ ∈ *list*(*M*)
length(*env*) = *m* ∪ *p*
∀ *i* . *i* ∈ *m* ⟶ *nth*(*f*‘*i*,*env*′) = *nth*(*i*,*env*)
∀ *j* . *j* ∈ *p* ⟶ *nth*(*g*‘*j*,*env*′) = *nth*(*j*,*env*)
shows ∀ *i* . *i* ∈ *m* ∪ *p* ⟶
nth(*i*,*env*) = *nth*(*union_fun*(*f*,*g*,*m*,*p*)‘*i*,*env*′)
⟨*proof*⟩

lemma *id_fn_type* :
assumes *n* ∈ *nat*
shows *id*(*n*) ∈ *n* → *n*
⟨*proof*⟩

lemma *id_fn_action*:
assumes $n \in nat$ $env \in list(M)$
shows $\bigwedge j\ .\ j < n \Longrightarrow nth(j,env) = nth(id(n)`j,env)$
⟨*proof*⟩

definition
$rsum :: [i,i,i,i,i] \Rightarrow i$ **where**
$rsum(f,g,m,n,p) \equiv \lambda j \in m+_\omega p\ \ .\ if\ j<m\ then\ f`j\ else\ (g`(j\#-m))+_\omega n$

lemma *sum_inl*:
assumes $m \in nat$ $n \in nat$
$f \in m \rightarrow n$ $x \in m$
shows $rsum(f,g,m,n,p)`x = f`x$
⟨*proof*⟩

lemma *sum_inr*:
assumes $m \in nat$ $n \in nat$ $p \in nat$
$g \in p \rightarrow q$ $m \leq x$ $x < m+_\omega p$
shows $rsum(f,g,m,n,p)`x = g`(x\#-m)+_\omega n$
⟨*proof*⟩

lemma *sum_action* :
assumes $m \in nat$ $n \in nat$ $p \in nat$ $q \in nat$
$f \in m \rightarrow n$ $g \in p \rightarrow q$
$env \in list(M)$
$env' \in list(M)$
$env1 \in list(M)$
$env2 \in list(M)$
$length(env) = m$
$length(env1) = p$
$length(env') = n$
$\bigwedge i\ .\ i < m \Longrightarrow nth(i,env) = nth(f`i,env')$
$\bigwedge j.\ j < p \Longrightarrow nth(j,env1) = nth(g`j,env2)$
shows $\forall\ i\ .\ i < m+_\omega p \longrightarrow$
$nth(i,env@env1) = nth(rsum(f,g,m,n,p)`i,env'@env2)$
⟨*proof*⟩

lemma *sum_type* :
assumes $m \in nat$ $n \in nat$ $p \in nat$ $q \in nat$
$f \in m \rightarrow n$ $g \in p \rightarrow q$
shows $rsum(f,g,m,n,p) \in (m+_\omega p) \rightarrow (n+_\omega q)$
⟨*proof*⟩

lemma *sum_type_id* :
assumes
$f \in length(env) \rightarrow length(env')$
$env \in list(M)$

$env' \in list(M)$
$env1 \in list(M)$
shows
$rsum(f,id(length(env1)),length(env),length(env'),length(env1)) \in$
$(length(env)+_\omega length(env1)) \to (length(env')+_\omega length(env1))$
⟨*proof*⟩

lemma *sum_type_id_aux2* :
assumes
$f \in m \to n$
$m \in nat\ n \in nat$
$env1 \in list(M)$
shows
$rsum(f,id(length(env1)),m,n,length(env1)) \in$
$(m+_\omega length(env1)) \to (n+_\omega length(env1))$
⟨*proof*⟩

lemma *sum_action_id* :
assumes
$env \in list(M)$
$env' \in list(M)$
$f \in length(env) \to length(env')$
$env1 \in list(M)$
$\bigwedge i\ .\ i < length(env) \Longrightarrow nth(i,env) = nth(f`i,env')$
shows $\bigwedge i\ .\ i < length(env)+_\omega length(env1) \Longrightarrow$
$nth(i,env@env1) = nth(rsum(f,id(length(env1)),length(env),length(env'),length(env1))`i,env'@env1)$
⟨*proof*⟩

lemma *sum_action_id_aux* :
assumes
$f \in m \to n$
$env \in list(M)$
$env' \in list(M)$
$env1 \in list(M)$
$length(env) = m$
$length(env') = n$
$length(env1) = p$
$\bigwedge i\ .\ i < m \Longrightarrow nth(i,env) = nth(f`i,env')$
shows $\bigwedge i\ .\ i < m+_\omega length(env1) \Longrightarrow$
$nth(i,env@env1) = nth(rsum(f,id(length(env1)),m,n,length(env1))`i,env'@env1)$
⟨*proof*⟩

definition
$sum\_id :: [i,i] \Rightarrow i$ **where**
$sum\_id(m,f) \equiv rsum(\lambda x \in 1.x,f,1,1,m)$

lemma *sum_id0* : $m \in nat \Longrightarrow sum\_id(m,f)`0 = 0$
⟨*proof*⟩

lemma *sum_idS* : $p \in nat \Longrightarrow q \in nat \Longrightarrow f \in p \rightarrow q \Longrightarrow x \in p \Longrightarrow sum\_id(p,f)\,`(succ(x)) = succ(f`x)$
⟨*proof*⟩

lemma *sum_id_tc_aux* :
$p \in nat \Longrightarrow q \in nat \Longrightarrow f \in p \rightarrow q \Longrightarrow sum\_id(p,f) \in 1+_\omega p \rightarrow 1+_\omega q$
⟨*proof*⟩

lemma *sum_id_tc* :
$n \in nat \Longrightarrow m \in nat \Longrightarrow f \in n \rightarrow m \Longrightarrow sum\_id(n,f) \in succ(n) \rightarrow succ(m)$
⟨*proof*⟩

4.2 Renaming of formulas

consts *ren* :: $i \Rightarrow i$
primrec
$ren(Member(x,y)) =$
$(\lambda\ n \in nat\ .\ \lambda\ m \in nat.\ \lambda f \in n \rightarrow m.\ Member\ (f`x,\ f`y))$

$ren(Equal(x,y)) =$
$(\lambda\ n \in nat\ .\ \lambda\ m \in nat.\ \lambda f \in n \rightarrow m.\ Equal\ (f`x,\ f`y))$

$ren(Nand(p,q)) =$
$(\lambda\ n \in nat\ .\ \lambda\ m \in nat.\ \lambda f \in n \rightarrow m.\ Nand\ (ren(p)`n`m`f,\ ren(q)`n`m`f))$

$ren(Forall(p)) =$
$(\lambda\ n \in nat\ .\ \lambda\ m \in nat.\ \lambda f \in n \rightarrow m.\ Forall\ (ren(p)`succ(n)`succ(m)`sum\_id(n,f)))$

lemma *arity_meml* : $l \in nat \Longrightarrow Member(x,y) \in formula \Longrightarrow arity(Member(x,y)) \leq l \Longrightarrow x \in l$
⟨*proof*⟩
lemma *arity_memr* : $l \in nat \Longrightarrow Member(x,y) \in formula \Longrightarrow arity(Member(x,y)) \leq l \Longrightarrow y \in l$
⟨*proof*⟩
lemma *arity_eql* : $l \in nat \Longrightarrow Equal(x,y) \in formula \Longrightarrow arity(Equal(x,y)) \leq l \Longrightarrow x \in l$
⟨*proof*⟩
lemma *arity_eqr* : $l \in nat \Longrightarrow Equal(x,y) \in formula \Longrightarrow arity(Equal(x,y)) \leq l \Longrightarrow y \in l$
⟨*proof*⟩
lemma *nand_ar1* : $p \in formula \Longrightarrow q \in formula \Longrightarrow arity(p) \leq arity(Nand(p,q))$
⟨*proof*⟩
lemma *nand_ar2* : $p \in formula \Longrightarrow q \in formula \Longrightarrow arity(q) \leq arity(Nand(p,q))$
⟨*proof*⟩

lemma *nand_ar1D* : $p \in formula \Longrightarrow q \in formula \Longrightarrow arity(Nand(p,q)) \leq n \Longrightarrow arity(p) \leq n$
⟨*proof*⟩

lemma *nand_ar2D* : $p \in formula \Longrightarrow q \in formula \Longrightarrow arity(Nand(p,q)) \leq n \Longrightarrow arity(q) \leq n$
⟨*proof*⟩

lemma *ren_tc* : $p \in formula \Longrightarrow$
$(\bigwedge n\ m\ f\ .\ n \in nat \Longrightarrow m \in nat \Longrightarrow f \in n \rightarrow m \Longrightarrow ren(p)`n`m`f \in formula)$
⟨*proof*⟩

lemma *arity_ren* :
fixes p
assumes $p \in formula$
shows $\bigwedge n\ m\ f\ .\ n \in nat \Longrightarrow m \in nat \Longrightarrow f \in n \rightarrow m \Longrightarrow arity(p) \leq n \Longrightarrow arity(ren(p)`n`m`f) \leq m$
⟨*proof*⟩

lemma *arity_forallE* : $p \in formula \Longrightarrow m \in nat \Longrightarrow arity(Forall(p)) \leq m \Longrightarrow arity(p) \leq succ(m)$
⟨*proof*⟩

lemma *env_coincidence_sum_id* :
assumes $m \in nat\ n \in nat$
$\varrho \in list(A)\ \varrho' \in list(A)$
$f \in n \rightarrow m$
$\bigwedge i\ .\ i < n \Longrightarrow nth(i,\varrho) = nth(f`i,\varrho')$
$a \in A\ j \in succ(n)$
shows $nth(j,Cons(a,\varrho)) = nth(sum\_id(n,f)`j,Cons(a,\varrho'))$
⟨*proof*⟩

lemma *sats_iff_sats_ren* :
assumes $\varphi \in formula$
shows $[\![\ n \in nat\ ;\ m \in nat\ ;\ \varrho \in list(M)\ ;\ \varrho' \in list(M)\ ;\ f \in n \rightarrow m\ ;$
$arity(\varphi) \leq n\ ;$
$\bigwedge i\ .\ i < n \Longrightarrow nth(i,\varrho) = nth(f`i,\varrho')\ ]\!] \Longrightarrow$
$sats(M,\varphi,\varrho) \longleftrightarrow sats(M,ren(\varphi)`n`m`f,\varrho')$
⟨*proof*⟩

end
theory *Utils*
imports *ZF-Constructible.Formula*
begin

This theory encapsulates some ML utilities

⟨*ML*⟩

end
theory *Renaming_Auto*
imports

Renaming
Utils
keywords
rename :: *thy_decl* % *ML*
and
simple_rename :: *thy_decl* % *ML*
and
src
and
tgt
abbrevs
simple_rename =

begin

lemmas *nat_succI* = *nat_succ_iff*[*THEN iffD2*]
⟨*ML*⟩
end
theory *M_Basic_No_Repl*
imports *ZF-Constructible.Relative*
begin

This locale is exactly *M_basic* without its only replacement instance.

locale *M_basic_no_repl* = *M_trivial* +
assumes *Inter_separation*:
M(*A*) ==> *separation*(*M*, λ*x*. ∀ *y*[*M*]. *y*∈*A* ⟶ *x*∈*y*)
and *Diff_separation*:
M(*B*) ==> *separation*(*M*, λ*x*. *x* ∉ *B*)
and *cartprod_separation*:
[| *M*(*A*); *M*(*B*) |]
==> *separation*(*M*, λ*z*. ∃ *x*[*M*]. *x*∈*A* & (∃ *y*[*M*]. *y*∈*B* & *pair*(*M*,*x*,*y*,*z*)))
and *image_separation*:
[| *M*(*A*); *M*(*r*) |]
==> *separation*(*M*, λ*y*. ∃ *p*[*M*]. *p*∈*r* & (∃ *x*[*M*]. *x*∈*A* & *pair*(*M*,*x*,*y*,*p*)))
and *converse_separation*:
M(*r*) ==> *separation*(*M*,
λ*z*. ∃ *p*[*M*]. *p*∈*r* & (∃ *x*[*M*]. ∃ *y*[*M*]. *pair*(*M*,*x*,*y*,*p*) & *pair*(*M*,*y*,*x*,*z*)))
and *restrict_separation*:
M(*A*) ==> *separation*(*M*, λ*z*. ∃ *x*[*M*]. *x*∈*A* & (∃ *y*[*M*]. *pair*(*M*,*x*,*y*,*z*)))
and *comp_separation*:
[| *M*(*r*); *M*(*s*) |]
==> *separation*(*M*, λ*xz*. ∃ *x*[*M*]. ∃ *y*[*M*]. ∃ *z*[*M*]. ∃ *xy*[*M*]. ∃ *yz*[*M*].
pair(*M*,*x*,*z*,*xz*) & *pair*(*M*,*x*,*y*,*xy*) & *pair*(*M*,*y*,*z*,*yz*) &
xy∈*s* & *yz*∈*r*)
and *pred_separation*:
[| *M*(*r*); *M*(*x*) |] ==> *separation*(*M*, λ*y*. ∃ *p*[*M*]. *p*∈*r* & *pair*(*M*,*y*,*x*,*p*))
and *Memrel_separation*:
separation(*M*, λ*z*. ∃ *x*[*M*]. ∃ *y*[*M*]. *pair*(*M*,*x*,*y*,*z*) & *x* ∈ *y*)
and *is_recfun_separation*:

— for well-founded recursion: used to prove *is_recfun_equal*
[| $M(r)$; $M(f)$; $M(g)$; $M(a)$; $M(b)$ |]
==> *separation*(M,
$\lambda x.\ \exists\, xa[M].\ \exists\, xb[M].$
pair(M,x,a,xa) & $xa \in r$ & *pair*(M,x,b,xb) & $xb \in r$ &
($\exists\, fx[M].\ \exists\, gx[M].$ *fun_apply*(M,f,x,fx) & *fun_apply*(M,g,x,gx) &
$fx \neq gx$))
and *power_ax*: *power_ax*(M)

lemma (**in** *M_basic_no_repl*) *cartprod_iff*:
[| $M(A)$; $M(B)$; $M(C)$ |]
==> *cartprod*(M,A,B,C) $\longleftrightarrow$
($\exists\, p1[M].\ \exists\, p2[M].$ *powerset*(M,$A \cup B$,$p1$) & *powerset*(M,$p1$,$p2$) &
$C = \{z \in p2.\ \exists\, x{\in}A.\ \exists\, y{\in}B.\ z = \langle x,y\rangle\}$)
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *cartprod_closed_lemma*:
[| $M(A)$; $M(B)$ |] ==> $\exists\, C[M].$ *cartprod*(M,A,B,C)
⟨*proof*⟩

All the lemmas above are necessary because Powerset is not absolute. I should have used Replacement instead!

lemma (**in** *M_basic_no_repl*) *cartprod_closed* [*intro*,*simp*]:
[| $M(A)$; $M(B)$ |] ==> $M(A{*}B)$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *sum_closed* [*intro*,*simp*]:
[| $M(A)$; $M(B)$ |] ==> $M(A{+}B)$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *sum_abs* [*simp*]:
[| $M(A)$; $M(B)$; $M(Z)$ |] ==> *is_sum*(M,A,B,Z) $\longleftrightarrow$ ($Z = A{+}B$)
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *M_converse_iff*:
$M(r)$ ==>
converse(r) =
$\{z \in \bigcup(\bigcup(r)) * \bigcup(\bigcup(r)).$
$\exists\, p{\in}r.\ \exists\, x[M].\ \exists\, y[M].\ p = \langle x,y\rangle$ & $z = \langle y,x\rangle\}$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *converse_closed* [*intro*,*simp*]:
$M(r)$ ==> M(*converse*(r))
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *converse_abs* [*simp*]:
[| $M(r)$; $M(z)$ |] ==> *is_converse*(M,r,z) $\longleftrightarrow$ z = *converse*(r)
⟨*proof*⟩

4.2.1 image, preimage, domain, range

lemma (**in** *M_basic_no_repl*) *image_closed* [*intro*,*simp*]:
[| $M(A)$; $M(r)$ |] ==> $M(r``A)$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *vimage_abs* [*simp*]:
[| $M(r)$; $M(A)$; $M(z)$ |] ==> $pre\_image(M,r,A,z) \longleftrightarrow z = r\text{-}``A$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *vimage_closed* [*intro*,*simp*]:
[| $M(A)$; $M(r)$ |] ==> $M(r\text{-}``A)$
⟨*proof*⟩

4.2.2 Domain, range and field

lemma (**in** *M_basic_no_repl*) *domain_closed* [*intro*,*simp*]:
$M(r)$ ==> $M(domain(r))$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *range_closed* [*intro*,*simp*]:
$M(r)$ ==> $M(range(r))$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *field_abs* [*simp*]:
[| $M(r)$; $M(z)$ |] ==> $is\_field(M,r,z) \longleftrightarrow z = field(r)$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *field_closed* [*intro*,*simp*]:
$M(r)$ ==> $M(field(r))$
⟨*proof*⟩

4.2.3 Relations, functions and application

lemma (**in** *M_basic_no_repl*) *apply_closed* [*intro*,*simp*]:
[|$M(f)$; $M(a)$|] ==> $M(f`a)$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *apply_abs* [*simp*]:
[| $M(f)$; $M(x)$; $M(y)$ |] ==> $fun\_apply(M,f,x,y) \longleftrightarrow f`x = y$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *injection_abs* [*simp*]:
[| $M(A)$; $M(f)$ |] ==> $injection(M,A,B,f) \longleftrightarrow f \in inj(A,B)$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *surjection_abs* [*simp*]:
[| $M(A)$; $M(B)$; $M(f)$ |] ==> $surjection(M,A,B,f) \longleftrightarrow f \in surj(A,B)$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *bijection_abs* [*simp*]:
[| $M(A)$; $M(B)$; $M(f)$ |] ==> $bijection(M,A,B,f) \longleftrightarrow f \in bij(A,B)$
⟨*proof*⟩

4.2.4 Composition of relations

lemma (**in** *M_basic_no_repl*) *M_comp_iff*:
[| $M(r)$; $M(s)$ |]
==> $r\ O\ s =$
$\{xz \in domain(s) * range(r).$
$\exists x[M].\ \exists y[M].\ \exists z[M].\ xz = \langle x,z\rangle\ \&\ \langle x,y\rangle \in s\ \&\ \langle y,z\rangle \in r\}$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *comp_closed* [*intro*,*simp*]:
[| $M(r)$; $M(s)$ |] ==> $M(r\ O\ s)$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *composition_abs* [*simp*]:
[| $M(r)$; $M(s)$; $M(t)$ |] ==> $composition(M,r,s,t) \longleftrightarrow t = r\ O\ s$
⟨*proof*⟩

no longer needed

lemma (**in** *M_basic_no_repl*) *restriction_is_function*:
[| $restriction(M,f,A,z)$; $function(f)$; $M(f)$; $M(A)$; $M(z)$ |]
==> $function(z)$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *restrict_closed* [*intro*,*simp*]:
[| $M(A)$; $M(r)$ |] ==> $M(restrict(r,A))$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *Inter_closed* [*intro*,*simp*]:
$M(A)$ ==> $M(\bigcap(A))$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *Int_closed* [*intro*,*simp*]:
[| $M(A)$; $M(B)$ |] ==> $M(A \cap B)$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *Diff_closed* [*intro*,*simp*]:
[|$M(A)$; $M(B)$|] ==> $M(A\text{-}B)$
⟨*proof*⟩

4.2.5 Some Facts About Separation Axioms

lemma (**in** *M_basic_no_repl*) *separation_conj*:
[|$separation(M,P)$; $separation(M,Q)$|] ==> $separation(M,\ \lambda z.\ P(z)\ \&\ Q(z))$
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *separation_disj*:

[|*separation*(*M*,*P*); *separation*(*M*,*Q*)|] ==> *separation*(*M*, λ*z*. *P*(*z*) | *Q*(*z*))
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *separation_neg*:
separation(*M*,*P*) ==> *separation*(*M*, λ*z*. ~*P*(*z*))
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *separation_imp*:
[|*separation*(*M*,*P*); *separation*(*M*,*Q*)|]
==> *separation*(*M*, λ*z*. *P*(*z*) ⟶ *Q*(*z*))
⟨*proof*⟩

This result is a hint of how little can be done without the Reflection Theorem. The quantifier has to be bounded by a set. We also need another instance of Separation!

lemma (**in** *M_basic_no_repl*) *separation_rall*:
[|*M*(*Y*); ∀ *y*[*M*]. *separation*(*M*, λ*x*. *P*(*x*,*y*));
∀ *z*[*M*]. *strong_replacement*(*M*, λ*x y*. *y* = {*u* ∈ *z* . *P*(*u*,*x*)})|]
==> *separation*(*M*, λ*x*. ∀ *y*[*M*]. *y*∈*Y* ⟶ *P*(*x*,*y*))
⟨*proof*⟩

4.2.6 Functions and function space

lemma (**in** *M_basic_no_repl*) *succ_fun_eq2*:
[|*M*(*B*); *M*(*n*->*B*)|] ==>
succ(*n*) -> *B* =
⋃{*z*. *p* ∈ (*n*->*B*)**B*, ∃ *f*[*M*]. ∃ *b*[*M*]. *p* = <*f*,*b*> & *z* = {*cons*(<*n*,*b*>, *f*)}}
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *list_case'_closed* [*intro*,*simp*]:
[|*M*(*k*); *M*(*a*); ∀ *x*[*M*]. ∀ *y*[*M*]. *M*(*b*(*x*,*y*))|] ==> *M*(*list_case'*(*a*,*b*,*k*))
⟨*proof*⟩

lemma (**in** *M_basic_no_repl*) *tl'_closed*: *M*(*x*) ==> *M*(*tl'*(*x*))
⟨*proof*⟩

sublocale *M_basic* ⊆ *mbnr*:*M_basic_no_repl*
⟨*proof*⟩

end

5 Some enhanced theorems on recursion

theory *Recursion_Thms*
imports *ZF-Constructible.Datatype_absolute*

begin

— Removing arities from inherited simpset
declare *arity_And* [*simp del*] *arity_Or*[*simp del*] *arity_Implies*[*simp del*]
arity_Exists[*simp del*] *arity_Iff*[*simp del*]
arity_subset_fm [*simp del*] *arity_ordinal_fm*[*simp del*] *arity_transset_fm*[*simp del*]

We prove results concerning definitions by well-founded recursion on some relation R and its transitive closure $R^{\widehat{}}*$

lemma *fld_restrict_eq* : $a \in A \Longrightarrow (r \cap A{\times}A)\text{-}``\{a\} = (r\text{-}``\{a\} \cap A)$
⟨*proof*⟩

lemma *fld_restrict_mono* : $relation(r) \Longrightarrow A \subseteq B \Longrightarrow r \cap A{\times}A \subseteq r \cap B{\times}B$
⟨*proof*⟩

lemma *fld_restrict_dom* :
assumes $relation(r)$ $domain(r) \subseteq A$ $range(r) \subseteq A$
shows $r \cap A{\times}A = r$
⟨*proof*⟩

definition *tr_down* :: $[i,i] \Rightarrow i$
where $tr\_down(r,a) = (r^{\widehat{}}+)\text{-}``\{a\}$

lemma *tr_downD* : $x \in tr\_down(r,a) \Longrightarrow \langle x,a\rangle \in r^{\widehat{}}+$
⟨*proof*⟩

lemma *pred_down* : $relation(r) \Longrightarrow r\text{-}``\{a\} \subseteq tr\_down(r,a)$
⟨*proof*⟩

lemma *tr_down_mono* : $relation(r) \Longrightarrow x \in r\text{-}``\{a\} \Longrightarrow tr\_down(r,x) \subseteq tr\_down(r,a)$
⟨*proof*⟩

lemma *rest_eq* :
assumes $relation(r)$ **and** $r\text{-}``\{a\} \subseteq B$ **and** $a \in B$
shows $r\text{-}``\{a\} = (r \cap B{\times}B)\text{-}``\{a\}$
⟨*proof*⟩

lemma *wfrec_restr_eq* : $r' = r \cap A{\times}A \Longrightarrow wfrec[A](r,a,H) = wfrec(r',a,H)$
⟨*proof*⟩

lemma *wfrec_restr* :
assumes *rr*: $relation(r)$ **and** *wfr*:$wf(r)$
shows $a \in A \Longrightarrow tr\_down(r,a) \subseteq A \Longrightarrow wfrec(r,a,H) = wfrec[A](r,a,H)$
⟨*proof*⟩

lemmas *wfrec_tr_down* = *wfrec_restr*[*OF _ _ _ subset_refl*]

lemma *wfrec_trans_restr* : $relation(r) \Longrightarrow wf(r) \Longrightarrow trans(r) \Longrightarrow r\text{-}``\{a\} \subseteq A \Longrightarrow$
$a \in A \Longrightarrow$

wfrec(*r*, *a*, *H*) = *wfrec*[*A*](*r*, *a*, *H*)
⟨*proof*⟩

lemma *field_trancl* : *field*(*r*^+) = *field*(*r*)
⟨*proof*⟩

definition
Rrel :: [*i*⇒*i*⇒*o*,*i*] ⇒ *i* **where**
Rrel(*R*,*A*) ≡ {*z*∈*A*×*A*. ∃ *x y*. *z* = ⟨*x*, *y*⟩ ∧ *R*(*x*,*y*)}

lemma *RrelI* : *x* ∈ *A* ⟹ *y* ∈ *A* ⟹ *R*(*x*,*y*) ⟹ ⟨*x*,*y*⟩ ∈ *Rrel*(*R*,*A*)
⟨*proof*⟩

lemma *Rrel_mem*: *Rrel*(*mem*,*x*) = *Memrel*(*x*)
⟨*proof*⟩

lemma *relation_Rrel*: *relation*(*Rrel*(*R*,*d*))
⟨*proof*⟩

lemma *field_Rrel*: *field*(*Rrel*(*R*,*d*)) ⊆ *d*
⟨*proof*⟩

lemma *Rrel_mono* : *A* ⊆ *B* ⟹ *Rrel*(*R*,*A*) ⊆ *Rrel*(*R*,*B*)
⟨*proof*⟩

lemma *Rrel_restr_eq* : *Rrel*(*R*,*A*) ∩ *B*×*B* = *Rrel*(*R*,*A*∩*B*)
⟨*proof*⟩

lemma *field_Memrel* : *field*(*Memrel*(*A*)) ⊆ *A*

⟨*proof*⟩

lemma *restrict_trancl_Rrel*:
assumes *R*(*w*,*y*)
shows *restrict*(*f*,*Rrel*(*R*,*d*)-‘‘{*y*})‘*w*
= *restrict*(*f*,(*Rrel*(*R*,*d*)^+)-‘‘{*y*})‘*w*
⟨*proof*⟩

lemma *restrict_trans_eq*:
assumes *w* ∈ *y*
shows *restrict*(*f*,*Memrel*(*eclose*({*x*}))-‘‘{*y*})‘*w*
= *restrict*(*f*,(*Memrel*(*eclose*({*x*}))^+)-‘‘{*y*})‘*w*
⟨*proof*⟩

lemma *wf_eq_trancl*:
assumes ⋀ *f y* . *H*(*y*,*restrict*(*f*,*R*-‘‘{*y*})) = *H*(*y*,*restrict*(*f*,*R*^+-‘‘{*y*}))
shows *wfrec*(*R*, *x*, *H*) = *wfrec*(*R*^+, *x*, *H*) (**is** *wfrec*(*?r*,_,_) = *wfrec*(*?r′*,_,_))

⟨*proof*⟩

lemma *transrec_equal_on_Ord*:
assumes
$\bigwedge x\ f\ .\ Ord(x) \Longrightarrow foo(x,f) = bar(x,f)$
$Ord(\alpha)$
shows
$transrec(\alpha,\ foo) = transrec(\alpha,\ bar)$
⟨*proof*⟩
lemma (**in** *M_eclose*) *transrec_equal_on_M*:
assumes
$\bigwedge x\ f\ .\ M(x) \Longrightarrow M(f) \Longrightarrow foo(x,f) = bar(x,f)$
$\bigwedge \beta.\ M(\beta) \Longrightarrow transrec\_replacement(M,is\_foo,\beta)\ relation2(M,is\_foo,foo)$
$strong\_replacement(M,\ \lambda x\ y.\ y = \langle x,\ transrec(x,\ foo)\rangle)$
$\forall\, x[M].\ \forall\, g[M].\ function(g) \longrightarrow M(foo(x,g))$
$M(\alpha)\ Ord(\alpha)$
shows
$transrec(\alpha,\ foo) = transrec(\alpha,\ bar)$
⟨*proof*⟩

lemma *ordermap_restr_eq*:
assumes $well\_ord(X,r)$
shows $ordermap(X,\ r) = ordermap(X,\ r \cap X\times X)$
⟨*proof*⟩

end

6 Automatic synthesis of formulas

theory *Synthetic_Definition*
imports
Utils
keywords
synthesize :: *thy_decl* % *ML*
and
synthesize_notc :: *thy_decl* % *ML*
and
generate_schematic :: *thy_decl* % *ML*
and
arity_theorem :: *thy_decl* % *ML*
and
manual_schematic :: *thy_goal_stmt* % *ML*
and
manual_arity :: *thy_goal_stmt* % *ML*
and
from_schematic
and
for

and
from_definition
and
assuming
and
intermediate

begin

named_theorems *fm_definitions Definitions of synthetized formulas.*

named_theorems *iff_sats Theorems for synthetising formulas.*

named_theorems *arity Theorems for arity of formulas.*

named_theorems *arity_aux Auxiliary theorems for calculating arities.*

⟨*ML*⟩

The `synthetic_def` function extracts definitions from schematic goals. A new definition is added to the context.

end

7 Aids to internalize formulas

theory *Internalizations*
imports
ZF-Constructible.DPow_absolute
Synthetic_Definition
Nat_Miscellanea
begin

definition
infinity_ax :: $(i \Rightarrow o) \Rightarrow o$ **where**
infinity_ax$(M) \equiv$
$(\exists\, I[M].\ (\exists\, z[M].\ empty(M,z) \wedge z{\in}I) \wedge (\forall\, y[M].\ y{\in}I \longrightarrow (\exists\, sy[M].\ successor(M,y,sy) \wedge sy{\in}I)))$

definition
wellfounded_trancl :: $[i{=>}o,i,i,i] => o$ **where**
wellfounded_trancl$(M,Z,r,p) \equiv$
$\exists\, w[M].\ \exists\, wx[M].\ \exists\, rp[M].$
$w \in Z\ \&\ pair(M,w,p,wx)\ \&\ tran\_closure(M,r,rp)\ \&\ wx \in rp$

lemma *empty_intf* :
infinity_ax$(M) \Longrightarrow$
$(\exists\, z[M].\ empty(M,z))$
⟨*proof*⟩

lemma *Transset_intf* :
Transset(*M*) $\Longrightarrow$ *y*$\in$*x* $\Longrightarrow$ *x* $\in$ *M* $\Longrightarrow$ *y* $\in$ *M*
⟨*proof*⟩

definition
choice_ax :: (*i*$\Rightarrow$*o*) $\Rightarrow$ *o* **where**
choice_ax(*M*) $\equiv$ $\forall$ *x*[*M*]. $\exists$ *a*[*M*]. $\exists$ *f*[*M*]. *ordinal*(*M*,*a*) $\wedge$ *surjection*(*M*,*a*,*x*,*f*)

lemma (**in** *M_basic*) *choice_ax_abs* :
choice_ax(*M*) $\longleftrightarrow$ ($\forall$ *x*[*M*]. $\exists$ *a*[*M*]. $\exists$ *f*[*M*]. *Ord*(*a*) $\wedge$ *f* $\in$ *surj*(*a*,*x*))
⟨*proof*⟩

Setting up notation for internalized formulas

abbreviation
dec10 :: *i* (*10*) **where** *10* $\equiv$ *succ*(*9*)
abbreviation
dec11 :: *i* (*11*) **where** *11* $\equiv$ *succ*(*10*)
abbreviation
dec12 :: *i* (*12*) **where** *12* $\equiv$ *succ*(*11*)
abbreviation
dec13 :: *i* (*13*) **where** *13* $\equiv$ *succ*(*12*)
abbreviation
dec14 :: *i* (*14*) **where** *14* $\equiv$ *succ*(*13*)
abbreviation
dec15 :: *i* (*15*) **where** *15* $\equiv$ *succ*(*14*)
abbreviation
dec16 :: *i* (*16*) **where** *16* $\equiv$ *succ*(*15*)
abbreviation
dec17 :: *i* (*17*) **where** *17* $\equiv$ *succ*(*16*)
abbreviation
dec18 :: *i* (*18*) **where** *18* $\equiv$ *succ*(*17*)
abbreviation
dec19 :: *i* (*19*) **where** *19* $\equiv$ *succ*(*18*)
abbreviation
dec20 :: *i* (*20*) **where** *20* $\equiv$ *succ*(*19*)
abbreviation
dec21 :: *i* (*21*) **where** *21* $\equiv$ *succ*(*20*)
abbreviation
dec22 :: *i* (*22*) **where** *22* $\equiv$ *succ*(*21*)
abbreviation
dec23 :: *i* (*23*) **where** *23* $\equiv$ *succ*(*22*)
abbreviation
dec24 :: *i* (*24*) **where** *24* $\equiv$ *succ*(*23*)
abbreviation
dec25 :: *i* (*25*) **where** *25* $\equiv$ *succ*(*24*)
abbreviation
dec26 :: *i* (*26*) **where** *26* $\equiv$ *succ*(*25*)
abbreviation
dec27 :: *i* (*27*) **where** *27* $\equiv$ *succ*(*26*)

abbreviation
  dec28 :: *i* (*28*) **where** *28* ≡ *succ*(*27*)
abbreviation
  dec29 :: *i* (*29*) **where** *29* ≡ *succ*(*28*)

notation *Member* (‹·_ ∈/ _·›)
notation *Equal* (‹·_ =/ _·›)
notation *Nand* (‹·¬′(_ ∧/ _′)·›)
notation *And* (‹·_ ∧/ _·›)
notation *Or* (‹·_ ∨/ _·›)
notation *Iff* (‹·_ ↔/ _·›)
notation *Implies* (‹·_ →/ _·›)
notation *Neg* (‹·¬_·›)
notation *Forall* (‹′(·∀(/_)·′)›)
notation *Exists* (‹′(·∃(/_)·′)›)

notation *subset_fm* (‹·_ ⊆/ _·›)
notation *succ_fm* (‹·*succ*′(_′) *is* _·›)
notation *empty_fm* (‹·_ *is empty*·›)
notation *fun_apply_fm* (‹·_‘_ *is* _·›)
notation *big_union_fm* (‹·⋃_ *is* _·›)
notation *upair_fm* (‹·{_,_} *is* _ ·›)
notation *ordinal_fm* (‹·_ *is ordinal*·›)

notation *pair_fm* (‹·⟨_,_⟩ *is* _ ·›)
notation *composition_fm* (‹·_ ∘ _ *is* _ ·›)
notation *domain_fm* (‹·*dom*′(_′) *is* _ ·›)
notation *range_fm* (‹·*ran*′(_′) *is* _ ·›)
notation *union_fm* (‹·_ ∪ _ *is* _ ·›)
notation *image_fm* (‹·_ ‘‘ _ *is* _ ·›)
notation *pre_image_fm* (‹·_ -‘‘ _ *is* _ ·›)
notation *field_fm* (‹·*fld*′(_′) *is* _ ·›)
notation *cons_fm* (‹·*cons*′(_,_′) *is* _ ·›)
notation *number1_fm* (‹·_ *is the number one*·›)
notation *function_fm* (‹·_ *is funct*·›)
notation *relation_fm* (‹·_ *is relat*·›)
notation *restriction_fm* (‹·_ ↾ _ *is* _ ·›)
notation *transset_fm* (‹·_ *is transitive*·›)
notation *limit_ordinal_fm* (‹·_ *is limit*·›)
notation *finite_ordinal_fm* (‹·_ *is finite ord*·›)
notation *omega_fm* (‹·_ *is* ω·›)
notation *cartprod_fm* (‹·_ × _ *is* _·›)
notation *Memrel_fm* (‹·*Memrel*′(_′) *is* _·›)
notation *quasinat_fm* (‹·_ *is qnat*·›)

notation *Inl_fm* (‹·*Inl*′(_′) *is* _ ·›)
notation *Inr_fm* (‹·*Inr*′(_′) *is* _ ·›)
notation *pred_set_fm* (‹·_-*predecessors of* _ *are* _·›)

abbreviation
fm_typedfun :: $[i,i,i] \Rightarrow i$ ($\langle\cdot$_ : _ $\rightarrow$ _$\cdot\rangle$) **where**
fm_typedfun$(f,A,B) \equiv$ *typed_function_fm*(A,B,f)

abbreviation
fm_surjection :: $[i,i,i] \Rightarrow i$ ($\langle\cdot$_ *surjects* _ *to* _$\cdot\rangle$) **where**
fm_surjection$(f,A,B) \equiv$ *surjection_fm*(A,B,f)

abbreviation
fm_injection :: $[i,i,i] \Rightarrow i$ ($\langle\cdot$_ *injects* _ *to* _$\cdot\rangle$) **where**
fm_injection$(f,A,B) \equiv$ *injection_fm*(A,B,f)

abbreviation
fm_bijection :: $[i,i,i] \Rightarrow i$ ($\langle\cdot$_ *bijects* _ *to* _$\cdot\rangle$) **where**
fm_bijection$(f,A,B) \equiv$ *bijection_fm*(A,B,f)

We found it useful to have slightly different versions of some results in ZF-Constructible:

lemma *nth_closed* :
assumes $env{\in}list(A)$ $0{\in}A$
shows $nth(n,env){\in}A$
⟨*proof*⟩

lemma *conj_setclass_model_iff_sats* [*iff_sats*]:
$[\!|\ 0 \in A;\ nth(i,env) = x;\ env \in list(A);$
$P \longleftrightarrow sats(A,p,env);\ env \in list(A)\ |\!]$
$\Longrightarrow (P \wedge (\#\#A)(x)) \longleftrightarrow sats(A,\ p,\ env)$
$[\!|\ 0 \in A;\ nth(i,env) = x;\ env \in list(A);$
$P \longleftrightarrow sats(A,p,env);\ env \in list(A)\ |\!]$
$\Longrightarrow ((\#\#A)(x) \wedge P) \longleftrightarrow sats(A,\ p,\ env)$
⟨*proof*⟩

lemma *conj_mem_model_iff_sats* [*iff_sats*]:
$[\!|\ 0 \in A;\ nth(i,env) = x;\ env \in list(A);$
$P \longleftrightarrow sats(A,p,env);\ env \in list(A)\ |\!]$
$\Longrightarrow (P \wedge x \in A) \longleftrightarrow sats(A,\ p,\ env)$
$[\!|\ 0 \in A;\ nth(i,env) = x;\ env \in list(A);$
$P \longleftrightarrow sats(A,p,env);\ env \in list(A)\ |\!]$
$\Longrightarrow (x \in A \wedge P) \longleftrightarrow sats(A,\ p,\ env)$
⟨*proof*⟩

lemma *mem_model_iff_sats* [*iff_sats*]:
$[\!|\ 0 \in A;\ nth(i,env) = x;\ env \in list(A)|\!]$
$\Longrightarrow (x{\in}A) \longleftrightarrow sats(A,\ Exists(Equal(0,0)),\ env)$
⟨*proof*⟩

lemma *subset_iff_sats*[*iff_sats*]:
$nth(i,\ env) = x \Longrightarrow nth(j,\ env) = y \Longrightarrow i{\in}nat \Longrightarrow j{\in}nat \Longrightarrow$
$env \in list(A) \Longrightarrow subset(\#\#A,\ x,\ y) \longleftrightarrow sats(A,\ subset\_fm(i,\ j),\ env)$
⟨*proof*⟩

lemma *not_mem_model_iff_sats* [*iff_sats*]:
$[|\ 0 \in A;\ nth(i,env) = x;\ env \in list(A)|]$
$\quad ==> (\forall\ x\ .\ x \notin A) \longleftrightarrow sats(A,\ Neg(Exists(Equal(0,0))),\ env)$
⟨*proof*⟩

lemma *top_iff_sats* [*iff_sats*]:
$env \in list(A) \Longrightarrow 0 \in A \Longrightarrow sats(A,\ Exists(Equal(0,0)),\ env)$
⟨*proof*⟩

lemma *prefix1_iff_sats*[*iff_sats*]:
assumes
$x \in nat\ env \in list(A)\ 0 \in A\ a \in A$
shows
$a = nth(x,env) \longleftrightarrow sats(A,\ Equal(0,x+_\omega 1),\ Cons(a,env))$
$nth(x,env) = a \longleftrightarrow sats(A,\ Equal(x+_\omega 1,0),\ Cons(a,env))$
$a \in nth(x,env) \longleftrightarrow sats(A,\ Member(0,x+_\omega 1),\ Cons(a,env))$
$nth(x,env) \in a \longleftrightarrow sats(A,\ Member(x+_\omega 1,0),\ Cons(a,env))$
⟨*proof*⟩

lemma *prefix2_iff_sats*[*iff_sats*]:
assumes
$x \in nat\ env \in list(A)\ 0 \in A\ a \in A\ b \in A$
shows
$b = nth(x,env) \longleftrightarrow sats(A,\ Equal(1,x+_\omega 2),\ Cons(a,Cons(b,env)))$
$nth(x,env) = b \longleftrightarrow sats(A,\ Equal(x+_\omega 2,1),\ Cons(a,Cons(b,env)))$
$b \in nth(x,env) \longleftrightarrow sats(A,\ Member(1,x+_\omega 2),\ Cons(a,Cons(b,env)))$
$nth(x,env) \in b \longleftrightarrow sats(A,\ Member(x+_\omega 2,1),\ Cons(a,Cons(b,env)))$
⟨*proof*⟩

lemma *prefix3_iff_sats*[*iff_sats*]:
assumes
$x \in nat\ env \in list(A)\ 0 \in A\ a \in A\ b \in A\ c \in A$
shows
$c = nth(x,env) \longleftrightarrow sats(A,\ Equal(2,x+_\omega 3),\ Cons(a,Cons(b,Cons(c,env))))$
$nth(x,env) = c \longleftrightarrow sats(A,\ Equal(x+_\omega 3,2),\ Cons(a,Cons(b,Cons(c,env))))$
$c \in nth(x,env) \longleftrightarrow sats(A,\ Member(2,x+_\omega 3),\ Cons(a,Cons(b,Cons(c,env))))$
$nth(x,env) \in c \longleftrightarrow sats(A,\ Member(x+_\omega 3,2),\ Cons(a,Cons(b,Cons(c,env))))$
⟨*proof*⟩

lemmas *FOL_sats_iff = sats_Nand_iff sats_Forall_iff sats_Neg_iff sats_And_iff sats_Or_iff sats_Implies_iff sats_Iff_iff sats_Exists_iff*

lemma *nth_ConsI*: $[\![nth(n,l) = x;\ n \in nat]\!] \Longrightarrow nth(succ(n),\ Cons(a,l)) = x$

⟨*proof*⟩

lemmas *nth_rules* = *nth_0 nth_ConsI nat_0I nat_succI*
lemmas *sep_rules* = *nth_0 nth_ConsI FOL_iff_sats function_iff_sats*
fun_plus_iff_sats successor_iff_sats
omega_iff_sats FOL_sats_iff Replace_iff_sats

Also a different compilation of lemmas (term*sep_rules*) used in formula synthesis

lemmas *fm_defs* =
omega_fm_def limit_ordinal_fm_def empty_fm_def typed_function_fm_def
pair_fm_def upair_fm_def domain_fm_def function_fm_def succ_fm_def
cons_fm_def fun_apply_fm_def image_fm_def big_union_fm_def union_fm_def
relation_fm_def composition_fm_def field_fm_def ordinal_fm_def range_fm_def
transset_fm_def subset_fm_def Replace_fm_def

lemmas *formulas_def* [*fm_definitions*] = *fm_defs*
is_iterates_fm_def iterates_MH_fm_def is_wfrec_fm_def is_recfun_fm_def
is_transrec_fm_def
is_nat_case_fm_def quasinat_fm_def number1_fm_def ordinal_fm_def finite_ordinal_fm_def
cartprod_fm_def sum_fm_def Inr_fm_def Inl_fm_def
formula_functor_fm_def
Memrel_fm_def transset_fm_def subset_fm_def pre_image_fm_def restriction_fm_def
list_functor_fm_def tl_fm_def quasilist_fm_def Cons_fm_def Nil_fm_def

lemmas *sep_rules'* [*iff_sats*] = *nth_0 nth_ConsI FOL_iff_sats function_iff_sats*
fun_plus_iff_sats omega_iff_sats

lemmas *more_iff_sats* [*iff_sats*] = *rtran_closure_iff_sats tran_closure_iff_sats*
is_eclose_iff_sats Inl_iff_sats Inr_iff_sats fun_apply_iff_sats cartprod_iff_sats
Collect_iff_sats

end

8 The binder *Least*

theory *Least*
imports
Internalizations

begin

We have some basic results on the least ordinal satisfying a predicate.

lemma *Least_Ord*: $(\mu\ \alpha.\ R(\alpha)) = (\mu\ \alpha.\ Ord(\alpha) \wedge R(\alpha))$
⟨*proof*⟩

lemma *Ord_Least_cong*:
assumes $\bigwedge y.\ Ord(y) \Longrightarrow R(y) \longleftrightarrow Q(y)$

shows $(\mu\ \alpha.\ R(\alpha)) = (\mu\ \alpha.\ Q(\alpha))$
⟨*proof*⟩

definition
$least :: [i⇒o,i⇒o,i] ⇒ o$ **where**
$least(M,Q,i) \equiv ordinal(M,i) \wedge ($
$(empty(M,i) \wedge (\forall\, b[M].\ ordinal(M,b) \longrightarrow \neg Q(b)))$
$\vee\ (Q(i) \wedge (\forall\, b[M].\ ordinal(M,b) \wedge b∈i \longrightarrow \neg Q(b))))$

definition
$least\_fm :: [i,i] ⇒ i$ **where**
$least\_fm(q,i) \equiv And(ordinal\_fm(i),$
$Or(And(empty\_fm(i),Forall(Implies(ordinal\_fm(0),Neg(q)))),$
$And(Exists(And(q,Equal(0,succ(i)))),$
$Forall(Implies(And(ordinal\_fm(0),Member(0,succ(i))),Neg(q))))))$

lemma $least\_fm\_type[TC] : i \in nat \Longrightarrow q∈formula \Longrightarrow least\_fm(q,i) \in formula$
⟨*proof*⟩

lemmas $basic\_fm\_simps = sats\_subset\_fm'\ sats\_transset\_fm'\ sats\_ordinal\_fm'$

lemma *sats_least_fm* :
assumes *p_iff_sats*:
$\bigwedge a.\ a \in A \Longrightarrow P(a) \longleftrightarrow sats(A,\ p,\ Cons(a,\ env))$
shows
$[\![y \in nat;\ env \in list(A)\ ;\ 0∈A]\!]$
$\Longrightarrow sats(A,\ least\_fm(p,y),\ env) \longleftrightarrow$
$least(\#\#A,\ P,\ nth(y,env))$
⟨*proof*⟩

lemma *least_iff_sats* [*iff_sats*]:
assumes *is_Q_iff_sats*:
$\bigwedge a.\ a \in A \Longrightarrow is\_Q(a) \longleftrightarrow sats(A,\ q,\ Cons(a,env))$
shows
$[\![nth(j,env) = y;\ j \in nat;\ env \in list(A);\ 0∈A]\!]$
$\Longrightarrow least(\#\#A,\ is\_Q,\ y) \longleftrightarrow sats(A,\ least\_fm(q,j),\ env)$
⟨*proof*⟩

lemma *least_conj*: $a∈M \Longrightarrow least(\#\#M,\ \lambda x.\ x∈M \wedge Q(x),a) \longleftrightarrow least(\#\#M,Q,a)$
⟨*proof*⟩

context *M_trivial*
begin

8.1 Uniqueness, absoluteness and closure under *Least*

lemma *unique_least*:

assumes $M(a)$ $M(b)$ $least(M,Q,a)$ $least(M,Q,b)$
shows $a=b$
⟨*proof*⟩

lemma *least_abs*:
assumes $\bigwedge x.\ Q(x) \Longrightarrow Ord(x) \Longrightarrow \exists\, y[M].\ Q(y) \wedge Ord(y)\ M(a)$
shows $least(M,Q,a) \longleftrightarrow a = (\mu\ x.\ Q(x))$
⟨*proof*⟩

lemma *Least_closed*:
assumes $\bigwedge x.\ Q(x) \Longrightarrow Ord(x) \Longrightarrow \exists\, y[M].\ Q(y) \wedge Ord(y)$
shows $M(\mu\ x.\ Q(x))$
⟨*proof*⟩

Older, easier to apply versions (with a simpler assumption on Q).

lemma *least_abs'*:
assumes $\bigwedge x.\ Q(x) \Longrightarrow M(x)\ M(a)$
shows $least(M,Q,a) \longleftrightarrow a = (\mu\ x.\ Q(x))$
⟨*proof*⟩

lemma *Least_closed'*:
assumes $\bigwedge x.\ Q(x) \Longrightarrow M(x)$
shows $M(\mu\ x.\ Q(x))$
⟨*proof*⟩

end — *M_trivial*

end

9 Fully relational versions of higher order construct

theory *Higher_Order_Constructs*
imports
Recursion_Thms
Least
begin

syntax
_sats :: $[i,\ i,\ i] \Rightarrow o$ $((\_,\ \_ \models \_)\ [36,36,36]\ 25)$
translations
$(M,env \models \varphi) \rightleftharpoons CONST\ sats(M,\varphi,env)$

definition
is_If :: $[i\Rightarrow o,o,i,i,i] \Rightarrow o$ **where**
$is\_If(M,b,t,f,r) \equiv (b \longrightarrow r=t) \wedge (\neg b \longrightarrow r=f)$

lemma (**in** *M_trans*) *If_abs*:
$is\_If(M,b,t,f,r) \longleftrightarrow r = If(b,t,f)$
⟨*proof*⟩

definition
is_If_fm :: $[i,i,i,i] \Rightarrow i$ **where**
is_If_fm(φ,t,f,r) $\equiv$ *Or*(*And*(φ,*Equal*(t,r)),*And*(*Neg*(φ),*Equal*(f,r)))

lemma *is_If_fm_type* [*TC*]: $\varphi \in formula \Longrightarrow t \in nat \Longrightarrow f \in nat \Longrightarrow r \in nat \Longrightarrow$
is_If_fm(φ,t,f,r) $\in$ *formula*
⟨*proof*⟩

lemma *sats_is_If_fm*:
assumes *Qsats*: $Q \longleftrightarrow A$, $env \models \varphi$ $env \in list(A)$
shows *is_If*(##*A*, *Q*, *nth*(t, *env*), *nth*(f, *env*), *nth*(r, *env*)) $\longleftrightarrow$ *A*, *env* $\models$ *is_If_fm*(φ,t,f,r)
⟨*proof*⟩

lemma *is_If_fm_iff_sats* [*iff_sats*]:
assumes *Qsats*: $Q \longleftrightarrow A$, $env \models \varphi$ **and**
$nth(t, env) = ta$ $nth(f, env) = fa$ $nth(r, env) = ra$
$t \in nat$ $f \in nat$ $r \in nat$ $env \in list(A)$
shows *is_If*(##*A*,*Q*,*ta*,*fa*,*ra*) $\longleftrightarrow$ *A*, *env* $\models$ *is_If_fm*(φ,t,f,r)
⟨*proof*⟩

lemma *arity_is_If_fm* [*arity*]:
$\varphi \in formula \Longrightarrow t \in nat \Longrightarrow f \in nat \Longrightarrow r \in nat \Longrightarrow$
$arity(is\_If\_fm(\varphi, t, f, r)) = arity(\varphi) \cup succ(t) \cup succ(r) \cup succ(f)$
⟨*proof*⟩

definition
is_The :: $[i \Rightarrow o, i \Rightarrow o, i] \Rightarrow o$ **where**
is_The(*M*,*Q*,*i*) $\equiv$ ($Q(i) \wedge (\exists\, x[M].\ Q(x) \wedge (\forall\, y[M].\ Q(y) \longrightarrow y = x))) \vee$
$(\neg(\exists\, x[M].\ Q(x) \wedge (\forall\, y[M].\ Q(y) \longrightarrow y = x))) \wedge empty(M,i)$

lemma (**in** *M_trans*) *The_abs*:
assumes $\bigwedge x.\ Q(x) \Longrightarrow M(x)\ M(a)$
shows *is_The*(*M*,*Q*,*a*) $\longleftrightarrow$ $a = (THE\ x.\ Q(x))$
⟨*proof*⟩

definition
is_recursor :: $[i \Rightarrow o, i, [i,i,i] \Rightarrow o, i, i] \Rightarrow o$ **where**
is_recursor(*M*,*a*,*is_b*,*k*,*r*) $\equiv$ *is_transrec*(*M*, $\lambda n\ f\ ntc.$ *is_nat_case*(*M*,*a*,
$\lambda m\ bmfm.$
$\exists\, fm[M].\ fun\_apply(M,f,m,fm) \wedge is\_b(m,fm,bmfm),n,ntc),k,r)$

lemma (**in** *M_eclose*) *recursor_abs*:
assumes *Ord*(*k*) **and**
types: *M*(*a*) *M*(*k*) *M*(*r*) **and**
b_iff: ⋀*m f bmf*. *M*(*m*) ⟹ *M*(*f*) ⟹ *M*(*bmf*) ⟹ *is_b*(*m*,*f*,*bmf*) ⟷ *bmf* = *b*(*m*,*f*) **and**
b_closed: ⋀*m f bmf*. *M*(*m*) ⟹ *M*(*f*) ⟹ *M*(*b*(*m*,*f*)) **and**
repl: *transrec_replacement*(*M*, λ*n f ntc*. *is_nat_case*(*M*, *a*,
λ*m bmfm*. ∃*fm*[*M*]. *fun_apply*(*M*, *f*, *m*, *fm*) ∧ *is_b*(*m*, *fm*, *bmfm*), *n*, *ntc*), *k*)
shows
is_recursor(*M*,*a*,*is_b*,*k*,*r*) ⟷ *r* = *recursor*(*a*,*b*,*k*)
⟨*proof*⟩

definition
is_wfrec_on :: [*i*=>*o*,[*i*,*i*,*i*]=>*o*,*i*,*i*,*i*, *i*] => *o* **where**
is_wfrec_on(*M*,*MH*,*A*,*r*,*a*,*z*) == *is_wfrec*(*M*,*MH*,*r*,*a*,*z*)

lemma (**in** *M_trancl*) *trans_wfrec_on_abs*:
[|*wf*(*r*); *trans*(*r*); *relation*(*r*); *M*(*r*); *M*(*a*); *M*(*z*);
wfrec_replacement(*M*,*MH*,*r*); *relation2*(*M*,*MH*,*H*);
∀ *x*[*M*]. ∀ *g*[*M*]. *function*(*g*) ⟶ *M*(*H*(*x*,*g*));
r-‘‘{*a*}⊆*A*; *a* ∈ *A*|]
==> *is_wfrec_on*(*M*,*MH*,*A*,*r*,*a*,*z*) ⟷ *z*=*wfrec*[*A*](*r*,*a*,*H*)
⟨*proof*⟩

end

10 Automatic relativization of terms and formulas

Relativization of terms and formulas. Relativization of formulas shares relativized terms as far as possible; assuming that the witnesses for the relativized terms are always unique.

theory *Relativization*
imports
ZF-Constructible.Datatype_absolute
Higher_Order_Constructs
keywords
relativize :: *thy_decl* % *ML*
and
relativize_tm :: *thy_decl* % *ML*
and
reldb_add :: *thy_decl* % *ML*
and
reldb_rem :: *thy_decl* % *ML*
and
relationalize :: *thy_decl* % *ML*
and
rel_closed :: *thy_goal_stmt* % *ML*

and
is_iff_rel :: *thy_goal_stmt* % *ML*
and
univalent :: *thy_goal_stmt* % *ML*
and
absolute
and
functional
and
relational
and
external
and
for

begin

⟨*ML*⟩

lemmas *relative_abs* =
M_trans.empty_abs
M_trans.pair_abs
M_trivial.cartprod_abs
M_trans.union_abs
M_trans.inter_abs
M_trans.setdiff_abs
M_trans.Union_abs
M_trivial.cons_abs

M_trivial.successor_abs
M_trans.Collect_abs
M_trans.Replace_abs
M_trivial.lambda_abs2
M_trans.image_abs

M_trivial.nat_case_abs

M_trivial.omega_abs
M_basic.sum_abs
M_trivial.Inl_abs
M_trivial.Inr_abs
M_basic.converse_abs
M_basic.vimage_abs
M_trans.domain_abs
M_trans.range_abs
M_basic.field_abs

M_basic.composition_abs

M_trans.restriction_abs
M_trans.Inter_abs
M_trivial.bool_of_o_abs
M_trivial.not_abs
M_trivial.and_abs
M_trivial.or_abs
M_trivial.Nil_abs
M_trivial.Cons_abs

M_trivial.list_case_abs
M_trivial.hd_abs
M_trivial.tl_abs
M_trivial.least_abs'
M_eclose.transrec_abs
M_trans.If_abs
M_trans.The_abs
M_eclose.recursor_abs
M_trancl.trans_wfrec_abs
M_trancl.trans_wfrec_on_abs

lemmas *datatype_abs* =
M_datatypes.list_N_abs
M_datatypes.list_abs
M_datatypes.formula_N_abs
M_datatypes.formula_abs
M_eclose.is_eclose_n_abs
M_eclose.eclose_abs
M_datatypes.length_abs
M_datatypes.nth_abs
M_trivial.Member_abs
M_trivial.Equal_abs
M_trivial.Nand_abs
M_trivial.Forall_abs
M_datatypes.depth_abs
M_datatypes.formula_case_abs

declare *relative_abs*[*absolut*]
declare *datatype_abs*[*absolut*]

⟨*ML*⟩

declare *relative_abs*[*Rel*]

declare *datatype_abs*[*Rel*]

⟨*ML*⟩

end
theory *Discipline_Base*

imports
ZF-Constructible.Rank
ZF_Miscellanea
M_Basic_No_Repl
Relativization

begin

declare [[*syntax_ambiguity_warning* = *false*]]

10.1 Discipline of relativization of basic concepts

definition
is_singleton :: [*i*⇒*o*,*i*,*i*] ⇒ *o* **where**
is_singleton(*A*,*x*,*z*) ≡ ∃ *c*[*A*]. *empty*(*A*,*c*) ∧ *is_cons*(*A*,*x*,*c*,*z*)

lemma (**in** *M_trivial*) *singleton_abs*[*simp*] :
⟦ *M*(*x*) ; *M*(*s*) ⟧ ⟹ *is_singleton*(*M*,*x*,*s*) ⟷ *s* = {*x*}
⟨*proof*⟩

⟨*ML*⟩
notation *singleton_fm* (‹·{_} *is* _·›)

lemma (**in** *M_trivial*) *singleton_closed* [*simp*]:
M(*x*) ⟹ *M*({*x*})
⟨*proof*⟩

lemma (**in** *M_trivial*) *Upair_closed*[*simp*]: *M*(*a*) ⟹ *M*(*b*) ⟹ *M*(*Upair*(*a*,*b*))
⟨*proof*⟩

The following named theorems gather instances of transitivity that arise from closure theorems

named_theorems *trans_closed*

definition
is_hcomp :: [*i*⇒*o*,*i*⇒*i*⇒*o*,*i*⇒*i*⇒*o*,*i*,*i*] ⇒ *o* **where**
is_hcomp(*M*,*is_f*,*is_g*,*a*,*w*) ≡ ∃ *z*[*M*]. *is_g*(*a*,*z*) ∧ *is_f*(*z*,*w*)

lemma (**in** *M_trivial*) *is_hcomp_abs*:
assumes
is_f_abs:⋀*a z*. *M*(*a*) ⟹ *M*(*z*) ⟹ *is_f*(*a*,*z*) ⟷ *z* = *f*(*a*) **and**
is_g_abs:⋀*a z*. *M*(*a*) ⟹ *M*(*z*) ⟹ *is_g*(*a*,*z*) ⟷ *z* = *g*(*a*) **and**
g_closed:⋀*a*. *M*(*a*) ⟹ *M*(*g*(*a*))
M(*a*) *M*(*w*)
shows
is_hcomp(*M*,*is_f*,*is_g*,*a*,*w*) ⟷ *w* = *f*(*g*(*a*))
⟨*proof*⟩

definition
hcomp_fm :: $[i⇒i⇒i,i⇒i⇒i,i,i] ⇒ i$ **where**
$hcomp\_fm(pf,pg,a,w) ≡ Exists(And(pg(succ(a),0),pf(0,succ(w))))$

lemma *sats_hcomp_fm*:
assumes
f_iff_sats:$⋀a\ b\ z.\ a∈nat ⟹ b∈nat ⟹ z∈M ⟹$
$is\_f(nth(a,Cons(z,env)),nth(b,Cons(z,env))) ⟷ sats(M,pf(a,b),Cons(z,env))$
and
g_iff_sats:$⋀a\ b\ z.\ a∈nat ⟹ b∈nat ⟹ z∈M ⟹$
$is\_g(nth(a,Cons(z,env)),nth(b,Cons(z,env))) ⟷ sats(M,pg(a,b),Cons(z,env))$
and
$a∈nat\ w∈nat\ env∈list(M)$
shows
$sats(M,hcomp\_fm(pf,pg,a,w),env) ⟷ is\_hcomp(\#\#M,is\_f,is\_g,nth(a,env),nth(w,env))$
⟨*proof*⟩

definition
hcomp_r :: $[i⇒o,[i⇒o,i,i]⇒o,[i⇒o,i,i]⇒o,i,i] ⇒ o$ **where**
$hcomp\_r(M,is\_f,is\_g,a,w) ≡ ∃ z[M].\ is\_g(M,a,z) ∧ is\_f(M,z,w)$

definition
is_hcomp2_2 :: $[i⇒o,[i⇒o,i,i,i]⇒o,[i⇒o,i,i,i]⇒o,[i⇒o,i,i,i]⇒o,i,i,i] ⇒ o$ **where**
$is\_hcomp2\_2(M,is\_f,is\_g1,is\_g2,a,b,w) ≡ ∃ g1ab[M].\ ∃ g2ab[M].$
$is\_g1(M,a,b,g1ab) ∧ is\_g2(M,a,b,g2ab) ∧ is\_f(M,g1ab,g2ab,w)$

lemma (**in** *M_trivial*) *hcomp_abs*:
assumes
is_f_abs:$⋀a\ z.\ M(a) ⟹ M(z) ⟹ is\_f(M,a,z) ⟷ z = f(a)$ **and**
is_g_abs:$⋀a\ z.\ M(a) ⟹ M(z) ⟹ is\_g(M,a,z) ⟷ z = g(a)$ **and**
g_closed:$⋀a.\ M(a) ⟹ M(g(a))$
$M(a)\ M(w)$
shows
$hcomp\_r(M,is\_f,is\_g,a,w) ⟷ w = f(g(a))$
⟨*proof*⟩

lemma *hcomp_uniqueness*:
assumes
$uniq\_is\_f$:
$⋀r\ d\ d'.\ M(r) ⟹ M(d) ⟹ M(d') ⟹ is\_f(M,\ r,\ d) ⟹ is\_f(M,\ r,\ d') ⟹$
$d = d'$
and
$uniq\_is\_g$:
$⋀r\ d\ d'.\ M(r) ⟹ M(d) ⟹ M(d') ⟹ is\_g(M,\ r,\ d) ⟹ is\_g(M,\ r,\ d') ⟹$
$d = d'$
and
$M(a)\ M(w)\ M(w')$
$hcomp\_r(M,is\_f,is\_g,a,w)$

hcomp_r(*M*,*is_f*,*is_g*,*a*,*w*′)
shows
w=*w*′
⟨*proof*⟩

lemma *hcomp_witness*:
assumes
wit_is_f: ⋀*r*. *M*(*r*) ⟹ ∃ *d*[*M*]. *is_f*(*M*,*r*,*d*) **and**
wit_is_g: ⋀*r*. *M*(*r*) ⟹ ∃ *d*[*M*]. *is_g*(*M*,*r*,*d*) **and**
M(*a*)
shows
∃ *w*[*M*]. *hcomp_r*(*M*,*is_f*,*is_g*,*a*,*w*)
⟨*proof*⟩

lemma (**in** *M_trivial*) *hcomp2_2_abs*:
assumes
is_f_abs:⋀*r1 r2 z*. *M*(*r1*) ⟹ *M*(*r2*) ⟹ *M*(*z*) ⟹ *is_f*(*M*,*r1*,*r2*,*z*) ⟷ *z* = *f*(*r1*,*r2*) **and**
is_g1_abs:⋀*r1 r2 z*. *M*(*r1*) ⟹ *M*(*r2*) ⟹ *M*(*z*) ⟹ *is_g1*(*M*,*r1*,*r2*,*z*) ⟷ *z* = *g1*(*r1*,*r2*) **and**
is_g2_abs:⋀*r1 r2 z*. *M*(*r1*) ⟹ *M*(*r2*) ⟹ *M*(*z*) ⟹ *is_g2*(*M*,*r1*,*r2*,*z*) ⟷ *z* = *g2*(*r1*,*r2*) **and**
types: *M*(*a*) *M*(*b*) *M*(*w*) *M*(*g1*(*a*,*b*)) *M*(*g2*(*a*,*b*))
shows
is_hcomp2_2(*M*,*is_f*,*is_g1*,*is_g2*,*a*,*b*,*w*) ⟷ *w* = *f*(*g1*(*a*,*b*),*g2*(*a*,*b*))
⟨*proof*⟩

lemma *hcomp2_2_uniqueness*:
assumes
uniq_is_f:
⋀*r1 r2 d d*′. *M*(*r1*) ⟹ *M*(*r2*) ⟹ *M*(*d*) ⟹ *M*(*d*′) ⟹
is_f(*M*, *r1*, *r2* , *d*) ⟹ *is_f*(*M*, *r1*, *r2*, *d*′) ⟹ *d* = *d*′
and
uniq_is_g1:
⋀*r1 r2 d d*′. *M*(*r1*) ⟹ *M*(*r2*)⟹ *M*(*d*) ⟹ *M*(*d*′) ⟹ *is_g1*(*M*, *r1*,*r2*, *d*) ⟹ *is_g1*(*M*, *r1*,*r2*, *d*′) ⟹
d = *d*′
and
uniq_is_g2:
⋀*r1 r2 d d*′. *M*(*r1*) ⟹ *M*(*r2*)⟹ *M*(*d*) ⟹ *M*(*d*′) ⟹ *is_g2*(*M*, *r1*,*r2*, *d*) ⟹ *is_g2*(*M*, *r1*,*r2*, *d*′) ⟹
d = *d*′
and
M(*a*) *M*(*b*) *M*(*w*) *M*(*w*′)
is_hcomp2_2(*M*,*is_f*,*is_g1*,*is_g2*,*a*,*b*,*w*)
is_hcomp2_2(*M*,*is_f*,*is_g1*,*is_g2*,*a*,*b*,*w*′)
shows
w=*w*′
⟨*proof*⟩

lemma *hcomp2_2_witness*:
assumes
wit_is_f: $\bigwedge r1\ r2.\ M(r1) \Longrightarrow M(r2) \Longrightarrow \exists\, d[M].\ is\_f(M,r1,r2,d)$ **and**
wit_is_g1: $\bigwedge r1\ r2.\ M(r1) \Longrightarrow M(r2) \Longrightarrow \exists\, d[M].\ is\_g1(M,r1,r2,d)$ **and**
wit_is_g2: $\bigwedge r1\ r2.\ M(r1) \Longrightarrow M(r2) \Longrightarrow \exists\, d[M].\ is\_g2(M,r1,r2,d)$ **and**
$M(a)\ M(b)$
shows
$\exists\, w[M].\ is\_hcomp2\_2(M,is\_f,is\_g1,is\_g2,a,b,w)$
⟨*proof*⟩

lemma (**in** *M_trivial*) *extensionality_trans*:
assumes
$M(d) \wedge (\forall\, x[M].\ x{\in}d \longleftrightarrow P(x))$
$M(d') \wedge (\forall\, x[M].\ x{\in}d' \longleftrightarrow P(x))$
shows
$d{=}d'$
⟨*proof*⟩

definition
$lt\_rel :: [i{\Rightarrow}o,i,i] \Rightarrow o$ **where**
$lt\_rel(M,a,b) \equiv a{\in}b \wedge ordinal(M,b)$

lemma (**in** *M_trans*) *lt_abs*[*absolut*]: $M(a) \Longrightarrow M(b) \Longrightarrow lt\_rel(M,a,b) \longleftrightarrow a{<}b$
⟨*proof*⟩

definition
$le\_rel :: [i{\Rightarrow}o,i,i] \Rightarrow o$ **where**
$le\_rel(M,a,b) \equiv \exists\, sb[M].\ successor(M,b,sb) \wedge lt\_rel(M,a,sb)$

lemma (**in** *M_trivial*) *le_abs*[*absolut*]: $M(a) \Longrightarrow M(b) \Longrightarrow le\_rel(M,a,b) \longleftrightarrow a{\le}b$
⟨*proof*⟩

10.2 Discipline for *Pow*

definition
$is\_Pow :: [i{\Rightarrow}o,i,i] \Rightarrow o$ **where**
$is\_Pow(M,A,z) \equiv M(z) \wedge (\forall\, x[M].\ x \in z \longleftrightarrow subset(M,x,A))$

definition
$Pow\_rel :: [i{\Rightarrow}o,i] \Rightarrow i$ (‹$Pow{-}'(\_')$›) **where**
$Pow\_rel(M,r) \equiv THE\ d.\ is\_Pow(M,r,d)$

abbreviation
$Pow\_r\_set :: [i,i] \Rightarrow i$ (‹$Pow{-}'(\_')$›) **where**
$Pow\_r\_set(M) \equiv Pow\_rel(\#\#M)$

context *M_basic_no_repl*

begin

lemma *is_Pow_uniqueness*:
  assumes
    $M(r)$
    is_Pow(M,r,d) *is_Pow*(M,r,d')
  shows
    $d{=}d'$
  ⟨*proof*⟩

lemma *is_Pow_witness*: $M(r) \Longrightarrow \exists\, d[M].$ *is_Pow*(M,r,d)
  ⟨*proof*⟩

lemma *is_Pow_closed* : $[\![\ M(r);$*is_Pow*$(M,r,d)\ ]\!] \Longrightarrow M(d)$
  ⟨*proof*⟩

lemma *Pow_rel_closed*[*intro*,*simp*]: $M(r) \Longrightarrow M($*Pow_rel*$(M,r))$
  ⟨*proof*⟩

lemmas *trans_Pow_rel_closed*[*trans_closed*] = *transM*[*OF* _ *Pow_rel_closed*]

The proof of *f_rel_iff* lemma is schematic and it can reused by copy-paste replacing appropriately.

lemma *Pow_rel_iff*:
  assumes $M(r)$ $M(d)$
  shows *is_Pow*$(M,r,d) \longleftrightarrow d =$ *Pow_rel*(M,r)
⟨*proof*⟩

The next "def_" result really corresponds to $?A \in$ *Pow*$(?B) \longleftrightarrow ?A \subseteq ?B$

lemma *def_Pow_rel*: $M(A) \Longrightarrow M(r) \Longrightarrow A{\in}$*Pow_rel*$(M,r) \longleftrightarrow A \subseteq r$
  ⟨*proof*⟩

lemma *Pow_rel_char*: $M(r) \Longrightarrow$ *Pow_rel*$(M,r) = \{A{\in}$*Pow*$(r).\ M(A)\}$
⟨*proof*⟩

lemma *mem_Pow_rel_abs*: $M(a) \Longrightarrow M(r) \Longrightarrow a \in$ *Pow_rel*$(M,r) \longleftrightarrow a \in$ *Pow*(r)
  ⟨*proof*⟩

end — *M_basic_no_repl*

10.3 Discipline for *PiP*

definition
  PiP_rel:: $[i{\Rightarrow}o,i,i]{\Rightarrow}o$ **where**
  PiP_rel$(M,A,f) \equiv \exists\, df[M].$ *is_domain*$(M,f,df) \wedge$ *subset*$(M,A,df) \wedge$
                    is_function(M,f)

context *M_basic*

begin

lemma *def_PiP_rel*:
  assumes
    $M(A)$ $M(f)$
  shows
    $PiP\_rel(M,A,f) \longleftrightarrow A \subseteq domain(f) \wedge function(f)$
  ⟨*proof*⟩

end — *M_basic*

definition — FIX THIS: not completely relational. Can it be?
  Sigfun :: $[i,i\Rightarrow i]\Rightarrow i$ **where**
  $Sigfun(x,B) \equiv \bigcup y{\in}B(x).\ \{\langle x,y\rangle\}$

lemma *Sigma_Sigfun*: $Sigma(A,B) = \bigcup\ \{Sigfun(x,B)\ .\ x{\in}A\}$
  ⟨*proof*⟩

definition — FIX THIS: not completely relational. Can it be?
  is_Sigfun :: $[i\Rightarrow o,i,i\Rightarrow i,i]\Rightarrow o$ **where**
  $is\_Sigfun(M,x,B,Sd) \equiv M(Sd) \wedge (\exists\, RB[M].\ is\_Replace(M,B(x),\lambda y\ z.\ z{=}\{\langle x,y\rangle\},RB)$
                $\wedge\ big\_union(M,RB,Sd))$

context *M_trivial*
begin

lemma *is_Sigfun_abs*:
  assumes
    $strong\_replacement(M,\lambda y\ z.\ z{=}\{\langle x,y\rangle\})$
    $M(x)$ $M(B(x))$ $M(Sd)$
  shows
    $is\_Sigfun(M,x,B,Sd) \longleftrightarrow Sd = Sigfun(x,B)$
⟨*proof*⟩

lemma *Sigfun_closed*:
  assumes
    $strong\_replacement(M,\ \lambda y\ z.\ y \in B(x) \wedge z = \{\langle x,\ y\rangle\})$
    $M(x)$ $M(B(x))$
  shows
    $M(Sigfun(x,B))$
  ⟨*proof*⟩

lemmas *trans_Sigfun_closed*[*trans_closed*] = *transM*[*OF* _ *Sigfun_closed*]

end — *M_trivial*

definition
is_Sigma :: [*i*⇒*o*,*i*,*i*⇒*i*,*i*]⇒*o* **where**
is_Sigma(*M*,*A*,*B*,*S*) ≡ *M*(*S*) ∧ (∃ *RSf*[*M*].
is_Replace(*M*,*A*,λ*x z*. *z*=*Sigfun*(*x*,*B*),*RSf*) ∧ *big_union*(*M*,*RSf*,*S*))

locale *M_Pi* = *M_basic* +
assumes
Pi_separation: *M*(*A*) ⟹ *separation*(*M*, *PiP_rel*(*M*,*A*))
and
Pi_replacement:
M(*x*) ⟹ *M*(*y*) ⟹
strong_replacement(*M*, λ*ya z*. *ya* ∈ *y* ∧ *z* = {⟨*x*, *ya*⟩})
M(*y*) ⟹
strong_replacement(*M*, λ*x z*. *z* = (⋃ *xa*∈*y*. {⟨*x*, *xa*⟩}))

locale *M_Pi_assumptions* = *M_Pi* +
fixes *A B*
assumes
Pi_assumptions:
M(*A*)
⋀*x*. *x*∈*A* ⟹ *M*(*B*(*x*))
∀ *x*∈*A*. *strong_replacement*(*M*, λ*y z*. *y* ∈ *B*(*x*) ∧ *z* = {⟨*x*, *y*⟩})
strong_replacement(*M*,λ*x z*. *z*=*Sigfun*(*x*,*B*))
begin

lemma *Sigma_abs*[*simp*]:
assumes
M(*S*)
shows
is_Sigma(*M*,*A*,*B*,*S*) ⟷ *S* = *Sigma*(*A*,*B*)
⟨*proof*⟩

lemma *Sigma_closed*[*intro*,*simp*]: *M*(*Sigma*(*A*,*B*))
⟨*proof*⟩

lemmas *trans_Sigma_closed*[*trans_closed*] = *transM*[*OF* _ *Sigma_closed*]

end — *M_Pi_assumptions*

10.4 Discipline for *Pi*

definition
is_Pi :: [*i*⇒*o*,*i*,*i*⇒*i*,*i*]⇒*o* **where**
is_Pi(*M*,*A*,*B*,*I*) ≡ *M*(*I*) ∧ (∃ *S*[*M*]. ∃ *PS*[*M*]. *is_Sigma*(*M*,*A*,*B*,*S*) ∧
is_Pow(*M*,*S*,*PS*) ∧
is_Collect(*M*,*PS*,*PiP_rel*(*M*,*A*),*I*))

definition
$Pi\_rel :: [i \Rightarrow o, i, i \Rightarrow i] \Rightarrow i$ $(\langle Pi\text{–}'(\_,\_')\rangle)$ **where**
$Pi\_rel(M,A,B) \equiv THE\ d.\ is\_Pi(M,A,B,d)$

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

context *M_basic*
begin

lemmas $Pow\_rel\_iff = mbnr.Pow\_rel\_iff$
lemmas $Pow\_rel\_char = mbnr.Pow\_rel\_char$
lemmas $mem\_Pow\_rel\_abs = mbnr.mem\_Pow\_rel\_abs$
lemmas $Pow\_rel\_closed = mbnr.Pow\_rel\_closed$
lemmas $def\_Pow\_rel = mbnr.def\_Pow\_rel$
lemmas $trans\_Pow\_rel\_closed = mbnr.trans\_Pow\_rel\_closed$

end — *M_basic*

context *M_Pi_assumptions*
begin

lemma *is_Pi_uniqueness*:
assumes
$is\_Pi(M,A,B,d)$ $is\_Pi(M,A,B,d')$
shows
$d{=}d'$
⟨*proof*⟩

lemma *is_Pi_witness*: $\exists\, d[M].\ is\_Pi(M,A,B,d)$
⟨*proof*⟩

lemma *is_Pi_closed* : $is\_Pi(M,A,B,d) \Longrightarrow M(d)$
⟨*proof*⟩

lemma $Pi\_rel\_closed[intro,simp]$: $M(Pi\_rel(M,A,B))$
⟨*proof*⟩

lemmas $trans\_Pi\_rel\_closed[trans\_closed] = transM[OF\ \_\ Pi\_rel\_closed]$

lemma *Pi_rel_iff*:
assumes $M(d)$
shows $is\_Pi(M,A,B,d) \longleftrightarrow d = Pi\_rel(M,A,B)$
⟨*proof*⟩

lemma *def_Pi_rel*:

Pi_rel(*M*,*A*,*B*) = {*f*∈*Pow_rel*(*M*,*Sigma*(*A*,*B*)). *A*⊆*domain*(*f*) ∧ *function*(*f*)}
⟨*proof*⟩

lemma *Pi_rel_char*: *Pi_rel*(*M*,*A*,*B*) = {*f*∈*Pi*(*A*,*B*). *M*(*f*)}
⟨*proof*⟩

lemma *mem_Pi_rel_abs*:
assumes *M*(*f*)
shows *f* ∈ *Pi_rel*(*M*,*A*,*B*) ⟷ *f* ∈ *Pi*(*A*,*B*)
⟨*proof*⟩

end — *M_Pi_assumptions*

The next locale (and similar ones below) are used to show the relationship between versions of simple (i.e. Σ_1^{ZF}, Π_1^{ZF}) concepts in two different transitive models.

locale *M_N_Pi_assumptions* = *M*:*M_Pi_assumptions* + *N*:*M_Pi_assumptions*
N **for** *N* +
assumes
M_imp_N:*M*(*x*) ⟹ *N*(*x*)
begin

lemma *Pi_rel_transfer*: $Pi^M(A,B) \subseteq Pi^N(A,B)$
⟨*proof*⟩

end — *M_N_Pi_assumptions*

locale *M_Pi_assumptions_0* = *M_Pi_assumptions* _ *0*
begin

This is used in the proof of *AC_Pi_rel*

lemma *Pi_rel_empty1*[*simp*]: $Pi^M(0,B) = \{0\}$
⟨*proof*⟩

end — *M_Pi_assumptions_0*

context *M_Pi_assumptions*
begin

10.5 Auxiliary ported results on *Pi_rel*, now unused

lemma *Pi_rel_iff'*:
assumes *types*:*M*(*f*)
shows
f ∈ *Pi_rel*(*M*,*A*,*B*) ⟷ *function*(*f*) ∧ *f* ⊆ *Sigma*(*A*,*B*) ∧ *A* ⊆ *domain*(*f*)
⟨*proof*⟩

lemma *lam_type_M*:
  assumes $M(A)$ $\bigwedge x.\ x{\in}A \Longrightarrow\ M(B(x))$
    $\bigwedge x.\ x \in A \Longrightarrow b(x){\in}B(x)$ *strong_replacement*$(M,\lambda x\ y.\ y{=}\langle x,\ b(x)\rangle)$
  shows $(\lambda x{\in}A.\ b(x)) \in$ *Pi_rel*(M,A,B)
⟨*proof*⟩

end — *M_Pi_assumptions*

locale *M_Pi_assumptions2* = *M_Pi_assumptions* +
  PiC: *M_Pi_assumptions* _ _ *C* **for** *C*
begin

lemma *Pi_rel_type*:
  assumes $f \in Pi^M(A,C)$ $\bigwedge x.\ x \in A \Longrightarrow f`x \in B(x)$
    and *types*: $M(f)$
  shows $f \in Pi^M(A,B)$
  ⟨*proof*⟩

lemma *Pi_rel_weaken_type*:
  assumes $f \in Pi^M(A,B)$ $\bigwedge x.\ x \in A \Longrightarrow B(x) \subseteq C(x)$
    and *types*: $M(f)$
  shows $f \in Pi^M(A,C)$
  ⟨*proof*⟩

end — *M_Pi_assumptions2*

end

11 Arities of internalized formulas

theory *Arities*
  imports
    Discipline_Base
begin

lemmas *FOL_arities* [*simp del*, *arity*] = *arity_And arity_Or arity_Implies arity_Iff arity_Exists*

declare *pred_Un_distrib*[*arity_aux*]

context
  notes *FOL_arities*[*simp*]
begin

lemma *arity_upair_fm* [*arity*] : ⟦ *t1*∈*nat* ; *t2*∈*nat* ; *up*∈*nat* ⟧ ⟹
  arity(*upair_fm*(*t1*,*t2*,*up*)) = ⋃ {*succ*(*t1*),*succ*(*t2*),*succ*(*up*)}

⟨*proof*⟩

end

lemma *Un_trasposition_aux1*: $r \cup s \cup r = r \cup s$ ⟨*proof*⟩

lemma *Un_trasposition_aux2*:
$r \cup (s \cup (r \cup u)) = r \cup (s \cup u)$
$r \cup (s \cup (t \cup (r \cup u))) = r \cup (s \cup (t \cup u))$ ⟨*proof*⟩

Using the previous lemmas to guide the automatic arity calculation.

context
notes *Un_assoc*[*symmetric*,*simp*] *Un_trasposition_aux1*[*simp*]
begin

⟨*ML*⟩

end — context

context
notes *FOL_arities*[*simp*]
begin

lemma *arity_is_recfun_fm* [*arity*]:
$[\![p \in formula\ ;\ v \in nat\ ;\ n \in nat;\ Z \in nat; i \in nat]\!] \Longrightarrow\ arity(p) = i \Longrightarrow$
$arity(is\_recfun\_fm(p,v,n,Z)) = succ(v) \cup succ(n) \cup succ(Z) \cup pred(pred(pred(pred(i))))$
⟨*proof*⟩

lemma *arity_is_wfrec_fm* [*arity*]:
$[\![p \in formula\ ;\ v \in nat\ ;\ n \in nat;\ Z \in nat\ ;\ i \in nat]\!] \Longrightarrow arity(p) = i \Longrightarrow$
$arity(is\_wfrec\_fm(p,v,n,Z)) = succ(v) \cup succ(n) \cup succ(Z) \cup pred(pred(pred(pred(pred(i)))))$
⟨*proof*⟩

lemma *arity_is_nat_case_fm* [*arity*]:
$[\![p \in formula\ ;\ v \in nat\ ;\ n \in nat;\ Z \in nat;\ i \in nat]\!] \Longrightarrow arity(p) = i \Longrightarrow$
$arity(is\_nat\_case\_fm(v,p,n,Z)) = succ(v) \cup succ(n) \cup succ(Z) \cup pred(pred(i))$
⟨*proof*⟩

lemma *arity_iterates_MH_fm* [*arity*]:
assumes $isF \in formula\ v \in nat\ n \in nat\ g \in nat\ z \in nat\ i \in nat$
$arity(isF) = i$
shows $arity(iterates\_MH\_fm(isF,v,n,g,z)) =$
$succ(v) \cup succ(n) \cup succ(g) \cup succ(z) \cup pred(pred(pred(pred(i))))$
⟨*proof*⟩

lemma *arity_is_iterates_fm* [*arity*]:
assumes $p \in formula\ v \in nat\ n \in nat\ Z \in nat\ i \in nat$
$arity(p) = i$
shows $arity(is\_iterates\_fm(p,v,n,Z)) = succ(v) \cup succ(n) \cup succ(Z) \cup$

$pred(pred(pred(pred(pred(pred(pred(pred(pred(pred(i)))))))))))$
⟨*proof*⟩

lemma *arity_eclose_n_fm* [*arity*]:
assumes $A{\in}nat$ $x{\in}nat$ $t{\in}nat$
shows $arity(eclose\_n\_fm(A,x,t)) = succ(A) \cup succ(x) \cup succ(t)$
⟨*proof*⟩

lemma *arity_mem_eclose_fm* [*arity*]:
assumes $x{\in}nat$ $t{\in}nat$
shows $arity(mem\_eclose\_fm(x,t)) = succ(x) \cup succ(t)$
⟨*proof*⟩

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

lemma *arity_Collect_fm* [*arity*]:
assumes $x \in nat$ $y \in nat$ $p{\in}formula$
shows $arity(Collect\_fm(x,p,y)) = succ(x) \cup succ(y) \cup pred(arity(p))$
⟨*proof*⟩

schematic_goal *arity_least_fm'*:
assumes
$i \in nat$ $q \in formula$
shows
$arity(least\_fm(q,i)) \equiv ?ar$
⟨*proof*⟩

lemma *arity_least_fm* [*arity*]:
assumes
$i \in nat$ $q \in formula$
shows
$arity(least\_fm(q,i)) = succ(i) \cup pred(arity(q))$
⟨*proof*⟩

lemma *arity_Replace_fm* [*arity*]:
$[\![p{\in}formula\ ;\ v{\in}nat\ ;\ n{\in}nat;\ i{\in}nat]\!] \Longrightarrow arity(p) = i \Longrightarrow$
$arity(Replace\_fm(v,p,n)) = succ(n) \cup succ(v) \cup pred(pred(i))$
⟨*proof*⟩

lemma *arity_lambda_fm* [*arity*]:
$[\![p{\in}formula;\ v{\in}nat\ ;\ n{\in}nat;\ i{\in}nat]\!] \Longrightarrow\ arity(p) = i \Longrightarrow$
$arity(lambda\_fm(p,v,n)) = succ(n) \cup (succ(v) \cup (pred\hat{}3(i)))$
⟨*proof*⟩

lemma *arity_transrec_fm* [*arity*]:
$[\![p{\in}formula\ ;\ v{\in}nat\ ;\ n{\in}nat;\ i{\in}nat]\!] \Longrightarrow arity(p) = i \Longrightarrow$
$arity(is\_transrec\_fm(p,v,n)) = succ(v) \cup succ(n) \cup (pred\hat{}8(i))$

⟨*proof*⟩

lemma *arity_wfrec_replacement_fm* :
⟦*p*∈*formula* ; *v*∈*nat* ; *n*∈*nat*; *Z*∈*nat* ; *i*∈*nat*⟧ ⟹ *arity*(*p*) = *i* ⟹
arity(*Exists*(*And*(*pair_fm*(1,0,2),*is_wfrec_fm*(*p*,*v*,*n*,*Z*))))
= 2 ∪ *v* ∪ *n* ∪ *Z* ∪ (*pred*^6(*i*))
⟨*proof*⟩

end — *FOL_arities*

declare *arity_subset_fm* [*simp del*] *arity_ordinal_fm*[*simp del*, *arity*] *arity_transset_fm*[*simp del*]

context
notes *Un_assoc*[*symmetric*,*simp*] *Un_trasposition_aux1*[*simp*]
begin
⟨*ML*⟩
end

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

⟨*ML*⟩

end
theory *Discipline_Function*
imports
Arities
begin

Discipline for *fst* ⟨*ML*⟩

definition
is_fst :: (*i*⇒*o*)⇒*i*⇒*i*⇒*o* **where**
is_fst(*M*,*x*,*t*) ≡ (∃ *z*[*M*]. *pair*(*M*,*t*,*z*,*x*)) ∨
(¬(∃ *z*[*M*]. ∃ *w*[*M*]. *pair*(*M*,*w*,*z*,*x*)) ∧ *empty*(*M*,*t*))
⟨*ML*⟩
notation *fst_fm* (‹·*fst*′(_′) *is* _·›)

⟨*ML*⟩

definition *fst_rel* :: [*i*⇒*o*,*i*] ⇒ *i* **where**
fst_rel(*M*,*p*) ≡ *THE d*. *M*(*d*) ∧ *is_fst*(*M*,*p*,*d*)

⟨*ML*⟩

definition

is_snd :: $(i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$ **where**
is_snd$(M,x,t) \equiv (\exists\, z[M].\ pair(M,z,t,x)) \vee$
$(\neg(\exists\, z[M].\ \exists\, w[M].\ pair(M,z,w,x)) \wedge empty(M,t))$
⟨*ML*⟩
notation *snd_fm* (‹·*snd*′(_′) *is* _·›)
⟨*ML*⟩

definition *snd_rel* :: $[i \Rightarrow o, i] \Rightarrow i$ **where**
snd_rel$(M,p) \equiv$ *THE d*. $M(d) \wedge$ *is_snd*(M,p,d)

⟨*ML*⟩

context *M_trans*
begin

lemma *fst_snd_closed*:
assumes $M(p)$
shows $M(fst(p)) \wedge M(snd(p))$
⟨*proof*⟩

lemma *fst_closed*[*intro*,*simp*]: $M(x) \Longrightarrow M(fst(x))$
⟨*proof*⟩

lemma *snd_closed*[*intro*,*simp*]: $M(x) \Longrightarrow M(snd(x))$
⟨*proof*⟩

lemma *fst_abs* [*absolut*]:
$[\![M(p);\ M(x)\ ]\!] \Longrightarrow$ *is_fst*$(M,p,x) \longleftrightarrow x = fst(p)$
⟨*proof*⟩

lemma *snd_abs* [*absolut*]:
$[\![M(p);\ M(y)\ ]\!] \Longrightarrow$ *is_snd*$(M,p,y) \longleftrightarrow y = snd(p)$
⟨*proof*⟩

lemma *empty_rel_abs* : $M(x) \Longrightarrow M(0) \Longrightarrow x = 0 \longleftrightarrow x =$ (*THE d*. $M(d) \wedge$ *empty*$(M,\ d)$)
⟨*proof*⟩

lemma *fst_rel_abs*:
assumes $M(p)$
shows $fst(p) =$ *fst_rel*(M,p)
⟨*proof*⟩

lemma *snd_rel_abs*:
assumes $M(p)$
shows $snd(p) =$ *snd_rel*(M,p)
⟨*proof*⟩

end — *M_trans*

⟨*ML*⟩
context *M_trans*
begin

lemma *minimum_closed*[*simp*,*intro*]:
  assumes *M*(*A*)
  shows *M*(*minimum*(*r*,*A*))
  ⟨*proof*⟩

lemma *first_abs* :
  assumes *M*(*B*)
  shows *first*(*z*,*B*,*r*) ⟷ *first_rel*(*M*,*z*,*B*,*r*)
  ⟨*proof*⟩

lemma *minimum_abs*:
  assumes *M*(*B*)
  shows *minimum*(*r*,*B*) = *minimum_rel*(*M*,*r*,*B*)
⟨*proof*⟩

end — *M_trans*

11.1 Discipline for $\lambda A\ B.\ A \to B$

definition
  is_function_space :: [*i*⇒*o*,*i*,*i*,*i*] ⇒ *o* **where**
  is_function_space(*M*,*A*,*B*,*fs*) ≡ *M*(*fs*) ∧ *is_funspace*(*M*,*A*,*B*,*fs*)

definition
  function_space_rel :: [*i*⇒*o*,*i*,*i*] ⇒ *i* **where**
  function_space_rel(*M*,*A*,*B*) ≡ *THE d. is_function_space*(*M*,*A*,*B*,*d*)

⟨*ML*⟩

abbreviation
  function_space_r :: [*i*,*i*⇒*o*,*i*] ⇒ *i* (‹_ →⁻ _› [*61*,*1*,*61*] *60*) **where**
  A →^{M} *B* ≡ *function_space_rel*(*M*,*A*,*B*)

abbreviation
  function_space_r_set :: [*i*,*i*,*i*] ⇒ *i* (‹_ →⁻ _› [*61*,*1*,*61*] *60*) **where**
  function_space_r_set(*A*,*M*) ≡ *function_space_rel*(##*M*,*A*)

context *M_Pi*
begin

lemma *is_function_space_uniqueness*:
  assumes

$M(r)$ $M(B)$
$is\_function\_space(M,r,B,d)$ $is\_function\_space(M,r,B,d')$
shows
$d{=}d'$
⟨*proof*⟩

lemma *is_function_space_witness*:
assumes $M(A)$ $M(B)$
shows $\exists\, d[M].\ is\_function\_space(M,A,B,d)$
⟨*proof*⟩

lemma *is_function_space_closed* :
$is\_function\_space(M,A,B,d) \Longrightarrow M(d)$
⟨*proof*⟩
lemma *function_space_rel_closed*[*intro*,*simp*]:
assumes $M(x)$ $M(y)$
shows $M(function\_space\_rel(M,x,y))$
⟨*proof*⟩

lemmas $trans\_function\_space\_rel\_closed[trans\_closed] = transM[OF\_\ function\_space\_rel\_closed]$

lemma *is_function_space_iff*:
assumes $M(x)$ $M(y)$ $M(d)$
shows $is\_function\_space(M,x,y,d) \longleftrightarrow d = function\_space\_rel(M,x,y)$
⟨*proof*⟩

lemma *def_function_space_rel*:
assumes $M(A)$ $M(y)$
shows $function\_space\_rel(M,A,y) = Pi\_rel(M,A,\lambda\_.\ y)$
⟨*proof*⟩

lemma *function_space_rel_char*:
assumes $M(A)$ $M(y)$
shows $function\_space\_rel(M,A,y) = \{f \in A \to y.\ M(f)\}$
⟨*proof*⟩

lemma *mem_function_space_rel_abs*:
assumes $M(A)$ $M(y)$ $M(f)$
shows $f \in function\_space\_rel(M,A,y) \longleftrightarrow\ f \in A \to y$
⟨*proof*⟩

end — *M_Pi*

locale $M\_N\_Pi = M{:}M\_Pi + N{:}M\_Pi\ N$ **for** N +
assumes
$M\_imp\_N{:}M(x) \Longrightarrow N(x)$
begin

lemma *function_space_rel_transfer*: $M(A) \Longrightarrow M(B) \Longrightarrow$
function_space_rel(M,A,B) $\subseteq$ *function_space_rel*(N,A,B)
⟨*proof*⟩

end — *M_N_Pi*

abbreviation
is_apply ≡ *fun_apply*
— It is not necessary to perform the Discipline for *is_apply* since it is absolute in this context

11.2 Discipline for *Collect* terms.

We have to isolate the predicate involved and apply the Discipline to it.

definition
injP_rel:: [i⇒o,i,i]⇒o **where**
injP_rel(M,A,f) ≡ ∀ w[M]. ∀ x[M]. ∀ fw[M]. ∀ fx[M]. w∈A ∧ x∈A ∧
is_apply(M,f,w,fw) ∧ *is_apply*(M,f,x,fx) ∧ fw=fx⟶ w=x

⟨*ML*⟩

context *M_basic*
begin

— I'm undecided on keeping the relative quantifiers here. Same with *surjP* below. It might relieve from changing *?P*(*?x*) ⟹ ∃ x. *?P*(x)
(⋀x. *?P*(x)) ⟹ ∀ x. *?P*(x) to ⟦*?P*(*?x*); *?M*(*?x*)⟧ ⟹ ∃ x[*?M*]. *?P*(x)
(⋀x. *?M*(x) ⟹ *?P*(x)) ⟹ ∀ x[*?M*]. *?P*(x) in some proofs. I wonder if this escalates well. Assuming that all terms appearing in the "def_" theorem are in M and using ⟦*?y* ∈ *?x*; *M*(*?x*)⟧ ⟹ *M*(*?y*), it might do.
lemma *def_injP_rel*:
assumes
M(A) *M*(f)
shows
injP_rel(M,A,f) ⟷ (∀ w[M]. ∀ x[M]. w∈A ∧ x∈A ∧ f‘w=f‘x ⟶ w=x)
⟨*proof*⟩

end — *M_basic*

11.3 Discipline for *inj*

definition
is_inj :: [i⇒o,i,i,i]⇒o **where**
is_inj(M,A,B,I) ≡ M(I) ∧ (∃ F[M]. *is_function_space*(M,A,B,F) ∧
is_Collect(M,F,*injP_rel*(M,A),I))

declare *typed_function_iff_sats Collect_iff_sats* [*iff_sats*]

⟨*ML*⟩
notation *is_function_space_fm* (‹·_ → _ *is* _·›)

⟨*ML*⟩
notation *is_inj_fm* (‹·*inj*′(_,_′) *is* _·›)

⟨*ML*⟩

lemma *arity_is_inj_fm*[*arity*]:
  A ∈ *nat* ⟹
    B ∈ *nat* ⟹ *I* ∈ *nat* ⟹ *arity*(*is_inj_fm*(*A*, *B*, *I*)) = *succ*(*A*) ∪ *succ*(*B*) ∪ *succ*(*I*)
  ⟨*proof*⟩

definition
  inj_rel :: [*i*⇒*o*,*i*,*i*] ⇒ *i* (‹*inj*–′(_,_′)›) **where**
  inj_rel(*M*,*A*,*B*) ≡ *THE d*. *is_inj*(*M*,*A*,*B*,*d*)

abbreviation
  inj_r_set :: [*i*,*i*,*i*] ⇒ *i* (‹*inj*–′(_,_′)›) **where**
  inj_r_set(*M*) ≡ *inj_rel*(##*M*)

locale *M_inj* = *M_Pi* +
  assumes
    injP_separation: *M*(*r*) ⟹ *separation*(*M*,*injP_rel*(*M*, *r*))
begin

lemma *is_inj_uniqueness*:
  assumes
    M(*r*) *M*(*B*)
    is_inj(*M*,*r*,*B*,*d*) *is_inj*(*M*,*r*,*B*,*d*′)
  shows
    d=*d*′
  ⟨*proof*⟩

lemma *is_inj_witness*: *M*(*r*) ⟹ *M*(*B*)⟹ ∃ *d*[*M*]. *is_inj*(*M*,*r*,*B*,*d*)
  ⟨*proof*⟩

lemma *is_inj_closed* :
  is_inj(*M*,*x*,*y*,*d*) ⟹ *M*(*d*)
  ⟨*proof*⟩

lemma *inj_rel_closed*[*intro*,*simp*]:
  assumes *M*(*x*) *M*(*y*)
  shows *M*(*inj_rel*(*M*,*x*,*y*))
⟨*proof*⟩

lemmas *trans_inj_rel_closed*[*trans_closed*] = *transM*[*OF* _ *inj_rel_closed*]

lemma *inj_rel_iff*:
  assumes *M*(*x*) *M*(*y*) *M*(*d*)
  shows *is_inj*(*M*,*x*,*y*,*d*) ⟷ *d* = *inj_rel*(*M*,*x*,*y*)
⟨*proof*⟩

lemma *def_inj_rel*:
  assumes *M*(*A*) *M*(*B*)
  shows *inj_rel*(*M*,*A*,*B*) =
        {*f* ∈ *function_space_rel*(*M*,*A*,*B*). ∀ *w*[*M*]. ∀ *x*[*M*]. *w*∈*A* ∧ *x*∈*A* ∧ *f*‘*w* = *f*‘*x* ⟶ *w*=*x*}
    (**is** _ = *Collect*(_,*?P*))
⟨*proof*⟩

lemma *inj_rel_char*:
  assumes *M*(*A*) *M*(*B*)
  shows *inj_rel*(*M*,*A*,*B*) = {*f* ∈ *inj*(*A*,*B*). *M*(*f*)}
⟨*proof*⟩

end — *M_inj*

locale *M_N_inj* = *M*:*M_inj* + *N*:*M_inj N* **for** *N* +
  assumes
    M_imp_N:*M*(*x*) ⟹ *N*(*x*)
begin

lemma *inj_rel_transfer*: *M*(*A*) ⟹ *M*(*B*) ⟹ *inj_rel*(*M*,*A*,*B*) ⊆ *inj_rel*(*N*,*A*,*B*)
  ⟨*proof*⟩

end — *M_N_inj*

definition
  surjP_rel:: [*i*⇒*o*,*i*,*i*,*i*]⇒*o* **where**
  surjP_rel(*M*,*A*,*B*,*f*) ≡
    ∀ *y*[*M*]. ∃ *x*[*M*]. ∃ *fx*[*M*]. *y*∈*B* ⟶ *x*∈*A* ∧ *is_apply*(*M*,*f*,*x*,*fx*) ∧ *fx*=*y*

⟨*ML*⟩

context *M_basic*
begin

lemma *def_surjP_rel*:

assumes
$M(A)$ $M(B)$ $M(f)$
shows
$surjP\_rel(M,A,B,f) \longleftrightarrow (\forall\, y[M].\ \exists\, x[M].\ y{\in}B \longrightarrow x{\in}A \wedge f`x{=}y)$
⟨*proof*⟩

end — *M_basic*

11.4 Discipline for *surj*

definition
$is\_surj :: [i{\Rightarrow}o,i,i,i]{\Rightarrow}o$ **where**
$is\_surj(M,A,B,I) \equiv M(I) \wedge (\exists\, F[M].\ is\_function\_space(M,A,B,F) \wedge$
$is\_Collect(M,F,surjP\_rel(M,A,B),I))$

⟨*ML*⟩
notation is_surj_fm $(\langle\cdot surj'(\_,\_')\ is\ \_\cdot\rangle)$

definition
$surj\_rel :: [i{\Rightarrow}o,i,i] \Rightarrow i$ $(\langle surj{-}'(\_,\_')\rangle)$ **where**
$surj\_rel(M,A,B) \equiv THE\ d.\ is\_surj(M,A,B,d)$

abbreviation
$surj\_r\_set :: [i,i,i] \Rightarrow i$ $(\langle surj{-}'(\_,\_')\rangle)$ **where**
$surj\_r\_set(M) \equiv surj\_rel(\#\#M)$

locale $M\_surj = M\_Pi +$
assumes
$surjP\_separation$: $M(A){\Longrightarrow}M(B){\Longrightarrow}separation(M,\lambda x.\ surjP\_rel(M,A,B,x))$
begin

lemma *is_surj_uniqueness*:
assumes
$M(r)$ $M(B)$
$is\_surj(M,r,B,d)$ $is\_surj(M,r,B,d')$
shows
$d{=}d'$
⟨*proof*⟩

lemma *is_surj_witness*: $M(r) \Longrightarrow M(B){\Longrightarrow} \exists\, d[M].\ is\_surj(M,r,B,d)$
⟨*proof*⟩

lemma *is_surj_closed* :
$is\_surj(M,x,y,d) \Longrightarrow M(d)$
⟨*proof*⟩

lemma *surj_rel_closed*[*intro*,*simp*]:
assumes $M(x)$ $M(y)$
shows $M(surj\_rel(M,x,y))$

⟨*proof*⟩

lemmas *trans_surj_rel_closed*[*trans_closed*] = *transM*[*OF* _ *surj_rel_closed*]

lemma *surj_rel_iff*:
  assumes *M*(*x*) *M*(*y*) *M*(*d*)
  shows *is_surj*(*M*,*x*,*y*,*d*) ⟷ *d* = *surj_rel*(*M*,*x*,*y*)
⟨*proof*⟩

lemma *def_surj_rel*:
  assumes *M*(*A*) *M*(*B*)
  shows *surj_rel*(*M*,*A*,*B*) =
        {*f* ∈ *function_space_rel*(*M*,*A*,*B*). ∀ *y*[*M*]. ∃ *x*[*M*]. *y*∈*B* ⟶ *x*∈*A* ∧ *f*‘*x*=*y* }
    (**is** _ = *Collect*(_,*?P*))
⟨*proof*⟩

lemma *surj_rel_char*:
  assumes *M*(*A*) *M*(*B*)
  shows *surj_rel*(*M*,*A*,*B*) = {*f* ∈ *surj*(*A*,*B*). *M*(*f*)}
⟨*proof*⟩

end — *M_surj*

locale *M_N_surj* = *M*:*M_surj* + *N*:*M_surj N* **for** *N* +
  assumes
    M_imp_N:*M*(*x*) ⟹ *N*(*x*)
begin

lemma *surj_rel_transfer*: *M*(*A*) ⟹ *M*(*B*) ⟹ *surj_rel*(*M*,*A*,*B*) ⊆ *surj_rel*(*N*,*A*,*B*)
  ⟨*proof*⟩

end — *M_N_surj*

definition
  is_Int :: [*i*⇒*o*,*i*,*i*,*i*]⇒*o* **where**
  is_Int(*M*,*A*,*B*,*I*) ≡ *M*(*I*) ∧ (∀ *x*[*M*]. *x* ∈ *I* ⟷ *x* ∈ *A* ∧ *x* ∈ *B*)

⟨*ML*⟩
notation *is_Int_fm* (‹_ ∩ _ *is* _›)

context *M_basic*
begin

lemma *is_Int_closed* :
  is_Int(*M*,*A*,*B*,*I*) ⟹ *M*(*I*)
  ⟨*proof*⟩

lemma *is_Int_abs*:
  assumes
    $M(A)$ $M(B)$ $M(I)$
  shows
    $is\_Int(M,A,B,I) \longleftrightarrow I = A \cap B$
  ⟨*proof*⟩

lemma *is_Int_uniqueness*:
  assumes
    $M(r)$ $M(B)$
    $is\_Int(M,r,B,d)$ $is\_Int(M,r,B,d')$
  shows
    $d{=}d'$
⟨*proof*⟩

Note: $[\![M(?A); M(?B)]\!] \Longrightarrow M(?A \cap ?B)$ already in *ZF-Constructible.Relative*.

end — *M_basic*

11.5 Discipline for *bij*

⟨*ML*⟩
notation *is_bij_fm* (‹·*bij*′(_,_′) *is* _·›)

abbreviation
  bij_r_class :: $[i{\Rightarrow}o,i,i] \Rightarrow i$ (‹*bij*–′(_,_′)›) **where**
  $bij\_r\_class \equiv bij\_rel$

abbreviation
  bij_r_set :: $[i,i,i] \Rightarrow i$ (‹*bij*–′(_,_′)›) **where**
  $bij\_r\_set(M) \equiv bij\_rel(\#\#M)$

locale $M\_Perm = M\_Pi + M\_inj + M\_surj$
begin

lemma $is\_bij\_closed : is\_bij(M,f,y,d) \Longrightarrow M(d)$
  ⟨*proof*⟩

lemma *bij_rel_closed*[*intro*,*simp*]:
  assumes $M(x)$ $M(y)$
  shows $M(bij\_rel(M,x,y))$
  ⟨*proof*⟩

lemmas $trans\_bij\_rel\_closed[trans\_closed] = transM[OF \_ bij\_rel\_closed]$

lemma *bij_rel_iff*:
  assumes $M(x)$ $M(y)$ $M(d)$
  shows $is\_bij(M,x,y,d) \longleftrightarrow d = bij\_rel(M,x,y)$
  ⟨*proof*⟩

lemma *def_bij_rel*:
assumes $M(A)$ $M(B)$
shows *bij_rel*(M,A,B) = *inj_rel*(M,A,B) ∩ *surj_rel*(M,A,B)
⟨*proof*⟩

lemma *bij_rel_char*:
assumes $M(A)$ $M(B)$
shows *bij_rel*(M,A,B) = {f ∈ *bij*(A,B). $M(f)$}
⟨*proof*⟩

end — *M_Perm*

locale *M_N_Perm* = *M_N_Pi* + *M_N_inj* + *M_N_surj* + M:*M_Perm* + N:*M_Perm* N

begin

lemma *bij_rel_transfer*: $M(A) \Longrightarrow M(B) \Longrightarrow$ *bij_rel*(M,A,B) ⊆ *bij_rel*(N,A,B)
⟨*proof*⟩

end — *M_N_Perm*

11.6 Discipline for (≈)

⟨*ML*⟩
notation *is_eqpoll_fm* (‹·_ ≈ _·›)

context *M_Perm* **begin**

⟨*ML*⟩
⟨*proof*⟩

end — *M_Perm*

abbreviation
eqpoll_r :: [i,i⇒o,i] => o (‹_ ≈[−] _› [51,1,51] 50) **where**
$A \approx^M B \equiv$ *eqpoll_rel*(M,A,B)

abbreviation
eqpoll_r_set :: [i,i,i] ⇒ o (‹_ ≈[−] _› [51,1,51] 50) **where**
eqpoll_r_set(A,M) ≡ *eqpoll_rel*(##M,A)

context *M_Perm*
begin

lemma *def_eqpoll_rel*:
assumes
$M(A)$ $M(B)$
shows

eqpoll_rel$(M,A,B) \longleftrightarrow (\exists f[M].\ f \in$ *bij_rel*$(M,A,B))$
⟨*proof*⟩

end — *M_Perm*

context *M_N_Perm*
begin

lemma *eqpoll_rel_transfer*: **assumes** $A \approx^M B$ $M(A)$ $M(B)$
shows $A \approx^N B$
⟨*proof*⟩

end — *M_N_Perm*

11.7 Discipline for $(\lesssim)$

⟨*ML*⟩
notation *is_lepoll_fm* $(\langle\cdot\_ \lesssim \_\cdot\rangle)$
⟨*ML*⟩

context *M_inj* **begin**

⟨*ML*⟩
⟨*proof*⟩

end — *M_inj*

abbreviation
lepoll_r :: $[i,i\Rightarrow o,i] \Rightarrow o$ $(\langle\_ \lesssim^{-} \_\rangle\ [51,1,51]\ 50)$ **where**
$A \lesssim^M B \equiv$ *lepoll_rel*(M,A,B)

abbreviation
lepoll_r_set :: $[i,i,i] \Rightarrow o$ $(\langle\_ \lesssim^{-} \_\rangle\ [51,1,51]\ 50)$ **where**
lepoll_r_set$(A,M) \equiv$ *lepoll_rel*$(\#\#M,A)$

context *M_Perm*
begin

lemma *def_lepoll_rel*:
assumes
$M(A)$ $M(B)$
shows
lepoll_rel$(M,A,B) \longleftrightarrow (\exists f[M].\ f \in$ *inj_rel*$(M,A,B))$
⟨*proof*⟩

end — *M_Perm*

context *M_N_Perm*

begin

lemma *lepoll_rel_transfer*: **assumes** $A \lesssim^M B$ $M(A)$ $M(B)$
shows $A \lesssim^N B$
⟨*proof*⟩

end — *M_N_Perm*

11.8 Discipline for $(\prec)$

⟨*ML*⟩
notation *is_lesspoll_fm* $(\langle\cdot\_ \prec \_\cdot\rangle)$
⟨*ML*⟩

context *M_Perm* **begin**

⟨*ML*⟩
⟨*proof*⟩

end — *M_Perm*

abbreviation
lesspoll_r :: $[i,i\Rightarrow o,i] \Rightarrow o$ $(\langle\_ \prec\!\!\!-\ \_\rangle\ [51,1,51]\ 50)$ **where**
$A \prec^M B \equiv$ *lesspoll_rel*(M,A,B)

abbreviation
lesspoll_r_set :: $[i,i,i] \Rightarrow o$ $(\langle\_ \prec\!\!\!-\ \_\rangle\ [51,1,51]\ 50)$ **where**
lesspoll_r_set$(A,M) \equiv$ *lesspoll_rel*$(\#\#M,A)$

Since *lesspoll_rel* is defined as a propositional combination of older terms, there is no need for a separate "def" theorem for it.

Note that *lesspoll_rel* is neither Σ_1^{ZF} nor Π_1^{ZF}, so there is no "transfer" theorem for it.

end

12 Replacements using Lambdas

theory *Lambda_Replacement*
imports
Discipline_Function
begin

In this theory we prove several instances of separation and replacement in *M_basic*. Moreover we introduce a new locale assuming two instances of separation and twelve instances of lambda replacements (ie, replacement of the form $\lambda xy.y = \langle x, f(x)\rangle$) we prove a bunch of other instances.

definition
lam_replacement :: $[i \Rightarrow o, i \Rightarrow i] \Rightarrow o$ **where**
lam_replacement$(M,b) \equiv$ *strong_replacement*$(M,\ \lambda x\ y.\ y = \langle x,\ b(x)\rangle)$

lemma *separation_univ* :
shows *separation*(M,M)
⟨*proof*⟩

context *M_basic*
begin

lemma *separation_iff'*:
assumes *separation*$(M,\lambda x\ .\ P(x))$ *separation*$(M,\lambda x\ .\ Q(x))$
shows *separation*$(M,\lambda x\ .\ P(x) \longleftrightarrow Q(x))$
⟨*proof*⟩

lemma *separation_in_constant* :
assumes $M(a)$
shows *separation*$(M,\lambda x\ .\ x{\in}a)$
⟨*proof*⟩

lemma *separation_equal* :
shows *separation*$(M,\lambda x\ .\ x{=}a)$
⟨*proof*⟩

lemma (**in** *M_basic*) *separation_in_rev*:
assumes $(M)(a)$
shows *separation*$(M,\lambda x\ .\ a{\in}x)$
⟨*proof*⟩

lemma *lam_replacement_iff_lam_closed*:
assumes $\forall\, x[M].\ M(b(x))$
shows *lam_replacement*$(M,\ b) \longleftrightarrow\ (\forall\, A[M].\ M(\lambda x{\in}A.\ b(x)))$
⟨*proof*⟩

lemma *lam_replacement_imp_lam_closed*:
assumes *lam_replacement*$(M,\ b)$ $M(A)$ $\forall\, x{\in}A.\ M(b(x))$
shows $M(\lambda x{\in}A.\ b(x))$
⟨*proof*⟩

lemma *lam_replacement_cong*:
assumes *lam_replacement*(M,f) $\forall\, x[M].\ f(x) = g(x)$ $\forall\, x[M].\ M(f(x))$
shows *lam_replacement*(M,g)
⟨*proof*⟩

lemma *converse_subset* : *converse*$(r) \subseteq \{\langle snd(x),fst(x)\rangle\ .\ x{\in}r\}$
⟨*proof*⟩

lemma *converse_eq_aux* :

assumes $<0,0>\in r$
shows $converse(r) = \{\langle snd(x),fst(x)\rangle \ .\ x\in r\}$
⟨*proof*⟩

lemma *converse_eq_aux'* :
assumes $<0,0>\notin r$
shows $converse(r) = \{\langle snd(x),fst(x)\rangle \ .\ x\in r\}$ - $\{<0,0>\}$
⟨*proof*⟩

lemma *diff_un* : $b\subseteq a \Longrightarrow (a\text{-}b) \cup b = a$
⟨*proof*⟩

lemma *converse_eq*: $converse(r) = (\{\langle snd(x),fst(x)\rangle \ .\ x\in r\}$ - $\{<0,0>\}) \cup (r\cap\{<0,0>\})$
⟨*proof*⟩

lemma *range_subset* : $range(r) \subseteq \{snd(x).\ x\in r\}$
⟨*proof*⟩

lemma *lam_replacement_imp_strong_replacement_aux*:
assumes $lam\_replacement(M,\ b)\ \forall x[M].\ M(b(x))$
shows $strong\_replacement(M,\ \lambda x\ y.\ y = b(x))$
⟨*proof*⟩

lemma *lam_replacement_imp_RepFun_Lam*:
assumes $lam\_replacement(M,\ f)\ M(A)$
shows $M(\{y\ .\ x\in A\ ,\ M(y) \wedge y=\langle x,f(x)\rangle\})$
⟨*proof*⟩

lemma *lam_closed_imp_closed*:
assumes $\forall A[M].\ M(\lambda x\in A.\ f(x))$
shows $\forall x[M].\ M(f(x))$
⟨*proof*⟩

lemma *lam_replacement_if*:
assumes $lam\_replacement(M,f)\ lam\_replacement(M,g)\ separation(M,b)$
$\forall x[M].\ M(f(x))\ \forall x[M].\ M(g(x))$
shows $lam\_replacement(M,\ \lambda x.\ if\ b(x)\ then\ f(x)\ else\ g(x))$
⟨*proof*⟩

lemma *lam_replacement_constant*: $M(b) \Longrightarrow lam\_replacement(M,\lambda\_.\ b)$
⟨*proof*⟩

12.1 Replacement instances obtained through Powerset

The next few lemmas provide bounds for certain constructions.

lemma *not_functional_Replace_0*:
assumes $\neg(\forall\ y\ y'.\ P(y) \wedge P(y') \longrightarrow y=y')$
shows $\{y\ .\ x\in A,\ P(y)\} = 0$
⟨*proof*⟩

lemma *Replace_in_Pow_rel*:
assumes $\bigwedge x\ b.\ x \in A \Longrightarrow P(x,b) \Longrightarrow b \in U\ \forall x{\in}A.\ \forall y\ y'.\ P(x,y) \wedge P(x,y') \longrightarrow y{=}y'$
separation$(M,\ \lambda y.\ \exists x[M].\ x \in A \wedge P(x,\ y))$
$M(U)\ M(A)$
shows $\{y\ .\ x \in A,\ P(x,\ y)\} \in Pow^M(U)$
⟨*proof*⟩

lemma *Replace_sing_0_in_Pow_rel*:
assumes $\bigwedge b.\ P(b) \Longrightarrow b \in U$
separation$(M,\ \lambda y.\ P(y))\ M(U)$
shows $\{y\ .\ x \in \{0\},\ P(y)\} \in Pow^M(U)$
⟨*proof*⟩

lemma *The_in_Pow_rel_Union*:
assumes $\bigwedge b.\ P(b) \Longrightarrow b \in U$ *separation*$(M,\ \lambda y.\ P(y))\ M(U)$
shows $(THE\ i.\ P(i)) \in Pow^M(\bigcup U)$
⟨*proof*⟩

lemma *separation_least*: *separation*$(M,\ \lambda y.\ Ord(y) \wedge P(y) \wedge (\forall j.\ j < y \longrightarrow \neg P(j)))$
⟨*proof*⟩

lemma *Least_in_Pow_rel_Union*:
assumes $\bigwedge b.\ P(b) \Longrightarrow b \in U$
$M(U)$
shows $(\mu\ i.\ P(i)) \in Pow^M(\bigcup U)$
⟨*proof*⟩

lemma *bounded_lam_replacement*:
fixes U
assumes $\forall X[M].\ \forall x{\in}X.\ f(x) \in U(X)$
and *separation_f*:$\forall A[M].$ *separation*$(M,\lambda y.\ \exists x[M].\ x{\in}A \wedge y = \langle x,\ f(x)\rangle)$
and *U_closed* [*intro*,*simp*]: $\bigwedge X.\ M(X) \Longrightarrow M(U(X))$
shows *lam_replacement*$(M,\ f)$
⟨*proof*⟩

lemma *lam_replacement_domain′*:
assumes $\forall A[M].$ *separation*$(M,\ \lambda y.\ \exists x{\in}A.\ y = \langle x,\ domain(x)\rangle)$
shows *lam_replacement*$(M,domain)$
⟨*proof*⟩
lemma *lam_replacement_fst′*:
assumes $\forall A[M].$ *separation*$(M,\ \lambda y.\ \exists x{\in}A.\ y = \langle x,\ fst(x)\rangle)$
shows *lam_replacement*(M,fst)
⟨*proof*⟩

lemma *lam_replacement_restrict*:
assumes $\forall A[M].$ *separation*$(M,\ \lambda y.\ \exists x{\in}A.\ y = \langle x,\ restrict(x,B)\rangle)\ \ M(B)$

shows *lam_replacement*(M, λr . *restrict*(r,B))
⟨*proof*⟩

end — *M_basic*

locale *M_replacement* = *M_basic* +
assumes
lam_replacement_domain: *lam_replacement*(M,*domain*)
and
lam_replacement_fst: *lam_replacement*(M,*fst*)
and
lam_replacement_snd: *lam_replacement*(M,*snd*)
and
lam_replacement_Union: *lam_replacement*(M,*Union*)
and
middle_del_replacement: *strong_replacement*(M, $\lambda x\ y.\ y$=⟨*fst*(*fst*(x)),*snd*(*snd*(x))⟩)
and
product_replacement:
strong_replacement(M, $\lambda x\ y.\ y$=⟨*snd*(*fst*(x)),⟨*fst*(*fst*(x)),*snd*(*snd*(x))⟩⟩)
and
lam_replacement_Upair:*lam_replacement*(M, $\lambda p.$ *Upair*(*fst*(p),*snd*(p)))
and
lam_replacement_Diff:*lam_replacement*(M, $\lambda p.$ *fst*(p) - *snd*(p))
and
lam_replacement_Image:*lam_replacement*(M, $\lambda p.$ *fst*(p) `` *snd*(p))
and
middle_separation: *separation*(M, $\lambda x.$ *snd*(*fst*(x))=*fst*(*snd*(x)))
and
separation_fst_in_snd: *separation*(M, $\lambda y.$ *fst*(*snd*(y)) ∈ *snd*(*snd*(y)))
and
lam_replacement_converse : *lam_replacement*(M,*converse*)
and
lam_replacement_comp: *lam_replacement*(M, $\lambda x.$ *fst*(x) *O* *snd*(x))
begin

lemma *lam_replacement_imp_strong_replacement*:
assumes *lam_replacement*(M, f)
shows *strong_replacement*(M, $\lambda x\ y.\ y = f(x)$)
⟨*proof*⟩

lemma *Collect_middle*: {p ∈ (λx∈A. $f(x)$) × (λx∈{$f(x)$. x∈A}. $g(x)$) . *snd*(*fst*(p))=*fst*(*snd*(p))}
= { ⟨x,$f(x)$⟩,⟨$f(x)$,$g(f(x))$⟩ . x∈A }
⟨*proof*⟩

lemma *RepFun_middle_del*: { ⟨*fst*(*fst*(p)),*snd*(*snd*(p))⟩ . p ∈ { ⟨⟨x,$f(x)$⟩,⟨$f(x)$,$g(f(x))$⟩⟩
. x∈A }}
= { ⟨x,$g(f(x))$⟩ . x∈A }
⟨*proof*⟩

lemma *lam_replacement_imp_RepFun*:
  assumes *lam_replacement*(M, f) $M(A)$
  shows $M(\{y\ .\ x{\in}A\ ,\ M(y) \wedge y{=}f(x)\})$
⟨*proof*⟩

lemma *lam_replacement_product*:
  assumes *lam_replacement*(M,f) *lam_replacement*(M,g)
  shows *lam_replacement*(M, $\lambda x.\ \langle f(x),g(x)\rangle$)
⟨*proof*⟩

lemma *lam_replacement_hcomp*:
  assumes *lam_replacement*(M,f) *lam_replacement*(M,g) $\forall\, x[M].\ M(f(x))$
  shows *lam_replacement*(M, $\lambda x.\ g(f(x))$)
⟨*proof*⟩

lemma *lam_replacement_Collect* :
  assumes $M(A)$ $\forall\, x[M].$ *separation*(M,$F(x)$)
    separation(M,$\lambda p\ .\ \forall\, x{\in}A.\ x{\in}snd(p) \longleftrightarrow F(fst(p),x)$)
  shows *lam_replacement*(M,$\lambda x.\ \{y{\in}A\ .\ F(x,y)\}$)
⟨*proof*⟩

lemma *lam_replacement_hcomp2*:
  assumes *lam_replacement*(M,f) *lam_replacement*(M,g)
    $\forall\, x[M].\ M(f(x))$ $\forall\, x[M].\ M(g(x))$
    lam_replacement(M, $\lambda p.\ h(fst(p),snd(p))$)
    $\forall\, x[M].\ \forall\, y[M].\ M(h(x,y))$
  shows *lam_replacement*(M, $\lambda x.\ h(f(x),g(x))$)
  ⟨*proof*⟩

lemma *lam_replacement_identity*: *lam_replacement*(M,$\lambda x.\ x$)
⟨*proof*⟩

lemma *lam_replacement_vimage* :
  shows *lam_replacement*(M, $\lambda x.\ fst(x)\text{-}``snd(x)$)
  ⟨*proof*⟩

lemma *strong_replacement_separation_aux* :
  assumes *strong_replacement*(M,$\lambda\ x\ y\ .\ y{=}f(x)$) *separation*($M$,$P$)
  shows *strong_replacement*(M, $\lambda x\ y\ .\ P(x) \wedge y{=}f(x)$)
⟨*proof*⟩

lemma *separation_in*:
  assumes $\forall\, x[M].\ M(f(x))$ *lam_replacement*(M,f)
    $\forall\, x[M].\ M(g(x))$ *lam_replacement*(M,g)
  shows *separation*(M,$\lambda x\ .\ f(x){\in}g(x)$)
⟨*proof*⟩

lemma *lam_replacement_swap*: *lam_replacement*(M, $\lambda x.\ \langle snd(x),fst(x)\rangle$)
  ⟨*proof*⟩

lemma *lam_replacement_range* : *lam_replacement*(*M*,*range*)
⟨*proof*⟩

lemma *separation_in_range* : *M*(*a*) ⟹ *separation*(*M*, λ*x*. *a*∈*range*(*x*))
⟨*proof*⟩

lemma *separation_in_domain* : *M*(*a*) ⟹ *separation*(*M*, λ*x*. *a*∈*domain*(*x*))
⟨*proof*⟩

lemma *lam_replacement_separation* :
assumes *lam_replacement*(*M*,*f*) *separation*(*M*,*P*)
shows *strong_replacement*(*M*, λ*x y* . *P*(*x*) ∧ *y*=⟨*x*,*f*(*x*)⟩)
⟨*proof*⟩

lemmas *strong_replacement_separation* =
strong_replacement_separation_aux[*OF lam_replacement_imp_strong_replacement*]

lemma *id_closed*: *M*(*A*) ⟹ *M*(*id*(*A*))
⟨*proof*⟩

lemma *relation_separation*: *separation*(*M*, λ*z*. ∃*x y*. *z* = ⟨*x*, *y*⟩)
⟨*proof*⟩

lemma *separation_pair*:
assumes *separation*(*M*, λ*y* . *P*(*fst*(*y*), *snd*(*y*)))
shows *separation*(*M*, λ*y*. ∃ *u v* . *y*=⟨*u*,*v*⟩ ∧ *P*(*u*,*v*))
⟨*proof*⟩

lemma *lam_replacement_Pair*:
shows *lam_replacement*(*M*, λ*x*. ⟨*fst*(*x*), *snd*(*x*)⟩)
⟨*proof*⟩

lemma *lam_replacement_Un*: *lam_replacement*(*M*, λ*p*. *fst*(*p*) ∪ *snd*(*p*))
⟨*proof*⟩

lemma *lam_replacement_cons*: *lam_replacement*(*M*, λ*p*. *cons*(*fst*(*p*),*snd*(*p*)))
⟨*proof*⟩

lemma *lam_replacement_sing*: *lam_replacement*(*M*, λ*x*. {*x*})
⟨*proof*⟩

lemmas *tag_replacement* = *lam_replacement_constant*[*unfolded lam_replacement_def*]

lemma *lam_replacement_id2*: *lam_replacement*(*M*, λ*x*. ⟨*x*, *x*⟩)
⟨*proof*⟩

lemmas *id_replacement* = *lam_replacement_id2*[*unfolded lam_replacement_def*]

lemma *lam_replacement_apply2*:*lam_replacement*(M, $\lambda p.\ fst(p)\ `\ snd(p)$)
⟨*proof*⟩

definition *map_snd* **where**
$map\_snd(X) = \{snd(z)\ .\ z{\in}X\}$

lemma *map_sndE*: $y{\in}map\_snd(X) \Longrightarrow \exists\, p{\in}X.\ y{=}snd(p)$
⟨*proof*⟩

lemma *map_sndI* : $\exists\, p{\in}X.\ y{=}snd(p) \Longrightarrow y{\in}map\_snd(X)$
⟨*proof*⟩

lemma *map_snd_closed*: $M(x) \Longrightarrow M(map\_snd(x))$
⟨*proof*⟩

lemma *lam_replacement_imp_lam_replacement_RepFun*:
assumes *lam_replacement*(M, f) $\forall\, x[M].\ M(f(x))$
separation(M, $\lambda x.\ ((\forall\, y{\in}snd(x).\ fst(y)\ \in\ fst(x))\ \wedge\ (\forall\, y{\in}fst(x).\ \exists\, u{\in}snd(x).\ y{=}fst(u))))$
and
lam_replacement_RepFun_snd:*lam_replacement*(M,*map_snd*)
shows *lam_replacement*(M, $\lambda x.\ \{f(y)\ .\ y{\in}x\}$)
⟨*proof*⟩

lemma *lam_replacement_apply*:$M(S) \Longrightarrow$ *lam_replacement*(M, $\lambda x.\ \ S\ `\ x$)
⟨*proof*⟩

lemma *apply_replacement*:$M(S) \Longrightarrow$ *strong_replacement*(M, $\lambda x\ y.\ y = S\ `\ x$)
⟨*proof*⟩

lemma *lam_replacement_id_const*: $M(b) \Longrightarrow$ *lam_replacement*(M, $\lambda x.\ \langle x,\ b\rangle$)
⟨*proof*⟩

lemmas *pospend_replacement* = *lam_replacement_id_const*[*unfolded lam_replacement_def*]

lemma *lam_replacement_const_id*: $M(b) \Longrightarrow$ *lam_replacement*(M, $\lambda z.\ \langle b,\ z\rangle$)
⟨*proof*⟩

lemmas *prepend_replacement* = *lam_replacement_const_id*[*unfolded lam_replacement_def*]

lemma *lam_replacement_apply_const_id*: $M(f) \Longrightarrow M(z) \Longrightarrow$
lam_replacement(M, $\lambda x.\ f\ `\ \langle z,\ x\rangle$)
⟨*proof*⟩

lemmas *apply_replacement2* = *lam_replacement_apply_const_id*[*unfolded lam_replacement_def*]

lemma *lam_replacement_Inl*: *lam_replacement*(M, *Inl*)
⟨*proof*⟩

lemma *lam_replacement_Inr*: *lam_replacement*(*M*, *Inr*)
⟨*proof*⟩

lemmas *Inl_replacement1* = *lam_replacement_Inl*[*unfolded lam_replacement_def*]

lemma *lam_replacement_Diff'*: *M*(*X*) ⟹ *lam_replacement*(*M*, λ*x*. *x* - *X*)
⟨*proof*⟩

lemmas *Pair_diff_replacement* = *lam_replacement_Diff'*[*unfolded lam_replacement_def*]

lemma *diff_Pair_replacement*: *M*(*p*) ⟹ *strong_replacement*(*M*, λ*x y* . *y*=⟨*x*,*x*-{*p*}⟩)
⟨*proof*⟩

lemma *swap_replacement*:*strong_replacement*(*M*, λ*x y*. *y* = ⟨*x*, (λ⟨*x*,*y*⟩. ⟨*y*, *x*⟩)(*x*)⟩)
⟨*proof*⟩

lemma *lam_replacement_Un_const*:*M*(*b*) ⟹ *lam_replacement*(*M*, λ*x*. *x* ∪ *b*)
⟨*proof*⟩

lemmas *tag_union_replacement* = *lam_replacement_Un_const*[*unfolded lam_replacement_def*]

lemma *lam_replacement_csquare*: *lam_replacement*(*M*,λ*p*. ⟨*fst*(*p*) ∪ *snd*(*p*), *fst*(*p*),
snd(*p*)⟩)
⟨*proof*⟩

lemma *csquare_lam_replacement*:*strong_replacement*(*M*, λ*x y*. *y* = ⟨*x*, (λ⟨*x*,*y*⟩.
⟨*x* ∪ *y*, *x*, *y*⟩)(*x*)⟩)
⟨*proof*⟩

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

lemma *assoc_replacement*:*strong_replacement*(*M*, λ*x y*. *y* = ⟨*x*, (λ⟨⟨*x*,*y*⟩,*z*⟩. ⟨*x*,
y, *z*⟩)(*x*)⟩)
⟨*proof*⟩

lemma *lam_replacement_prod_fun*: *M*(*f*) ⟹ *M*(*g*) ⟹ *lam_replacement*(*M*,λ*x*.
⟨*f* ‘ *fst*(*x*), *g* ‘ *snd*(*x*)⟩)
⟨*proof*⟩

lemma *prod_fun_replacement*:*M*(*f*) ⟹ *M*(*g*) ⟹
strong_replacement(*M*, λ*x y*. *y* = ⟨*x*, (λ⟨*w*,*y*⟩. ⟨*f* ‘ *w*, *g* ‘ *y*⟩)(*x*)⟩)
⟨*proof*⟩

lemma *lam_replacement_vimage_sing*: *lam_replacement*(*M*, λ*p*. *fst*(*p*) -‘‘ {*snd*(*p*)})
⟨*proof*⟩

lemma *lam_replacement_vimage_sing_fun*: $M(f) \Longrightarrow$ *lam_replacement*$(M,\ \lambda x.\ f\ \text{-``}\ \{x\})$
⟨*proof*⟩
lemma *lam_replacement_image_sing_fun*: $M(f) \Longrightarrow$ *lam_replacement*$(M,\ \lambda x.\ f\ \text{``}\ \{x\})$
⟨*proof*⟩

lemma *converse_apply_projs*: $\forall\, x[M].\ \bigcup\ (fst(x)\ \text{-``}\ \{snd(x)\}) = converse(fst(x))\ \text{`}\ (snd(x))$
⟨*proof*⟩

lemma *lam_replacement_converse_app*: *lam_replacement*$(M, \lambda p.\ converse(fst(p))\ \text{`}\ snd(p))$
⟨*proof*⟩

lemmas *cardinal_lib_assms4* = *lam_replacement_vimage_sing_fun*[*unfolded lam_replacement_def*]

lemma *lam_replacement_sing_const_id*:
$M(x) \Longrightarrow$ *lam_replacement*$(M,\ \lambda y.\ \{\langle x,\ y\rangle\})$
⟨*proof*⟩

lemma *tag_singleton_closed*: $M(x) \Longrightarrow M(z) \Longrightarrow M(\{\{\langle z,\ y\rangle\}\ .\ y \in x\})$
⟨*proof*⟩

lemma *separation_eq*:
assumes $\forall\, x[M].\ M(f(x))$ *lam_replacement*(M,f)
$\forall\, x[M].\ M(g(x))$ *lam_replacement*(M,g)
shows *separation*$(M,\lambda x\ .\ f(x) = g(x))$
⟨*proof*⟩

lemma *separation_subset*:
assumes $\forall\, x[M].\ M(f(x))$ *lam_replacement*(M,f)
$\forall\, x[M].\ M(g(x))$ *lam_replacement*(M,g)
shows *separation*$(M,\lambda x\ .\ f(x) \subseteq g(x))$
⟨*proof*⟩

lemma *separation_ball*:
assumes *separation*$(M,\ \lambda y.\ f(fst(y),snd(y)))\ M(X)$
shows *separation*$(M,\ \lambda y.\ \forall\, u{\in}X.\ f(y,u))$
⟨*proof*⟩

lemma *lam_replacement_twist*: *lam_replacement*$(M,\lambda\langle\langle x,y\rangle,z\rangle.\ \langle x,y,z\rangle)$
⟨*proof*⟩

lemma *twist_closed*[*intro*,*simp*]: $M(x) \Longrightarrow M((\lambda\langle\langle x,y\rangle,z\rangle.\ \langle x,y,z\rangle)(x))$
⟨*proof*⟩

lemma *lam_replacement_Lambda*:
assumes *lam_replacement*$(M,\ \lambda y.\ b(fst(y),\ snd(y)))$

$\forall\, w[M].\ \forall\, y[M].\ M(b(w,\ y))\ M(W)$
shows *lam_replacement*$(M,\ \lambda x.\ \lambda w{\in}W.\ b(x,\ w))$
⟨*proof*⟩

lemma *lam_replacement_apply_Pair*:
assumes $M(y)$
shows *lam_replacement*$(M,\ \lambda x.\ y\ `\ \langle fst(x),\ snd(x)\rangle)$
⟨*proof*⟩

lemma *lam_replacement_apply_fst_snd*:
shows *lam_replacement*$(M,\ \lambda w.\ fst(w)\ `\ fst(snd(w))\ `\ snd(snd(w)))$
⟨*proof*⟩

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

lemma *lam_replacement_if_mem*:
lam_replacement$(M,\ \lambda x.\ if\ snd(x) \in fst(x)\ then\ 1\ else\ 0)$
⟨*proof*⟩

lemma *lam_replacement_Lambda_apply_fst_snd*:
assumes $M(X)$
shows *lam_replacement*$(M,\ \lambda x.\ \lambda w{\in}X.\ x\ `\ fst(w)\ `\ snd(w))$
⟨*proof*⟩

lemma *lam_replacement_Lambda_apply_Pair*:
assumes $M(X)\ M(y)$
shows *lam_replacement*$(M,\ \lambda x.\ \lambda w{\in}X.\ y\ `\ \langle x,\ w\rangle)$
⟨*proof*⟩

lemma *lam_replacement_Lambda_if_mem*:
assumes $M(X)$
shows *lam_replacement*$(M,\ \lambda x.\ \lambda xa{\in}X.\ if\ xa \in x\ then\ 1\ else\ 0)$
⟨*proof*⟩

lemma *lam_replacement_comp′*:
$M(f) \Longrightarrow M(g) \Longrightarrow$ *lam_replacement*$(M,\ \lambda x\ .\ f\ O\ x\ O\ g)$
⟨*proof*⟩

lemma *separation_bex*:
assumes *separation*$(M,\ \lambda y.\ f(fst(y),snd(y)))\ M(X)$
shows *separation*$(M,\ \lambda y.\ \exists\, u{\in}X.\ f(y,u))$
⟨*proof*⟩

lemma *case_closed* :
assumes $\forall\, x[M].\ M(f(x))\ \forall\, x[M].\ M(g(x))$
shows $\forall\, x[M].\ M(case(f,g,x))$
⟨*proof*⟩

lemma *separation_fst_equal* : $M(a) \Longrightarrow$ *separation*(M,λx . *fst*(x)=a)
⟨*proof*⟩

lemma *lam_replacement_case* :
assumes *lam_replacement*(M,f) *lam_replacement*(M,g)
$\forall\, x[M].\ M(f(x))\ \forall\, x[M].\ M(g(x))$
shows *lam_replacement*(M, λx . *case*(f,g,x))
⟨*proof*⟩

lemma *Pi_replacement1*: $M(x) \Longrightarrow M(y) \Longrightarrow$ *strong_replacement*(M, $\lambda ya\ z.\ ya \in y \wedge z = \{\langle x,\ ya\rangle\}$)
⟨*proof*⟩

lemma *surj_imp_inj_replacement1*:
$M(f) \Longrightarrow M(x) \Longrightarrow$ *strong_replacement*(M, $\lambda y\ z.\ y \in f$ -`` $\{x\} \wedge z = \{\langle x,\ y\rangle\}$)
⟨*proof*⟩

lemmas *domain_replacement* = *lam_replacement_domain*[*unfolded lam_replacement_def*]

lemma *domain_replacement_simp*: *strong_replacement*(M, $\lambda x\ y.\ y$=*domain*(x))
⟨*proof*⟩

lemma *un_Pair_replacement*: $M(p) \Longrightarrow$ *strong_replacement*(M, $\lambda x\, y\ .\ y = x \cup \{p\}$)
⟨*proof*⟩

lemma *diff_replacement*: $M(X) \Longrightarrow$ *strong_replacement*(M, $\lambda x\ y.\ y = x$ - X)
⟨*proof*⟩

lemma *lam_replacement_succ*:
lam_replacement(M,λz . *succ*(z))
⟨*proof*⟩

lemma *lam_replacement_hcomp_Least*:
assumes *lam_replacement*(M, g) *lam_replacement*(M,$\lambda x.\ \mu\ i.\ x \in F(i,x)$)
$\forall\, x[M].\ M(g(x))\ \bigwedge x\ i.\ M(x) \Longrightarrow i \in F(i,\ x) \Longrightarrow M(i)$
shows *lam_replacement*(M,$\lambda x.\ \mu\ i.\ g(x) \in F(i,g(x))$)
⟨*proof*⟩

lemma *domain_mem_separation*: $M(A) \Longrightarrow$ *separation*(M, λx . *domain*(x)$\in A$)
⟨*proof*⟩

lemma *domain_eq_separation*: $M(p) \Longrightarrow$ *separation*(M, λx . *domain*(x) = p)
⟨*proof*⟩

lemma *lam_replacement_Int*:
shows *lam_replacement*(M, $\lambda x.$ *fst*(x) $\cap$ *snd*(x))
⟨*proof*⟩

lemma *lam_replacement_CartProd*:

assumes *lam_replacement*(M,f) *lam_replacement*(M,g)
$\forall\, x[M].\ M(f(x))\ \forall\, x[M].\ M(g(x))$
shows *lam_replacement*(M, $\lambda x.\ f(x) \times g(x)$)
⟨*proof*⟩

lemma *restrict_eq_separation'*: $M(B) \Longrightarrow \forall\, A[M].\ separation(M,\ \lambda y.\ \exists\, x{\in}A.\ y = \langle x,\ restrict(x,\ B)\rangle)$
⟨*proof*⟩

lemmas *lam_replacement_restrict'* = *lam_replacement_restrict*[*OF restrict_eq_separation'*]

lemma *restrict_strong_replacement*: $M(A) \Longrightarrow strong\_replacement(M, \lambda x\ y.\ y{=}restrict(x,A))$
⟨*proof*⟩

lemma *restrict_eq_separation*: $M(r) \Longrightarrow M(p) \Longrightarrow separation(M, \lambda x\ .\ restrict(x,r) = p)$
⟨*proof*⟩

lemma *separation_equal_fst2* : $M(a) \Longrightarrow separation(M,\lambda x\ .\ fst(fst(x))=a)$
⟨*proof*⟩

lemma *separation_equal_apply*: $M(f) \Longrightarrow M(a) \Longrightarrow separation(M,\lambda x.\ f`x=a)$
⟨*proof*⟩

lemma *lam_apply_replacement*: $M(A) \Longrightarrow M(f) \Longrightarrow lam\_replacement(M,\ \lambda x\ .\ \lambda n{\in}A.\ f\ `\ \langle x,\ n\rangle)$
⟨*proof*⟩

lemma *separation_all*:
assumes *separation*(M, $\lambda x\ .P(fst(x),snd(x))$)
shows *separation*(M, $\lambda z.\ \forall\, x{\in}z.\ P(z,x)$)
⟨*proof*⟩

lemma *separation_Transset*: *separation*(M,*Transset*)
⟨*proof*⟩

lemma *separation_comp* :
assumes *separation*(M,P) *lam_replacement*(M,f) $\forall\, x[M].\ M(f(x))$
shows *separation*(M,$\lambda x.\ P(f(x))$)
⟨*proof*⟩

lemma *separation_Ord*: *separation*(M,*Ord*)
⟨*proof*⟩

end — *M_replacement*

locale *M_replacement_extra* = *M_replacement* +
assumes
lam_replacement_minimum:*lam_replacement*(M, $\lambda p.\ minimum(fst(p),snd(p))$)

and
lam_replacement_RepFun_cons:*lam_replacement*(M, λp. *RepFun*(*fst*(p), λx. $\{\langle$*snd*(p),$x\rangle\}$))
— This one is too particular: It is for *Sigfun*. I would like greater modularity here.

begin
lemma *lam_replacement_Sigfun*:
assumes *lam_replacement*(M,f) $\forall\, y[M].\ M(f(y))$
shows *lam_replacement*(M, λx. *Sigfun*(x,f))
⟨*proof*⟩

12.2 Particular instances

lemma *surj_imp_inj_replacement2*:
$M(f) \Longrightarrow$ *strong_replacement*(M, $\lambda x\ z.\ z =$ *Sigfun*(x, $\lambda y.\ f$ -‘‘ $\{y\}$))
⟨*proof*⟩

lemma *lam_replacement_minimum_vimage*:
$M(f) \Longrightarrow M(r) \Longrightarrow$ *lam_replacement*(M, $\lambda x.$ *minimum*(r, f -‘‘ $\{x\}$))
⟨*proof*⟩

lemmas *surj_imp_inj_replacement4* = *lam_replacement_minimum_vimage*[*unfolded lam_replacement_def*]

lemma *lam_replacement_Pi*: $M(y) \Longrightarrow$ *lam_replacement*(M, $\lambda x.\ \bigcup xa{\in}y.\ \{\langle x,$ $xa\rangle\}$)
⟨*proof*⟩

lemma *Pi_replacement2*: $M(y) \Longrightarrow$ *strong_replacement*(M, $\lambda x\ z.\ z = (\bigcup xa{\in}y.$ $\{\langle x,\ xa\rangle\})$)
⟨*proof*⟩

lemma *if_then_Inj_replacement*:
shows $M(A) \Longrightarrow$ *strong_replacement*(M, $\lambda x\ y.\ y = \langle x,$ *if* $x \in A$ *then* *Inl*(x) *else* *Inr*(x)$\rangle$)
⟨*proof*⟩

lemma *lam_if_then_replacement*:
$M(b) \Longrightarrow$
$M(a) \Longrightarrow M(f) \Longrightarrow$ *strong_replacement*(M, $\lambda y\ ya.\ ya = \langle y,$ *if* $y = a$ *then* b *else* f ‘ $y\rangle$)
⟨*proof*⟩

lemma *if_then_replacement*:
$M(A) \Longrightarrow M(f) \Longrightarrow M(g) \Longrightarrow$ *strong_replacement*(M, $\lambda x\ y.\ y = \langle x,$ *if* $x \in A$ *then* f ‘ x *else* g ‘ $x\rangle$)
⟨*proof*⟩

lemma *ifx_replacement*:
M(*f*) ⟹
M(*b*) ⟹ *strong_replacement*(*M*, λ*x y*. *y* = ⟨*x*, *if x* ∈ *range*(*f*) *then converse*(*f*) ‘ *x else b*⟩)
⟨*proof*⟩

lemma *if_then_range_replacement2*:
M(*A*) ⟹ *M*(*C*) ⟹ *strong_replacement*(*M*, λ*x y*. *y* = ⟨*x*, *if x* = *Inl*(*A*) *then C else x*⟩)
⟨*proof*⟩

lemma *if_then_range_replacement*:
M(*u*) ⟹
M(*f*) ⟹
strong_replacement
(*M*,
λ*z y*. *y* = ⟨*z*, *if z* = *u then f* ‘ *0 else if z* ∈ *range*(*f*) *then f* ‘ *succ*(*converse*(*f*) ‘ *z*) *else z*⟩)
⟨*proof*⟩

lemma *Inl_replacement2*:
M(*A*) ⟹
strong_replacement(*M*, λ*x y*. *y* = ⟨*x*, *if fst*(*x*) = *A then Inl*(*snd*(*x*)) *else Inr*(*x*)⟩)
⟨*proof*⟩

lemma *case_replacement1*:
strong_replacement(*M*, λ*z y*. *y* = ⟨*z*, *case*(*Inr*, *Inl*, *z*)⟩)
⟨*proof*⟩

lemma *case_replacement2*:
strong_replacement(*M*, λ*z y*. *y* = ⟨*z*, *case*(*case*(*Inl*, λ*y*. *Inr*(*Inl*(*y*))), λ*y*. *Inr*(*Inr*(*y*)), *z*)⟩)
⟨*proof*⟩

lemma *case_replacement4*:
M(*f*) ⟹ *M*(*g*) ⟹ *strong_replacement*(*M*, λ*z y*. *y* = ⟨*z*, *case*(λ*w*. *Inl*(*f* ‘ *w*), λ*y*. *Inr*(*g* ‘ *y*), *z*)⟩)
⟨*proof*⟩

lemma *case_replacement5*:
strong_replacement(*M*, λ*x y*. *y* = ⟨*x*, (λ⟨*x*,*z*⟩. *case*(λ*y*. *Inl*(⟨*y*, *z*⟩), λ*y*. *Inr*(⟨*y*, *z*⟩), *x*))(*x*)⟩)
⟨*proof*⟩

end — *M_replacement_extra*

— To be used in the relativized treatment of Cohen posets
definition
— ”domain collect F”

$dC\_F :: i \Rightarrow i \Rightarrow i$ **where**
$dC\_F(A,d) \equiv \{p \in A.\ domain(p) = d\ \}$

definition
— "domain restrict SepReplace Y"
$drSR\_Y :: i \Rightarrow i \Rightarrow i \Rightarrow i \Rightarrow i$ **where**
$drSR\_Y(B,D,A,x) \equiv \{y\ .\ r{\in}A\ ,\ restrict(r,B) = x \wedge y = domain(r) \wedge domain(r) \in D\}$

lemma *drSR_Y_equality*: $drSR\_Y(B,D,A,x) = \{\ dr{\in}D\ .\ (\exists\, r{\in}A\ .\ restrict(r,B) = x \wedge dr{=}domain(r))\ \}$
⟨*proof*⟩

context *M_replacement_extra*
begin

lemma *separation_restrict_eq_dom_eq*:$\forall\, x[M].separation(M, \lambda dr.\ \exists\, r{\in}A\ .\ restrict(r,B) = x \wedge dr{=}domain(r))$
if $M(A)$ **and** $M(B)$ **for** $A\ B$
⟨*proof*⟩

lemma *separation_is_insnd_restrict_eq_dom* : $separation(M,\ \lambda p.\ \forall\, x{\in}D.\ x \in snd(p) \longleftrightarrow (\exists\, r{\in}A.\ restrict(r,\ B) = fst(p) \wedge x = domain(r)))$
if $M(B)\ M(D)\ M(A)$ **for** $A\ B\ D$
⟨*proof*⟩

lemma *lam_replacement_drSR_Y*:
assumes
$M(B)\ M(D)\ M(A)$
shows $lam\_replacement(M,\ drSR\_Y(B,D,A))$
⟨*proof*⟩

lemma *drSR_Y_closed*:
assumes
$M(B)\ M(D)\ M(A)\ M(f)$
shows $M(drSR\_Y(B,D,A,f))$
⟨*proof*⟩

lemma *lam_if_then_apply_replacement*: $M(f) \Longrightarrow M(v) \Longrightarrow M(u) \Longrightarrow$
$lam\_replacement(M,\ \lambda x.\ if\ f\ `\ x = v\ then\ f\ `\ u\ else\ f\ `\ x)$
⟨*proof*⟩

lemma *lam_if_then_apply_replacement2*: $M(f) \Longrightarrow M(m) \Longrightarrow M(y) \Longrightarrow$
$lam\_replacement(M,\ \lambda z\ .\ if\ f\ `\ z = m\ then\ y\ else\ f\ `\ z)$
⟨*proof*⟩

lemma *lam_if_then_replacement2*: $M(A) \Longrightarrow M(f) \Longrightarrow$
$lam\_replacement(M,\ \lambda x\ .\ if\ x \in A\ then\ f\ `\ x\ else\ x)$

⟨*proof*⟩

lemma *lam_if_then_replacement_apply*: $M(G) \Longrightarrow$ *lam_replacement*$(M,\ \lambda x.$ *if* $M(x)$ *then* G ‘ x *else* $0)$
⟨*proof*⟩

lemma *lam_replacement_dC_F*:
assumes $M(A)$
shows *lam_replacement*$(M,$ *dC_F*$(A))$
⟨*proof*⟩

lemma *dCF_closed*:
assumes $M(A)$ $M(f)$
shows $M($*dC_F*$(A,f))$
⟨*proof*⟩

lemma *lam_replacement_min*: $M(f) \Longrightarrow M(r) \Longrightarrow$ *lam_replacement*$(M,\ \lambda x\ .$ *minimum*$(r,$ f -‘‘ $\{x\}))$
⟨*proof*⟩

lemma *lam_replacement_Collect_ball_Pair*:
assumes *separation*$(M,\ \lambda p.\ \forall\, x{\in}G.\ x \in snd(p) \longleftrightarrow (\forall\, s{\in}fst(p).\ \langle s,\ x\rangle \in Q))$ $M(G)$ $M(Q)$
shows *lam_replacement*$(M,\ \lambda x\ .\ \{a \in G\ .\ \forall\, s{\in}x.\ \langle s,\ a\rangle \in Q\})$
⟨*proof*⟩

lemma *surj_imp_inj_replacement3*:
$(\bigwedge x.\ M(x) \Longrightarrow$ *separation*$(M,\ \lambda y.\ \forall\, s{\in}x.\ \langle s,\ y\rangle \in Q)) \Longrightarrow M(G) \Longrightarrow M(Q) \Longrightarrow M(x) \Longrightarrow$
strong_replacement$(M,\ \lambda y\ z.\ y \in \{a \in G\ .\ \forall\, s{\in}x.\ \langle s,\ a\rangle \in Q\} \wedge z = \{\langle x,\ y\rangle\})$
⟨*proof*⟩

lemmas *replacements* = *Pair_diff_replacement id_replacement tag_replacement*
pospend_replacement prepend_replacement
Inl_replacement1 diff_Pair_replacement
swap_replacement tag_union_replacement csquare_lam_replacement
assoc_replacement prod_fun_replacement
cardinal_lib_assms4 domain_replacement
apply_replacement
un_Pair_replacement restrict_strong_replacement diff_replacement
if_then_Inj_replacement lam_if_then_replacement if_then_replacement
ifx_replacement if_then_range_replacement2 if_then_range_replacement
Inl_replacement2
case_replacement1 case_replacement2 case_replacement4 case_replacement5

end — *M_replacement_extra*

end
theory *Discipline_Cardinal*

imports
Discipline_Function
begin

declare [[*syntax_ambiguity_warning* = *false*]]

⟨*ML*⟩

notation *is_cardinal_fm* (‹*cardinal'*(_') *is* _›)

abbreviation
cardinal_r :: [*i*,*i*⇒*o*] ⇒ *i* (‹|_|−›) **where**
$|x|^M \equiv$ *cardinal_rel*(*M*,*x*)

abbreviation
cardinal_r_set :: [*i*,*i*]⇒*i* (‹|_|−›) **where**
$|x|^M \equiv$ *cardinal_rel*(##*M*,*x*)

context *M_trivial* **begin**
⟨*ML*⟩
⟨*proof*⟩
end

⟨*ML*⟩
⟨*proof*⟩

⟨*ML*⟩

lemma *arity_is_surj_fm* [*arity*] :
$A \in nat \Longrightarrow B \in nat \Longrightarrow I \in nat \Longrightarrow arity(is\_surj\_fm(A,\ B,\ I)) = succ(A) \cup succ(B) \cup succ(I)$
⟨*proof*⟩

⟨*ML*⟩

lemma *arity_is_inj_fm* [*arity*]:
$A \in nat \Longrightarrow B \in nat \Longrightarrow I \in nat \Longrightarrow arity(is\_inj\_fm(A,\ B,\ I)) = succ(A) \cup succ(B) \cup succ(I)$
⟨*proof*⟩

⟨*ML*⟩

context *M_Perm* **begin**

⟨*ML*⟩
⟨*proof*⟩
end

⟨*ML*⟩

notation *lt_rel_fm* (‹·_ < _·›)
⟨*ML*⟩

lemma *arity_lt_rel_fm*[*arity*]: $a \in nat \Longrightarrow b \in nat \Longrightarrow arity(lt\_rel\_fm(a,\ b)) = succ(a) \cup succ(b)$
⟨*proof*⟩

⟨*ML*⟩
notation *is_Card_fm* (‹·*Card*′(_′)·›)
⟨*ML*⟩

notation *Card_rel* (‹*Card*–′(_′)›)

lemma (**in** *M_Perm*) *is_Card_iff*: $M(A) \Longrightarrow is\_Card(M,\ A) \longleftrightarrow Card^M(A)$
⟨*proof*⟩

abbreviation
Card_r_set :: [*i*,*i*]⇒*o* (‹*Card*–′(_′)›) **where**
$Card^M(i) \equiv Card\_rel(\#\#M,i)$

⟨*ML*⟩
notation *is_InfCard_fm* (‹·*InfCard*′(_′)·›)
⟨*ML*⟩

notation *InfCard_rel* (‹*InfCard*–′(_′)›)

abbreviation
InfCard_r_set :: [*i*,*i*]⇒*o* (‹*InfCard*–′(_′)›) **where**
$InfCard^M(i) \equiv InfCard\_rel(\#\#M,i)$

⟨*ML*⟩

abbreviation
cadd_r :: [*i*,*i*⇒*o*,*i*] ⇒ *i* (‹_ ⊕– _› [66,1,66] 65) **where**
$A \oplus^M B \equiv cadd\_rel(M,A,B)$

context *M_basic* **begin**
⟨*ML*⟩
⟨*proof*⟩
end

⟨*ML*⟩
⟨*proof*⟩
⟨*ML*⟩

context *M_Perm* **begin**
⟨*ML*⟩

⟨*proof*⟩
end

⟨*ML*⟩

abbreviation
cmult_r :: [*i*,*i*⇒*o*,*i*] ⇒ *i* (‹_ ⊗– _› [66,1,66] 65) **where**
$A \otimes^M B \equiv cmult\_rel(M,A,B)$

⟨*ML*⟩

declare *cartprod_iff_sats* [*iff_sats*]

⟨*ML*⟩

context *M_Perm* **begin**

⟨*ML*⟩
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

end

end

13 Relativization of the cumulative hierarchy

theory *Univ_Relative*
imports
ZF-Constructible.Rank
ZF.Univ
Discipline_Cardinal

begin

declare *arity_ordinal_fm*[*arity*]

context *M_trivial*
begin
declare *powerset_abs*[*simp*]

lemma *family_union_closed*: ⟦*strong_replacement*(*M*, λ*x* *y*. *y* = *f*(*x*)); *M*(*A*);
∀ *x*∈*A*. *M*(*f*(*x*))⟧
⟹ *M*(⋃ *x*∈*A*. *f*(*x*))
⟨*proof*⟩

lemma *family_union_closed'*: ⟦*strong_replacement*(*M*, λ*x* *y*. *x*∈*A* ∧ *y* = *f*(*x*));
M(*A*); ∀ *x*∈*A*. *M*(*f*(*x*))⟧
⟹ *M*(⋃ *x*∈*A*. *f*(*x*))
⟨*proof*⟩

end — *M_trivial*

definition
Powapply :: [*i*,*i*] ⇒ *i* **where**
Powapply(*f*,*y*) ≡ *Pow*(*f*‘*y*)

⟨*ML*⟩

declare *Replace_iff_sats*[*iff_sats*]
⟨*ML*⟩

notation *Powapply_rel* (‹*Powapply*–'(_,_')›)

context *M_basic*
begin

⟨*ML*⟩
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

end — *M_basic*

definition
HVfrom :: [*i*,*i*,*i*] ⇒ *i* **where**
HVfrom(*A*,*x*,*f*) ≡ *A* ∪ (⋃ *y*∈*x*. *Powapply*(*f*,*y*))

⟨*ML*⟩

lemma *arity_is_HVfrom_fm*:
A ∈ *nat* ⟹
x ∈ *nat* ⟹
f ∈ *nat* ⟹
d ∈ *nat* ⟹
arity(*is_HVfrom_fm*(*A*, *x*, *f*, *d*)) = *succ*(*A*) ∪ *succ*(*d*) ∪ (*succ*(*x*) ∪ *succ*(*f*))
⟨*proof*⟩

notation *HVfrom_rel* (‹*HVfrom*–'(_,_,_')›)

locale *M_HVfrom* = *M_eclose* +
assumes
Powapply_replacement:
M(*f*) ⟹ *strong_replacement*(*M*,λ*y* *z*. *z* = *Powapply*^{M}(*f*,*y*))

begin

⟨*ML*⟩
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

end — *M_HVfrom*

definition
Vfrom_rel :: [*i*⇒*o*,*i*,*i*] ⇒ *i* (‹*Vfrom*–'(_,_')›) **where**
$Vfrom^M(A,i) = transrec(i,\ HVfrom\_rel(M,A))$

definition
is_Vfrom :: [*i*⇒*o*,*i*,*i*,*i*] ⇒ *o* **where**
is_Vfrom(*M*,*A*,*i*,*z*) ≡ *is_transrec*(*M*,*is_HVfrom*(*M*,*A*),*i*,*z*)

definition
Hrank :: [*i*,*i*] ⇒ *i* **where**
Hrank(*x*,*f*) ≡ $(\bigcup y{\in}x.\ succ(f`y))$

definition
rrank :: *i* ⇒ *i* **where**
rrank(*a*) ≡ *Memrel*(*eclose*({*a*}))^+

⟨*ML*⟩

lemma *arity_is_Hrank_fm* : *x* ∈ *nat* ⟹
f ∈ *nat* ⟹
d ∈ *nat* ⟹
arity(*is_Hrank_fm*(*x*, *f*, *d*)) =
succ(*d*) ∪ *succ*(*x*) ∪ *succ*(*f*)
⟨*proof*⟩

locale *M_Vfrom* = *M_HVfrom* +
assumes
trepl_HVfrom : ⟦ *M*(*A*);*M*(*i*) ⟧ ⟹ *transrec_replacement*(*M*,*is_HVfrom*(*M*,*A*),*i*)
and
Hrank_replacement : *M*(*f*) ⟹ *strong_replacement*(*M*,λ*x y* . *y* = *succ*(*f*`*x*))
and
is_Hrank_replacement : *M*(*x*) ⟹ *wfrec_replacement*(*M*,*is_Hrank*(*M*),*rrank*(*x*))
and
HVfrom_replacement : ⟦ *M*(*i*) ; *M*(*A*) ⟧ ⟹
transrec_replacement(*M*,*is_HVfrom*(*M*,*A*),*i*)

begin

lemma *Vfrom_rel_iff* :

assumes $M(A)$ $M(i)$ $M(z)$ $Ord(i)$
shows $is\_Vfrom(M,A,i,z) \longleftrightarrow z = Vfrom^M(A,i)$
⟨*proof*⟩

lemma *relation2_HVfrom*: $M(A) \Longrightarrow relation2(M,is\_HVfrom(M,A),HVfrom\_rel(M,A))$
⟨*proof*⟩

lemma *HVfrom_closed* :
$M(A) \Longrightarrow \forall\, x[M].\ \forall\, g[M].\ function(g) \longrightarrow M(HVfrom\_rel(M,A,x,g))$
⟨*proof*⟩

lemma *Vfrom_rel_closed*:
assumes $M(A)$ $M(i)$ $Ord(i)$
shows $M(transrec(i,\ HVfrom\_rel(M,\ A)))$
⟨*proof*⟩

lemma *transrec_HVfrom*:
assumes $M(A)$
shows $Ord(i) \Longrightarrow M(i) \Longrightarrow \{x{\in}Vfrom(A,i).\ M(x)\} = transrec(i,HVfrom\_rel(M,A))$
⟨*proof*⟩

lemma *Vfrom_abs*: $[\![\ M(A);\ M(i);\ M(V);\ Ord(i)\ ]\!] \Longrightarrow is\_Vfrom(M,A,i,V) \longleftrightarrow$
$V = \{x{\in}Vfrom(A,i).\ M(x)\}$
⟨*proof*⟩

lemma *Vfrom_closed*: $[\![\ M(A);\ M(i);\ Ord(i)\ ]\!] \Longrightarrow M(\{x{\in}Vfrom(A,i).\ M(x)\})$
⟨*proof*⟩

end — *M_Vfrom*

13.1 Formula synthesis

context *M_Vfrom*
begin

⟨*ML*⟩
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

lemma *relation2_Hrank* :
$relation2(M,is\_Hrank(M),Hrank)$
⟨*proof*⟩

lemma *Hrank_closed* :
$\forall\, x[M].\ \forall\, g[M].\ function(g) \longrightarrow M(Hrank(x,g))$
⟨*proof*⟩

end — *M_basic*

context *M_eclose*
begin

lemma *wf_rrank* : $M(x) \Longrightarrow wf(rrank(x))$
⟨*proof*⟩

lemma *trans_rrank* : $M(x) \Longrightarrow trans(rrank(x))$
⟨*proof*⟩

lemma *relation_rrank* : $M(x) \Longrightarrow relation(rrank(x))$
⟨*proof*⟩

lemma *rrank_in_M* : $M(x) \Longrightarrow M(rrank(x))$
⟨*proof*⟩

end — *M_eclose*

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

lemma *rank_trancl*: $rank(x) = wfrec(rrank(x),\ x,\ Hrank)$
⟨*proof*⟩

definition
$Vset'$ :: $[i] \Rightarrow i$ **where**
$Vset'(A) \equiv Vfrom(0,A)$

⟨*ML*⟩

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

⟨*ML*⟩
context *M_Vfrom*
begin

lemma *Vset_abs*: $[\![\ M(i);\ M(V);\ Ord(i)\ ]\!] \Longrightarrow is\_Vset(M,i,V) \longleftrightarrow V = \{x{\in}Vset(i).\ M(x)\}$
⟨*proof*⟩

lemma *Vset_closed*: $[\![\ M(i);\ Ord(i)\ ]\!] \Longrightarrow M(\{x{\in}Vset(i).\ M(x)\})$

⟨*proof*⟩

lemma *rank_closed*: $M(a) \Longrightarrow M(rank(a))$
⟨*proof*⟩

lemma *M_into_Vset*:
assumes $M(a)$
shows $\exists\, i[M].\ \exists\, V[M].\ ordinal(M,i) \wedge is\_Vset(M,i,V) \wedge a \in V$
⟨*proof*⟩

end — *M_HVfrom*

end

14 Relative, Choice-less Cardinal Numbers

theory *Cardinal_Relative*
imports
Lambda_Replacement
Univ_Relative
begin

The following command avoids that a commonly used one-letter variable be captured by the definition of the constructible universe L.

hide_const (**open**) L

We also return to the old notation for $(+)$ to preserve the old Constructibility code.

no_notation *oadd* (**infixl** ‹+› *65*)
notation *sum* (**infixr** ‹+› *65*)

definition
Finite_rel :: $[i \Rightarrow o, i] => o$ **where**
$Finite\_rel(M,A) \equiv \exists\, om[M].\ \exists\, n[M].\ omega(M,om) \wedge n \in om \wedge eqpoll\_rel(M,A,n)$

definition
banach_functor :: $[i,i,i,i,i] \Rightarrow i$ **where**
$banach\_functor(X,Y,f,g,W) \equiv X - g``(Y - f``W)$

definition
is_banach_functor :: $[i \Rightarrow o,i,i,i,i,i,i] \Rightarrow o$ **where**
$is\_banach\_functor(M,X,Y,f,g,W,b) \equiv$
$\exists\, fW[M].\ \exists\, YfW[M].\ \exists\, gYfW[M].\ image(M,f,W,fW) \wedge setdiff(M,Y,fW,YfW)$
$\wedge$
$image(M,g,YfW,gYfW) \wedge setdiff(M,X,gYfW,b)$

lemma (**in** *M_basic*) *banach_functor_abs* :
assumes $M(X)\ M(Y)\ M(f)\ M(g)$
shows $relation1(M, is\_banach\_functor(M,X,Y,f,g), banach\_functor(X,Y,f,g))$

⟨*proof*⟩

lemma (**in** *M_basic*) *banach_functor_closed*:
assumes $M(X)$ $M(Y)$ $M(f)$ $M(g)$
shows $\forall\, W[M].\ M(banach\_functor(X,Y,f,g,W))$
⟨*proof*⟩

locale *M_cardinals* = *M_ordertype* + *M_trancl* + *M_Perm* + *M_replacement_extra*
+
assumes
radd_separation: $M(R) \Longrightarrow M(S) \Longrightarrow$
separation(M, λz.
$(\exists\, x\ y.\ z = \langle Inl(x),\ Inr(y)\rangle) \vee$
$(\exists\, x'\ x.\ z = \langle Inl(x'),\ Inl(x)\rangle \wedge \langle x',\ x\rangle \in R) \vee$
$(\exists\, y'\ y.\ z = \langle Inr(y'),\ Inr(y)\rangle \wedge \langle y',\ y\rangle \in S))$
and
rmult_separation: $M(b) \Longrightarrow M(d) \Longrightarrow$ *separation*(M,
$\lambda z.\ \exists\, x'\ y'\ x\ y.\ z = \langle\langle x',\ y'\rangle,\ x,\ y\rangle \wedge ((\langle x',\ x\rangle \in b \vee x' = x \wedge \langle y',\ y\rangle \in d))$
and
banach_repl_iter: $M(X) \Longrightarrow M(Y) \Longrightarrow M(f) \Longrightarrow M(g) \Longrightarrow$
strong_replacement(M, $\lambda x\ y.\ x{\in}nat \wedge y = banach\_functor(X,\ Y,\ f,$
$g)\hat{}x\ (0)$)
begin

lemma *rvimage_separation*: $M(f) \Longrightarrow M(r) \Longrightarrow$
separation(M, $\lambda z.\ \exists\, x\ y.\ z = \langle x,\ y\rangle \wedge \langle f\ `\ x,\ f\ `\ y\rangle \in r$)
⟨*proof*⟩

lemma *radd_closed*[*intro*,*simp*]: $M(a) \Longrightarrow M(b) \Longrightarrow M(c) \Longrightarrow M(d) \Longrightarrow M(radd(a,b,c,d))$
⟨*proof*⟩

lemma *rmult_closed*[*intro*,*simp*]: $M(a) \Longrightarrow M(b) \Longrightarrow M(c) \Longrightarrow M(d) \Longrightarrow M(rmult(a,b,c,d))$
⟨*proof*⟩

end — *M_cardinals*

lemma (**in** *M_cardinals*) *is_cardinal_iff_Least*:
assumes $M(A)$ $M(\kappa)$
shows $is\_cardinal(M,A,\kappa) \longleftrightarrow \kappa = (\mu\ i.\ M(i) \wedge i \approx^M A)$
⟨*proof*⟩

14.1 The Schroeder-Bernstein Theorem

See Davey and Priestly, page 106

context *M_cardinals*
begin

lemma *bnd_mono_banach_functor*: *bnd_mono*(X, *banach_functor*(X,Y,f,g))
⟨*proof*⟩

lemma *inj_Inter*:
assumes $g \in inj(Y,X)$ $A \neq 0$ $\forall a \in A.\ a \subseteq Y$
shows $g``(\bigcap A) = (\bigcap a \in A.\ g``a)$
⟨*proof*⟩

lemma *contin_banach_functor*:
assumes $g \in inj(Y,X)$
shows *contin*(*banach_functor*(X,Y,f,g))
⟨*proof*⟩

lemma *lfp_banach_functor*:
assumes $g \in inj(Y,X)$
shows *lfp*(X, *banach_functor*(X,Y,f,g)) =
$(\bigcup n \in nat.\ banach\_functor(X,Y,f,g)\hat{}n\ (0))$
⟨*proof*⟩

lemma *lfp_banach_functor_closed*:
assumes $M(g)$ $M(X)$ $M(Y)$ $M(f)$ $g \in inj(Y,X)$
shows $M(lfp(X,\ banach\_functor(X,Y,f,g)))$
⟨*proof*⟩

lemma *banach_decomposition_rel*:
$[|\ M(f);\ M(g);\ M(X);\ M(Y);\ f \in X \to Y;\ \ g \in inj(Y,X)\ |] \Longrightarrow$
$\exists XA[M].\ \exists XB[M].\ \exists YA[M].\ \exists YB[M].$
$(XA \cap XB = 0)\ \&\ (XA \cup XB = X)\ \&$
$(YA \cap YB = 0)\ \&\ (YA \cup YB = Y)\ \&$
$f``XA = YA\ \&\ g``YB = XB$
⟨*proof*⟩

lemma *schroeder_bernstein_closed*:
$[|\ M(f);\ M(g);\ M(X);\ M(Y);\ f \in inj(X,Y);\ \ g \in inj(Y,X)\ |] \Longrightarrow \exists h[M].\ h \in bij(X,Y)$
⟨*proof*⟩

lemma *mem_Pow_rel*: $M(r) \Longrightarrow a \in Pow\_rel(M,r) \Longrightarrow a \in Pow(r) \wedge M(a)$
⟨*proof*⟩

lemma *mem_bij_abs*[*simp*]: $[\![M(f);M(A);M(B)]\!] \Longrightarrow f \in bij^M(A,B) \longleftrightarrow f \in bij(A,B)$
⟨*proof*⟩

lemma *mem_inj_abs*[*simp*]: $[\![M(f);M(A);M(B)]\!] \Longrightarrow f \in inj^M(A,B) \longleftrightarrow f \in inj(A,B)$
⟨*proof*⟩

lemma *mem_surj_abs*: $[\![M(f);M(A);M(B)]\!] \Longrightarrow f \in surj^M(A,B) \longleftrightarrow f \in surj(A,B)$

⟨*proof*⟩

lemma *bij_imp_eqpoll_rel*:
assumes $f \in bij(A,B)$ $M(f)$ $M(A)$ $M(B)$
shows $A \approx^M B$
⟨*proof*⟩

lemma *eqpoll_rel_refl*: $M(A) \Longrightarrow A \approx^M A$
⟨*proof*⟩

lemma *eqpoll_rel_sym*: $X \approx^M Y \Longrightarrow M(X) \Longrightarrow M(Y) \Longrightarrow Y \approx^M X$
⟨*proof*⟩

lemma *eqpoll_rel_trans* [*trans*]:
$[|X \approx^M Y;\ Y \approx^M Z;\ M(X);\ M(Y)\ ;\ M(Z)\ |] ==> X \approx^M Z$
⟨*proof*⟩

lemma *subset_imp_lepoll_rel*: $X \subseteq Y \Longrightarrow M(X) \Longrightarrow M(Y) \Longrightarrow X \lesssim^M Y$
⟨*proof*⟩

lemmas *lepoll_rel_refl* = *subset_refl* [*THEN subset_imp_lepoll_rel*, *simp*]

lemmas *le_imp_lepoll_rel* = *le_imp_subset* [*THEN subset_imp_lepoll_rel*]

lemma *eqpoll_rel_imp_lepoll_rel*: $X \approx^M Y ==> M(X) \Longrightarrow M(Y) \Longrightarrow X \lesssim^M Y$
⟨*proof*⟩

lemma *lepoll_rel_trans* [*trans*]:
assumes
$X \lesssim^M Y$ $Y \lesssim^M Z$ $M(X)$ $M(Y)$ $M(Z)$
shows
$X \lesssim^M Z$
⟨*proof*⟩

lemma *eq_lepoll_rel_trans* [*trans*]:
assumes
$X \approx^M Y$ $Y \lesssim^M Z$ $M(X)$ $M(Y)$ $M(Z)$
shows
$X \lesssim^M Z$
⟨*proof*⟩

lemma *lepoll_rel_eq_trans* [*trans*]:
assumes $X \lesssim^M Y$ $Y \approx^M Z$ $M(X)$ $M(Y)$ $M(Z)$
shows $X \lesssim^M Z$
⟨*proof*⟩

lemma *eqpoll_relI*: $[\![X \lesssim^M Y;\ Y \lesssim^M X;\ M(X)\ ;\ M(Y)\ ]\!] \Longrightarrow X \approx^M Y$
⟨*proof*⟩

lemma *eqpoll_relE*:
$[\!| X \approx^M Y;\ [\!| X \lesssim^M Y;\ Y \lesssim^M X\ |\!] ==> P\ ;\ \ M(X)\ ;\ M(Y)\ |\!] ==> P$
⟨*proof*⟩

lemma *eqpoll_rel_iff*: $M(X) \Longrightarrow M(Y) \Longrightarrow X \approx^M Y \longleftrightarrow X \lesssim^M Y\ \&\ Y \lesssim^M X$
⟨*proof*⟩

lemma *lepoll_rel_0_is_0*: $A \lesssim^M 0 \Longrightarrow M(A) \Longrightarrow A = 0$
⟨*proof*⟩

lemmas *empty_lepoll_relI* = *empty_subsetI* [*THEN subset_imp_lepoll_rel*, *OF nonempty*]

lemma *lepoll_rel_0_iff*: $M(A) \Longrightarrow A \lesssim^M 0 \longleftrightarrow A=0$
⟨*proof*⟩

lemma *Un_lepoll_rel_Un*:
$[\!| A \lesssim^M B;\ C \lesssim^M D;\ B \cap D = 0;\ M(A);\ M(B);\ M(C);\ M(D)\ |\!] ==> A \cup C \lesssim^M B \cup D$
⟨*proof*⟩

lemma *eqpoll_rel_0_is_0*: $A \approx^M 0 \Longrightarrow M(A) \Longrightarrow A = 0$
⟨*proof*⟩

lemma *eqpoll_rel_0_iff*: $M(A) \Longrightarrow A \approx^M 0 \longleftrightarrow A=0$
⟨*proof*⟩

lemma *eqpoll_rel_disjoint_Un*:
$[\!| A \approx^M B;\ \ C \approx^M D;\ \ A \cap C = 0;\ \ B \cap D = 0;\ M(A);\ M(B);\ M(C)\ ;\ M(D)\ |\!]$
$==> A \cup C \approx^M B \cup D$
⟨*proof*⟩

14.2 lesspoll_rel: contributions by Krzysztof Grabczewski

lemma *lesspoll_rel_not_refl*: $M(i) \Longrightarrow\ ^\sim (i \prec^M i)$
⟨*proof*⟩

lemma *lesspoll_rel_irrefl*: $i \prec^M i ==> M(i) \Longrightarrow P$
⟨*proof*⟩

lemma *lesspoll_rel_imp_lepoll_rel*: $[\![A \prec^M B;\ M(A);\ M(B)]\!] \Longrightarrow A \lesssim^M B$
⟨*proof*⟩

lemma *rvimage_closed* [*intro*,*simp*]:
assumes

$M(A)$ $M(f)$ $M(r)$
shows
$M(rvimage(A,f,r))$
⟨*proof*⟩

lemma *lepoll_rel_well_ord*: [| $A \lesssim^M B$; *well_ord*(B,r); $M(A)$; $M(B)$; $M(r)$ |] ==> $\exists s[M].$ *well_ord*(A,s)
⟨*proof*⟩

lemma *lepoll_rel_iff_leqpoll_rel*: $[\![M(A);\ M(B)]\!] \Longrightarrow A \lesssim^M B \longleftrightarrow A \prec^M B \mid A \approx^M B$
⟨*proof*⟩

end — *M_cardinals*

context *M_cardinals*
begin

lemma *inj_rel_is_fun_M*: $f \in inj^M(A,B) \Longrightarrow M(f) \Longrightarrow M(A) \Longrightarrow M(B) \Longrightarrow f \in A \rightarrow^M B$
⟨*proof*⟩
lemma *inj_rel_not_surj_rel_succ*:
notes *mem_inj_abs*[*simp del*]
assumes *fi*: $f \in inj^M(A,\ succ(m))$ **and** *fns*: $f \notin surj^M(A,\ succ(m))$
and *types*: $M(f)$ $M(A)$ $M(m)$
shows $\exists f[M].\ f \in inj^M(A,m)$
⟨*proof*⟩

lemma *lesspoll_rel_trans* [*trans*]:
[| $X \prec^M Y$; $Y \prec^M Z$; $M(X)$; $M(Y)$; $M(Z)$ |] ==> $X \prec^M Z$
⟨*proof*⟩

lemma *lesspoll_rel_trans1* [*trans*]:
[| $X \lesssim^M Y$; $Y \prec^M Z$; $M(X)$; $M(Y)$; $M(Z)$ |] ==> $X \prec^M Z$
⟨*proof*⟩

lemma *lesspoll_rel_trans2* [*trans*]:
[| $X \prec^M Y$; $Y \lesssim^M Z$; $M(X)$; $M(Y)$; $M(Z)$|] ==> $X \prec^M Z$
⟨*proof*⟩

lemma *eq_lesspoll_rel_trans* [*trans*]:
[| $X \approx^M Y$; $Y \prec^M Z$; $M(X)$; $M(Y)$; $M(Z)$ |] ==> $X \prec^M Z$
⟨*proof*⟩

lemma *lesspoll_rel_eq_trans* [*trans*]:
[| $X \prec^M Y$; $Y \approx^M Z$; $M(X)$; $M(Y)$; $M(Z)$ |] ==> $X \prec^M Z$
⟨*proof*⟩

lemma *is_cardinal_cong*:
  assumes $X \approx^M Y$ $M(X)$ $M(Y)$
  shows $\exists \kappa[M].\ is\_cardinal(M,X,\kappa) \wedge is\_cardinal(M,Y,\kappa)$
⟨*proof*⟩
lemma *cardinal_rel_cong*: $X \approx^M Y \Longrightarrow M(X) \Longrightarrow M(Y) \Longrightarrow |X|^M = |Y|^M$
  ⟨*proof*⟩

lemma *well_ord_is_cardinal_eqpoll_rel*:
  assumes *well_ord*(A,r) **shows** *is_cardinal*$(M,A,\kappa) \Longrightarrow M(A) \Longrightarrow M(\kappa) \Longrightarrow M(r) \Longrightarrow \kappa \approx^M A$
⟨*proof*⟩

lemmas *Ord_is_cardinal_eqpoll_rel* = *well_ord_Memrel*[*THEN well_ord_is_cardinal_eqpoll_rel*]

15 Porting from *ZF.Cardinal*

The following results were ported more or less directly from *ZF.Cardinal*

lemma *well_ord_cardinal_rel_eqpoll_rel*:
  assumes r: *well_ord*(A,r) **and** $M(A)$ $M(r)$ **shows** $|A|^M \approx^M A$
  ⟨*proof*⟩

lemmas *Ord_cardinal_rel_eqpoll_rel* = *well_ord_Memrel*[*THEN well_ord_cardinal_rel_eqpoll_rel*]

lemma *Ord_cardinal_rel_idem*: *Ord*$(A) \Longrightarrow M(A) \Longrightarrow ||A|^M|^M = |A|^M$
  ⟨*proof*⟩

lemma *well_ord_cardinal_rel_eqE*:
  assumes *woX*: *well_ord*(X,r) **and** *woY*: *well_ord*(Y,s) **and** *eq*: $|X|^M = |Y|^M$
    and *types*: $M(X)$ $M(r)$ $M(Y)$ $M(s)$
  shows $X \approx^M Y$
⟨*proof*⟩

lemma *well_ord_cardinal_rel_eqpoll_rel_iff*:
  [| *well_ord*(X,r); *well_ord*(Y,s); $M(X)$; $M(r)$; $M(Y)$; $M(s)$ |] ==> $|X|^M = |Y|^M \longleftrightarrow X \approx^M Y$
  ⟨*proof*⟩

lemma *Ord_cardinal_rel_le*: *Ord*$(i) \Longrightarrow M(i) ==> |i|^M \leq i$
  ⟨*proof*⟩

lemma *Card_rel_cardinal_rel_eq*: *Card*$^M(K) ==> M(K) \Longrightarrow |K|^M = K$
  ⟨*proof*⟩

lemma *Card_relI*: [| *Ord*(i); !!j. $j<i \Longrightarrow M(j) ==> \sim(j \approx^M i)$; $M(i)$ |] ==> *Card*$^M(i)$
  ⟨*proof*⟩

lemma *Card_rel_is_Ord*: $Card^M(i) ==> M(i) \Longrightarrow Ord(i)$
⟨*proof*⟩

lemma *Card_rel_cardinal_rel_le*: $Card^M(K) ==> M(K) \Longrightarrow K \leq |K|^M$
⟨*proof*⟩

lemma *Ord_cardinal_rel* [*simp*,*intro*!]: $M(A) \Longrightarrow Ord(|A|^M)$
⟨*proof*⟩

lemma *Card_rel_iff_initial*: **assumes** *types*:$M(K)$
shows $Card^M(K) \longleftrightarrow Ord(K)$ & $(\forall j[M].\ j<K \longrightarrow \sim (j \approx^M K))$
⟨*proof*⟩

lemma *lt_Card_rel_imp_lesspoll_rel*: [| $Card^M(a)$; $i<a$; $M(a)$; $M(i)$ |] ==> $i \prec^M a$
⟨*proof*⟩

lemma *Card_rel_0*: $Card^M(0)$
⟨*proof*⟩

lemma *Card_rel_Un*: [| $Card^M(K)$; $Card^M(L)$; $M(K)$; $M(L)$ |] ==> $Card^M(K \cup L)$
⟨*proof*⟩

lemma *Card_rel_cardinal_rel* [*iff*]: **assumes** *types*:$M(A)$ **shows** $Card^M(|A|^M)$
⟨*proof*⟩

lemma *cardinal_rel_eq_lemma*:
assumes *i*:$|i|^M \leq j$ **and** *j*: $j \leq i$ **and** *types*: $M(i)$ $M(j)$
shows $|j|^M = |i|^M$
⟨*proof*⟩

lemma *cardinal_rel_mono*:
assumes *ij*: $i \leq j$ **and** *types*:$M(i)$ $M(j)$ **shows** $|i|^M \leq |j|^M$
⟨*proof*⟩

lemma *cardinal_rel_lt_imp_lt*: [| $|i|^M < |j|^M$; $Ord(i)$; $Ord(j)$; $M(i)$; $M(j)$ |] ==> $i < j$
⟨*proof*⟩

lemma *Card_rel_lt_imp_lt*: [| $|i|^M < K$; $Ord(i)$; $Card^M(K)$; $M(i)$; $M(K)$|] ==> $i < K$
⟨*proof*⟩

lemma *Card_rel_lt_iff*: [| $Ord(i)$; $Card^M(K)$; $M(i)$; $M(K)$ |] ==> $(|i|^M < K) \longleftrightarrow (i < K)$
⟨*proof*⟩

lemma *Card_rel_le_iff*: [| $Ord(i)$; $Card^M(K)$; $M(i)$; $M(K)$ |] ==> $(K \leq |i|^M)$

$\longleftrightarrow (K \leq i)$
⟨*proof*⟩

lemma *well_ord_lepoll_rel_imp_cardinal_rel_le*:
assumes *wB*: *well_ord*(*B*,*r*) **and** *AB*: $A \lesssim^M B$
and
types: $M(B)$ $M(r)$ $M(A)$
shows $|A|^M \leq |B|^M$
⟨*proof*⟩

lemma *lepoll_rel_cardinal_rel_le*: [| $A \lesssim^M i$; *Ord*(*i*); $M(A)$; $M(i)$ |] ==> $|A|^M \leq i$
⟨*proof*⟩

lemma *lepoll_rel_Ord_imp_eqpoll_rel*: [| $A \lesssim^M i$; *Ord*(*i*); $M(A)$; $M(i)$ |] ==> $|A|^M \approx^M A$
⟨*proof*⟩

lemma *lesspoll_rel_imp_eqpoll_rel*: [| $A \prec^M i$; *Ord*(*i*); $M(A)$; $M(i)$ |] ==> $|A|^M \approx^M A$
⟨*proof*⟩

lemma *lesspoll_cardinal_lt_rel*:
shows [| $A \prec^M i$; *Ord*(*i*); $M(i)$; $M(A)$ |] ==> $|A|^M < i$
⟨*proof*⟩

lemma *cardinal_rel_subset_Ord*: [|*A*<=*i*; *Ord*(*i*); $M(A)$; $M(i)$|] ==> $|A|^M \subseteq i$
⟨*proof*⟩
lemma *cons_lepoll_rel_consD*:
[| *cons*(*u*,*A*) $\lesssim^M$ *cons*(*v*,*B*); $u \notin A$; $v \notin B$; $M(u)$; $M(A)$; $M(v)$; $M(B)$ |] ==> $A \lesssim^M B$
⟨*proof*⟩

lemma *cons_eqpoll_rel_consD*: [| *cons*(*u*,*A*) $\approx^M$ *cons*(*v*,*B*); $u \notin A$; $v \notin B$; $M(u)$; $M(A)$; $M(v)$; $M(B)$ |] ==> $A \approx^M B$
⟨*proof*⟩

lemma *succ_lepoll_rel_succD*: *succ*(*m*) $\lesssim^M$ *succ*(*n*) $\Longrightarrow$ $M(m)$ $\Longrightarrow$ $M(n)$ ==> $m \lesssim^M n$
⟨*proof*⟩

lemma *nat_lepoll_rel_imp_le*:
$m \in nat$ ==> $n \in nat$ $\Longrightarrow$ $m \lesssim^M n$ $\Longrightarrow$ $M(m)$ $\Longrightarrow$ $M(n)$ $\Longrightarrow$ $m \leq n$
⟨*proof*⟩

lemma *nat_eqpoll_rel_iff*: [| $m \in nat$; $n \in nat$; $M(m)$; $M(n)$ |] ==> $m \approx^M n \longleftrightarrow m = n$
⟨*proof*⟩

lemma *nat_into_Card_rel*:
assumes *n*: $n \in nat$ **and** *types*: $M(n)$ **shows** $Card^M(n)$
⟨*proof*⟩

lemmas *cardinal_rel_0* = *nat_0I* [*THEN nat_into_Card_rel*, *THEN Card_rel_cardinal_rel_eq*, *simplified*, *iff*]
lemmas *cardinal_rel_1* = *nat_1I* [*THEN nat_into_Card_rel*, *THEN Card_rel_cardinal_rel_eq*, *simplified*, *iff*]

lemma *succ_lepoll_rel_natE*: [| $succ(n) \lesssim^M n$; $n \in nat$ |] ==> P
⟨*proof*⟩

lemma *nat_lepoll_rel_imp_ex_eqpoll_rel_n*:
[| $n \in nat$; $nat \lesssim^M X$; $M(n)$; $M(X)$|] ==> $\exists Y[M]. Y \subseteq X \ \& \ n \approx^M Y$
⟨*proof*⟩

lemma *lepoll_rel_succ*: $M(i) \Longrightarrow i \lesssim^M succ(i)$
⟨*proof*⟩

lemma *lepoll_rel_imp_lesspoll_rel_succ*:
assumes *A*: $A \lesssim^M m$ **and** *m*: $m \in nat$
and *types*: $M(A)$ $M(m)$
shows $A \prec^M succ(m)$
⟨*proof*⟩

lemma *lesspoll_rel_succ_imp_lepoll_rel*:
[| $A \prec^M succ(m)$; $m \in nat$; $M(A)$; $M(m)$ |] ==> $A \lesssim^M m$
⟨*proof*⟩

lemma *lesspoll_rel_succ_iff*: $m \in nat \Longrightarrow M(A)$ ==> $A \prec^M succ(m) \longleftrightarrow A \lesssim^M m$
⟨*proof*⟩

lemma *lepoll_rel_succ_disj*: [| $A \lesssim^M succ(m)$; $m \in nat$; $M(A)$; $M(m)$|] ==> $A \lesssim^M m \mid A \approx^M succ(m)$
⟨*proof*⟩

lemma *lesspoll_rel_cardinal_rel_lt*: [| $A \prec^M i$; $Ord(i)$; $M(A)$; $M(i)$ |] ==> $|A|^M < i$
⟨*proof*⟩

lemma *lt_not_lepoll_rel*:
assumes *n*: $n<i$ $n \in nat$
and *types*:$M(n)$ $M(i)$ **shows** $\sim i \lesssim^M n$
⟨*proof*⟩

A slightly weaker version of *nat_eqpoll_rel_iff*

lemma *Ord_nat_eqpoll_rel_iff*:

assumes i: $Ord(i)$ **and** n: $n \in nat$
and *types*: $M(i)$ $M(n)$
shows $i \approx^M n \longleftrightarrow i{=}n$
⟨*proof*⟩

lemma *Card_rel_nat*: $Card^M(nat)$
⟨*proof*⟩

lemma *nat_le_cardinal_rel*: $nat \le i \Longrightarrow M(i) \Longrightarrow nat \le |i|^M$
⟨*proof*⟩

lemma *n_lesspoll_rel_nat*: $n \in nat \Longrightarrow n \prec^M nat$
⟨*proof*⟩

lemma *cons_lepoll_rel_cong*:
$[\!|\ A \lesssim^M B;\ \ b \notin B;\ M(A);\ M(B);\ M(b);\ M(a)\ |\!] \Longrightarrow cons(a,A) \lesssim^M cons(b,B)$
⟨*proof*⟩

lemma *cons_eqpoll_rel_cong*:
$[\!|\ A \approx^M B;\ \ a \notin A;\ \ b \notin B;\ \ M(A);\ M(B);\ M(a)\ ;\ M(b)\ |\!] \Longrightarrow cons(a,A) \approx^M cons(b,B)$
⟨*proof*⟩

lemma *cons_lepoll_rel_cons_iff*:
$[\!|\ a \notin A;\ \ b \notin B;\ M(a);\ M(A);\ M(b);\ M(B)\ |\!] \Longrightarrow cons(a,A) \lesssim^M cons(b,B) \longleftrightarrow A \lesssim^M B$
⟨*proof*⟩

lemma *cons_eqpoll_rel_cons_iff*:
$[\!|\ a \notin A;\ \ b \notin B;\ M(a);\ M(A);\ M(b);\ M(B)\ |\!] \Longrightarrow cons(a,A) \approx^M cons(b,B) \longleftrightarrow A \approx^M B$
⟨*proof*⟩

lemma *singleton_eqpoll_rel_1*: $M(a) \Longrightarrow \{a\} \approx^M 1$
⟨*proof*⟩

lemma *cardinal_rel_singleton*: $M(a) \Longrightarrow |\{a\}|^M = 1$
⟨*proof*⟩

lemma *not_0_is_lepoll_rel_1*: $A \neq 0 \Longrightarrow M(A) \Longrightarrow 1 \lesssim^M A$
⟨*proof*⟩

lemma *succ_eqpoll_rel_cong*: $A \approx^M B \Longrightarrow M(A) \Longrightarrow M(B) \Longrightarrow succ(A) \approx^M succ(B)$
⟨*proof*⟩

The next result was not straightforward to port, and even a different statement was needed.

lemma *sum_bij_rel*:
[| $f \in bij^M(A,C)$; $g \in bij^M(B,D)$; $M(f)$; $M(A)$; $M(C)$; $M(g)$; $M(B)$; $M(D)$|]
==> $(\lambda z{\in}A+B.\ case(\%x.\ Inl(f`x),\ \%y.\ Inr(g`y),\ z)) \in bij^M(A+B,\ C+D)$
⟨*proof*⟩

lemma *sum_bij_rel'*:
assumes $f \in bij^M(A,C)$ $g \in bij^M(B,D)$ $M(f)$
$M(A)$ $M(C)$ $M(g)$ $M(B)$ $M(D)$
shows
$(\lambda z{\in}A+B.\ case(\lambda x.\ Inl(f`x),\ \lambda y.\ Inr(g`y),\ z)) \in bij(A+B,\ C+D)$
$M(\lambda z{\in}A+B.\ case(\lambda x.\ Inl(f`x),\ \lambda y.\ Inr(g`y),\ z))$
⟨*proof*⟩

lemma *sum_eqpoll_rel_cong*:
assumes $A \approx^M C$ $B \approx^M D$ $M(A)$ $M(C)$ $M(B)$ $M(D)$
shows $A+B \approx^M C+D$
⟨*proof*⟩

lemma *prod_bij_rel'*:
assumes $f \in bij^M(A,C)$ $g \in bij^M(B,D)$ $M(f)$
$M(A)$ $M(C)$ $M(g)$ $M(B)$ $M(D)$
shows
$(\lambda{<}x,y{>}{\in}A{*}B.\ {<}f`x,\ g`y{>}) \in bij(A{*}B,\ C{*}D)$
$M(\lambda{<}x,y{>}{\in}A{*}B.\ {<}f`x,\ g`y{>})$
⟨*proof*⟩

lemma *prod_eqpoll_rel_cong*:
assumes $A \approx^M C$ $B \approx^M D$ $M(A)$ $M(C)$ $M(B)$ $M(D)$
shows $A{\times}B \approx^M C{\times}D$
⟨*proof*⟩

lemma *inj_rel_disjoint_eqpoll_rel*:
[| $f \in inj^M(A,B)$; $A \cap B = 0$;$M(f)$; $M(A)$;$M(B)$ |] ==> $A \cup (B - range(f)) \approx^M B$
⟨*proof*⟩

lemma *Diff_sing_lepoll_rel*:
[| $a \in A$; $A \lesssim^M succ(n)$; $M(a)$; $M(A)$; $M(n)$ |] ==> $A - \{a\} \lesssim^M n$
⟨*proof*⟩

lemma *lepoll_rel_Diff_sing*:
assumes A: $succ(n) \lesssim^M A$
and *types*: $M(n)$ $M(A)$ $M(a)$
shows $n \lesssim^M A - \{a\}$
⟨*proof*⟩

lemma *Diff_sing_eqpoll_rel*: [| $a \in A$; $A \approx^M succ(n)$; $M(a)$; $M(A)$; $M(n)$ |] ==> $A - \{a\} \approx^M n$
⟨*proof*⟩

lemma *lepoll_rel_1_is_sing*: [| $A \lesssim^M 1$; $a \in A$;$M(a)$; $M(A)$ |] ==> $A = \{a\}$
⟨*proof*⟩

lemma *Un_lepoll_rel_sum*: $M(A) \Longrightarrow M(B) \Longrightarrow A \cup B \lesssim^M A+B$
⟨*proof*⟩

lemma *well_ord_Un_M*:
assumes *well_ord*(X,R) *well_ord*(Y,S)
and *types*: $M(X)$ $M(R)$ $M(Y)$ $M(S)$
shows $\exists T[M]$. *well_ord*($X \cup Y$, T)
⟨*proof*⟩

lemma *disj_Un_eqpoll_rel_sum*: $M(A) \Longrightarrow M(B) \Longrightarrow A \cap B = 0 \Longrightarrow A \cup B \approx^M A + B$
⟨*proof*⟩

lemma *eqpoll_rel_imp_Finite_rel_iff*: $A \approx^M B$ ==> $M(A) \Longrightarrow M(B) \Longrightarrow$ *Finite_rel*(M,A) $\longleftrightarrow$ *Finite_rel*(M,B)
⟨*proof*⟩

lemma *Finite_abs*[*simp*]: **assumes** $M(A)$ **shows** *Finite_rel*(M,A) $\longleftrightarrow$ *Finite*(A)
⟨*proof*⟩

lemma *lepoll_rel_nat_imp_Finite_rel*:
assumes A: $A \lesssim^M n$ **and** n: $n \in nat$
and *types*: $M(A)$ $M(n)$
shows *Finite_rel*(M,A)
⟨*proof*⟩

lemma *lesspoll_rel_nat_is_Finite_rel*:
$A \prec^M nat \Longrightarrow M(A) \Longrightarrow$ *Finite_rel*(M,A)
⟨*proof*⟩

lemma *lepoll_rel_Finite_rel*:
assumes Y: $Y \lesssim^M X$ **and** X: *Finite_rel*(M,X)
and *types*:$M(Y)$ $M(X)$
shows *Finite_rel*(M,Y)
⟨*proof*⟩

lemma *succ_lepoll_rel_imp_not_empty*: *succ*(x) $\lesssim^M y$ ==> $M(x) \Longrightarrow M(y) \Longrightarrow y \neq 0$
⟨*proof*⟩

lemma *eqpoll_rel_succ_imp_not_empty*: $x \approx^M$ *succ*(n) ==> $M(x) \Longrightarrow M(n) \Longrightarrow x \neq 0$
⟨*proof*⟩

lemma *Finite_subset_closed*:
  assumes *Finite*(*B*) *B*⊆*A* *M*(*A*)
  shows *M*(*B*)
⟨*proof*⟩

lemma *Finite_Pow_abs*:
  assumes *Finite*(*A*) *M*(*A*)
  shows *Pow*(*A*) = *Pow_rel*(*M*,*A*)
  ⟨*proof*⟩

lemma *Finite_Pow_rel*:
  assumes *Finite*(*A*) *M*(*A*)
  shows *Finite*(*Pow_rel*(*M*,*A*))
  ⟨*proof*⟩

lemma *Pow_rel_0* [*simp*]: *Pow_rel*(*M*,*0*) = {*0*}
  ⟨*proof*⟩

lemma *eqpoll_rel_imp_Finite*: *A* ≈^{M} *B* ⟹ *Finite*(*A*) ⟹ *M*(*A*) ⟹ *M*(*B*) ⟹ *Finite*(*B*)
⟨*proof*⟩

lemma *eqpoll_rel_imp_Finite_iff*: *A* ≈^{M} *B* ⟹ *M*(*A*) ⟹ *M*(*B*) ⟹ *Finite*(*A*) ⟷ *Finite*(*B*)
  ⟨*proof*⟩

end — *M_cardinals*

end

16 Relative, Choice-less Cardinal Arithmetic

theory *CardinalArith_Relative*
  imports
    Cardinal_Relative

begin

⟨*ML*⟩

definition
  csquare_lam :: *i*⇒*i* **where**
  csquare_lam(*K*) ≡ λ⟨*x*,*y*⟩∈*K*×*K*. ⟨*x* ∪ *y*, *x*, *y*⟩

— Can't do the next thing because split is a missing HOC

⟨*ML*⟩

definition
is_csquare_lam :: [*i*⇒*o*,*i*,*i*]⇒*o* **where**
is_csquare_lam(*M*,*K*,*l*) ≡ ∃ *K2*[*M*]. *cartprod*(*M*,*K*,*K*,*K2*) ∧
is_lambda(*M*,*K2*,*is_csquare_lam_body*(*M*),*l*)

definition *jump_cardinal_body* :: [*i*⇒*o*,*i*] ⇒ *i* **where**
jump_cardinal_body(*M*,*X*) ≡
{*z* . *r* ∈ *Pow*^{M}(*X* × *X*), *M*(*z*) ∧ *M*(*r*) ∧ *well_ord*(*X*, *r*) ∧ *z* = *ordertype*(*X*, *r*)}

lemma (**in** *M_cardinals*) *csquare_lam_closed*[*intro*,*simp*]: *M*(*K*) ⟹ *M*(*csquare_lam*(*K*))
⟨*proof*⟩

locale *M_pre_cardinal_arith* = *M_cardinals* +
assumes
wfrec_pred_replacement:*M*(*A*) ⟹ *M*(*r*) ⟹
wfrec_replacement(*M*, λ*x f z*. *z* = *f* ‘‘ *Order.pred*(*A*, *x*, *r*), *r*)
begin

lemma *ord_iso_separation*: *M*(*A*) ⟹ *M*(*r*) ⟹ *M*(*s*) ⟹
separation(*M*, λ*f*. ∀ *x*∈*A*. ∀ *y*∈*A*. ⟨*x*, *y*⟩ ∈ *r* ⟷ ⟨*f* ‘ *x*, *f* ‘ *y*⟩ ∈ *s*)
⟨*proof*⟩

end

locale *M_cardinal_arith* = *M_pre_cardinal_arith* +
assumes
ordertype_replacement :
M(*X*) ⟹ *strong_replacement*(*M*,λ *x z* . *M*(*z*) ∧ *M*(*x*) ∧ *x*∈*Pow_rel*(*M*,*X*×*X*) ∧ *well_ord*(*X*, *x*) ∧ *z*=*ordertype*(*X*,*x*))
and
strong_replacement_jc_body :
strong_replacement(*M*,λ *x z* . *M*(*z*) ∧ *M*(*x*) ∧ *z* = *jump_cardinal_body*(*M*,*x*))

lemmas (**in** *M_cardinal_arith*) *surj_imp_inj_replacement* =
surj_imp_inj_replacement1 surj_imp_inj_replacement2 surj_imp_inj_replacement4
lam_replacement_vimage_sing_fun[*THEN lam_replacement_imp_strong_replacement*]

⟨*ML*⟩

lemma (**in** *M_trivial*) *rmultP_abs* [*absolut*]: ⟦ *M*(*r*); *M*(*s*); *M*(*z*) ⟧ ⟹ *is_rmultP*(*M*,*s*,*r*,*z*) ⟷
(∃ *x′ y′ x y*. *z* = ⟨⟨*x′*, *y′*⟩, *x*, *y*⟩ ∧ (⟨*x′*, *x*⟩ ∈ *r* ∨ *x′* = *x* ∧ ⟨*y′*, *y*⟩ ∈ *s*))
⟨*proof*⟩

definition
is_csquare_rel :: [*i*⇒*o*,*i*,*i*]⇒*o* **where**

$is\_csquare\_rel(M,K,cs) \equiv \exists\, K2[M].\ \exists\, la[M].\ \exists\, memK[M].$
$\exists\, rmKK[M].\ \exists\, rmKK2[M].$
$cartprod(M,K,K,K2) \wedge is\_csquare\_lam(M,K,la) \wedge$
$membership(M,K,memK) \wedge is\_rmult(M,K,memK,K,memK,rmKK) \wedge$
$is\_rmult(M,K,memK,K2,rmKK,rmKK2) \wedge is\_rvimage(M,K2,la,rmKK2,cs)$

context *M_basic*
begin

lemma *rvimage_abs*[*absolut*]:
assumes $M(A)\ M(f)\ M(r)\ M(z)$
shows $is\_rvimage(M,A,f,r,z) \longleftrightarrow z = rvimage(A,f,r)$
⟨*proof*⟩

lemma *rmult_abs* [*absolut*]: $[\![\ M(A);\ M(r);\ M(B);\ M(s);\ M(z)\ ]\!] \Longrightarrow$
$is\_rmult(M,A,r,B,s,z) \longleftrightarrow z{=}rmult(A,r,B,s)$
⟨*proof*⟩

lemma *csquare_lam_body_abs*[*absolut*]: $M(x) \Longrightarrow M(z) \Longrightarrow$
$is\_csquare\_lam\_body(M,x,z) \longleftrightarrow z = <fst(x) \cup snd(x),\ fst(x),\ snd(x)>$
⟨*proof*⟩

lemma *csquare_lam_abs*[*absolut*]: $M(K) \Longrightarrow M(l) \Longrightarrow$
$is\_csquare\_lam(M,K,l) \longleftrightarrow l = (\lambda x{\in}K{\times}K.\ \langle fst(x) \cup snd(x),\ fst(x),\ snd(x)\rangle)$
⟨*proof*⟩

lemma *csquare_lam_eq_lam*:$csquare\_lam(K) = (\lambda z{\in}K{\times}K.\ <fst(z) \cup snd(z),$
$fst(z),\ snd(z)>)$
⟨*proof*⟩

end — *M_basic*

context *M_pre_cardinal_arith*
begin

lemma *csquare_rel_closed*[*intro*,*simp*]: $M(K) \Longrightarrow M(csquare\_rel(K))$
⟨*proof*⟩

lemma *csquare_rel_abs*[*absolut*]: $[\![\ M(K);\ M(cs)]\!] \Longrightarrow$
$is\_csquare\_rel(M,K,cs) \longleftrightarrow cs = csquare\_rel(K)$
⟨*proof*⟩

end — *M_pre_cardinal_arith*

⟨*ML*⟩

abbreviation

csucc_r :: $[i,i \Rightarrow o] \Rightarrow i$ ($\langle$'(_$^+$')–$\rangle$) **where**
csucc_r$(x,M) \equiv$ *csucc_rel*(M,x)

abbreviation
csucc_r_set :: $[i,i] \Rightarrow i$ ($\langle$'(_$^+$')–$\rangle$) **where**
csucc_r_set$(x,M) \equiv$ *csucc_rel*$(\#\#M,x)$

context *M_Perm*
begin

⟨*ML*⟩
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

end — *M_Perm*

notation *csucc_rel* ($\langle$*csucc*–'(_')$\rangle$)

context *M_cardinals*
begin

lemma *Card_rel_Union* [*simp*,*intro*,*TC*]:
assumes *A*: $\bigwedge x.\ x{\in}A \Longrightarrow Card^M(x)$ **and**
types:*M*(*A*)
shows $Card^M(\bigcup(A))$
⟨*proof*⟩

lemma *in_Card_imp_lesspoll*: [| $Card^M(K)$; $b \in K$; *M*(*K*); *M*(*b*) |] ==> $b \prec^M K$
⟨*proof*⟩

16.1 Cardinal addition

Note (Paulson): Could omit proving the algebraic laws for cardinal addition and multiplication. On finite cardinals these operations coincide with addition and multiplication of natural numbers; on infinite cardinals they coincide with union (maximum). Either way we get most laws for free.

16.1.1 Cardinal addition is commutative

lemma *sum_commute_eqpoll_rel*: $M(A) \Longrightarrow M(B) \Longrightarrow A{+}B \approx^M B{+}A$
⟨*proof*⟩

lemma *cadd_rel_commute*: $M(i) \Longrightarrow M(j) \Longrightarrow i \oplus^M j = j \oplus^M i$
⟨*proof*⟩

16.1.2 Cardinal addition is associative

lemma *sum_assoc_eqpoll_rel*: $M(A) \Longrightarrow M(B) \Longrightarrow M(C) \Longrightarrow (A+B)+C \approx^M A+(B+C)$
⟨*proof*⟩

Unconditional version requires AC

lemma *well_ord_cadd_rel_assoc*:
assumes *i*: *well_ord*(*i*,*ri*) **and** *j*: *well_ord*(*j*,*rj*) **and** *k*: *well_ord*(*k*,*rk*)
and
types: $M(i)$ $M(ri)$ $M(j)$ $M(rj)$ $M(k)$ $M(rk)$
shows $(i \oplus^M j) \oplus^M k = i \oplus^M (j \oplus^M k)$
⟨*proof*⟩

16.1.3 0 is the identity for addition

lemma *case_id_eq*: $x \in sum(A,B) \Longrightarrow case(\lambda z\ .\ z,\ \lambda z.\ z\ ,x) = snd(x)$
⟨*proof*⟩

lemma *lam_case_id*: $(\lambda z \in 0 + A.\ case(\lambda x.\ x,\ \lambda y.\ y,\ z)) = (\lambda z \in 0 + A\ .\ snd(z))$
⟨*proof*⟩

lemma *sum_0_eqpoll_rel*: $M(A) \Longrightarrow 0+A \approx^M A$
⟨*proof*⟩

lemma *cadd_rel_0* [*simp*]: $Card^M(K) \Longrightarrow M(K) \Longrightarrow 0 \oplus^M K = K$
⟨*proof*⟩

16.1.4 Addition by another cardinal

lemma *sum_lepoll_rel_self*: $M(A) \Longrightarrow M(B) \Longrightarrow A \lesssim^M A+B$
⟨*proof*⟩

lemma *cadd_rel_le_self*:
assumes *K*: $Card^M(K)$ **and** *L*: *Ord*(*L*) **and**
types:$M(K)$ $M(L)$
shows $K \le (K \oplus^M L)$
⟨*proof*⟩

16.1.5 Monotonicity of addition

lemma *sum_lepoll_rel_mono*:
$[|\ A \lesssim^M C;\ \ B \lesssim^M D;\ M(A);\ M(B);\ M(C);\ M(D)\ |] ==> A + B \lesssim^M C + D$
⟨*proof*⟩

lemma *cadd_rel_le_mono*:
[| $K' \leq K$; $L' \leq L$;$M(K')$;$M(K)$;$M(L')$;$M(L)$ |] ==> $(K' \oplus^M L') \leq (K \oplus^M L)$
⟨*proof*⟩

16.1.6 Addition of finite cardinals is "ordinary" addition

lemma *sum_succ_eqpoll_rel*: $M(A) \Longrightarrow M(B) \Longrightarrow succ(A)+B \approx^M succ(A+B)$
⟨*proof*⟩

lemma *cadd_succ_lemma*:
assumes *Ord*(m) *Ord*(n) **and**
types: $M(m)$ $M(n)$
shows $succ(m) \oplus^M n = |succ(m \oplus^M n)|^M$
⟨*proof*⟩

lemma *nat_cadd_rel_eq_add*:
assumes m: $m \in nat$ **and** [*simp*]: $n \in nat$ **shows**$m \oplus^M n = m +_\omega n$
⟨*proof*⟩

16.2 Cardinal multiplication

16.2.1 Cardinal multiplication is commutative

lemma *prod_commute_eqpoll_rel*: $M(A) \Longrightarrow M(B) \Longrightarrow A*B \approx^M B*A$
⟨*proof*⟩

lemma *cmult_rel_commute*: $M(i) \Longrightarrow M(j) \Longrightarrow i \otimes^M j = j \otimes^M i$
⟨*proof*⟩

16.2.2 Cardinal multiplication is associative

lemma *prod_assoc_eqpoll_rel*: $M(A) \Longrightarrow M(B) \Longrightarrow M(C) \Longrightarrow (A*B)*C \approx^M A*(B*C)$
⟨*proof*⟩

Unconditional version requires AC

lemma *well_ord_cmult_rel_assoc*:
assumes i: *well_ord*(i,ri) **and** j: *well_ord*(j,rj) **and** k: *well_ord*(k,rk)
and
types: $M(i)$ $M(ri)$ $M(j)$ $M(rj)$ $M(k)$ $M(rk)$
shows $(i \otimes^M j) \otimes^M k = i \otimes^M (j \otimes^M k)$
⟨*proof*⟩

16.2.3 Cardinal multiplication distributes over addition

lemma *sum_prod_distrib_eqpoll_rel*: $M(A) \Longrightarrow M(B) \Longrightarrow M(C) \Longrightarrow (A+B)*C \approx^M (A*C)+(B*C)$
⟨*proof*⟩

lemma *well_ord_cadd_cmult_distrib*:
assumes *i*: *well_ord*(*i*,*ri*) **and** *j*: *well_ord*(*j*,*rj*) **and** *k*: *well_ord*(*k*,*rk*)
and
types: $M(i)$ $M(ri)$ $M(j)$ $M(rj)$ $M(k)$ $M(rk)$
shows $(i \oplus^M j) \otimes^M k = (i \otimes^M k) \oplus^M (j \otimes^M k)$
⟨*proof*⟩

16.2.4 Multiplication by 0 yields 0

lemma *prod_0_eqpoll_rel*: $M(A) \Longrightarrow 0*A \approx^M 0$
⟨*proof*⟩

lemma *cmult_rel_0* [*simp*]: $M(i) \Longrightarrow 0 \otimes^M i = 0$
⟨*proof*⟩

16.2.5 1 is the identity for multiplication

lemma *prod_singleton_eqpoll_rel*: $M(x) \Longrightarrow M(A) \Longrightarrow \{x\}*A \approx^M A$
⟨*proof*⟩

lemma *cmult_rel_1* [*simp*]: $Card^M(K) \Longrightarrow M(K) \Longrightarrow 1 \otimes^M K = K$
⟨*proof*⟩

16.3 Some inequalities for multiplication

lemma *prod_square_lepoll_rel*: $M(A) \Longrightarrow A \lesssim^M A*A$
⟨*proof*⟩

lemma *cmult_rel_square_le*: $Card^M(K) \Longrightarrow M(K) \Longrightarrow K \leq K \otimes^M K$
⟨*proof*⟩

16.3.1 Multiplication by a non-zero cardinal

lemma *prod_lepoll_rel_self*: $b \in B \Longrightarrow M(b) \Longrightarrow M(B) \Longrightarrow M(A) \Longrightarrow A \lesssim^M A*B$
⟨*proof*⟩

lemma *cmult_rel_le_self*:
[| $Card^M(K)$; $Ord(L)$; $0<L$; $M(K)$;$M(L)$ |] ==> $K \leq (K \otimes^M L)$
⟨*proof*⟩

16.3.2 Monotonicity of multiplication

lemma *prod_lepoll_rel_mono*:
[| $A \lesssim^M C$; $B \lesssim^M D$; $M(A)$; $M(B)$; $M(C)$; $M(D)$|] ==> $A * B \lesssim^M C * D$
⟨*proof*⟩

lemma *cmult_rel_le_mono*:
[| $K' \leq K$; $L' \leq L$;$M(K')$;$M(K)$;$M(L')$;$M(L)$ |] ==> $(K' \otimes^M L') \leq (K \otimes^M L)$
⟨*proof*⟩

16.4 Multiplication of finite cardinals is "ordinary" multiplication

lemma *prod_succ_eqpoll_rel*: $M(A) \Longrightarrow M(B) \Longrightarrow succ(A)*B \approx^M B + A*B$
⟨*proof*⟩

lemma *cmult_rel_succ_lemma*:
[| $Ord(m)$; $Ord(n)$; $M(m)$; $M(n)$ |] ==> $succ(m) \otimes^M n = n \oplus^M (m \otimes^M n)$
⟨*proof*⟩

lemma *nat_cmult_rel_eq_mult*: [| $m \in nat$; $n \in nat$ |] ==> $m \otimes^M n = m\#*n$
⟨*proof*⟩

lemma *cmult_rel_2*: $Card^M(n) \Longrightarrow M(n) \Longrightarrow 2 \otimes^M n = n \oplus^M n$
⟨*proof*⟩

lemma *sum_lepoll_rel_prod*:
assumes C: $2 \lesssim^M C$ **and**
types:$M(C)$ $M(B)$
shows $B+B \lesssim^M C*B$
⟨*proof*⟩

lemma *lepoll_imp_sum_lepoll_prod*: [| $A \lesssim^M B$; $2 \lesssim^M A$; $M(A)$;$M(B)$ |] ==>
$A+B \lesssim^M A*B$
⟨*proof*⟩

end — *M_cardinals*

16.5 Infinite Cardinals are Limit Ordinals

context *M_pre_cardinal_arith*
begin

lemma *nat_cons_lepoll_rel*: $nat \lesssim^M A \Longrightarrow M(A) \Longrightarrow M(u) ==> cons(u,A) \lesssim^M A$
⟨*proof*⟩

lemma *nat_cons_eqpoll_rel*: $nat \lesssim^M A ==> M(A) \Longrightarrow M(u) \Longrightarrow cons(u,A) \approx^M A$
⟨*proof*⟩

lemma *nat_succ_eqpoll_rel*: $nat \subseteq A ==> M(A) \Longrightarrow succ(A) \approx^M A$
⟨*proof*⟩

lemma *InfCard_rel_nat*: $InfCard^M(nat)$

⟨*proof*⟩

lemma *InfCard_rel_is_Card_rel*: $M(K) \Longrightarrow \mathit{InfCard}^M(K) \Longrightarrow \mathit{Card}^M(K)$
⟨*proof*⟩

lemma *InfCard_rel_Un*:
[| $\mathit{InfCard}^M(K)$; $\mathit{Card}^M(L)$; $M(K)$; $M(L)$ |] ==> $\mathit{InfCard}^M(K \cup L)$
⟨*proof*⟩

lemma *InfCard_rel_is_Limit*: $\mathit{InfCard}^M(K)$ ==> $M(K) \Longrightarrow \mathit{Limit}(K)$
⟨*proof*⟩

end — *M_pre_cardinal_arith*

lemma (**in** *M_ordertype*) *ordertype_abs*[*absolut*]:
assumes *wellordered*(M,A,r) $M(A)$ $M(r)$ $M(i)$
shows *otype*(M,A,r,i) $\longleftrightarrow$ i = *ordertype*(A,r)
— Awful proof, it essentially repeats the same argument twice
⟨*proof*⟩

lemma (**in** *M_ordertype*) *ordertype_closed*[*intro*,*simp*]: ⟦ *wellordered*(M,A,r);$M(A)$;$M(r)$⟧
$\Longrightarrow$ M(*ordertype*(A,r))
⟨*proof*⟩

⟨*ML*⟩

lemma (**in** *M_trivial*) *is_transitive_iff_transitive_rel*:
$M(A)$$\Longrightarrow$ $M(r)$ $\Longrightarrow$ *transitive_rel*(M, A, r) $\longleftrightarrow$ *is_transitive*(M,A, r)
⟨*proof*⟩

⟨*ML*⟩

lemma (**in** *M_trivial*) *is_linear_iff_linear_rel*:
$M(A)$$\Longrightarrow$ $M(r)$ $\Longrightarrow$ *is_linear*(M,A, r) $\longleftrightarrow$ *linear_rel*(M, A, r)
⟨*proof*⟩

⟨*ML*⟩

lemma (**in** *M_trivial*) *is_wellfounded_on_iff_wellfounded_on*:
$M(A)$$\Longrightarrow$ $M(r)$ $\Longrightarrow$ *is_wellfounded_on*(M,A, r) $\longleftrightarrow$ *wellfounded_on*(M, A, r)
⟨*proof*⟩

definition
is_well_ord :: [i=>o,i,i]=>o **where**

— linear and wellfounded on A
is_well_ord(M,A,r) ==
is_transitive(M,A,r) ∧ *is_linear*(M,A,r) ∧ *is_wellfounded_on*(M,A,r)

lemma (**in** *M_trivial*) *is_well_ord_iff_wellordered*:
$M(A) \Longrightarrow M(r) \Longrightarrow$ *is_well_ord*(M,A, r) ⟷ *wellordered*(M, A, r)
⟨*proof*⟩

⟨*ML*⟩

context *M_pre_cardinal_arith*
begin

⟨*ML*⟩
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

end — *M_pre_cardinal_arith*

⟨*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* ⟹
$A \in nat \Longrightarrow$
$r \in nat \Longrightarrow$
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
[|$x \in nat$; $y \in nat$; $z \in nat$; *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*⟩

lemma *trans_on_iff_trans*: *trans*[*A*](*r*) ⟷ *trans*(*r* ∩ *A*×*A*)
⟨*proof*⟩

lemma *trans_on_subset*: *trans*[*A*](*r*) ⟹ *B* ⊆ *A* ⟹ *trans*[*B*](*r*)
⟨*proof*⟩

lemma *relation_Int*: *relation*(*r* ∩ *B*×*B*)
⟨*proof*⟩

Discipline for *ordermap*

⟨*ML*⟩

context *M_pre_cardinal_arith*
begin

lemma *wfrec_on_pred_eq*:
assumes *r* ∈ *Pow*(*A*×*A*) *M*(*A*) *M*(*r*)
shows *wfrec*[*A*](*r*, *x*, λ*x f*. *f* `` *Order.pred*(*A*, *x*, *r*)) = *wfrec*(*r*, *x*, λ*x f*. *f* ``
Order.pred(*A*, *x*, *r*))
⟨*proof*⟩

lemma *wfrec_on_pred_closed*:
assumes *wf*[*A*](*r*) *trans*[*A*](*r*) *r* ∈ *Pow*(*A*×*A*) *M*(*A*) *M*(*r*) *x* ∈ *A*
shows *M*(*wfrec*(*r*, *x*, λ*x f*. *f* `` *Order.pred*(*A*, *x*, *r*)))

⟨*proof*⟩

lemma *wfrec_on_pred_closed'*:
  assumes *wf*[*A*](*r*) *trans*[*A*](*r*) *r* ∈ *Pow*(*A*×*A*) *M*(*A*) *M*(*r*) *x* ∈ *A*
  shows *M*(*wfrec*[*A*](*r*, *x*, λ*x f*. *f* `` *Order.pred*(*A*, *x*, *r*)))
  ⟨*proof*⟩

lemma *ordermap_rel_closed'*:
  assumes *wf*[*A*](*r*) *trans*[*A*](*r*) *r* ∈ *Pow*(*A*×*A*) *M*(*A*) *M*(*r*)
  shows *M*(*ordermap_rel*(*M*, *A*, *r*))
⟨*proof*⟩

lemma *ordermap_rel_closed*[*intro*,*simp*]:
  assumes *wf*[*A*](*r*) *trans*[*A*](*r*) *r* ∈ *Pow*(*A*×*A*)
  shows *M*(*A*) ⟹ *M*(*r*) ⟹ *M*(*ordermap_rel*(*M*, *A*, *r*))
  ⟨*proof*⟩

lemma *is_ordermap_iff*:
  assumes *r* ∈ *Pow*(*A*×*A*) *wf*[*A*](*r*) *trans*[*A*](*r*)
    M(*A*) *M*(*r*) *M*(*res*)
  shows *is_ordermap*(*M*, *A*, *r*, *res*) ⟷ *res* = *ordermap_rel*(*M*, *A*, *r*)
⟨*proof*⟩

end — *M_pre_cardinal_arith*

⟨*ML*⟩

Discipline for *ordertype*

⟨*ML*⟩

context *M_pre_cardinal_arith*
begin

lemma *is_ordertype_iff*:
  assumes *r* ∈ *Pow*(*A*×*A*) *wf*[*A*](*r*) *trans*[*A*](*r*)
  shows *M*(*A*) ⟹ *M*(*r*) ⟹ *M*(*res*) ⟹ *is_ordertype*(*M*, *A*, *r*, *res*) ⟷ *res* = *ordertype_rel*(*M*, *A*, *r*)
  ⟨*proof*⟩

lemma *is_ordertype_iff'*:
  assumes *r* ∈ *Pow_rel*(*M*,*A*×*A*) *well_ord*(*A*,*r*)
  shows *M*(*A*) ⟹ *M*(*r*) ⟹ *M*(*res*) ⟹ *is_ordertype*(*M*, *A*, *r*, *res*) ⟷ *res* = *ordertype_rel*(*M*, *A*, *r*)
  ⟨*proof*⟩

lemma *is_ordertype_iff''*:
  assumes *well_ord*(*A*,*r*) *r*⊆*A*×*A*
  shows *M*(*A*) ⟹ *M*(*r*) ⟹ *M*(*res*) ⟹ *is_ordertype*(*M*, *A*, *r*, *res*) ⟷ *res* =

ordertype_rel(M, A, r)
⟨*proof*⟩

end — *M_pre_cardinal_arith*

⟨*ML*⟩
definition
jump_cardinal′ :: $i \Rightarrow i$ **where**
jump_cardinal′(K) ≡
$\bigcup X \in Pow(K).\ \{z.\ r \in Pow(X*X),\ well\_ord(X,r)\ \&\ z = ordertype(X,r)\}$

⟨*ML*⟩
definition *jump_cardinal_body′* **where**
jump_cardinal_body′(X) ≡ $\{z\ .\ r \in Pow(X \times X),\ \ well\_ord(X,\ r) \wedge z = ordertype(X,\ r)\}$

⟨*ML*⟩

context *M_pre_cardinal_arith*
begin

lemma *ordertype_rel_closed′*:
assumes *wf*[*A*](r) *trans*[*A*](r) $r \in Pow(A \times A)$ $M(r)$ $M(A)$
shows $M(ordertype\_rel(M,A,r))$
⟨*proof*⟩

lemma *ordertype_rel_closed*[*intro*,*simp*]:
assumes $well\_ord(A,r)$ $r \in Pow\_rel(M,A \times A)$ $M(A)$
shows $M(ordertype\_rel(M,A,r))$
⟨*proof*⟩

lemma *ordertype_rel_abs*:
assumes $wellordered(M,X,r)$ $M(X)$ $M(r)$
shows $ordertype\_rel(M,X,r) = ordertype(X,r)$
⟨*proof*⟩

lemma *univalent_aux1*: $M(X) \Longrightarrow univalent(M,Pow\_rel(M,X \times X),$
$\lambda r\ z.\ M(z) \wedge M(r) \wedge r \in Pow\_rel(M,X \times X) \wedge is\_well\_ord(M,\ X,\ r) \wedge is\_ordertype(M,\ X,\ r,\ z))$
⟨*proof*⟩

lemma *jump_cardinal_body_eq* :
$M(X) \Longrightarrow jump\_cardinal\_body(M,X) = jump\_cardinal\_body'\_rel(M,X)$
⟨*proof*⟩

end — *M_pre_cardinal_arith*

context *M_cardinal_arith*
begin

lemma *jump_cardinal_closed_aux1*:
assumes $M(X)$
shows
$M(jump\_cardinal\_body(M,X))$
⟨*proof*⟩

lemma *univalent_jc_body*: $M(X) \Longrightarrow univalent(M,X,\lambda\ x\ z\ .\ M(z) \wedge M(x) \wedge z = jump\_cardinal\_body(M,x))$
⟨*proof*⟩

lemma *jump_cardinal_body_closed*:
assumes $M(K)$
shows $M(\{a\ .\ X \in Pow^M(K),\ M(a) \wedge M(X) \wedge a = jump\_cardinal\_body(M,X)\})$
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

⟨*ML*⟩
⟨*proof*⟩

end

context *M_cardinal_arith*
begin

lemma (**in** *M_ordertype*) *ordermap_closed*[*intro*,*simp*]:
assumes *wellordered*(M,A,r) **and** *types*:$M(A)$ $M(r)$
shows $M(ordermap(A,r))$
⟨*proof*⟩

lemma *ordermap_eqpoll_pred*:
[| *well_ord*(A,r); $x \in A$; $M(A)$;$M(r)$;$M(x)$|] ==> $ordermap(A,r)\ `x \approx^M Order.pred(A,x,r)$
⟨*proof*⟩

Kunen: "each $\langle x,\ y\rangle \in K \times K$ has no more than $z \times z$ predecessors..." (page 29)

lemma *ordermap_csquare_le*:
assumes K: $Limit(K)$ **and** x: $x<K$ **and** y: $y<K$
and *types*: $M(K)$ $M(x)$ $M(y)$
shows $|ordermap(K \times K,\ csquare\_rel(K))\ `\langle x,y\rangle|^M \le |succ(succ(x \cup y))|^M \otimes^M |succ(succ(x \cup y))|^M$
⟨*proof*⟩

Kunen: "... so the order type is ≤ K"

lemma *ordertype_csquare_le_M*:

assumes *IK*: $InfCard^M(K)$ **and** *eq*: $\bigwedge y.\ y{\in}K \Longrightarrow InfCard^M(y) \Longrightarrow M(y) \Longrightarrow y \otimes^M y = y$
— Note the weakened hypothesis $[\![?y \in K;\ InfCard^M(?y);\ M(?y)]\!] \Longrightarrow ?y \otimes^M ?y = ?y$
and *types*: $M(K)$
shows $ordertype(K{*}K,\ csquare\_rel(K)) \leq K$
⟨*proof*⟩

lemma *InfCard_rel_csquare_eq*:
assumes *IK*: $InfCard^M(K)$ **and**
types: $M(K)$
shows $K \otimes^M K = K$
⟨*proof*⟩

lemma *well_ord_InfCard_rel_square_eq*:
assumes *r*: $well\_ord(A,r)$ **and** *I*: $InfCard^M(|A|^M)$ **and**
types: $M(A)\ M(r)$
shows $A \times A \approx^M A$
⟨*proof*⟩

lemma *InfCard_rel_square_eqpoll*:
assumes $InfCard^M(K)$ **and** *types*:$M(K)$ **shows** $K \times K \approx^M K$
⟨*proof*⟩

lemma *Inf_Card_rel_is_InfCard_rel*: [| $Card^M(i)$; ~ $Finite\_rel(M,i)$; $M(i)$ |] ==> $InfCard^M(i)$
⟨*proof*⟩

16.5.1 Toward's Kunen's Corollary 10.13 (1)

lemma *InfCard_rel_le_cmult_rel_eq*: [| $InfCard^M(K)$; $L \leq K$; $0{<}L$; $M(K)$; $M(L)$ |] ==> $K \otimes^M L = K$
⟨*proof*⟩

lemma *InfCard_rel_cmult_rel_eq*: [| $InfCard^M(K)$; $InfCard^M(L)$; $M(K)$; $M(L)$ |] ==> $K \otimes^M L = K \cup L$
⟨*proof*⟩

lemma *InfCard_rel_cdouble_eq*: $InfCard^M(K) \Longrightarrow M(K) \Longrightarrow K \oplus^M K = K$
⟨*proof*⟩

lemma *InfCard_rel_le_cadd_rel_eq*: [| $InfCard^M(K)$; $L \leq K$; $M(K)$; $M(L)$|] ==> $K \oplus^M L = K$
⟨*proof*⟩

lemma *InfCard_rel_cadd_rel_eq*: [| $InfCard^M(K)$; $InfCard^M(L)$; $M(K)$; $M(L)$ |] ==> $K \oplus^M L = K \cup L$
⟨*proof*⟩

end — *M_cardinal_arith*

16.6 For Every Cardinal Number There Exists A Greater One

This result is Kunen's Theorem 10.16, which would be trivial using AC

locale *M_cardinal_arith_jump* = *M_cardinal_arith* + *M_ordertype*
begin

lemma *well_ord_restr*: $well\_ord(X, r) \Longrightarrow well\_ord(X, r \cap X \times X)$
⟨*proof*⟩

lemma *ordertype_restr_eq* :
assumes $well\_ord(X,r)$
shows $ordertype(X, r) = ordertype(X, r \cap X \times X)$
⟨*proof*⟩

lemma *def_jump_cardinal_rel_aux*:
$X \in Pow^M(K) \Longrightarrow well\_ord(X, w) \Longrightarrow M(K) \Longrightarrow$
$\{z . r \in Pow^M(X \times X), M(z) \wedge well\_ord(X, r) \wedge z = ordertype(X, r)\} =$
$\{z . r \in Pow^M(K \times K), M(z) \wedge well\_ord(X, r) \wedge z = ordertype(X, r)\}$
⟨*proof*⟩

lemma *def_jump_cardinal_rel*:
assumes $M(K)$
shows *jump_cardinal'_rel*$(M,K) =$
$(\bigcup X \in Pow\_rel(M,K).\ \{z.\ r \in Pow\_rel(M,K*K),\ well\_ord(X,r)\ \&\ z = ordertype(X,r)\})$
⟨*proof*⟩

notation *jump_cardinal'_rel* (‹*jump'_cardinal'_rel*›)

lemma *Ord_jump_cardinal_rel*: $M(K) \Longrightarrow Ord(jump\_cardinal\_rel(M,K))$
⟨*proof*⟩

declare *conj_cong* [*cong del*]
— incompatible with some of the proofs of the original theory

lemma *jump_cardinal_rel_iff_old*:
$M(i) \Longrightarrow M(K) \Longrightarrow i \in jump\_cardinal\_rel(M,K) \longleftrightarrow$
$(\exists\, r[M].\ \exists\, X[M].\ r \subseteq K*K\ \&\ X \subseteq K\ \&\ well\_ord(X,r)\ \&\ i = ordertype(X,r))$
⟨*proof*⟩

lemma *K_lt_jump_cardinal_rel*: $Ord(K) ==> M(K) \Longrightarrow K < jump\_cardinal\_rel(M,K)$
⟨*proof*⟩

lemma *Card_rel_jump_cardinal_rel_lemma*:
[| $well\_ord(X,r)$; $r \subseteq K * K$; $X \subseteq K$;
$f \in bij(ordertype(X,r), jump\_cardinal\_rel(M,K))$;
$M(X)$; $M(r)$; $M(K)$; $M(f)$ |]
==> $jump\_cardinal\_rel(M,K) \in jump\_cardinal\_rel(M,K)$
⟨*proof*⟩

lemma *Card_rel_jump_cardinal_rel*: $M(K) \Longrightarrow Card\_rel(M,jump\_cardinal\_rel(M,K))$
⟨*proof*⟩

16.7 Basic Properties of Successor Cardinals

lemma *csucc_rel_basic*: $Ord(K) ==> M(K) \Longrightarrow Card\_rel(M,csucc\_rel(M,K))$
& $K < csucc\_rel(M,K)$
⟨*proof*⟩

lemmas *Card_rel_csucc_rel* = *csucc_rel_basic* [*THEN conjunct1*]

lemmas *lt_csucc_rel* = *csucc_rel_basic* [*THEN conjunct2*]

lemma *Ord_0_lt_csucc_rel*: $Ord(K) ==> M(K) \Longrightarrow 0 < csucc\_rel(M,K)$
⟨*proof*⟩

lemma *csucc_rel_le*: [| $Card\_rel(M,L)$; $K<L$; $M(K)$; $M(L)$ |] ==> $csucc\_rel(M,K)$
$\leq L$
⟨*proof*⟩

lemma *lt_csucc_rel_iff*: [| $Ord(i)$; $Card\_rel(M,K)$; $M(K)$; $M(i)$|] ==> $i <$
$csucc\_rel(M,K) \longleftrightarrow |i|^M \leq K$
⟨*proof*⟩

lemma *Card_rel_lt_csucc_rel_iff*:
[| $Card\_rel(M,K')$; $Card\_rel(M,K)$; $M(K')$; $M(K)$ |] ==> $K' < csucc\_rel(M,K)$
$\longleftrightarrow K' \leq K$
⟨*proof*⟩

lemma *InfCard_rel_csucc_rel*: $InfCard\_rel(M,K) \Longrightarrow M(K) ==> InfCard\_rel(M,csucc\_rel(M,K))$
⟨*proof*⟩

16.7.1 Theorems by Krzysztof Grabczewski, proofs by lcp

lemma *nat_sum_eqpoll_rel_sum*:
assumes *m*: $m \in nat$ **and** *n*: $n \in nat$ **shows** $m + n \approx^M m +_\omega n$

⟨*proof*⟩

lemma *Ord_nat_subset_into_Card_rel*: [| *Ord*(*i*); *i* ⊆ *nat* |] ==> $Card^M(i)$
⟨*proof*⟩

end — *M_cardinal_arith_jump*
end
theory *Aleph_Relative*
imports
CardinalArith_Relative
begin

definition
HAleph :: [*i*,*i*] ⇒ *i* **where**
HAleph(*i*,*r*) ≡ *if*(¬(*Ord*(*i*)),*i*,*if*(*i*=*0*, *nat*, *if*(¬*Limit*(*i*) ∧ *i*≠*0*,
csucc(*r*‘(⋃ *i*)),
⋃*j*∈*i*. *r*‘*j*)))

⟨*ML*⟩

lemma *arity_is_HAleph_fm_aux*:
assumes
i ∈ *nat* *r* ∈ *nat*
— NOTE: assumptions are **not** used, but if omitted, next lemma fails!
shows
arity(*Replace_fm*(*8* +ω *i*, *·10* +ω *r*‘*0 is 1*·, *3*)) = *9* +ω *i* ∪ *pred*(*pred*(*11* +ω *r*))
⟨*proof*⟩

lemma *arity_is_HAleph_fm*[*arity*]:
assumes
i ∈ *nat* *r* ∈ *nat* *l* ∈ *nat*
shows
arity(*is_HAleph_fm*(*i*, *r*, *l*)) = *succ*(*i*) ∪ *succ*(*l*) ∪ *succ*(*r*)
⟨*proof*⟩

definition
Aleph′ :: *i* => *i* **where**
Aleph′(*a*) == *transrec*(*a*,λ*i* *r*. *HAleph*(*i*,*r*))

⟨*ML*⟩

The extra assumptions *a* < *length*(*env*) and *c* < *length*(*env*) in this schematic goal (and the following results on synthesis that depend on it) are imposed by ⟦⋀*a0 a1 a2 a3 a4 a5 a6 a7*. ⟦*a0* ∈ *?A*; *a1* ∈ *?A*; *a2* ∈ *?A*; *a3* ∈ *?A*; *a4* ∈ *?A*; *a5* ∈ *?A*; *a6* ∈ *?A*; *a7* ∈ *?A*⟧ ⟹ *?MH*(*a2*, *a1*, *a0*) ⟷ *?A*, *Cons*(*a0*, *Cons*(*a1*, *Cons*(*a2*, *Cons*(*a3*, *Cons*(*a4*, *Cons*(*a5*, *Cons*(*a6*, *Cons*(*a7*, *?env*)))))))) ⊨ *?p*; *nth*(*?i*, *?env*) = *?x*; *nth*(*?k*, *?env*) = *?z*; *?i* < *length*(*?env*); *?k* < *length*(*?env*); *?env* ∈ *list*(*?A*)⟧ ⟹ *is_transrec*(*##?A*, *?MH*, *?x*, *?z*) ⟷ *?A*, *?env* ⊨ *is_transrec_fm*(*?p*, *?i*, *?k*).

schematic_goal *sats_is_Aleph_fm_auto*:
$a \in nat \Longrightarrow c \in nat \Longrightarrow env \in list(A) \Longrightarrow$
$a < length(env) \Longrightarrow c < length(env) \Longrightarrow 0 \in A \Longrightarrow$
$is\_Aleph(\#\#A,\ nth(a,\ env),\ nth(c,\ env)) \longleftrightarrow A,\ env \models ?fm(a,\ c)$
⟨*proof*⟩

⟨*ML*⟩

notation *is_Aleph_fm* (‹·א′(_′) *is* _·›)

lemma *is_Aleph_fm_type* [*TC*]: $a \in nat \Longrightarrow c \in nat \Longrightarrow is\_Aleph\_fm(a,\ c) \in formula$
⟨*proof*⟩

lemma *sats_is_Aleph_fm*:
assumes $f{\in}nat\ r{\in}nat\ env \in list(A)\ 0{\in}A\ f < length(env)\ r{<}\ length(env)$
shows $is\_Aleph(\#\#A,\ nth(f,\ env),\ nth(r,\ env)) \longleftrightarrow A,\ env \models is\_Aleph\_fm(f,r)$
⟨*proof*⟩

lemma *is_Aleph_iff_sats* [*iff_sats*]:
assumes
$nth(f,\ env) = fa\ nth(r,\ env) = ra\ f < length(env)\ r{<}\ length(env)$
$f \in nat\ r \in nat\ env \in list(A)\ 0{\in}A$
shows $is\_Aleph(\#\#A,fa,ra) \longleftrightarrow A,\ env \models is\_Aleph\_fm(f,r)$
⟨*proof*⟩

⟨*ML*⟩

lemma (**in** *M_cardinal_arith_jump*) *is_Limit_iff*:
assumes $M(a)$
shows $is\_Limit(M,a) \longleftrightarrow Limit(a)$
⟨*proof*⟩

lemma *HAleph_eq_Aleph_recursive*:
$Ord(i) \Longrightarrow HAleph(i,r) = (if\ i = 0\ then\ nat$
$else\ if\ \exists j.\ i = succ(j)\ then\ csucc(r\ `\ (THE\ j.\ i = succ(j)))\ else\ \bigcup j{<}i.\ r\ `\ j)$
⟨*proof*⟩

lemma *Aleph′_eq_Aleph*: $Ord(a) \Longrightarrow Aleph'(a) = Aleph(a)$
⟨*proof*⟩

⟨*ML*⟩

abbreviation
$Aleph\_r :: [i,i{\Rightarrow}o] \Rightarrow i$ (‹א_—›) **where**
$Aleph\_r(a,M) \equiv Aleph\_rel(M,a)$

abbreviation

Aleph_r_set :: $[i,i] \Rightarrow i$ ($\langle\aleph\_-\rangle$) **where**
Aleph_r_set$(a,M) \equiv$ *Aleph_rel*$(\#\#M,a)$

lemma *Aleph_rel_def'*: *Aleph_rel*$(M,a) \equiv$ *transrec*$(a, \lambda i\ r.$ *HAleph_rel*$(M,\ i,\ r))$
⟨*proof*⟩

lemma *succ_mem_Limit*: *Limit*$(j) \Longrightarrow i \in j \Longrightarrow$ *succ*$(i) \in j$
⟨*proof*⟩

locale *M_pre_aleph* = *M_eclose* + *M_cardinal_arith_jump* +
assumes
haleph_transrec_replacement: $M(a) \Longrightarrow$ *transrec_replacement*$(M,$*is_HAleph*$(M),a)$

begin

lemma *aux_ex_Replace_funapply*:
assumes $M(a)$ $M(f)$
shows $\exists\, x[M].$ *is_Replace*$(M,\ a,\ \lambda j\ y.\ f\ `\ j = y,\ x)$
⟨*proof*⟩

lemma *is_HAleph_zero*:
assumes $M(f)$
shows *is_HAleph*$(M,0,f,res) \longleftrightarrow res =$ *nat*
⟨*proof*⟩

lemma *is_HAleph_succ*:
assumes $M(f)$ $M(x)$ *Ord*(x) $M(res)$
shows *is_HAleph*$(M,$*succ*$(x),f,res) \longleftrightarrow res =$ *csucc_rel*$(M,f`x)$
⟨*proof*⟩

lemma *is_HAleph_limit*:
assumes $M(f)$ $M(x)$ *Limit*(x) $M(res)$
shows *is_HAleph*$(M,x,f,res) \longleftrightarrow res = (\bigcup\{y\ .\ i{\in}x\ ,M(i) \wedge M(y) \wedge y = f`i\})$
⟨*proof*⟩

lemma *is_HAleph_iff*:
assumes $M(a)$ $M(f)$ $M(res)$
shows *is_HAleph*$(M,\ a,\ f,\ res) \longleftrightarrow res =$ *HAleph_rel*$(M,\ a,\ f)$
⟨*proof*⟩

lemma *HAleph_rel_closed* [*intro*,*simp*]:
assumes *function*(f) $M(a)$ $M(f)$
shows $M($*HAleph_rel*$(M,a,f))$
⟨*proof*⟩

lemma *Aleph_rel_closed*[*intro*, *simp*]:
assumes *Ord*(a) $M(a)$
shows $M($*Aleph_rel*$(M,a))$
⟨*proof*⟩

lemma *Aleph_rel_zero*: $\aleph_0{}^M = nat$
⟨*proof*⟩

lemma *Aleph_rel_succ*: $Ord(\alpha) \Longrightarrow M(\alpha) \Longrightarrow \aleph_{succ(\alpha)}{}^M = (\aleph_\alpha{}^{M+})^M$
⟨*proof*⟩

lemma *Aleph_rel_limit*:
assumes $Limit(\alpha)$ $M(\alpha)$
shows $\aleph_\alpha{}^M = \bigcup\{\aleph_j{}^M \,.\, j \in \alpha\}$
⟨*proof*⟩

lemma *is_Aleph_iff*:
assumes $Ord(a)$ $M(a)$ $M(res)$
shows $is\_Aleph(M,\ a,\ res) \longleftrightarrow res = \aleph_a{}^M$
⟨*proof*⟩

end — *M_pre_aleph*

locale *M_aleph* = *M_pre_aleph* +
assumes
aleph_rel_replacement: $strong\_replacement(M,\ \lambda x\ y.\ Ord(x) \wedge y = \aleph_x{}^M)$
begin

lemma *Aleph_rel_cont*: $Limit(l) \Longrightarrow M(l) \Longrightarrow \aleph_l{}^M = (\bigcup i{<}l.\ \aleph_i{}^M)$
⟨*proof*⟩

lemma *Ord_Aleph_rel*:
assumes $Ord(a)$
shows $M(a) \Longrightarrow Ord(\aleph_a{}^M)$
⟨*proof*⟩

lemma *Card_rel_Aleph_rel* [*simp*, *intro*]:
assumes $Ord(a)$ **and** *types*: $M(a)$ **shows** $Card^M(\aleph_a{}^M)$
⟨*proof*⟩

lemma *Aleph_rel_increasing*:
assumes $a < b$ **and** *types*: $M(a)$ $M(b)$
shows $\aleph_a{}^M < \aleph_b{}^M$
⟨*proof*⟩

lemmas *nat_subset_Aleph_rel_1* =
Ord_lt_subset[*OF Ord_Aleph_rel*[*of 1*] *Aleph_rel_increasing*[*of 0 1*,*simplified*],*simplified*]

end — *M_aleph*

end

17 Relative, Cardinal Arithmetic Using AC

theory *Cardinal_AC_Relative*
imports
CardinalArith_Relative

begin

locale *M_AC* =
fixes *M*
assumes
choice_ax: *choice_ax*(*M*)

locale *M_cardinal_AC* = *M_cardinal_arith* + *M_AC*
begin

lemma *well_ord_surj_imp_lepoll_rel*:
assumes *well_ord*(*A*,*r*) *h* ∈ *surj*(*A*,*B*) **and**
types:*M*(*A*) *M*(*r*) *M*(*h*) *M*(*B*)
shows *B* $\lesssim^{M}$ *A*
⟨*proof*⟩

lemma *surj_imp_well_ord_M*:
assumes *wos*: *well_ord*(*A*,*r*) *h* ∈ *surj*(*A*,*B*)
and
types: *M*(*A*) *M*(*r*) *M*(*h*) *M*(*B*)
shows ∃ *s*[*M*]. *well_ord*(*B*,*s*)
⟨*proof*⟩

lemma *choice_ax_well_ord*: *M*(*S*) ⟹ ∃ *r*[*M*]. *well_ord*(*S*,*r*)
⟨*proof*⟩

lemma *Finite_cardinal_rel_Finite*:
assumes *Finite*($|i|^{M}$) *M*(*i*)
shows *Finite*(*i*)
⟨*proof*⟩

end — *M_cardinal_AC*

locale *M_Pi_assumptions_choice* = *M_Pi_assumptions* + *M_cardinal_AC* +
assumes
B_replacement: *strong_replacement*(*M*, λ*x y*. *y* = *B*(*x*))
and
— The next one should be derivable from (some variant) of B_replacement. Proving both instances each time seems inconvenient.
minimum_replacement: *M*(*r*) ⟹ *strong_replacement*(*M*, λ*x y*. *y* = ⟨*x*, *minimum*(*r*, *B*(*x*))⟩)
begin

lemma *AC_M*:
  assumes $a \in A$ $\bigwedge x.\ x \in A \Longrightarrow \exists\, y.\ y \in B(x)$
  shows $\exists\, z[M].\ z \in Pi^M(A,\ B)$
⟨*proof*⟩

lemma *AC_Pi_rel*: **assumes** $\bigwedge x.\ x \in A \Longrightarrow \exists\, y.\ y \in B(x)$
  shows $\exists\, z[M].\ z \in Pi^M(A,\ B)$
⟨*proof*⟩

end — *M_Pi_assumptions_choice*

context *M_cardinal_AC*
begin

17.1 Strengthened Forms of Existing Theorems on Cardinals

lemma *cardinal_rel_eqpoll_rel*: $M(A) \Longrightarrow |A|^M \approx^M A$
  ⟨*proof*⟩

lemmas *cardinal_rel_idem* = *cardinal_rel_eqpoll_rel* [*THEN cardinal_rel_cong*, *simp*]

lemma *cardinal_rel_eqE*: $|X|^M = |Y|^M$ ==> $M(X) \Longrightarrow M(Y) \Longrightarrow X \approx^M Y$
  ⟨*proof*⟩

lemma *cardinal_rel_eqpoll_rel_iff*: $M(X) \Longrightarrow M(Y) \Longrightarrow |X|^M = |Y|^M \longleftrightarrow X \approx^M Y$
  ⟨*proof*⟩

lemma *cardinal_rel_disjoint_Un*:
  [| $|A|^M$=$|B|^M$; $|C|^M$=$|D|^M$; $A \cap C = 0$; $B \cap D = 0$; $M(A)$; $M(B)$; $M(C)$; $M(D)$|]
    ==> $|A \cup C|^M = |B \cup D|^M$
  ⟨*proof*⟩

lemma *lepoll_rel_imp_cardinal_rel_le*: $A \lesssim^M B$ ==> $M(A) \Longrightarrow M(B) \Longrightarrow |A|^M \le |B|^M$
  ⟨*proof*⟩

lemma *cadd_rel_assoc*: $[\![M(i);\ M(j);\ M(k)]\!] \Longrightarrow (i \oplus^M j) \oplus^M k = i \oplus^M (j \oplus^M k)$
  ⟨*proof*⟩

lemma *cmult_rel_assoc*: $[\![M(i);\ M(j);\ M(k)]\!] \Longrightarrow (i \otimes^M j) \otimes^M k = i \otimes^M (j \otimes^M k)$
  ⟨*proof*⟩

lemma *cadd_cmult_distrib*: $[\![M(i);\ M(j);\ M(k)]\!] \Longrightarrow (i \oplus^M j) \otimes^M k = (i \otimes^M k) \oplus^M (j \otimes^M k)$
⟨*proof*⟩

lemma *InfCard_rel_square_eq*: $InfCard^M(|A|^M) \Longrightarrow M(A) \Longrightarrow A \times A \approx^M A$
⟨*proof*⟩

17.2 The relationship between cardinality and le-pollence

lemma *Card_rel_le_imp_lepoll_rel*:
assumes $|A|^M \le |B|^M$
and *types*: $M(A)$ $M(B)$
shows $A \lesssim^M B$
⟨*proof*⟩

lemma *le_Card_rel_iff*: $Card^M(K) ==> M(K) \Longrightarrow M(A) \Longrightarrow |A|^M \le K \longleftrightarrow A \lesssim^M K$
⟨*proof*⟩

lemma *cardinal_rel_0_iff_0* [*simp*]: $M(A) \Longrightarrow |A|^M = 0 \longleftrightarrow A = 0$
⟨*proof*⟩

lemma *cardinal_rel_lt_iff_lesspoll_rel*:
assumes *i*: $Ord(i)$ **and**
types: $M(i)$ $M(A)$
shows $i < |A|^M \longleftrightarrow i \prec^M A$
⟨*proof*⟩

lemma *cardinal_rel_le_imp_lepoll_rel*: $i \le |A|^M ==> M(i) \Longrightarrow M(A) \Longrightarrow i \lesssim^M A$
⟨*proof*⟩

17.3 Other Applications of AC

We have an example of instantiating a locale involving higher order variables inside a proof, by using the assumptions of the first order, active locale.

lemma *surj_rel_implies_inj_rel*:
assumes *f*: $f \in surj^M(X, Y)$ **and**
types: $M(f)$ $M(X)$ $M(Y)$
shows $\exists\, g[M].\ g \in inj^M(Y, X)$
⟨*proof*⟩

Kunen's Lemma 10.20

lemma *surj_rel_implies_cardinal_rel_le*:
assumes *f*: $f \in surj^M(X, Y)$ **and**
types: $M(f)$ $M(X)$ $M(Y)$
shows $|Y|^M \le |X|^M$
⟨*proof*⟩

end — *M_cardinal_AC*

The set-theoretic universe.

abbreviation
Universe :: $i \Rightarrow o$ ($\langle \mathcal{V} \rangle$) **where**
$\mathcal{V}(x) \equiv True$

lemma *separation_absolute*: *separation*($\mathcal{V}$, P)
⟨*proof*⟩

lemma *univalent_absolute*:
assumes *univalent*($\mathcal{V}$, A, P) $P(x, b)$ $x \in A$
shows $P(x, y) \Longrightarrow y = b$
⟨*proof*⟩

lemma *replacement_absolute*: *strong_replacement*($\mathcal{V}$, P)
⟨*proof*⟩

lemma *Union_ax_absolute*: *Union_ax*($\mathcal{V}$)
⟨*proof*⟩

lemma *upair_ax_absolute*: *upair_ax*($\mathcal{V}$)
⟨*proof*⟩

lemma *power_ax_absolute*:*power_ax*($\mathcal{V}$)
⟨*proof*⟩

locale *M_cardinal_UN* = *M_Pi_assumptions_choice* _ *K X* **for** *K X* +
assumes
— The next assumption is required by ($\bigwedge x.\ [\![?Q(x);\ Ord(x)]\!] \Longrightarrow \exists\, y[M].\ ?Q(y) \wedge Ord(y)) \Longrightarrow M(\mu\ x.\ ?Q(x))$
X_witness_in_M: $w \in X(x) \Longrightarrow M(x)$
and
lam_m_replacement:$M(f) \Longrightarrow$ *strong_replacement*(M,
$\lambda x\ y.\ y = \langle x,\ \mu\ i.\ x \in X(i),\ f\ `\ (\mu\ i.\ x \in X(i))\ `\ x \rangle$)
and
inj_replacement:
$M(x) \Longrightarrow$ *strong_replacement*(M, $\lambda y\ z.\ y \in inj^M(X(x),\ K) \wedge z = \{\langle x,\ y \rangle\}$)
strong_replacement(M, $\lambda x\ y.\ y = inj^M(X(x),\ K)$)
strong_replacement(M,
$\lambda x\ z.\ z = Sigfun(x,\ \lambda i.\ inj^M(X(i),\ K))$)
$M(r) \Longrightarrow$ *strong_replacement*(M,
$\lambda x\ y.\ y = \langle x,\ minimum(r,\ inj^M(X(x),\ K)) \rangle$)

begin

lemma *UN_closed*: $M(\bigcup i \in K.\ X(i))$
⟨*proof*⟩

Kunen's Lemma 10.21

lemma *cardinal_rel_UN_le*:
assumes K: $InfCard^M(K)$
shows $(\bigwedge i.\ i{\in}K \Longrightarrow |X(i)|^M \leq K) \Longrightarrow |\bigcup i{\in}K.\ X(i)|^M \leq K$
⟨*proof*⟩

end — *M_cardinal_UN*

end

18 Relativization of Finite Functions

theory *FiniteFun_Relative*
imports
Lambda_Replacement
begin

lemma *FiniteFunI* :
assumes $f{\in}Fin(A{\times}B)$ $function(f)$
shows $f \in A$ -||> B
⟨*proof*⟩

18.1 The set of finite binary sequences

We implement the poset for adding one Cohen real, the set $2^{<\omega}$ of finite binary sequences.

definition
$seqspace :: [i,i] \Rightarrow i$ ($‹\_^{<}\!\!-›$ $[100,1]100$) **where**
$B^{<\alpha} \equiv \bigcup n{\in}\alpha.\ (n{\rightarrow}B)$

schematic_goal *seqspace_fm_auto*:
assumes
$i \in nat$ $j \in nat$ $h{\in}nat$ $env \in list(A)$
shows
$(\exists\, om{\in}A.\ omega(\#\#A,om) \wedge nth(i,env) \in om \wedge is\_funspace(\#\#A,\ nth(i,env),\ nth(h,env),\ nth(j,env))) \longleftrightarrow (A,\ env \models (?sqsprp(i,j,h)))$
⟨*proof*⟩

⟨*ML*⟩

lemma *seqspaceI*[*intro*]: $n{\in}\alpha \Longrightarrow f{:}n{\rightarrow}B \Longrightarrow f{\in}B^{<\alpha}$
⟨*proof*⟩

lemma *seqspaceD*[*dest*]: $f{\in}B^{<\alpha} \Longrightarrow \exists\, n{\in}\alpha.\ f{:}n{\rightarrow}B$
⟨*proof*⟩

locale *M_seqspace* = *M_trancl* + *M_replacement* +
assumes

seqspace_replacement: $M(B) \Longrightarrow$ *strong_replacement*(M,$\lambda n\ z.\ n{\in}nat \wedge$ *is_funspace*(M,n,B,z))
begin

lemma *seqspace_closed*:
$M(B) \Longrightarrow M(B^{<\omega})$
⟨*proof*⟩
end

18.2 Representation of finite functions

A function $f \in A \rightarrow_{fin} B$ can be represented by a function $g \in |f| \rightarrow A \times B$. It is clear that f can be represented by any $g' = g{\cdot}\pi$, where π is a permutation $\pi \in dom(g) \rightarrow dom(g)$. We use this representation of $A \rightarrow_{fin} B$ to prove that our model is closed under $\_ \rightarrow_{fin} \_$.

A function $g \in n \rightarrow A \times B$ that is functional in the first components.

definition *cons_like* :: $i \Rightarrow o$ **where**
cons_like(f) $\equiv \forall\ i{\in}domain(f)\ .\ \forall\, j{\in}i\ .\ fst(f`i) \neq fst(f`j)$

⟨*ML*⟩

lemma (**in** *M_seqspace*) *cons_like_abs*:
$M(f) \Longrightarrow$ *cons_like*(f) $\longleftrightarrow$ *cons_like_rel*(M,f)
⟨*proof*⟩

definition *FiniteFun_iso* :: [i,i,i,i,i] $\Rightarrow o$ **where**
FiniteFun_iso(A,B,n,g,f) $\equiv\ (\forall\ i{\in}n\ .\ g`i \in f) \wedge (\forall\ ab{\in}f.\ (\exists\ i{\in}n.\ g`i{=}ab))$

From a function $g \in n \rightarrow A \times B$ we obtain a finite function in A -||> B.

definition *to_FiniteFun* :: $i \Rightarrow i$ **where**
to_FiniteFun(f) $\equiv \{f`i.\ i{\in}domain(f)\}$

definition *FiniteFun_Repr* :: [i,i] $\Rightarrow i$ **where**
FiniteFun_Repr(A,B) $\equiv \{f \in (A{\times}B)^{<\omega}\ .\ cons\_like(f)\ \}$

locale *M_FiniteFun* = *M_seqspace* +
assumes
cons_like_separation : *separation*(M,$\lambda f.$ *cons_like_rel*(M,f))
and
separation_is_function : *separation*(M, *is_function*(M))
begin

lemma *supset_separation*: *separation*(M, $\lambda\ x.\ \exists\, a.\ \exists\, b.\ x = \langle a,b\rangle \wedge b \subseteq a$)
⟨*proof*⟩

lemma *to_finiteFun_replacement*: *strong_replacement*(M, $\lambda x\ y.\ y = range(x)$)
⟨*proof*⟩

lemma *fun_range_eq*: $f \in A \to B \Longrightarrow \{f`i \;.\; i \in domain(f)\;\} = range(f)$
⟨*proof*⟩

lemma *FiniteFun_fst_type*:
assumes $h \in A \text{-}||> B$ $p \in h$
shows $fst(p) \in domain(h)$
⟨*proof*⟩

lemma *FinFun_closed*:
$M(A) \Longrightarrow M(B) \Longrightarrow M(\bigcup\{n \to A \times B \;.\; n \in \omega\})$
⟨*proof*⟩

lemma *cons_like_lt* :
assumes $n \in \omega$ $f \in succ(n) \to A \times B$ *cons_like*(f)
shows *restrict*$(f,n) \in n \to A \times B$ *cons_like*(*restrict*(f,n))
⟨*proof*⟩

A finite function $f \in A \text{-}||> B$ can be represented by a function $g \in n \to A \times B$, with $n = |f|$.

lemma *FiniteFun_iso_intro1*:
assumes $f \in (A \text{-}||> B)$
shows $\exists\, n \in \omega \;.\; \exists\, g \in n \to A \times B.$ *FiniteFun_iso*$(A,B,n,g,f) \wedge$ *cons_like*(g)
⟨*proof*⟩

All the representations of $f \in A \text{-}||> B$ are equal.

lemma *FiniteFun_isoD* :
assumes $n \in \omega$ $g \in n \to A \times B$ $f \in A \text{-}||> B$ *FiniteFun_iso*(A,B,n,g,f)
shows *to_FiniteFun*$(g) = f$
⟨*proof*⟩

lemma *to_FiniteFun_succ_eq* :
assumes $n \in \omega$ $f \in succ(n) \to A$
shows *to_FiniteFun*(f) = *cons*$(f`n$,*to_FiniteFun*(*restrict*$(f,n)))$
⟨*proof*⟩

If $g \in n \to A \times B$ is *cons_like*, then it is a representation of *to_FiniteFun*(g).

lemma *FiniteFun_iso_intro_to*:
assumes $n \in \omega$ $g \in n \to A \times B$ *cons_like*(g)
shows *to_FiniteFun*$(g) \in (A \text{-}||> B) \wedge$ *FiniteFun_iso*$(A,B,n,g,$*to_FiniteFun*$(g))$
⟨*proof*⟩

lemma *FiniteFun_iso_intro2*:
assumes $n \in \omega$ $f \in n \to A \times B$ *cons_like*(f)
shows $\exists\;\; g \in (A \text{-}||> B)$. *FiniteFun_iso*(A,B,n,f,g)
⟨*proof*⟩

lemma *FiniteFun_eq_range_Repr* :
shows $\{$*range*(h) . $h \in$ *FiniteFun_Repr*$(A,B)\;\}$ = $\{$*to_FiniteFun*(h) . $h \in$ *FiniteFun_Repr*$(A,B)\;\}$

⟨*proof*⟩

lemma *FiniteFun_eq_to_FiniteFun_Repr* :
shows A-||>B = {*to_FiniteFun*(h) . $h \in$ *FiniteFun_Repr*(A,B) }
(**is** *?Y*=*?X*)
⟨*proof*⟩

lemma *FiniteFun_Repr_closed* :
assumes $M(A)$ $M(B)$
shows M(*FiniteFun_Repr*(A,B))
⟨*proof*⟩

lemma *to_FiniteFun_closed*:
assumes $M(A)$ $f \in A$
shows M(*range*(f))
⟨*proof*⟩

lemma *To_FiniteFun_Repr_closed* :
assumes $M(A)$ $M(B)$
shows M({*range*(h) . $h \in$ *FiniteFun_Repr*(A,B) })
⟨*proof*⟩

lemma *FiniteFun_closed*[*intro*,*simp*] :
assumes $M(A)$ $M(B)$
shows M(A -||> B)
⟨*proof*⟩

end — *M_FiniteFun*

end

19 Library of basic *ZF* results

theory *ZF_Library_Relative*
imports
Aleph_Relative— must be before Cardinal_AC_Relative!
Cardinal_AC_Relative
FiniteFun_Relative
begin

no_notation *sum* (**infixr** ‹+› *65*)
notation *oadd* (**infixl** ‹+› *65*)

lemma (**in** *M_cardinal_arith_jump*) *csucc_rel_cardinal_rel*:
assumes *Ord*(κ) $M(\kappa)$
shows $(|\kappa|^{M+})^M = (\kappa^+)^M$
⟨*proof*⟩

lemma (**in** *M_cardinal_arith_jump*) *csucc_rel_le_mono*:
assumes $\kappa \leq \nu$ $M(\kappa)$ $M(\nu)$
shows $(\kappa^+)^M \leq (\nu^+)^M$
⟨*proof*⟩

lemma (**in** *M_cardinal_AC*) *cardinal_rel_succ_not_0*: $|A|^M = succ(n) \Longrightarrow M(A) \Longrightarrow M(n) \Longrightarrow A \neq 0$
⟨*proof*⟩

⟨*ML*⟩

notation *Finite_to_one_rel* (‹*Finite'_to'_one–'*(_,_')›)

abbreviation
Finite_to_one_r_set :: [*i*,*i*,*i*] ⇒ *i* (‹*Finite'_to'_one–'*(_,_')›) **where**
$Finite\_to\_one^M(X,Y) \equiv Finite\_to\_one\_rel(\#\#M,X,Y)$

locale *M_ZF_library* = *M_cardinal_arith* + *M_aleph* + *M_FiniteFun* + *M_replacement_extra*
begin

lemma *Finite_Collect_imp*: $Finite(\{x{\in}X\ .\ Q(x)\}) \Longrightarrow Finite(\{x{\in}X\ .\ M(x) \wedge Q(x)\})$
(**is** *Finite*(*?A*) ⟹ *Finite*(*?B*))
⟨*proof*⟩

lemma *Finite_to_one_relI*[*intro*]:
assumes $f{:}X{\rightarrow}^M Y$ $\bigwedge y.\ y{\in}Y \Longrightarrow Finite(\{x{\in}X\ .\ f`x = y\})$
and *types*:$M(f)$ $M(X)$ $M(Y)$
shows $f \in Finite\_to\_one^M(X,Y)$
⟨*proof*⟩

lemma *Finite_to_one_relI'*[*intro*]:
assumes $f{:}X{\rightarrow}^M Y$ $\bigwedge y.\ y{\in}Y \Longrightarrow Finite(\{x{\in}X\ .\ M(x) \wedge f`x = y\})$
and *types*:$M(f)$ $M(X)$ $M(Y)$
shows $f \in Finite\_to\_one^M(X,Y)$
⟨*proof*⟩

lemma *Finite_to_one_relD*[*dest*]:
$f \in Finite\_to\_one^M(X,Y) \Longrightarrow f{:}X{\rightarrow}^M Y$
$f \in Finite\_to\_one^M(X,Y) \Longrightarrow y{\in}Y \Longrightarrow M(Y) \Longrightarrow Finite(\{x{\in}X\ .\ M(x) \wedge f`x = y\})$
⟨*proof*⟩

lemma *Diff_bij_rel*:
assumes $\forall A{\in}F.\ X \subseteq A$
and *types*: $M(F)$ $M(X)$ **shows** $(\lambda A{\in}F.\ A\text{-}X) \in bij^M(F, \{A\text{-}X.\ A{\in}F\})$
⟨*proof*⟩

lemma *function_space_rel_nonempty*:
  assumes $b{\in}B$ **and** *types*: $M(B)$ $M(A)$
  shows $(\lambda x{\in}A.\ b) : A \to^M B$
⟨*proof*⟩

lemma *mem_function_space_rel*:
  assumes $f \in A \to^M y$ $M(A)$ $M(y)$
  shows $f \in A \to y$
  ⟨*proof*⟩

lemmas *range_fun_rel_subset_codomain* = *range_fun_subset_codomain*[*OF mem_function_space_rel*]

end — *M_ZF_library*

context *M_Pi_assumptions*
begin

lemma *mem_Pi_rel*: $f \in Pi^M(A,B) \Longrightarrow f \in Pi(A,\ B)$
  ⟨*proof*⟩

lemmas *Pi_rel_rangeD* = *Pi_rangeD*[*OF mem_Pi_rel*]

lemmas *rel_apply_Pair* = *apply_Pair*[*OF mem_Pi_rel*]

lemmas *rel_apply_rangeI* = *apply_rangeI*[*OF mem_Pi_rel*]

lemmas *Pi_rel_range_eq* = *Pi_range_eq*[*OF mem_Pi_rel*]

lemmas *Pi_rel_vimage_subset* = *Pi_vimage_subset*[*OF mem_Pi_rel*]

end — *M_Pi_assumptions*

context *M_ZF_library*
begin

lemma *mem_bij_rel*: $[\![f \in bij^M(A,B);\ M(A);\ M(B)]\!] \Longrightarrow f{\in}bij(A,B)$
  ⟨*proof*⟩

lemma *mem_inj_rel*: $[\![f \in inj^M(A,B);\ M(A);\ M(B)]\!] \Longrightarrow f{\in}inj(A,B)$
  ⟨*proof*⟩

lemma *mem_surj_rel*: $[\![f \in surj^M(A,B);\ M(A);\ M(B)]\!] \Longrightarrow f{\in}surj(A,B)$
  ⟨*proof*⟩

lemmas *rel_apply_in_range* = *apply_in_range*[*OF _ _ mem_function_space_rel*]

lemmas *rel_range_eq_image* = *ZF_Library.range_eq_image*[*OF mem_function_space_rel*]

lemmas *rel_Image_sub_codomain* = *Image_sub_codomain*[*OF mem_function_space_rel*]

lemma *rel_inj_to_Image*: ⟦*f*:*A*→^{M}*B*; *f* ∈ *inj*^{M}(*A*,*B*); *M*(*A*); *M*(*B*)⟧ ⟹ *f* ∈ *inj*^{M}(*A*,*f*‘‘*A*)
⟨*proof*⟩

lemma *inj_rel_imp_surj_rel*:
fixes *f b*
defines [*simp*]: *ifx*(*x*) ≡ *if x*∈*range*(*f*) *then converse*(*f*)‘*x else b*
assumes *f* ∈ *inj*^{M}(*B*,*A*) *b*∈*B* **and** *types*: *M*(*f*) *M*(*B*) *M*(*A*)
shows (λ*x*∈*A*. *ifx*(*x*)) ∈ *surj*^{M}(*A*,*B*)
⟨*proof*⟩

lemma *function_space_rel_disjoint_Un*:
assumes *f* ∈ *A*→^{M}*B g* ∈ *C*→^{M}*D* *A* ∩ *C* = *0*
and *types*:*M*(*A*) *M*(*B*) *M*(*C*) *M*(*D*)
shows *f* ∪ *g* ∈ (*A* ∪ *C*)→^{M} (*B* ∪ *D*)
⟨*proof*⟩

lemma *restrict_eq_imp_Un_into_function_space_rel*:
assumes *f* ∈ *A*→^{M}*B g* ∈ *C*→^{M}*D* *restrict*(*f*, *A* ∩ *C*) = *restrict*(*g*, *A* ∩ *C*)
and *types*:*M*(*A*) *M*(*B*) *M*(*C*) *M*(*D*)
shows *f* ∪ *g* ∈ (*A* ∪ *C*)→^{M} (*B* ∪ *D*)
⟨*proof*⟩

lemma *lepoll_relD*[*dest*]: *A* ≲^{M} *B* ⟹ ∃*f*[*M*]. *f* ∈ *inj*^{M}(*A*, *B*)
⟨*proof*⟩
lemma *lepoll_relI*[*intro*]: *f* ∈ *inj*^{M}(*A*, *B*) ⟹ *M*(*f*) ⟹ *A* ≲^{M} *B*
⟨*proof*⟩

lemma *eqpollD*[*dest*]: *A* ≈^{M} *B* ⟹ ∃*f*[*M*]. *f* ∈ *bij*^{M}(*A*, *B*)
⟨*proof*⟩
lemma *bij_rel_imp_eqpoll_rel*[*intro*]: *f* ∈ *bij*^{M}(*A*,*B*) ⟹ *M*(*f*) ⟹ *A* ≈^{M} *B*
⟨*proof*⟩

lemma *restrict_bij_rel*:— Unused
assumes *f* ∈ *inj*^{M}(*A*,*B*) *C*⊆*A*
and *types*:*M*(*A*) *M*(*B*) *M*(*C*)
shows *restrict*(*f*,*C*)∈ *bij*^{M}(*C*, *f*‘‘*C*)
⟨*proof*⟩

lemma *range_of_subset_eqpoll_rel*:
assumes *f* ∈ *inj*^{M}(*X*, *Y*) *S* ⊆ *X*
and *types*:*M*(*X*) *M*(*Y*) *M*(*S*)
shows *S* ≈^{M} *f* ‘‘ *S*
⟨*proof*⟩

lemmas *inj_rel_is_fun* = *inj_is_fun*[*OF mem_inj_rel*]

lemma *inj_rel_bij_rel_range*: $f \in inj^M(A,B) \Longrightarrow M(A) \Longrightarrow M(B) \Longrightarrow f \in bij^M(A,range(f))$
⟨*proof*⟩

lemma *bij_rel_is_inj_rel*: $f \in bij^M(A,B) \Longrightarrow M(A) \Longrightarrow M(B) \Longrightarrow f \in inj^M(A,B)$
⟨*proof*⟩

lemma *inj_rel_weaken_type*: $[|\ f \in inj^M(A,B);\ \ B \subseteq D;\ M(A);\ M(B);\ M(D)\ |] ==> f \in inj^M(A,D)$
⟨*proof*⟩

lemma *bij_rel_converse_bij_rel* [*TC*]: $f \in bij^M(A,B) \Longrightarrow M(A) \Longrightarrow M(B) ==> converse(f): bij^M(B,A)$
⟨*proof*⟩

lemma *bij_rel_is_fun_rel*: $f \in bij^M(A,B) \Longrightarrow M(A) \Longrightarrow M(B) \Longrightarrow f \in A \rightarrow^M B$
⟨*proof*⟩

lemmas *bij_rel_is_fun* = *bij_rel_is_fun_rel*[*THEN mem_function_space_rel*]

lemma *comp_bij_rel*:
$g \in bij^M(A,B) \Longrightarrow f \in bij^M(B,C) \Longrightarrow M(A) \Longrightarrow M(B) \Longrightarrow M(C) \Longrightarrow (f\ O\ g) \in bij^M(A,C)$
⟨*proof*⟩

lemma *inj_rel_converse_fun*: $f \in inj^M(A,B) \Longrightarrow M(A) \Longrightarrow M(B) \Longrightarrow converse(f) \in range(f) \rightarrow^M A$
⟨*proof*⟩

lemma *fg_imp_bijective_rel*:
assumes $f \in A \rightarrow^M B\ \ g \in B \rightarrow^M A\ \ f\ O\ g = id(B)\ g\ O\ f = id(A)\ M(A)\ M(B)$
shows $f \in bij^M(A,B)$
⟨*proof*⟩

end — *M_ZF_library*

⟨*ML*⟩

context *M_ZF_library*
begin

⟨*ML*⟩
⟨*proof*⟩

⟨*ML*⟩ ⟨*proof*⟩

end — *M_ZF_library*

⟨*ML*⟩
notation *is_cexp_fm* (‹·_ ↑− *is* _·›)
⟨*ML*⟩

abbreviation
  cexp_r :: [*i*,*i*,*i*⇒*o*] ⇒ *i* (‹_ ↑−,−›) **where**
  cexp_r(*x*,*y*,*M*) ≡ *cexp_rel*(*M*,*x*,*y*)

abbreviation
  cexp_r_set :: [*i*,*i*,*i*] ⇒ *i* (‹_ ↑−,−›) **where**
  cexp_r_set(*x*,*y*,*M*) ≡ *cexp_rel*(##*M*,*x*,*y*)

context *M_ZF_library*
begin

lemma *Card_rel_cexp_rel*: *M*(*κ*) ⟹ *M*(*ν*) ⟹ *Card*^{M}(*κ*$^{\uparrow\nu,M}$)
  ⟨*proof*⟩
declare *conj_cong*[*cong*]

lemma *eq_csucc_rel_ord*:
  Ord(*i*) ⟹ *M*(*i*) ⟹ (*i*$^+$)^M = (|*i*|$^{M+}$)^M
  ⟨*proof*⟩

lemma *lesspoll_succ_rel*:
  assumes *Ord*(*κ*) *M*(*κ*)
  shows *κ* ≲^{M} (*κ*$^+$)^M
  ⟨*proof*⟩

lemma *lesspoll_rel_csucc_rel*:
  assumes *Ord*(*κ*)
    and *types*:*M*(*κ*) *M*(*d*)
  shows *d* ≺^{M} (*κ*$^+$)^M ⟷ *d* ≲^{M} *κ*
⟨*proof*⟩

lemma *Infinite_imp_nats_lepoll*:
  assumes *Infinite*(*X*) *n* ∈ *ω*
  shows *n* ≲ *X*
  ⟨*proof*⟩

lemma *nepoll_imp_nepoll_rel* :
  assumes ¬ *x* ≈ *X* *M*(*x*) *M*(*X*)
  shows ¬ (*x* ≈^{M} *X*)
  ⟨*proof*⟩

lemma *Infinite_imp_nats_lepoll_rel*:
  assumes *Infinite*(*X*) *n* ∈ *ω*
    and *types*: *M*(*X*)
  shows *n* ≲^{M} *X*
  ⟨*proof*⟩

lemma *lepoll_rel_imp_lepoll*: $A \lesssim^M B \Longrightarrow M(A) \Longrightarrow M(B) \Longrightarrow A \lesssim B$
⟨*proof*⟩

lemma *zero_lesspoll_rel*: **assumes** $0<\kappa$ $M(\kappa)$ **shows** $0 \prec^M \kappa$
⟨*proof*⟩

lemma *lepoll_rel_nat_imp_Infinite*: $\omega \lesssim^M X \Longrightarrow M(X) \Longrightarrow Infinite(X)$
⟨*proof*⟩

lemma *InfCard_rel_imp_Infinite*: $InfCard^M(\kappa) \Longrightarrow M(\kappa) \Longrightarrow Infinite(\kappa)$
⟨*proof*⟩

lemma *lt_surj_rel_empty_imp_Card_rel*:
assumes $Ord(\kappa)$ $\bigwedge\alpha.\ \alpha < \kappa \Longrightarrow surj^M(\alpha,\kappa) = 0$
and *types*:$M(\kappa)$
shows $Card^M(\kappa)$
⟨*proof*⟩

end — *M_ZF_library*

⟨*ML*⟩

notation *mono_map_rel* (‹*mono′_map–′*(_,_,_,_′)›)

abbreviation
mono_map_r_set :: [*i*,*i*,*i*,*i*,*i*]⇒*i* (‹*mono′_map–′*(_,_,_,_′)›) **where**
$mono\_map^M(a,r,b,s) \equiv mono\_map\_rel(\#\#M,a,r,b,s)$

context *M_ZF_library*
begin

lemma *mono_map_rel_char*:
assumes $M(a)$ $M(b)$
shows $mono\_map^M(a,r,b,s) = \{f \in mono\_map(a,r,b,s)\ .\ M(f)\}$
⟨*proof*⟩

Just a sample of porting results on *mono_map*

lemma *mono_map_rel_mono*:
assumes
$f \in mono\_map^M(A,r,B,s)$ $B \subseteq C$
and *types*:$M(A)$ $M(B)$ $M(C)$
shows
$f \in mono\_map^M(A,r,C,s)$
⟨*proof*⟩

lemma *nats_le_InfCard_rel*:
assumes $n \in \omega$ $InfCard^M(\kappa)$
shows $n \leq \kappa$

⟨*proof*⟩

lemma *nat_into_InfCard_rel*:
assumes $n \in \omega$ $InfCard^M(\kappa)$
shows $n \in \kappa$
⟨*proof*⟩

lemma *Finite_lesspoll_rel_nat*:
assumes *Finite*(*x*) $M(x)$
shows $x \prec^M nat$
⟨*proof*⟩

lemma *Finite_cardinal_rel_in_nat* [*simp*]:
assumes *Finite*(*A*) $M(A)$ **shows** $|A|^M \in \omega$
⟨*proof*⟩

lemma *Finite_cardinal_rel_eq_cardinal*:
assumes *Finite*(*A*) $M(A)$ **shows** $|A|^M = |A|$
⟨*proof*⟩

lemma *Finite_imp_cardinal_rel_cons*:
assumes *FA*: *Finite*(*A*) **and** *a*: $a \notin A$ **and** *types*:$M(A)$ $M(a)$
shows $|cons(a,A)|^M = succ(|A|^M)$
⟨*proof*⟩

lemma *Finite_imp_succ_cardinal_rel_Diff*:
assumes *Finite*(*A*) $a \in A$ $M(A)$
shows $succ(|A\text{-}\{a\}|^M) = |A|^M$
⟨*proof*⟩

lemma *InfCard_rel_Aleph_rel*:
notes *Aleph_rel_zero*[*simp*]
assumes *Ord*(*α*)
and *types*: $M(\alpha)$
shows $InfCard^M(\aleph_\alpha{}^M)$
⟨*proof*⟩

lemmas *Limit_Aleph_rel* = *InfCard_rel_Aleph_rel*[*THEN InfCard_rel_is_Limit*]

bundle *Ord_dests* = *Limit_is_Ord*[*dest*] *Card_rel_is_Ord*[*dest*]
bundle *Aleph_rel_dests* = *Aleph_rel_cont*[*dest*]
bundle *Aleph_rel_intros* = *Aleph_rel_increasing*[*intro*!]
bundle *Aleph_rel_mem_dests* = *Aleph_rel_increasing*[*OF ltI*, *THEN ltD*, *dest*]

lemma *f_imp_injective_rel*:
assumes $f \in A \rightarrow^M B$ $\forall x{\in}A.\ d(f\ `\ x) = x$ $M(A)$ $M(B)$
shows $f \in inj^M(A, B)$
⟨*proof*⟩

lemma *lam_injective_rel*:
assumes $\bigwedge x.\ x \in A \Longrightarrow c(x) \in B$
$\bigwedge x.\ x \in A \Longrightarrow d(c(x)) = x$
$\forall\, x[M].\ M(c(x))\ lam\_replacement(M,c)$
$M(A)\ M(B)$
shows $(\lambda x{\in}A.\ c(x)) \in inj^M(A,\ B)$
⟨*proof*⟩

lemma *f_imp_surjective_rel*:
assumes $f \in A \to^M B \bigwedge y.\ y \in B \Longrightarrow d(y) \in A \bigwedge y.\ y \in B \Longrightarrow f\ `\ d(y) = y$
$M(A)\ M(B)$
shows $f \in surj^M(A,\ B)$
⟨*proof*⟩

lemma *lam_surjective_rel*:
assumes $\bigwedge x.\ x \in A \Longrightarrow c(x) \in B$
$\bigwedge y.\ y \in B \Longrightarrow d(y) \in A$
$\bigwedge y.\ y \in B \Longrightarrow c(d(y)) = y$
$\forall\, x[M].\ M(c(x))\ lam\_replacement(M,c)$
$M(A)\ M(B)$
shows $(\lambda x{\in}A.\ c(x)) \in surj^M(A,\ B)$
⟨*proof*⟩

lemma *lam_bijective_rel*:
assumes $\bigwedge x.\ x \in A \Longrightarrow c(x) \in B$
$\bigwedge y.\ y \in B \Longrightarrow d(y) \in A$
$\bigwedge x.\ x \in A \Longrightarrow d(c(x)) = x$
$\bigwedge y.\ y \in B \Longrightarrow c(d(y)) = y$
$\forall\, x[M].\ M(c(x))\ lam\_replacement(M,c)$
$M(A)\ M(B)$
shows $(\lambda x{\in}A.\ c(x)) \in bij^M(A,\ B)$
⟨*proof*⟩

lemma *function_space_rel_eqpoll_rel_cong*:
assumes
$A \approx^M A'\ B \approx^M B'\ M(A)\ M(A')\ M(B)\ M(B')$
shows
$A \to^M B \approx^M A' \to^M B'$
⟨*proof*⟩

lemma *curry_eqpoll_rel*:
fixes $\nu 1\ \nu 2\ \ \kappa$
assumes $M(\nu 1)\ M(\nu 2)\ M(\kappa)$
shows $\nu 1 \to^M (\nu 2 \to^M \kappa) \approx^M \nu 1 \times \nu 2 \to^M \kappa$
⟨*proof*⟩

lemma *Pow_rel_eqpoll_rel_function_space_rel*:
fixes $d\ X$
notes *bool_of_o_def* [*simp*]

defines [*simp*]:$d(A) \equiv (\lambda x{\in}X.\ bool\_of\_o(x{\in}A))$
— the witnessing map for the thesis:
assumes $M(X)$
shows $Pow^M(X) \approx^M X \rightarrow^M 2$
⟨*proof*⟩

lemma *Pow_rel_bottom*: $M(B) \Longrightarrow 0 \in Pow^M(B)$
⟨*proof*⟩

lemma *cantor_surj_rel*:
assumes $M(f)$ $M(A)$
shows $f \notin surj^M(A,Pow^M(A))$
⟨*proof*⟩

lemma *cantor_inj_rel*: $M(f) \Longrightarrow M(A) \Longrightarrow f \notin inj^M(Pow^M(A),A)$
⟨*proof*⟩

end — *M_ZF_library*

end

20 Lambda-replacements required for cardinal inequalities

theory *Replacement_Lepoll*
imports
ZF_Library_Relative
begin

definition
lepoll_assumptions1 :: $[i{\Rightarrow}o,i,[i,i]{\Rightarrow}i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions1$(M,A,F,S,fa,K,x,f,r) \equiv \forall x{\in}S.\ strong\_replacement(M,\ \lambda y\ z.\ y \in F(A,\ x) \wedge z = \{\langle x,\ y\rangle\})$

definition
lepoll_assumptions2 :: $[i{\Rightarrow}o,i,[i,i]{\Rightarrow}i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions2$(M,A,F,S,fa,K,x,f,r) \equiv strong\_replacement(M,\ \lambda x\ z.\ z = Sigfun(x,\ F(A)))$

definition
lepoll_assumptions3 :: $[i{\Rightarrow}o,i,[i,i]{\Rightarrow}i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions3$(M,A,F,S,fa,K,x,f,r) \equiv strong\_replacement(M,\ \lambda x\ y.\ y = F(A,\ x))$

definition
lepoll_assumptions4 :: $[i{\Rightarrow}o,i,[i,i]{\Rightarrow}i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions4$(M,A,F,S,fa,K,x,f,r) \equiv strong\_replacement(M,\ \lambda x\ y.\ y = \langle x,\ minimum(r,\ F(A,\ x))\rangle)$

definition

lepoll_assumptions5 :: $[i \Rightarrow o,i,[i,i] \Rightarrow i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions5$(M,A,F,S,fa,K,x,f,r) \equiv$
strong_replacement$(M, \lambda x\ y.\ y = \langle x, \mu\ i.\ x \in F(A,\ i), f\ `\ (\mu\ i.\ x \in F(A,\ i))\ `\ x\rangle)$

definition
lepoll_assumptions6 :: $[i \Rightarrow o,i,[i,i] \Rightarrow i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions6$(M,A,F,S,fa,K,x,f,r) \equiv$ *strong_replacement*$(M, \lambda y\ z.\ y \in inj^M(F(A,\ x),S) \wedge z = \{\langle x,\ y\rangle\})$

definition
lepoll_assumptions7 :: $[i \Rightarrow o,i,[i,i] \Rightarrow i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions7$(M,A,F,S,fa,K,x,f,r) \equiv$ *strong_replacement*$(M, \lambda x\ y.\ y = inj^M(F(A,\ x),S))$

definition
lepoll_assumptions8 :: $[i \Rightarrow o,i,[i,i] \Rightarrow i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions8$(M,A,F,S,fa,K,x,f,r) \equiv$ *strong_replacement*$(M, \lambda x\ z.\ z = Sigfun(x, \lambda i.\ inj^M(F(A,\ i),S)))$

definition
lepoll_assumptions9 :: $[i \Rightarrow o,i,[i,i] \Rightarrow i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions9$(M,A,F,S,fa,K,x,f,r) \equiv$ *strong_replacement*$(M, \lambda x\ y.\ y = \langle x,\ minimum(r,\ inj^M(F(A,\ x),S))\rangle)$

definition
lepoll_assumptions10 :: $[i \Rightarrow o,i,[i,i] \Rightarrow i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions10$(M,A,F,S,fa,K,x,f,r) \equiv$ *strong_replacement*
$(M, \lambda x\ z.\ z = Sigfun(x, \lambda k.\ if\ k \in range(f)\ then\ F(A,\ converse(f)\ `\ k)\ else\ 0))$

definition
lepoll_assumptions11 :: $[i \Rightarrow o,i,[i,i] \Rightarrow i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions11$(M,A,F,S,fa,K,x,f,r) \equiv$ *strong_replacement*$(M, \lambda x\ y.\ y = (if\ x \in range(f)\ then\ F(A,\ converse(f)\ `\ x)\ else\ 0))$

definition
lepoll_assumptions12 :: $[i \Rightarrow o,i,[i,i] \Rightarrow i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions12$(M,A,F,S,fa,K,x,f,r) \equiv$ *strong_replacement*$(M, \lambda y\ z.\ y \in F(A,\ converse(f)\ `\ x) \wedge z = \{\langle x,\ y\rangle\})$

definition
lepoll_assumptions13 :: $[i \Rightarrow o,i,[i,i] \Rightarrow i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions13$(M,A,F,S,fa,K,x,f,r) \equiv$ *strong_replacement*
$(M, \lambda x\ y.\ y = \langle x,\ minimum(r,\ if\ x \in range(f)\ then\ F(A,converse(f)\ `\ x)\ else\ 0)\rangle)$

definition
lepoll_assumptions14 :: $[i \Rightarrow o,i,[i,i] \Rightarrow i,i,i,i,i,i,i] \Rightarrow o$ **where**
lepoll_assumptions14$(M,A,F,S,fa,K,x,f,r) \equiv$ *strong_replacement*

$(M,\ \lambda x\ y.\ y = \langle x,\ \mu\ i.\ x \in (\mathit{if}\ i \in \mathit{range}(f)\ \mathit{then}\ F(A,\ \mathit{converse}(f)\ `\ i)\ \mathit{else}\ 0),$
$\mathit{fa}\ `\ (\mu\ i.\ x \in (\mathit{if}\ i \in \mathit{range}(f)\ \mathit{then}\ F(A,\ \mathit{converse}(f)\ `\ i)\ \mathit{else}\ 0))\ `\ x\rangle)$

definition
$\mathit{lepoll\_assumptions15} :: [i \Rightarrow o, i, [i,i] \Rightarrow i, i, i, i, i, i, i] \Rightarrow o$ **where**
$\mathit{lepoll\_assumptions15}(M,A,F,S,\mathit{fa},K,x,f,r) \equiv \mathit{strong\_replacement}$
$(M,\ \lambda y\ z.\ y \in \mathit{inj}^M(\mathit{if}\ x \in \mathit{range}(f)\ \mathit{then}\ F(A,\ \mathit{converse}(f)\ `\ x)\ \mathit{else}\ 0,K) \wedge z = \{\langle x,\ y\rangle\})$

definition
$\mathit{lepoll\_assumptions16} :: [i \Rightarrow o, i, [i,i] \Rightarrow i, i, i, i, i, i, i] \Rightarrow o$ **where**
$\mathit{lepoll\_assumptions16}(M,A,F,S,\mathit{fa},K,x,f,r) \equiv \mathit{strong\_replacement}(M,\ \lambda x\ y.\ y = \mathit{inj}^M(\mathit{if}\ x \in \mathit{range}(f)\ \mathit{then}\ F(A,\ \mathit{converse}(f)\ `\ x)\ \mathit{else}\ 0,K))$

definition
$\mathit{lepoll\_assumptions17} :: [i \Rightarrow o, i, [i,i] \Rightarrow i, i, i, i, i, i, i] \Rightarrow o$ **where**
$\mathit{lepoll\_assumptions17}(M,A,F,S,\mathit{fa},K,x,f,r) \equiv \mathit{strong\_replacement}$
$(M,\ \lambda x\ z.\ z = \mathit{Sigfun}(x,\ \lambda i.\ \mathit{inj}^M(\mathit{if}\ i \in \mathit{range}(f)\ \mathit{then}\ F(A,\ \mathit{converse}(f)\ `\ i)\ \mathit{else}\ 0,K)))$

definition
$\mathit{lepoll\_assumptions18} :: [i \Rightarrow o, i, [i,i] \Rightarrow i, i, i, i, i, i, i] \Rightarrow o$ **where**
$\mathit{lepoll\_assumptions18}(M,A,F,S,\mathit{fa},K,x,f,r) \equiv \mathit{strong\_replacement}$
$(M,\ \lambda x\ y.\ y = \langle x,\ \mathit{minimum}(r,\ \mathit{inj}^M(\mathit{if}\ x \in \mathit{range}(f)\ \mathit{then}\ F(A,\ \mathit{converse}(f)\ `\ x)\ \mathit{else}\ 0,K))\rangle)$

lemmas $\mathit{lepoll\_assumptions\_defs}[\mathit{simp}] = \mathit{lepoll\_assumptions1\_def}$
$\mathit{lepoll\_assumptions2\_def}\ \mathit{lepoll\_assumptions3\_def}\ \mathit{lepoll\_assumptions4\_def}$
$\mathit{lepoll\_assumptions5\_def}\ \mathit{lepoll\_assumptions6\_def}\ \mathit{lepoll\_assumptions7\_def}$
$\mathit{lepoll\_assumptions8\_def}\ \mathit{lepoll\_assumptions9\_def}\ \mathit{lepoll\_assumptions10\_def}$
$\mathit{lepoll\_assumptions11\_def}\ \mathit{lepoll\_assumptions12\_def}\ \mathit{lepoll\_assumptions13\_def}$
$\mathit{lepoll\_assumptions14\_def}\ \mathit{lepoll\_assumptions15\_def}\ \mathit{lepoll\_assumptions16\_def}$
$\mathit{lepoll\_assumptions17\_def}\ \mathit{lepoll\_assumptions18\_def}$

definition $\mathit{if\_range\_F}$ **where**
$[\mathit{simp}]$: $\mathit{if\_range\_F}(H,f,i) \equiv \mathit{if}\ i \in \mathit{range}(f)\ \mathit{then}\ H(\mathit{converse}(f)\ `\ i)\ \mathit{else}\ 0$

definition $\mathit{if\_range\_F\_else\_F}$ **where**
$\mathit{if\_range\_F\_else\_F}(H,b,f,i) \equiv \mathit{if}\ b{=}0\ \mathit{then}\ \mathit{if\_range\_F}(H,f,i)\ \mathit{else}\ H(i)$

lemma (**in** M_basic) $\mathit{lam\_Least\_assumption\_general}$:
assumes
$\mathit{separations}$:
$\forall A'[M].\ \mathit{separation}(M,\ \lambda y.\ \exists x{\in}A'.\ y = \langle x,\ \mu\ i.\ x \in \mathit{if\_range\_F\_else\_F}(F(A),b,f,i)\rangle)$
and
$\mathit{mem\_F\_bound}$:$\bigwedge x\ c.\ x{\in}F(A,c) \Longrightarrow c \in \mathit{range}(f) \cup U(A)$
and

types:$M(A)$ $M(b)$ $M(f)$ $M(U(A))$
shows *lam_replacement*(M,λx . μ *i*. $x \in$ *if_range_F_else_F*($F(A)$,b,f,i))
⟨*proof*⟩

lemma (**in** *M_basic*) *lam_Least_assumption_ifM_b0*:
fixes F
defines $F \equiv \lambda\_$ *x*. *if* $M(x)$ *then x else 0*
assumes
separations:
$\forall A'[M]$. *separation*(M, λy. $\exists x{\in}A'$. $y = \langle x, \mu\ i.\ x \in$ *if_range_F_else_F*($F(A)$,*0*,f,i)$\rangle$)
and
types:$M(A)$ $M(f)$
shows *lam_replacement*(M,λx . μ *i*. $x \in$ *if_range_F_else_F*($F(A)$,*0*,f,i))
(**is** *lam_replacement*(M,λx . *Least*(*?P*(x))))
⟨*proof*⟩

lemma (**in** *M_replacement_extra*) *lam_Least_assumption_ifM_bnot0*:
fixes F
defines $F \equiv \lambda\_$ *x*. *if* $M(x)$ *then x else 0*
assumes
separations:
$\forall A'[M]$. *separation*(M, λy. $\exists x{\in}A'$. $y = \langle x, \mu\ i.\ x \in$ *if_range_F_else_F*($F(A)$,b,f,i)$\rangle$)
separation(M,*Ord*)
and
types:$M(A)$ $M(f)$
and
$b{\neq}0$
shows *lam_replacement*(M,λx . μ *i*. $x \in$ *if_range_F_else_F*($F(A)$,b,f,i))
(**is** *lam_replacement*(M,λx . *Least*(*?P*(x))))
⟨*proof*⟩

lemma (**in** *M_replacement_extra*) *lam_Least_assumption_drSR_Y*:
fixes F r' D
defines $F \equiv$ *drSR_Y*(r',D)
assumes $\forall A'[M]$. *separation*(M, λy. $\exists x{\in}A'$. $y = \langle x, \mu\ i.\ x \in$ *if_range_F_else_F*($F(A)$,b,f,i)$\rangle$)
$M(A)$ $M(b)$ $M(f)$ $M(r')$
shows *lam_replacement*(M,λx . μ *i*. $x \in$ *if_range_F_else_F*($F(A)$,b,f,i))
⟨*proof*⟩

locale *M_replacement_lepoll* = *M_replacement_extra* + *M_inj* +
fixes F
assumes
F_type[*simp*]: $M(A) \Longrightarrow \forall x[M].\ M(F(A,x))$
and
lam_lepoll_assumption_F:$M(A) \Longrightarrow$ *lam_replacement*(M,$F(A)$)
and
— Here b is a Boolean.
lam_Least_assumption:$M(A) \Longrightarrow M(b) \Longrightarrow M(f) \Longrightarrow$
lam_replacement(M,λx . μ *i*. $x \in$ *if_range_F_else_F*($F(A)$,b,f,i))

and
F_args_closed: $M(A) \Longrightarrow M(x) \Longrightarrow x \in F(A,i) \Longrightarrow M(i)$
and
lam_replacement_inj_rel:*lam_replacement*$(M,\ \lambda p.\ inj^M(fst(p),snd(p)))$
begin

declare *if_range_F_else_F_def*[*simp*]

lemma *lepoll_assumptions1*:
assumes *types*[*simp*]:$M(A)\ M(S)$
shows *lepoll_assumptions1*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions2*:
assumes *types*[*simp*]:$M(A)\ M(S)$
shows *lepoll_assumptions2*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions3*:
assumes *types*[*simp*]:$M(A)$
shows *lepoll_assumptions3*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions4*:
assumes *types*[*simp*]:$M(A)\ M(r)$
shows *lepoll_assumptions4*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lam_Least_closed* :
assumes $M(A)\ M(b)\ M(f)$
shows $\forall\, x[M].\ M(\mu\ i.\ x \in$ *if_range_F_else_F*$(F(A),b,f,i))$
⟨*proof*⟩

lemma *lepoll_assumptions5*:
assumes
types[*simp*]:$M(A)\ M(f)$
shows *lepoll_assumptions5*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions6*:
assumes *types*[*simp*]:$M(A)\ M(S)\ M(x)$
shows *lepoll_assumptions6*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions7*:
assumes *types*[*simp*]:$M(A)\ M(S)\ M(x)$
shows *lepoll_assumptions7*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions8*:
assumes *types*[*simp*]:$M(A)$ $M(S)$
shows *lepoll_assumptions8*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions9*:
assumes *types*[*simp*]:$M(A)$ $M(S)$ $M(r)$
shows *lepoll_assumptions9*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions10*:
assumes *types*[*simp*]:$M(A)$ $M(f)$
shows *lepoll_assumptions10*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions11*:
assumes *types*[*simp*]:$M(A)$ $M(f)$
shows *lepoll_assumptions11*$(M,\ A,\ F,\ S,\ fa,\ K,\ x,\ f,\ r)$
⟨*proof*⟩

lemma *lepoll_assumptions12*:
assumes *types*[*simp*]:$M(A)$ $M(x)$ $M(f)$
shows *lepoll_assumptions12*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions13*:
assumes *types*[*simp*]:$M(A)$ $M(r)$ $M(f)$
shows *lepoll_assumptions13*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions14*:
assumes *types*[*simp*]:$M(A)$ $M(f)$ $M(fa)$
shows *lepoll_assumptions14*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions15*:
assumes *types*[*simp*]:$M(A)$ $M(x)$ $M(f)$ $M(K)$
shows *lepoll_assumptions15*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions16*:
assumes *types*[*simp*]:$M(A)$ $M(f)$ $M(K)$
shows *lepoll_assumptions16*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions17*:
assumes *types*[*simp*]:$M(A)$ $M(f)$ $M(K)$
shows *lepoll_assumptions17*(M,A,F,S,fa,K,x,f,r)
⟨*proof*⟩

lemma *lepoll_assumptions18*:
  assumes *types*[*simp*]:$M(A)$ $M(K)$ $M(f)$ $M(r)$
  shows *lepoll_assumptions18*(M,A,F,S,fa,K,x,f,r)
  ⟨*proof*⟩

lemmas *lepoll_assumptions* = *lepoll_assumptions1 lepoll_assumptions2*
  lepoll_assumptions3 lepoll_assumptions4 lepoll_assumptions5
  lepoll_assumptions6 lepoll_assumptions7 lepoll_assumptions8
  lepoll_assumptions9 lepoll_assumptions10 lepoll_assumptions11
  lepoll_assumptions12 lepoll_assumptions13 lepoll_assumptions14
  lepoll_assumptions15 lepoll_assumptions16
  lepoll_assumptions17 lepoll_assumptions18

end — *M_replacement_lepoll*

end

21 Cardinal Arithmetic under Choice

theory *Cardinal_Library_Relative*
  imports
    Replacement_Lepoll
begin

locale *M_library* = *M_ZF_library* + *M_cardinal_AC* +
  assumes
    separation_cardinal_rel_lesspoll_rel: $M(\kappa) \Longrightarrow$ *separation*(M, λx . $x \prec^M \kappa$)
begin

declare *eqpoll_rel_refl* [*simp*]

21.1 Miscellaneous

lemma *cardinal_rel_RepFun_apply_le*:
  assumes $S \in A \to B$ $M(S)$ $M(A)$ $M(B)$
  shows $|\{S`a$. $a \in A\}|^M \leq |A|^M$
⟨*proof*⟩

lemma *cardinal_rel_RepFun_le*:
  assumes *lrf*:*lam_replacement*(M,f) **and** *f_closed*:$\forall\, x[M].\ M(f(x))$ **and** $M(X)$
  shows $|\{f(x)$. $x \in X\}|^M \leq |X|^M$
  ⟨*proof*⟩

lemma *subset_imp_le_cardinal_rel*: $A \subseteq B \Longrightarrow M(A) \Longrightarrow M(B) \Longrightarrow |A|^M \leq |B|^M$
  ⟨*proof*⟩

lemma *lt_cardinal_rel_imp_not_subset*: $|A|^M < |B|^M \Longrightarrow M(A) \Longrightarrow M(B) \Longrightarrow \neg\ B \subseteq A$
⟨*proof*⟩

lemma *cardinal_rel_lt_csucc_rel_iff*:
$Card\_rel(M,K) \Longrightarrow M(K) \Longrightarrow M(K') \Longrightarrow |K'|^M < (K^+)^M \longleftrightarrow |K'|^M \leq K$
⟨*proof*⟩

end — *M_library*

locale *M_cardinal_UN_nat* = *M_cardinal_UN* _ ω X **for** X
begin
lemma *cardinal_rel_UN_le_nat*:
assumes $\bigwedge i.\ i\in\omega \Longrightarrow |X(i)|^M \leq \omega$
shows $|\bigcup i\in\omega.\ X(i)|^M \leq \omega$
⟨*proof*⟩

end — *M_cardinal_UN_nat*

locale *M_cardinal_UN_inj* = *M_library* +
j:*M_cardinal_UN* _ J +
y:*M_cardinal_UN* _ K $\lambda k.$ *if* $k\in range(f)$ *then* $X(converse(f)`k)$ *else* 0 **for** J K f +
assumes
f_inj: $f \in inj\_rel(M,J,K)$
begin

lemma *inj_rel_imp_cardinal_rel_UN_le*:
notes [*dest*] = *InfCard_is_Card Card_is_Ord*
fixes Y
defines $Y(k) \equiv$ *if* $k\in range(f)$ *then* $X(converse(f)`k)$ *else* 0
assumes $InfCard^M(K)$ $\bigwedge i.\ i\in J \Longrightarrow |X(i)|^M \leq K$
shows $|\bigcup i\in J.\ X(i)|^M \leq K$
⟨*proof*⟩

end — *M_cardinal_UN_inj*

locale *M_cardinal_UN_lepoll* = *M_library* + *M_replacement_lepoll* _ $\lambda\_.\ X$ +
j:*M_cardinal_UN* _ J **for** J
begin

lemma *leqpoll_rel_imp_cardinal_rel_UN_le*:
notes [*dest*] = *InfCard_is_Card Card_is_Ord*
assumes $InfCard^M(K)$ $J \lesssim^M K$ $\bigwedge i.\ i\in J \Longrightarrow |X(i)|^M \leq K$
$M(K)$
shows $|\bigcup i\in J.\ X(i)|^M \leq K$
⟨*proof*⟩

end — *M_cardinal_UN_lepoll*

context *M_library*
begin

lemma *cardinal_rel_lt_csucc_rel_iff'*:
  includes *Ord_dests*
  assumes *Card_rel*(M,κ)
    and *types*:$M(\kappa)$ $M(X)$
  shows $\kappa < |X|^M \longleftrightarrow (\kappa^+)^M \leq |X|^M$
  ⟨*proof*⟩

lemma *lepoll_rel_imp_subset_bij_rel*:
  assumes $M(X)$ $M(Y)$
  shows $X \lesssim^M Y \longleftrightarrow (\exists\, Z[M].\ Z \subseteq Y \wedge Z \approx^M X)$
⟨*proof*⟩

The following result proves to be very useful when combining *cardinal_rel* and *eqpoll_rel* in a calculation.

lemma *cardinal_rel_Card_rel_eqpoll_rel_iff*:
  Card_rel(M,κ) $\Longrightarrow$ $M(\kappa)$ $\Longrightarrow$ $M(X)$ $\Longrightarrow$ $|X|^M = \kappa \longleftrightarrow X \approx^M \kappa$
  ⟨*proof*⟩

lemma *lepoll_rel_imp_lepoll_rel_cardinal_rel*:
  assumes $X \lesssim^M Y$ $M(X)$ $M(Y)$
  shows $X \lesssim^M |Y|^M$
  ⟨*proof*⟩

lemma *lepoll_rel_Un*:
  assumes *InfCard_rel*(M,κ) $A \lesssim^M \kappa$ $B \lesssim^M \kappa$ $M(A)$ $M(B)$ $M(\kappa)$
  shows $A \cup B \lesssim^M \kappa$
⟨*proof*⟩

lemma *cardinal_rel_Un_le*:
  assumes *InfCard_rel*(M,κ) $|A|^M \leq \kappa$ $|B|^M \leq \kappa$ $M(\kappa)$ $M(A)$ $M(B)$
  shows $|A \cup B|^M \leq \kappa$
  ⟨*proof*⟩

lemma *Finite_cardinal_rel_iff'*: $M(i) \Longrightarrow$ *Finite*($|i|^M$) $\longleftrightarrow$ *Finite*(i)
  ⟨*proof*⟩

lemma *cardinal_rel_subset_of_Card_rel*:
  assumes *Card_rel*(M,γ) $a \subseteq \gamma$ $M(a)$ $M(\gamma)$
  shows $|a|^M < \gamma \vee |a|^M = \gamma$
⟨*proof*⟩

lemma *cardinal_rel_cases*:
  includes *Ord_dests*
  assumes $M(\gamma)$ $M(X)$

shows *Card_rel*(M,γ) $\Longrightarrow$ $|X|^M < \gamma \longleftrightarrow \neg\ |X|^M \geq \gamma$
⟨*proof*⟩

end — *M_library*

21.2 Countable and uncountable sets

definition
countable :: $i \Rightarrow o$ **where**
countable(X) $\equiv$ $X \lesssim \omega$

⟨*ML*⟩

notation *countable_rel* (‹*countable*–'(_')›)

abbreviation
countable_r_set :: $[i,i] \Rightarrow o$ (‹*countable*–'(_')›) **where**
$countable^M(i) \equiv$ *countable_rel*(##M,i)

context *M_library*
begin

lemma *countableI*[*intro*]: $X \lesssim^M \omega \Longrightarrow$ *countable_rel*(M,X)
⟨*proof*⟩

lemma *countableD*[*dest*]: *countable_rel*(M,X) $\Longrightarrow$ $X \lesssim^M \omega$
⟨*proof*⟩

lemma *countable_rel_iff_cardinal_rel_le_nat*: $M(X)$ $\Longrightarrow$ *countable_rel*(M,X)
$\longleftrightarrow$ $|X|^M \leq \omega$
⟨*proof*⟩

lemma *lepoll_rel_countable_rel*: $X \lesssim^M Y$ $\Longrightarrow$ *countable_rel*(M,Y) $\Longrightarrow$ $M(X)$
$\Longrightarrow$ $M(Y)$ $\Longrightarrow$ *countable_rel*(M,X)
⟨*proof*⟩
lemma *surj_rel_countable_rel*:
countable_rel(M,X) $\Longrightarrow$ $f \in$ *surj_rel*(M,X,Y) $\Longrightarrow$ $M(X)$ $\Longrightarrow$ $M(Y)$ $\Longrightarrow$ $M(f)$
$\Longrightarrow$ *countable_rel*(M,Y)
⟨*proof*⟩

lemma *Finite_imp_countable_rel*: *Finite_rel*(M,X) $\Longrightarrow$ $M(X)$ $\Longrightarrow$ *countable_rel*(M,X)
⟨*proof*⟩

end — *M_library*

lemma (**in** *M_cardinal_UN_lepoll*) *countable_rel_imp_countable_rel_UN*:
assumes *countable_rel*(M,J) $\bigwedge i.\ i \in J \Longrightarrow$ *countable_rel*(M,$X(i)$)
shows *countable_rel*(M,$\bigcup i \in J.\ X(i)$)
⟨*proof*⟩

locale *M_cardinal_library* = *M_library* + *M_replacement* +
  assumes
    lam_replacement_inj_rel:*lam_replacement*(*M*, $\lambda x.\ inj^M(fst(x),snd(x))$)
    and
    cdlt_assms: $M(G) \Longrightarrow M(Q) \Longrightarrow$ *separation*(*M*, $\lambda p.\ \forall\, x{\in}G.\ x \in snd(p) \longleftrightarrow (\forall\, s{\in}fst(p).\ \langle s,\ x\rangle \in Q)$)
    and
    cardinal_lib_assms1:
    $M(A) \Longrightarrow M(b) \Longrightarrow M(f) \Longrightarrow$
      separation(*M*, $\lambda y.\ \exists\, x{\in}A.\ y = \langle x,\ \mu\ i.\ x \in$ *if_range_F_else_F*$(\lambda x.$ *if* $M(x)$ *then* x *else* $0,b,f,i)\rangle$)
    and
    cardinal_lib_assms2:
    $M(A') \Longrightarrow M(G) \Longrightarrow M(b) \Longrightarrow M(f) \Longrightarrow$
      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* $G`a$ *else* $0,b,f,i)\rangle$)
    and
    cardinal_lib_assms3:
    $M(A') \Longrightarrow M(b) \Longrightarrow M(f) \Longrightarrow M(F) \Longrightarrow$
      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* $F\text{-}``\{a\}$ *else* $0,b,f,i)\rangle$)
    and
    lam_replacement_cardinal_rel : *lam_replacement*(*M*, *cardinal_rel*(*M*))
    and
    cardinal_lib_assms6:
    $M(f) \Longrightarrow M(\beta) \Longrightarrow Ord(\beta) \Longrightarrow$
     strong_replacement(*M*, $\lambda x\ y.\ x{\in}\beta \wedge y = \langle x,$ *transrec*$(x,\ \lambda a\ g.\ f\ `\ (g\ ``\ a))\rangle$)

begin

lemma *cardinal_lib_assms5* :
  $M(\gamma) \Longrightarrow Ord(\gamma) \Longrightarrow$ *separation*(*M*, $\lambda Z\ .$ *cardinal_rel*$(M,Z) < \gamma$)
  ⟨*proof*⟩

lemma *separation_dist*: *separation*(*M*, $\lambda\ x\ .\ \exists\, a.\ \exists\, b\ .\ x{=}\langle a,b\rangle \wedge a{\neq}b$)
  ⟨*proof*⟩

lemma *cdlt_assms'*: $M(x) \Longrightarrow M(Q) \Longrightarrow$ *separation*(*M*, $\lambda a\ .\ \ \forall\, s{\in}x.\ \langle s,\ a\rangle \in Q$)
  ⟨*proof*⟩

lemma *countable_rel_union_countable_rel*:
  assumes $\bigwedge x.\ x \in C \Longrightarrow$ *countable_rel*(M,x) *countable_rel*(M,C) $M(C)$
  shows *countable_rel*$(M,\bigcup C)$
⟨*proof*⟩

end — *M_cardinal_library*

abbreviation

uncountable_rel :: $[i \Rightarrow o, i] \Rightarrow o$ **where**
$uncountable\_rel(M,X) \equiv \neg\ countable\_rel(M,X)$

context *M_cardinal_library*
begin

lemma *uncountable_rel_iff_nat_lt_cardinal_rel*:
$M(X) \Longrightarrow uncountable\_rel(M,X) \longleftrightarrow \omega < |X|^M$
⟨*proof*⟩

lemma *uncountable_rel_not_empty*: $uncountable\_rel(M,X) \Longrightarrow X \neq 0$
⟨*proof*⟩

lemma *uncountable_rel_imp_Infinite*: $uncountable\_rel(M,X) \Longrightarrow M(X) \Longrightarrow Infinite(X)$
⟨*proof*⟩

lemma *uncountable_rel_not_subset_countable_rel*:
assumes $countable\_rel(M,X)$ $uncountable\_rel(M,Y)$ $M(X)$ $M(Y)$
shows $\neg\ (Y \subseteq X)$
⟨*proof*⟩

21.3 Results on Aleph_rels

lemma *nat_lt_Aleph_rel1*: $\omega < \aleph_1{}^M$
⟨*proof*⟩

lemma *zero_lt_Aleph_rel1*: $0 < \aleph_1{}^M$
⟨*proof*⟩

lemma *le_Aleph_rel1_nat*: $M(k) \Longrightarrow Card\_rel(M,k) \Longrightarrow k < \aleph_1{}^M \Longrightarrow k \leq \omega$
⟨*proof*⟩

lemma *lesspoll_rel_Aleph_rel_succ*:
assumes $Ord(\alpha)$
and *types*:$M(\alpha)$ $M(d)$
shows $d \prec^M \aleph_{succ(\alpha)}{}^M \longleftrightarrow d \lesssim^M \aleph_\alpha{}^M$
⟨*proof*⟩

lemma *cardinal_rel_Aleph_rel* [*simp*]: $Ord(\alpha) \Longrightarrow M(\alpha) \Longrightarrow |\aleph_\alpha{}^M|^M = \aleph_\alpha{}^M$
⟨*proof*⟩
lemma *Aleph_rel_lesspoll_rel_increasing*:
includes *Aleph_rel_intros*
assumes $M(b)$ $M(a)$
shows $a < b \Longrightarrow \aleph_a{}^M \prec^M \aleph_b{}^M$
⟨*proof*⟩

lemma *uncountable_rel_iff_subset_eqpoll_rel_Aleph_rel1*:
includes *Ord_dests*
assumes $M(X)$

notes *Aleph_rel_zero*[*simp*] *Card_rel_nat*[*simp*] *Aleph_rel_succ*[*simp*]
shows *uncountable_rel*(M,X) $\longleftrightarrow$ ($\exists\, S[M].\ S \subseteq X \wedge S \approx^M \aleph_1{}^M$)
⟨*proof*⟩

lemma *UN_if_zero*: $M(K) \Longrightarrow (\bigcup x{\in}K.\ \mathit{if}\ M(x)\ \mathit{then}\ G\ `\ x\ \mathit{else}\ 0) = (\bigcup x{\in}K.\ G\ `\ x)$
⟨*proof*⟩

lemma *mem_F_bound1*:
fixes F G
defines $F \equiv \lambda\_\ x.\ \mathit{if}\ M(x)\ \mathit{then}\ G`x\ \mathit{else}\ 0$
shows $x{\in}F(A,c) \Longrightarrow c \in (\mathit{range}(f) \cup \mathit{domain}(G))$
⟨*proof*⟩

lemma *lt_Aleph_rel_imp_cardinal_rel_UN_le_nat*: *function*(G) $\Longrightarrow$ *domain*(G) $\lesssim^M \omega \Longrightarrow$
$\forall\, n{\in}\mathit{domain}(G).\ |G`n|^M{<}\aleph_1{}^M \Longrightarrow M(G) \Longrightarrow |\bigcup n{\in}\mathit{domain}(G).\ G`n|^M{\le}\omega$
⟨*proof*⟩

lemma *Aleph_rel1_eq_cardinal_rel_vimage*: $f{:}\aleph_1{}^M{\to}^M\omega \Longrightarrow \exists\, n{\in}\omega.\ |f\text{-}``\{n\}|^M = \aleph_1{}^M$
⟨*proof*⟩

lemma *eqpoll_rel_Aleph_rel1_cardinal_rel_vimage*:
assumes $Z \approx^M (\aleph_1{}^M)$ $f \in Z \to^M \omega$ $M(Z)$
shows $\exists\, n{\in}\omega.\ |f\text{-}``\{n\}|^M = \aleph_1{}^M$
⟨*proof*⟩

21.4 Applications of transfinite recursive constructions

definition
rec_constr :: $[i,i] \Rightarrow i$ **where**
rec_constr(f,α) $\equiv$ *transrec*(α,$\lambda a\ g.\ f`(g``a)$)

The function *rec_constr* allows to perform *recursive constructions*: given a choice function on the powerset of some set, a transfinite sequence is created by successively choosing some new element.

The next result explains its use.

lemma *rec_constr_unfold*: *rec_constr*(f,α) = $f`(\{\mathit{rec\_constr}(f,\beta).\ \beta{\in}\alpha\})$
⟨*proof*⟩

lemma *rec_constr_type*:
assumes $f{:}\mathit{Pow\_rel}(M,G){\to}^M\ G$ *Ord*(α) $M(G)$
shows $M(\alpha) \Longrightarrow \mathit{rec\_constr}(f,\alpha) \in G$
⟨*proof*⟩

lemma *rec_constr_closed* :
assumes $f{:}\mathit{Pow\_rel}(M,G){\to}^M\ G$ *Ord*(α) $M(G)$ $M(\alpha)$
shows $M(\mathit{rec\_constr}(f,\alpha))$

⟨*proof*⟩

lemma *lambda_rec_constr_closed* :
assumes *Ord*(γ) *M*(γ) *M*(*f*) *f*:*Pow_rel*(*M*,*G*)→^{M} *G* *M*(*G*)
shows *M*(λα∈γ . *rec_constr*(*f*,α))
⟨*proof*⟩

The next lemma is an application of recursive constructions. It works under the assumption that whenever the already constructed subsequence is small enough, another element can be added.

lemma *bounded_cardinal_rel_selection*:
includes *Ord_dests*
assumes
⋀*Z*. |*Z*|^M < γ ⟹ *Z* ⊆ *G* ⟹ *M*(*Z*) ⟹ ∃ *a*∈*G*. ∀ *s*∈*Z*. <*s*,*a*>∈*Q* *b*∈*G*
Card_rel(*M*,γ)
M(*G*) *M*(*Q*) *M*(γ)
shows
∃ *S*[*M*]. *S* : γ →^{M} *G* ∧ (∀ α ∈ γ. ∀ β ∈ γ. α<β ⟶ <*S*‘α,*S*‘β>∈*Q*)
⟨*proof*⟩

The following basic result can, in turn, be proved by a bounded-cardinal_rel selection.

lemma *Infinite_iff_lepoll_rel_nat*: *M*(*Z*) ⟹ *Infinite*(*Z*) ⟷ ω ≲^{M} *Z*
⟨*proof*⟩

lemma *Infinite_InfCard_rel_cardinal_rel*: *Infinite*(*Z*) ⟹ *M*(*Z*) ⟹ *InfCard_rel*(*M*,|*Z*|^M)
⟨*proof*⟩

lemma (**in** *M_trans*) *mem_F_bound2*:
fixes *F A*
defines *F* ≡ λ_ *x*. *if M*(*x*) *then A*-‘‘{*x*} *else* 0
shows *x*∈*F*(*A*,*c*) ⟹ *c* ∈ (*range*(*f*) ∪ *range*(*A*))
⟨*proof*⟩

lemma *Finite_to_one_rel_surj_rel_imp_cardinal_rel_eq*:
assumes *F* ∈ *Finite_to_one_rel*(*M*,*Z*,*Y*) ∩ *surj_rel*(*M*,*Z*,*Y*) *Infinite*(*Z*) *M*(*Z*)
M(*Y*)
shows |*Y*|^M = |*Z*|^M
⟨*proof*⟩

lemma *cardinal_rel_map_Un*:
assumes *Infinite*(*X*) *Finite*(*b*) *M*(*X*) *M*(*b*)
shows |{*a* ∪ *b* . *a* ∈ *X*}|^M = |*X*|^M
⟨*proof*⟩

21.5 Results on relative cardinal exponentiation

lemma *cexp_rel_eqpoll_rel_cong*:
assumes

$A \approx^M A'$ $B \approx^M B'$ $M(A)$ $M(A')$ $M(B)$ $M(B')$
shows
$A^{\uparrow B,M} = A'^{\uparrow B',M}$
⟨*proof*⟩

lemma *cexp_rel_cexp_rel_cmult*:
assumes $M(\kappa)$ $M(\nu 1)$ $M(\nu 2)$
shows $(\kappa^{\uparrow \nu 1,M})^{\uparrow \nu 2,M} = \kappa^{\uparrow \nu 2 \otimes^M \nu 1,M}$
⟨*proof*⟩

lemma *cardinal_rel_Pow_rel*: $M(X) \Longrightarrow |Pow\_rel(M,X)|^M = 2^{\uparrow X,M}$ — Perhaps it's better with |X|
⟨*proof*⟩

lemma *cantor_cexp_rel*:
assumes *Card_rel*(M,ν) $M(\nu)$
shows $\nu < 2^{\uparrow \nu,M}$
⟨*proof*⟩

lemma *countable_iff_lesspoll_rel_Aleph_rel_one*:
notes *iff_trans*[*trans*]
assumes $M(C)$
shows *countable*$^M(C) \longleftrightarrow C \prec^M \aleph_1{}^M$
⟨*proof*⟩

lemma *countable_iff_le_rel_Aleph_rel_one*:
notes *iff_trans*[*trans*]
assumes $M(C)$
shows *countable*$^M(C) \longleftrightarrow |C|^M \prec^M \aleph_1{}^M$
⟨*proof*⟩

end — *M_cardinal_library*

lemma (**in** *M_cardinal_library*) *countable_fun_imp_countable_image*:
assumes $f{:}C \rightarrow^M B$ *countable*$^M(C)$ $\bigwedge c.\ c{\in}C \Longrightarrow$ *countable*$^M(f`c)$
$M(C)$ $M(B)$
shows *countable*$^M(\bigcup (f``C))$
⟨*proof*⟩

end

22 The Delta System Lemma, Relativized

theory *Delta_System_Relative*
imports
Cardinal_Library_Relative
begin

definition
delta_system :: $i \Rightarrow o$ **where**
$delta\_system(D) \equiv \exists r.\ \forall A{\in}D.\ \forall B{\in}D.\ A \neq B \longrightarrow A \cap B = r$

lemma *delta_systemI*[*intro*]:
assumes $\forall A{\in}D.\ \forall B{\in}D.\ A \neq B \longrightarrow A \cap B = r$
shows *delta_system*(*D*)
⟨*proof*⟩

lemma *delta_systemD*[*dest*]:
delta_system(*D*) $\Longrightarrow \exists r.\ \forall A{\in}D.\ \forall B{\in}D.\ A \neq B \longrightarrow A \cap B = r$
⟨*proof*⟩

lemma *delta_system_root_eq_Inter*:
assumes *delta_system*(*D*)
shows $\forall A{\in}D.\ \forall B{\in}D.\ A \neq B \longrightarrow A \cap B = \bigcap D$
⟨*proof*⟩

⟨*ML*⟩

locale *M_delta* = *M_cardinal_library* +
assumes
countable_lepoll_assms:
$M(G) \Longrightarrow M(A) \Longrightarrow M(b) \Longrightarrow M(f) \Longrightarrow separation(M,\ \lambda y.\ \exists x{\in}A.$
$y = \langle x,\ \mu\ i.\ x \in if\_range\_F\_else\_F(\lambda x.\ \{xa \in G\ .\ x \in xa\},$
$b,\ f,\ i)\rangle)$
begin

lemmas *cardinal_replacement* = *lam_replacement_cardinal_rel*[*unfolded lam_replacement_def*]

lemma *disjoint_separation*: $M(c) \Longrightarrow separation(M,\ \lambda\ x.\ \exists a.\ \exists b.\ x{=}\langle a,b\rangle \wedge a \cap b = c)$
⟨*proof*⟩

lemma *insnd_ball*: $M(G) \Longrightarrow separation(M,\ \lambda p.\ \forall x{\in}G.\ x \in snd(p) \longleftrightarrow fst(p) \in x)$
⟨*proof*⟩

lemma (**in** *M_trans*) *mem_F_bound6*:
fixes *F G*
defines $F \equiv \lambda\_\ x.\ Collect(G,\ (\in)(x))$
shows $x{\in}F(G,c) \Longrightarrow c \in (range(f) \cup \bigcup G)$
⟨*proof*⟩

lemma *delta_system_Aleph_rel1*:
assumes $\forall A{\in}F.\ Finite(A)\ F \approx^M \aleph_1{}^M\ M(F)$
shows $\exists D[M].\ D \subseteq F \wedge delta\_system(D) \wedge D \approx^M \aleph_1{}^M$

⟨*proof*⟩

lemma *delta_system_uncountable_rel*:
  assumes $\forall A{\in}F.\ Finite(A)\ uncountable\_rel(M,F)\ M(F)$
  shows $\exists D[M].\ D \subseteq F \wedge delta\_system(D) \wedge D \approx^M \aleph_1{}^M$
⟨*proof*⟩

end — *M_delta*

end

23 Relative DC

theory *Pointed_DC_Relative*
  imports
    Cardinal_Library_Relative

begin

consts *dc_witness* :: $i \Rightarrow i \Rightarrow i \Rightarrow i \Rightarrow i \Rightarrow i$
primrec
  wit0 : $dc\_witness(0,A,a,s,R) = a$
  witrec : $dc\_witness(succ(n),A,a,s,R) = s`\{x{\in}A.\ \langle dc\_witness(n,A,a,s,R),x\rangle{\in}R\}$

lemmas $dc\_witness\_def = dc\_witness\_nat\_def$

⟨*ML*⟩

schematic_goal *sats_is_dc_witness_fm_auto*:
  assumes $na < length(env)\ e < length(env)$
  shows
      $na \in \omega \Longrightarrow$
    $A \in \omega \Longrightarrow$
    $a \in \omega \Longrightarrow$
    $s \in \omega \Longrightarrow$
    $R \in \omega \Longrightarrow$
    $e \in \omega \Longrightarrow$
    $env \in list(Aa) \Longrightarrow$
    $0 \in Aa \Longrightarrow$
     $is\_dc\_witness(\#\#Aa,\ nth(na,\ env),\ nth(A,\ env),\ nth(a,\ env),\ nth(s,\ env),$
$nth(R,\ env),\ nth(e,\ env)) \longleftrightarrow$
    $Aa,\ env \models ?fm(nat,\ A,\ a,\ s,\ R,\ e)$
  ⟨*proof*⟩

⟨*ML*⟩
  ⟨*proof*⟩

definition *dcwit_body* :: $[i,i,i,i,i] \Rightarrow o$ **where**
  $dcwit\_body(A,a,g,R) \equiv \lambda p.\ snd(p) = dc\_witness(fst(p),\ A,\ a,\ g,\ R)$

⟨*ML*⟩

context *M_replacement*
begin

lemma *dc_witness_closed*[*intro*,*simp*]:
  assumes *M*(*n*) *M*(*A*) *M*(*a*) *M*(*s*) *M*(*R*) *n*∈*nat*
  shows *M*(*dc_witness*(*n*,*A*,*a*,*s*,*R*))
  ⟨*proof*⟩

lemma *dc_witness_rel_char*:
  assumes *M*(*A*)
  shows *dc_witness_rel*(*M*,*n*,*A*,*a*,*s*,*R*) = *dc_witness*(*n*,*A*,*a*,*s*,*R*)
⟨*proof*⟩

lemma (**in** *M_basic*) *first_section_closed*:
  assumes
    M(*A*) *M*(*a*) *M*(*R*)
  shows *M*({*x* ∈ *A* . ⟨*a*, *x*⟩ ∈ *R*})
⟨*proof*⟩

lemma *witness_into_A* [*TC*]:
  assumes *a*∈*A*
    ∀ *X*[*M*]. *X*≠*0* ∧ *X*⊆*A* ⟶ *s*‘*X*∈*A*
    ∀ *y*∈*A*. {*x*∈*A*. ⟨*y*,*x*⟩∈*R* } ≠ *0* *n*∈*nat*
    M(*A*) *M*(*a*) *M*(*s*) *M*(*R*)
  shows *dc_witness*(*n*, *A*, *a*, *s*, *R*)∈*A*
  ⟨*proof*⟩

end — *M_replacement*

locale *M_DC* = *M_trancl* + *M_replacement* + *M_eclose* +
  assumes
    separation_is_dcwit_body:
    M(*A*) ⟹ *M*(*a*) ⟹ *M*(*g*) ⟹ *M*(*R*) ⟹ *separation*(*M*,*is_dcwit_body*(*M*, *A*,
a, *g*, *R*))
    and
    dcwit_replacement:*Ord*(*na*) ⟹
    M(*na*) ⟹
    M(*A*) ⟹
    M(*a*) ⟹
    M(*s*) ⟹
    M(*R*) ⟹
    transrec_replacement
     (*M*, λ*n f ntc*.
          is_nat_case
           (*M*, *a*,
            λ*m bmfm*.

$\exists\, fm[M].\ \exists\, cp[M].$
$is\_apply(M,\ f,\ m,\ fm)\ \wedge$
$is\_Collect(M,\ A,\ \lambda x.\ \exists\, fmx[M].\ (M(x)\ \wedge\ fmx \in R)\ \wedge\ pair(M,\ fm,$
$x,\ fmx),\ cp)\ \wedge$
$is\_apply(M,\ s,\ cp,\ bmfm),$
$n,\ ntc),na)$
begin

lemma *is_dc_witness_iff*:
assumes $Ord(na)\ M(na)\ M(A)\ M(a)\ M(s)\ M(R)\ M(res)$
shows $is\_dc\_witness(M,\ na,\ A,\ a,\ s,\ R,\ res) \longleftrightarrow res = dc\_witness\_rel(M,\ na,$
$A,\ a,\ s,\ R)$
⟨*proof*⟩

lemma *dcwit_body_abs*:
$fst(x) \in \omega \Longrightarrow M(A) \Longrightarrow M(a) \Longrightarrow M(g) \Longrightarrow M(R) \Longrightarrow M(x) \Longrightarrow$
$is\_dcwit\_body(M,A,a,g,R,x) \longleftrightarrow dcwit\_body(A,a,g,R,x)$
⟨*proof*⟩

lemma *separation_eq_dc_witness*:
$M(A) \Longrightarrow$
$M(a) \Longrightarrow$
$M(g) \Longrightarrow$
$M(R) \Longrightarrow\ separation(M,\lambda p.\ fst(p)\in\omega \longrightarrow snd(p) = dc\_witness(fst(p),\ A,\ a,$
$g,\ R))$
⟨*proof*⟩

lemma *Lambda_dc_witness_closed*:
assumes $g \in Pow^M(A)\text{-}\{0\} \to A\ a\in A\ \forall\, y\in A.\ \{x \in A\ .\ \langle y,\ x\rangle \in R\} \neq 0$
$M(g)\ M(A)\ M(a)\ M(R)$
shows $M(\lambda n\in nat.\ dc\_witness(n,A,a,g,R))$
⟨*proof*⟩

lemma *witness_related*:
assumes $a\in A$
$\forall\, X[M].\ X\neq 0\ \wedge\ X\subseteq A \longrightarrow s`X\in X$
$\forall\, y\in A.\ \{x\in A.\ \langle y,x\rangle\in R\ \} \neq 0\ n\in nat$
$M(a)\ M(A)\ M(s)\ M(R)\ M(n)$
shows $\langle dc\_witness(n,\ A,\ a,\ s,\ R),dc\_witness(succ(n),\ A,\ a,\ s,\ R)\rangle\in R$
⟨*proof*⟩

lemma *witness_funtype*:
assumes $a\in A$
$\forall\, X[M].\ X\neq 0\ \wedge\ X\subseteq A \longrightarrow s`X \in A$
$\forall\, y\in A.\ \{x\in A.\ \langle y,x\rangle\in R\} \neq 0$
$M(A)\ M(a)\ M(s)\ M(R)$
shows $(\lambda n\in nat.\ dc\_witness(n,\ A,\ a,\ s,\ R)) \in nat \to A$ (**is** *?f* $\in$ _ $\to$ _)
⟨*proof*⟩

lemma *witness_to_fun*:
assumes $a{\in}A$
$\forall\, X[M].\ X{\neq}0 \wedge X{\subseteq}A \longrightarrow s`X{\in}A$
$\forall\, y{\in}A.\ \{x{\in}A.\ \langle y,x\rangle{\in}R\ \} \neq 0$
$M(A)\ M(a)\ M(s)\ M(R)$
shows $\exists f \in nat{\rightarrow}A.\ \forall\, n{\in}nat.\ f`n = dc\_witness(n,A,a,s,R)$
⟨*proof*⟩

end — *M_DC*

locale *M_library_DC* = *M_library* + *M_DC*
begin

lemma *AC_M_func*:
assumes $\bigwedge x.\ x \in A \Longrightarrow (\exists\, y.\ y \in x)\ M(A)$
shows $\exists f \in A \rightarrow^M \bigcup(A).\ \forall\, x \in A.\ f`x \in x$
⟨*proof*⟩

lemma *non_empty_family*: $[|\ 0 \notin A;\ \ x \in A\ |] ==> \exists\, y.\ y \in x$
⟨*proof*⟩

lemma *AC_M_func0*: $0 \notin A \Longrightarrow M(A) \Longrightarrow \exists f \in A \rightarrow^M \bigcup(A).\ \forall\, x \in A.\ f`x \in x$
⟨*proof*⟩

lemma *AC_M_func_Pow_rel*:
assumes $M(C)$
shows $\exists f \in (Pow^M(C)\text{-}\{0\}) \rightarrow^M C.\ \forall\, x \in Pow^M(C)\text{-}\{0\}.\ f`x \in x$
⟨*proof*⟩

theorem *pointed_DC*:
assumes $\forall\, x{\in}A.\ \exists\, y{\in}A.\ \langle x,y\rangle{\in}\ R\ M(A)\ M(R)$
shows $\forall\, a{\in}A.\ \exists f \in nat{\rightarrow}^M A.\ f`0 = a \wedge (\forall\, n \in nat.\ \langle f`n,f`succ(n)\rangle{\in}R)$
⟨*proof*⟩

lemma *aux_DC_on_AxNat2* : $\forall\, x{\in}A{\times}nat.\ \exists\, y{\in}A.\ \langle x,\langle y,succ(snd(x))\rangle\rangle \in R \Longrightarrow$
$\forall\, x{\in}A{\times}nat.\ \exists\, y{\in}A{\times}nat.\ \langle x,y\rangle \in \{\langle a,b\rangle{\in}R.\ snd(b) = succ(snd(a))\}$
⟨*proof*⟩

lemma *infer_snd* : $c{\in}\ A{\times}B \Longrightarrow snd(c) = k \Longrightarrow c{=}\langle fst(c),k\rangle$
⟨*proof*⟩

corollary *DC_on_A_x_nat* :
assumes $(\forall\, x{\in}A{\times}nat.\ \exists\, y{\in}A.\ \langle x,\langle y,succ(snd(x))\rangle\rangle \in R)\ a{\in}A\ M(A)\ M(R)$
shows $\exists f \in nat{\rightarrow}^M A.\ f`0 = a \wedge (\forall\, n \in nat.\ \langle\langle f`n,n\rangle,\langle f`succ(n),succ(n)\rangle\rangle{\in}R)$ (**is** $\exists\, x{\in}\_.?P(x)$)
⟨*proof*⟩

lemma *aux_sequence_DC* :

assumes $\forall\, x{\in}A.\ \forall\, n{\in}nat.\ \exists\, y{\in}A.\ \langle x,y\rangle \in S`n$
$R{=}\{\langle\langle x,n\rangle,\langle y,m\rangle\rangle \in (A{\times}nat){\times}(A{\times}nat).\ \langle x,y\rangle{\in}S`m\ \}$
shows $\forall\ x{\in}A{\times}nat\ .\ \exists\, y{\in}A.\ \langle x,\langle y,succ(snd(x))\rangle\rangle \in R$
⟨*proof*⟩

lemma *aux_sequence_DC2* : $\forall\, x{\in}A.\ \forall\, n{\in}nat.\ \exists\, y{\in}A.\ \langle x,y\rangle \in S`n \Longrightarrow$
$\forall\, x{\in}A{\times}nat.\ \exists\, y{\in}A.\ \langle x,\langle y,succ(snd(x))\rangle\rangle \in \{\langle\langle x,n\rangle,\langle y,m\rangle\rangle{\in}(A{\times}nat){\times}(A{\times}nat).$
$\langle x,y\rangle{\in}S`m\ \}$
⟨*proof*⟩

lemma *sequence_DC*:
assumes $\forall\, x{\in}A.\ \forall\, n{\in}nat.\ \exists\, y{\in}A.\ \langle x,y\rangle \in S`n\ M(A)\ M(S)$
shows $\forall\, a{\in}A.\ (\exists f \in nat{\rightarrow}^{M} A.\ f`0 = a \wedge (\forall\, n \in nat.\ \langle f`n,f`succ(n)\rangle{\in}S`succ(n)))$
⟨*proof*⟩

end — *M_library_DC*

end

24 Cohen forcing notions

theory *Partial_Functions_Relative*
imports
Cardinal_Library_Relative
begin

In this theory we introduce bounded partial functions and its relative version; for historical reasons the relative version is based on a proper definition of partial functions.

We note that finite partial functions are easier and are used to prove some lemmas about finite sets in the theory *Transitive_Models.ZF_Library_Relative*.

definition
$Fn :: [i,i,i] \Rightarrow i$ **where**
$Fn(\kappa,I,J) \equiv \bigcup\{y\ .\ d \in Pow(I),\ y{=}(d{\rightarrow}J) \wedge d{\prec}\kappa\}$

lemma *domain_function_lepoll* :
assumes *function*(r)
shows *domain*$(r) \lesssim r$
⟨*proof*⟩

lemma *function_lepoll*:
assumes $r{:}d{\rightarrow}J$
shows $r \lesssim d$
⟨*proof*⟩

lemma *function_eqpoll* :
assumes $r{:}d{\rightarrow}J$
shows $r \approx d$

⟨*proof*⟩

lemma *Fn_char* : $Fn(\kappa,I,J) = \{f \in Pow(I\times J)\ .\ function(f) \wedge f \prec \kappa\}$ (**is** *?L=?R*)
⟨*proof*⟩

lemma *zero_in_Fn*:
assumes $0 < \kappa$
shows $0 \in Fn(\kappa,\ I,\ J)$
⟨*proof*⟩

lemma *Fn_nat_eq_FiniteFun*: $Fn(nat,I,J) = I$ -||> J
⟨*proof*⟩

lemma *Fn_nat_subset_Pow*: $Fn(\kappa,I,J) \subseteq Pow(I\times J)$
⟨*proof*⟩

lemma *FnI*:
assumes $p : d \rightarrow J$ $d \subseteq I$ $d \prec \kappa$
shows $p \in Fn(\kappa,I,J)$
⟨*proof*⟩

lemma *FnD*[*dest*]:
assumes $p \in Fn(\kappa,I,J)$
shows $\exists\, d.\ p : d \rightarrow J \wedge d \subseteq I \wedge d \prec \kappa$
⟨*proof*⟩

lemma *Fn_is_function*: $p \in Fn(\kappa,I,J) \Longrightarrow function(p)$
⟨*proof*⟩

lemma *Fn_csucc*:
assumes $Ord(\kappa)$
shows $Fn(csucc(\kappa),I,J) = \bigcup\{y\ .\ d \in Pow(I),\ y=(d\rightarrow J) \wedge d \lesssim \kappa\}$
⟨*proof*⟩

definition
$FnleR :: i \Rightarrow i \Rightarrow o$ (**infixl** ‹$\supseteq$› *50*) **where**
$f \supseteq g \equiv g \subseteq f$

lemma *FnleR_iff_subset* [*iff*]: $f \supseteq g \longleftrightarrow g \subseteq f$
⟨*proof*⟩

definition
$Fnlerel :: i \Rightarrow i$ **where**
$Fnlerel(A) \equiv Rrel(\lambda x\ y.\ x \supseteq y, A)$

definition
$Fnle :: [i,i,i] \Rightarrow i$ **where**
$Fnle(\kappa,I,J) \equiv Fnlerel(Fn(\kappa,I,J))$

lemma *FnleI*[*intro*]:
assumes $p \in Fn(\kappa,I,J)$ $q \in Fn(\kappa,I,J)$ $p \supseteq q$
shows $\langle p,q\rangle \in Fnle(\kappa,I,J)$
⟨*proof*⟩

lemma *FnleD*[*dest*]:
assumes $\langle p,q\rangle \in Fnle(\kappa,I,J)$
shows $p \in Fn(\kappa,I,J)$ $q \in Fn(\kappa,I,J)$ $p \supseteq q$
⟨*proof*⟩

definition *PFun_Space_Rel* :: $[i,i\Rightarrow o,\ i] \Rightarrow i$ $(\_ \rightharpoonup^{\_} \_)$
where $A \rightharpoonup^M B \equiv \{f \in Pow(A\times B)\ .\ M(f) \wedge function(f)\}$

lemma (**in** *M_library*) *PFun_Space_subset_Powrel* :
assumes $M(A)$ $M(B)$
shows $A \rightharpoonup^M B = \{f \in Pow^M(A\times B)\ .\ function(f)\}$
⟨*proof*⟩

lemma (**in** *M_library*) *PFun_Space_closed* :
assumes $M(A)$ $M(B)$
shows $M(A \rightharpoonup^M B)$
⟨*proof*⟩

lemma *Un_filter_fun_space_closed*:
assumes $G \subseteq I \to J \bigwedge f\ g\ .\ f\in G \Longrightarrow g\in G \Longrightarrow \exists\, d\in I\to J\ .\ d \supseteq f \wedge d \supseteq g$
shows $\bigcup G \in Pow(I\times J)$ $function(\bigcup G)$
⟨*proof*⟩

lemma *Un_filter_is_fun* :
assumes $G \subseteq I \to J \bigwedge f\ g\ .\ f\in G \Longrightarrow g\in G \Longrightarrow \exists\, d\in I\to J\ .\ d\supseteq f \wedge d\supseteq g\ G\neq 0$
shows $\bigcup G \in I \to J$
⟨*proof*⟩

context *M_cardinals*
begin

lemma *mem_function_space_relD*:
assumes $f \in function\_space\_rel(M,A,y)$ $M(A)$ $M(y)$
shows $f \in A \to y$ **and** $M(f)$
⟨*proof*⟩

lemma *pfunI* :
assumes $C\subseteq A$ $f \in C \to^M B$ $M(C)$ $M(B)$
shows $f\in A \rightharpoonup^M B$
⟨*proof*⟩

lemma *zero_in_PFun_rel*:
assumes $M(I)$ $M(J)$
shows $0 \in I \rightharpoonup^M J$

⟨*proof*⟩

lemma *pfun_subsetI* :
assumes $f \in A \rightharpoonup^M B$ $g \subseteq f$ $M(g)$
shows $g \in A \rightharpoonup^M B$
⟨*proof*⟩

lemma *pfun_is_function* :
$f \in A \rightharpoonup^M B \Longrightarrow function(f)$
⟨*proof*⟩

lemma *pfun_Un_filter_closed*:
assumes $G \subseteq I \rightharpoonup^M J$ $\bigwedge f\ g\ .\ f \in G \Longrightarrow g \in G \Longrightarrow \exists\, d \in I \rightharpoonup^M J\ .\ d \supseteq f \wedge d \supseteq g$
shows $\bigcup G \in Pow(I \times J)$ $function(\bigcup G)$
⟨*proof*⟩

lemma *pfun_Un_filter_closed''*:
assumes $G \subseteq I \rightharpoonup^M J$ $\bigwedge f\ g\ .\ f \in G \Longrightarrow g \in G \Longrightarrow \exists\, d \in G\ .\ d \supseteq f \wedge d \supseteq g$
shows $\bigcup G \in Pow(I \times J)$ $function(\bigcup G)$
⟨*proof*⟩

lemma *pfun_Un_filter_closed'*:
assumes $G \subseteq I \rightharpoonup^M J$ $\bigwedge f\ g\ .\ f \in G \Longrightarrow g \in G \Longrightarrow \exists\, d \in G\ .\ d \supseteq f \wedge d \supseteq g$ $M(G)$
shows $\bigcup G \in I \rightharpoonup^M J$
⟨*proof*⟩

lemma *pfunD* :
assumes $f \in A \rightharpoonup^M B$
shows $\exists\, C[M].\ C \subseteq A \wedge f \in C \rightarrow B$
⟨*proof*⟩

lemma *pfunD_closed* :
assumes $f \in A \rightharpoonup^M B$
shows $M(f)$
⟨*proof*⟩

lemma *pfun_singletonI* :
assumes $x \in A$ $b \in B$ $M(A)$ $M(B)$
shows $\{\langle x,b\rangle\} \in A \rightharpoonup^M B$
⟨*proof*⟩

lemma *pfun_unionI* :
assumes $f \in A \rightharpoonup^M B$ $g \in A \rightharpoonup^M B$ $domain(f) \cap domain(g) = 0$
shows $f \cup g \in A \rightharpoonup^M B$
⟨*proof*⟩

lemma (**in** *M_library*) *pfun_restrict_eq_imp_compat*:
assumes $f \in I \rightharpoonup^M J$ $g \in I \rightharpoonup^M J$ $M(J)$
$restrict(f,\ domain(f) \cap domain(g)) = restrict(g,\ domain(f) \cap domain(g))$

shows $f \cup g \in I \rightharpoonup^M J$
⟨*proof*⟩

lemma *FiniteFun_pfunI* :
assumes $f \in A\text{-}||{>}\ B\ M(A)\ M(B)$
shows $f \in A \rightharpoonup^M B$
⟨*proof*⟩

lemma *PFun_FiniteFunI* :
assumes $f \in A \rightharpoonup^M B$ *Finite*(f)
shows $f \in A\text{-}||{>}\ B$
⟨*proof*⟩

end — *M_cardinals*

definition
Fn_rel :: $[i \Rightarrow o, i, i, i] \Rightarrow i$ (‹*Fn*–'(_,_,_')›) **where**
Fn_rel(M,κ,I,J) $\equiv \{f \in I \rightharpoonup^M J\ .\ f \prec^M \kappa\}$

context *M_library*
begin

lemma *Fn_rel_subset_PFun_rel* : $Fn^M(\kappa,I,J) \subseteq I \rightharpoonup^M J$
⟨*proof*⟩

lemma *Fn_relI*[*intro*]:
assumes $f : d \to J\ d \subseteq I\ f \prec^M \kappa\ M(d)\ M(J)\ M(f)$
shows $f \in$ *Fn_rel*(M,κ,I,J)
⟨*proof*⟩

lemma *Fn_relD*[*dest*]:
assumes $p \in$ *Fn_rel*(M,κ,I,J)
shows $\exists\, C[M].\ C \subseteq I \wedge p : C \to J \wedge p \prec^M \kappa$
⟨*proof*⟩

lemma *Fn_rel_is_function*:
assumes $p \in$ *Fn_rel*(M,κ,I,J)
shows *function*(p) $M(p)\ p \prec^M \kappa\ p \in I \rightharpoonup^M J$
⟨*proof*⟩

lemma *Fn_rel_mono*:
assumes $p \in$ *Fn_rel*(M,κ,I,J) $\kappa \prec^M \kappa'\ M(\kappa)\ M(\kappa')$
shows $p \in$ *Fn_rel*(M,κ',I,J)
⟨*proof*⟩

lemma *Fn_rel_mono'*:
assumes $p \in$ *Fn_rel*(M,κ,I,J) $\kappa \lesssim^M \kappa'\ M(\kappa)\ M(\kappa')$
shows $p \in$ *Fn_rel*(M,κ',I,J)

⟨*proof*⟩

lemma *Fn_csucc*:
assumes *Ord*(κ) *M*(κ)
shows *Fn_rel*(*M*,(κ⁺)^*M*,*I*,*J*) = {*p*∈ *I*⇁^*M* *J* . *p* ≲^*M* κ} (**is** *?L*=*?R*)
⟨*proof*⟩

lemma *Finite_imp_lesspoll_nat*:
assumes *Finite*(*A*)
shows *A* ≺ *nat*
⟨*proof*⟩

lemma *FinD_Finite* :
assumes *a*∈*Fin*(*A*)
shows *Finite*(*a*)
⟨*proof*⟩

lemma *Fn_rel_nat_eq_FiniteFun*:
assumes *M*(*I*) *M*(*J*)
shows *I* -||> *J* = *Fn_rel*(*M*,ω,*I*,*J*)
⟨*proof*⟩

lemma *Fn_nat_abs*:
assumes *M*(*I*) *M*(*J*)
shows *Fn*(*nat*,*I*,*J*) = *Fn_rel*(*M*,ω,*I*,*J*)
⟨*proof*⟩

lemma *Fn_rel_singletonI*:
assumes *x* ∈ *I* *j* ∈ *J* 1 ≺^*M* κ *M*(κ) *M*(*I*) *M*(*J*)
shows {⟨*x*,*j*⟩} ∈ *Fn*^*M*(κ,*I*,*J*)
⟨*proof*⟩

end — *M_library*

definition
Fnle_rel :: [*i*⇒*o*,*i*,*i*,*i*] ⇒ *i* (‹*Fnle*–'(_,_,_')›) **where**
Fnle_rel(*M*,κ,*I*,*J*) ≡ *Fnlerel*(*Fn*^*M*(κ,*I*,*J*))

abbreviation
Fn_r_set :: [*i*,*i*,*i*,*i*] ⇒ *i* (‹*Fn*–'(_,_,_')›) **where**
Fn_r_set(*M*) ≡ *Fn_rel*(##*M*)

abbreviation
Fnle_r_set :: [*i*,*i*,*i*,*i*] ⇒ *i* (‹*Fnle*–'(_,_,_')›) **where**
Fnle_r_set(*M*) ≡ *Fnle_rel*(##*M*)

context *M_library*
begin

lemma *Fnle_relI*[*intro*]:
assumes $p \in Fn\_rel(M,\kappa,I,J)$ $q \in Fn\_rel(M,\kappa,I,J)$ $p \supseteq q$
shows $<p,q> \in Fnle\_rel(M,\kappa,I,J)$
⟨*proof*⟩

lemma *Fnle_relD*[*dest*]:
assumes $<p,q> \in Fnle\_rel(M,\kappa,I,J)$
shows $p \in Fn\_rel(M,\kappa,I,J)$ $q \in Fn\_rel(M,\kappa,I,J)$ $p \supseteq q$
⟨*proof*⟩

lemma *Fn_rel_closed*[*intro*,*simp*]:
assumes $M(\kappa)$ $M(I)$ $M(J)$
shows $M(Fn^M(\kappa,I,J))$
⟨*proof*⟩

lemma *Fn_rel_subset_Pow*:
assumes $M(\kappa)$ $M(I)$ $M(J)$
shows $Fn^M(\kappa,I,J) \subseteq Pow(I\times J)$
⟨*proof*⟩

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

lemma *zero_in_Fn_rel*:
assumes $0<\kappa$ $M(\kappa)$ $M(I)$ $M(J)$
shows $0 \in Fn^M(\kappa,\ I,\ J)$
⟨*proof*⟩

lemma *zero_top_Fn_rel*:
assumes $p\in Fn^M(\kappa,\ I,\ J)$ $0<\kappa$ $M(\kappa)$ $M(I)$ $M(J)$
shows $\langle p,\ 0\rangle \in Fnle^M(\kappa,\ I,\ J)$
⟨*proof*⟩

lemma *preorder_on_Fnle_rel*:
assumes $M(\kappa)$ $M(I)$ $M(J)$
shows *preorder_on*$(Fn^M(\kappa,\ I,\ J),\ Fnle^M(\kappa,\ I,\ J))$
⟨*proof*⟩

end — *M_library*

end

References

[1] 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).

[2] L.C. Paulson, The relative consistency of the axiom of choice mechanized using Isabelle/ZF, *LMS J. Comput. Math.* **6**: 198–248 (2003). Appendix A available electronically at http://www.lms.ac.uk/jcm/6/lms2003-001/appendix-a/.