

# The Independence of the Continuum Hypothesis in Isabelle/ZF

Emmanuel Gunther\*   Miguel Pagano\*   Pedro Sánchez Terraf\*<sup>†</sup>  
Matías Steinberg\*

May 20, 2026

## Abstract

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

## Contents

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

---

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

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

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

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

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

## 1 Introduction

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

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

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

### Release notes

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

## 2 Forcing notions

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

```

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

```

```

hide_const (open) Order.pred

```

## 2.1 Basic concepts

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

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

**lemma**  $compat\_inI$  :  
 $[[ d \in A ; \langle d, p \rangle \in r ; \langle d, q \rangle \in r ]] \Longrightarrow compat\_in(A, r, p, q)$   
 $\langle proof \rangle$

**lemma**  $refl\_compat$ :  
 $[[ refl(A, r) ; \langle p, q \rangle \in r \mid p = q \mid \langle q, p \rangle \in r ; p \in A ; q \in A ]] \Longrightarrow compat\_in(A, r, p, q)$   
 $\langle proof \rangle$

**lemma**  $chain\_compat$ :  
 $refl(A, r) \Longrightarrow linear(A, r) \Longrightarrow (\forall p \in A. \forall q \in A. compat\_in(A, r, p, q))$   
 $\langle proof \rangle$

**lemma**  $subset\_fun\_image$ :  $f: N \rightarrow P \Longrightarrow f''N \subseteq P$   
 $\langle proof \rangle$

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

**locale**  $forcing\_notion =$   
**fixes**  $P$  ( $\langle \mathbb{P} \rangle$ ) **and**  $leq$  **and**  $one$  ( $\langle \mathbf{1} \rangle$ )  
**assumes**  $one\_in\_P$ :  $\mathbf{1} \in \mathbb{P}$   
**and**  $leq\_preord$ :  $preorder\_on(\mathbb{P}, leq)$   
**and**  $one\_max$ :  $\forall p \in \mathbb{P}. \langle p, \mathbf{1} \rangle \in leq$   
**begin**

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

**lemma**  $refl\_leq$ :  
 $r \in \mathbb{P} \Longrightarrow r \preceq r$   
 $\langle proof \rangle$

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

**definition**

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

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

**definition**

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

**lemma** *P\_dense*: *dense*( $\mathbb{P}$ )  
 <proof>

**definition**

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

**definition**

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

**lemma** *leq\_transD*:  $a \preceq b \Longrightarrow b \preceq c \Longrightarrow a \in \mathbb{P} \Longrightarrow b \in \mathbb{P} \Longrightarrow c \in \mathbb{P} \Longrightarrow a \preceq c$   
 <proof>

**lemma** *leq\_transD'*:  $A \subseteq \mathbb{P} \Longrightarrow a \preceq b \Longrightarrow b \preceq c \Longrightarrow a \in A \Longrightarrow b \in \mathbb{P} \Longrightarrow c \in \mathbb{P} \Longrightarrow a \preceq c$   
 <proof>

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

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

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

**lemma** *Incompatible\_imp\_not\_eq*:  $\llbracket p \perp q; p \in \mathbb{P}; q \in \mathbb{P} \rrbracket \Longrightarrow p \neq q$   
 <proof>

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

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

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

**lemma** *dense\_belowI* [intro!]:

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

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

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

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

**lemma** *dense\_below\_under*:

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

**lemma** *ideal\_dense\_below*:

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

**lemma** *dense\_below\_dense\_below*:

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

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

**definition**

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

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

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

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

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

**lemma** *low\_bound\_filter*: — says the compatibility is attained inside  $G$

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

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

**definition**

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

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

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

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

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

**lemma**  $\text{upclosure\_in\_P}$ :  $A \subseteq \mathbb{P} \Rightarrow \text{upclosure}(A) \subseteq \mathbb{P}$   
 $\langle \text{proof} \rangle$

**lemma**  $A\_sub\_upclosure$ :  $A \subseteq \mathbb{P} \Rightarrow A \subseteq \text{upclosure}(A)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{elem\_upclosure}$ :  $A \subseteq \mathbb{P} \Rightarrow x \in A \Rightarrow x \in \text{upclosure}(A)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{closure\_compat\_filter}$ :  
**assumes**  $A \subseteq \mathbb{P} (\forall p \in A. \forall q \in A. \text{compat\_in}(A, \text{leq}, p, q))$   
**shows**  $\text{filter}(\text{upclosure}(A))$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{aux\_RS1}$ :  $f \in N \rightarrow \mathbb{P} \Rightarrow n \in N \Rightarrow f^n \in \text{upclosure}(f^{“N})$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{decr\_succ\_decr}$ :  
**assumes**  $f \in \text{nat} \rightarrow \mathbb{P}$   $\text{preorder\_on}(\mathbb{P}, \text{leq})$   
 $\forall n \in \text{nat}. \langle f^{\text{‘ succ}(n)}, f^{\text{‘ } n} \rangle \in \text{leq}$   
 $m \in \text{nat}$   
**shows**  $n \in \text{nat} \Rightarrow n \leq m \Rightarrow \langle f^{\text{‘ } m}, f^{\text{‘ } n} \rangle \in \text{leq}$   
 $\langle \text{proof} \rangle$

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

**end** — *forcing\_notion*

## 2.2 Towards Rasiowa-Sikorski Lemma (RSL)

**locale** *countable\_generic* = *forcing\_notion* +  
**fixes**  $\mathcal{D}$   
**assumes** *countable\_subs\_of\_P*:  $\mathcal{D} \in \text{nat} \rightarrow \text{Pow}(\mathbb{P})$   
**and** *seq\_of\_denses*:  $\forall n \in \text{nat}. \text{dense}(\mathcal{D} \text{ ' } n)$

**begin**

**definition**

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

The next lemma identifies a sufficient condition for obtaining RSL.

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

**end** — *countable\_generic*

Now, the following recursive definition will fulfill the requirements of lemma *RS\_sequence\_imp\_rasiowa\_sikorski*

**consts** *RS\_seq* ::  $[i, i, i, i, i, i] \Rightarrow i$

**primrec**

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

**context** *countable\_generic*

**begin**

**lemma** *countable\_RS\_sequence\_aux*:

**fixes**  $p$  *enum*

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

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

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

**lemma** *countable\_RS\_sequence*:

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

**lemma** *RS\_seq\_type*:

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

**lemma** *RS\_seq\_funtype*:

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

**lemmas** *countable\_rasiowa\_sikorski* =

$\text{RS\_sequence\_imp\_rasiowa\_sikorski}[\text{OF\_RS\_seq\_funtype countable\_RS\_sequence}(1, 2)]$

**end** — *countable\_generic*

**end**

### 3 Cohen forcing notions

**theory** *Cohen\_Posets\_Relative*

**imports**

*Forcing\_Notions*

*Transitive\_Models.Delta\_System\_Relative*

*Transitive\_Models.Partial\_Functions\_Relative*

**begin**

**locale** *cohen\_data* =

**fixes**  $\kappa I J :: i$

**assumes** *zero\_lt\_kappa*:  $0 < \kappa$

**begin**

**lemmas** *zero\_lesspoll\_kappa* =  $\text{zero\_lesspoll}[\text{OF } \text{zero\_lt\_kappa}]$

**end** — *cohen\_data*

**abbreviation**

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

**lemma** *dense\_inj\_dense*:

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

**locale** *add\_reals* = *cohen\_data* *nat* \_ 2

**3.1 Combinatorial results on Cohen posets**

**sublocale** *cohen\_data*  $\subseteq$  *forcing\_notion*  $\text{Fn}(\kappa, I, J) \ \text{Fnle}(\kappa, I, J) \ 0$   
 $\langle \text{proof} \rangle$

**context** *cohen\_data*

**begin**

**lemma** *compat\_imp\_Un\_is\_function*:

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

**lemma** *Un\_filter\_is\_function*:  $\text{filter}(G) \Longrightarrow \text{function}(\bigcup G)$ 

$\langle \text{proof} \rangle$

**end** — *cohen\_data*

**locale** *M\_cohen* = *M\_delta* +

**assumes**

*countable\_lepoll\_assms2*:

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

**and**

*countable\_lepoll\_assms3*:

$M(A) \Longrightarrow M(f) \Longrightarrow M(b) \Longrightarrow M(D) \Longrightarrow M(r') \Longrightarrow M(A') \Longrightarrow$   
 $\text{separation}(M, \lambda y. \exists x \in A'. y = \langle x, \mu \ i. x \in \text{if\_range\_F\_else\_F}(\text{drSR\_Y}(r', D, A), b, f, i))$

**lemma** (**in** *M\_library*) *Fnle\_rel\_Aleph\_rel1\_closed*[*intro, simp*]:

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

$\langle \text{proof} \rangle$

**locale** *M\_add\_reals* = *M\_cohen* + *add\_reals*

**begin**

**lemmas** *zero\_lesspoll\_rel\_kappa* = *zero\_lesspoll\_rel*[*OF zero\_lt\_kappa*]

**end** — *M\_add\_reals*

⟨*ML*⟩

**context**

**notes** *Un\_assoc*[*simp*] *Un\_transposition\_aux2*[*simp*]

**begin**

⟨*ML*⟩

**end**

**lemma** (in *M\_trivial*) *compat\_in\_abs*[*absolut*]:

**assumes**

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

**shows**

$is\_compat\_in(M, A, r, p, q) \longleftrightarrow compat\_in(A, r, p, q)$

⟨*proof*⟩

**definition**

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

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

⟨*ML*⟩

**definition**

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

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

**abbreviation**

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

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

**abbreviation**

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

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

**context** *M\_trivial*

**begin**

**lemma** *antichain\_abs* [*absolut*]:

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

⟨*proof*⟩

**end** — *M\_trivial*

⟨*ML*⟩

**abbreviation**

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

**abbreviation**

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

**context**  $M\_cardinals$ **begin****lemma**  $def\_ccc\_rel$ :**shows**

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

**end** —  $M\_cardinals$ **context**  $M\_FiniteFun$ **begin****lemma**  $Fnle\_nat\_closed[intro, simp]$ :

**assumes**  $M(I) M(J)$

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

$\langle proof \rangle$

**lemma**  $Fn\_nat\_closed$ :

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

$\langle proof \rangle$

**end** —  $M\_FiniteFun$ **context**  $M\_add\_reals$ **begin**

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

$\langle proof \rangle$

**lemma** (**in**  $M\_trans$ )  $mem\_F\_bound3$ :

**fixes**  $F A$

**defines**  $F \equiv dC\_F$

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

$\langle proof \rangle$

**lemma**  $ccc\_rel\_Fn\_nat$ :

**assumes**  $M(I)$

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

*<proof>*

**end** — *M\_add\_reals*

**end**

**theory** *Edrel*

**imports**

*Transitive\_Models.ZF\_Miscellanea*

*Transitive\_Models.Recursion\_Thms*

**begin**

### 3.2 The well-founded relation *ed*

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

*<proof>*

**lemma** *ecloseE* :

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

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

*<proof>*

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

*<proof>*

**lemma** *in\_eclose\_sing* :

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

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

*<proof>*

**lemma** *in\_dom\_in\_eclose* :

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

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

*<proof>*

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

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

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

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

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

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

*<proof>*

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

*<proof>*

**lemma** *rank\_ed*:

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

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

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

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

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

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

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

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

**lemma**  $edrel\_sub\_memrel$ :  $edrel(A) \subseteq trancl(Memrel(eclose(A)))$   
 $\langle proof \rangle$

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

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

**lemma**  $dom\_under\_edrel\_eclose$ :  $edrel(eclose(\{x\})) -'' \{x\} = domain(x)$   
 $\langle proof \rangle$

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

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

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

```

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

end

```

## 4 Well-founded relation on names

```

theory FrecR
  imports
    Transitive_Models.Discipline_Function
    Edrel
  begin

```

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

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

lemma components_in_eclose :

```

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

**lemmas**  $\text{names\_simp} = \text{components\_simp}(2) \text{ components\_simp}(3)$

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

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

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

**definition**  
 $\text{is\_ftype} :: (i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$  **where**  
 $\text{is\_ftype} \equiv \text{is\_fst}$

**definition**  
 $\text{ftype\_fm} :: [i, i] \Rightarrow i$  **where**  
 $\text{ftype\_fm} \equiv \text{fst\_fm}$

**lemma**  $\text{is\_ftype\_iff\_sats}$  [ $\text{iff\_sats}$ ]:  
**assumes**  
 $\text{nth}(a, \text{env}) = x \text{ nth}(b, \text{env}) = y \text{ } a \in \text{nat} \text{ } b \in \text{nat} \text{ } \text{env} \in \text{list}(A)$   
**shows**  
 $\text{is\_ftype}(\#\#A, x, y) \longleftrightarrow \text{sats}(A, \text{ftype\_fm}(a, b), \text{env})$   
 $\langle \text{proof} \rangle$

**definition**  
 $\text{is\_name1} :: (i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$  **where**  
 $\text{is\_name1}(M, x, t2) \equiv \text{is\_hcomp}(M, \text{is\_fst}(M), \text{is\_snd}(M), x, t2)$

**definition**  
 $\text{name1\_fm} :: [i, i] \Rightarrow i$  **where**  
 $\text{name1\_fm}(x, t) \equiv \text{hcomp\_fm}(\text{fst\_fm}, \text{snd\_fm}, x, t)$

**lemma**  $\text{sats\_name1\_fm}$  [ $\text{simp}$ ]:  
 $\llbracket x \in \text{nat}; y \in \text{nat}; \text{env} \in \text{list}(A) \rrbracket \Longrightarrow$   
 $(A, \text{env} \models \text{name1\_fm}(x, y)) \longleftrightarrow \text{is\_name1}(\#\#A, \text{nth}(x, \text{env}), \text{nth}(y, \text{env}))$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{is\_name1\_iff\_sats}$  [ $\text{iff\_sats}$ ]:  
**assumes**  
 $\text{nth}(a, \text{env}) = x \text{ nth}(b, \text{env}) = y \text{ } a \in \text{nat} \text{ } b \in \text{nat} \text{ } \text{env} \in \text{list}(A)$

**shows**

$is\_name1(\#\#A, x, y) \longleftrightarrow A, env \models name1\_fm(a, b)$   
*<proof>*

**definition**

$is\_snd\_snd :: (i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$  **where**  
 $is\_snd\_snd(M, x, t) \equiv is\_hcomp(M, is\_snd(M), is\_snd(M), x, t)$

**definition**

$snd\_snd\_fm :: [i, i] \Rightarrow i$  **where**  
 $snd\_snd\_fm(x, t) \equiv hcomp\_fm(snd\_fm, snd\_fm, x, t)$

**lemma sats\_snd2\_fm [simp]:**

$\llbracket x \in nat; y \in nat; env \in list(A) \rrbracket \Longrightarrow$   
 $(A, env \models snd\_snd\_fm(x, y)) \longleftrightarrow is\_snd\_snd(\#\#A, nth(x, env), nth(y, env))$   
*<proof>*

**definition**

$is\_name2 :: (i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$  **where**  
 $is\_name2(M, x, t3) \equiv is\_hcomp(M, is\_fst(M), is\_snd\_snd(M), x, t3)$

**definition**

$name2\_fm :: [i, i] \Rightarrow i$  **where**  
 $name2\_fm(x, t3) \equiv hcomp\_fm(fst\_fm, snd\_snd\_fm, x, t3)$

**lemma sats\_name2\_fm :**

$\llbracket x \in nat; y \in nat; env \in list(A) \rrbracket$   
 $\Longrightarrow (A, env \models name2\_fm(x, y)) \longleftrightarrow is\_name2(\#\#A, nth(x, env), nth(y, env))$   
*<proof>*

**lemma is\_name2\_iff\_sats [iff\_sats]:**

**assumes**

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

**shows**

$is\_name2(\#\#A, x, y) \longleftrightarrow A, env \models name2\_fm(a, b)$   
*<proof>*

**definition**

$is\_cond\_of :: (i \Rightarrow o) \Rightarrow i \Rightarrow i \Rightarrow o$  **where**  
 $is\_cond\_of(M, x, t4) \equiv is\_hcomp(M, is\_snd(M), is\_snd\_snd(M), x, t4)$

**definition**

$cond\_of\_fm :: [i, i] \Rightarrow i$  **where**  
 $cond\_of\_fm(x, t4) \equiv hcomp\_fm(snd\_fm, snd\_snd\_fm, x, t4)$

**lemma sats\_cond\_of\_fm :**

$\llbracket x \in nat; y \in nat; env \in list(A) \rrbracket \Longrightarrow$   
 $(A, env \models cond\_of\_fm(x, y)) \longleftrightarrow is\_cond\_of(\#\#A, nth(x, env), nth(y, env))$   
*<proof>*

**lemma** *is\_cond\_of\_iff\_sats* [*iff\_sats*]:  
**assumes**  
 $nth(a,env) = x \quad nth(b,env) = y \quad a \in nat \quad b \in nat \quad env \in list(A)$   
**shows**  
 $is\_cond\_of(\#\#A,x,y) \longleftrightarrow A, env \models cond\_of\_fm(a,b)$   
*<proof>*

**lemma** *components\_type*[*TC*] :  
**assumes**  $a \in nat \quad b \in nat$   
**shows**  
 $f\_type\_fm(a,b) \in formula$   
 $name1\_fm(a,b) \in formula$   
 $name2\_fm(a,b) \in formula$   
 $cond\_of\_fm(a,b) \in formula$   
*<proof>*

**lemmas** *components\_iff\_sats* = *is\_f\_type\_iff\_sats is\_name1\_iff\_sats is\_name2\_iff\_sats is\_cond\_of\_iff\_sats*

**lemmas** *components\_defs* = *f\_type\_fm\_def snd\_snd\_fm\_def hcomp\_fm\_def name1\_fm\_def name2\_fm\_def cond\_of\_fm\_def*

**definition**  
 $is\_eclose\_n :: [i \Rightarrow o, [i \Rightarrow o, i, i] \Rightarrow o, i, i] \Rightarrow o$  **where**  
 $is\_eclose\_n(N, is\_name, en, t) \equiv$   
 $\exists n1[N]. \exists s1[N]. is\_name(N, t, n1) \wedge is\_singleton(N, n1, s1) \wedge is\_eclose(N, s1, en)$

**definition**  
 $eclose\_n1\_fm :: [i, i] \Rightarrow i$  **where**  
 $eclose\_n1\_fm(m, t) \equiv Exists(Exists(And(And(name1\_fm(t+\omega 2, 0), singleton\_fm(0, 1)), is\_eclose\_fm(1, m+\omega 2))))$

**definition**  
 $eclose\_n2\_fm :: [i, i] \Rightarrow i$  **where**  
 $eclose\_n2\_fm(m, t) \equiv Exists(Exists(And(And(name2\_fm(t+\omega 2, 0), singleton\_fm(0, 1)), is\_eclose\_fm(1, m+\omega 2))))$

**definition**  
 $is\_ecloseN :: [i \Rightarrow o, i, i] \Rightarrow o$  **where**  
 $is\_ecloseN(N, t, en) \equiv \exists en1[N]. \exists en2[N].$   
 $is\_eclose\_n(N, is\_name1, en1, t) \wedge is\_eclose\_n(N, is\_name2, en2, t) \wedge$   
 $union(N, en1, en2, en)$

**definition**  
 $ecloseN\_fm :: [i, i] \Rightarrow i$  **where**  
 $ecloseN\_fm(en, t) \equiv Exists(Exists(And(eclose\_n1\_fm(1, t+\omega 2), And(eclose\_n2\_fm(0, t+\omega 2), union\_fm(1, 0, en+\omega 2))))))$

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

**lemma** *sats\_ecloseN\_fm* [simp]:  
 $\llbracket en \in nat; t \in nat ; env \in list(A) \rrbracket$   
 $\implies (A, env \models ecloseN\_fm(en,t)) \longleftrightarrow is\_ecloseN(\#\#A,nth(t,env),nth(en,env))$   
 ⟨proof⟩

**lemma** *is\_ecloseN\_iff\_sats* [iff\_sats]:  
 $\llbracket nth(en, env) = ena; nth(t, env) = ta; en \in nat; t \in nat ; env \in list(A) \rrbracket$   
 $\implies is\_ecloseN(\#\#A,ta,ena) \longleftrightarrow A, env \models ecloseN\_fm(en,t)$   
 ⟨proof⟩

**definition**

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

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

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

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

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

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

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

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

*<proof>*

**lemma** *frecR\_D1* :

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

**lemma** *frecR\_D2* :

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

**lemma** *frecR\_DI* :

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

*<ML>*

**schematic\_goal** *sats\_frecR\_fm\_auto*:

**assumes**  
 $i \in nat \ j \in nat \ env \in list(A)$   
**shows**  
 $is\_frecR(\#\#A, nth(i, env), nth(j, env)) \longleftrightarrow A, env \models ?fr\_fm(i, j)$   
*<proof>*

*<ML>*

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

**lemma** *eq\_ftypep\_not\_frecR*:

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

**definition**

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

**lemma** *rank\_names\_types* [TC]:

**shows**  $Ord(rank\_names(x))$   
*<proof>*

**definition**

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

**definition**

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

**lemma** *type\_form\_tc* [TC]:  
**shows**  $\text{type\_form}(x) \in \mathcal{I}$   
 ⟨*proof*⟩

**lemma** *frecR\_le\_rnk\_names* :  
**assumes**  $frecR(x,y)$   
**shows**  $\text{rank\_names}(x) \leq \text{rank\_names}(y)$   
 ⟨*proof*⟩

**definition**  
 $\Gamma :: i \Rightarrow i$  **where**  
 $\Gamma(x) = \mathcal{I} ** \text{rank\_names}(x) ++ \text{type\_form}(x)$

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

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

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

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

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

**lemma** *wf\_frecrel* :  
**shows**  $wf(\text{frecrel}(A))$   
 ⟨*proof*⟩

**lemma** *core\_induction\_aux*:

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

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

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

**end**

**theory**  $FrecR\_Arities$

**imports**

$FrecR$

**begin**

**context**

**notes**  $FOL\_arities[simp]$

**begin**

$\langle ML \rangle$

**lemma**  $arity\_fst\_fm\ [arity] :$   
 $[[x \in nat ; t \in nat] \implies arity(fst\_fm(x, t)) = succ(x) \cup succ(t)$   
 $\langle proof \rangle$

$\langle ML \rangle$

**lemma**  $arity\_snd\_fm\ [arity] :$   
 $[[x \in nat ; t \in nat] \implies arity(snd\_fm(x, t)) = succ(x) \cup succ(t)$   
 $\langle proof \rangle$

**lemma**  $arity\_snd\_snd\_fm\ [arity] :$   
 $[[x \in nat ; t \in nat] \implies arity(snd\_snd\_fm(x, t)) = succ(x) \cup succ(t)$   
 $\langle proof \rangle$

**lemma**  $arity\_ftype\_fm\ [arity] :$

```

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

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

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

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

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

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

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

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

end — FOL_arities

end

```

## 5 Concepts involved in instances of Replacement

```

theory Fm_Definitions
  imports
    Transitive_Models.Renaming_Auto
    Transitive_Models.Aleph_Relative
    Frecl_Arities
begin

no_notation Aleph (⟨ℵ_⟩ [90] 90)

```

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

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

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

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

⟨ML⟩

**definition** is\_minimum' **where**

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

⟨ML⟩

**lemma** is\_lambda\_iff\_sats[iff\_sats]:

**assumes** is\_F\_iff\_sats:

!!a0 a1 a2.

[[a0∈Aa; a1∈Aa; a2∈Aa]

==> is\_F(a1, a0) ↔ sats(Aa, is\_F\_fm, Cons(a0, Cons(a1, Cons(a2, env))))

**shows**

nth(A, env) = Ab ==>

nth(r, env) = ra ==>

A ∈ nat ==>

r ∈ nat ==>

env ∈ list(Aa) ==>

is\_lambda(##Aa, Ab, is\_F, ra) ↔ Aa, env ⊨ lambda\_fm(is\_F\_fm, A, r)

⟨proof⟩

**lemma** sats\_is\_wfrec\_fm':

**assumes** MH\_iff\_sats:

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

**lemma**  $\text{is\_wfrec\_iff\_sats}'[\text{iff\_sats}]$ :

**assumes**  $MH\_iff\_sats$ :

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

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

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

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

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

**shows**

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

$\langle \text{proof} \rangle$

**lemma**  $\text{is\_wfrec\_on\_iff\_sats}[\text{iff\_sats}]$ :

**assumes**  $MH\_iff\_sats$ :

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

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

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

**shows**

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

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

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

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

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

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

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

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

$\langle \text{proof} \rangle$

Formulas for particular replacement instances

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

**definition**

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

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

$\langle ML \rangle$

**definition**

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

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

$\langle ML \rangle$

**definition**

$is\_Hcheck :: [i \Rightarrow o, i, i, i, i] \Rightarrow o$  **where**  
 $is\_Hcheck(M, o, z, f, hc) \equiv is\_Replace(M, z, PHcheck(M, o, f), hc)$

$\langle ML \rangle$

**lemma** *arity\_is\_Hcheck\_fm*:

**assumes**  $m \in nat$   $n \in nat$   $p \in nat$   $o \in nat$

**shows**  $arity(is\_Hcheck\_fm(m, n, p, o)) = succ(o) \cup succ(n) \cup succ(p) \cup succ(m)$

$\langle proof \rangle$

**definition**

$is\_check :: [i \Rightarrow o, i, i, i] \Rightarrow o$  **where**

$is\_check(M, o, x, z) \equiv \exists rh[M]. is\_rcheck(M, x, rh) \wedge$

$is\_wfreq(M, is\_Hcheck(M, o), rh, x, z)$

— Finally, we internalize the formula.

**definition**

$check\_fm :: [i, i, i] \Rightarrow i$  **where**

$check\_fm(o, x, z) \equiv Exists(And(is\_rcheck\_fm(1+\omega x, 0),$

$is\_wfreq\_fm(is\_Hcheck\_fm(6+\omega o, 2, 1, 0), 0, 1+\omega x, 1+\omega z)))$

**lemma** *check\_fm\_type[TC]*:  $x \in nat \Rightarrow o \in nat \Rightarrow z \in nat \Rightarrow check\_fm(x, o, z) \in formula$

$\langle proof \rangle$

**lemma** *sats\_check\_fm* :

**assumes**

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

**shows**

$(M, env \models check\_fm(o, x, z)) \longleftrightarrow is\_check(\#\#M, nth(o, env), nth(x, env), nth(z, env))$

$\langle proof \rangle$

**lemma** *iff\_sats\_check\_fm[iff\_sats]* :

**assumes**

$nth(o, env) = oa$   $nth(x, env) = xa$   $nth(z, env) = za$   $o \in nat$   $x \in nat$   $z \in nat$   
 $env \in list(A)$   $0 \in A$

**shows**  $is\_check(\#\#A, oa, xa, za) \longleftrightarrow A, env \models check\_fm(o, x, z)$

$\langle proof \rangle$

**lemma** *arity\_check\_fm[arity]*:

**assumes**  $m \in nat$   $n \in nat$   $o \in nat$

**shows**  $arity(check\_fm(m, n, o)) = succ(o) \cup succ(n) \cup succ(m)$

$\langle proof \rangle$

**notation**  $check\_fm (\langle \_ \_ \_ \_ \_ \rangle)$

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

**definition**

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

$\langle ML \rangle$

**abbreviation**

$fm\_leq :: [i, i, i] \Rightarrow i$  ( $\langle \cdot \_ \leq \_ \cdot \rangle$ ) **where**  
 $fm\_leq(A, l, B) \equiv is\_leq\_fm(l, A, B)$

**5.1 Formulas used to prove some generic instances.**

**definition**  $\varrho\_repl :: i \Rightarrow i$  **where**

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

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

**hide\_fact**  $Internalize.sum\_type$

**lemma**  $ren\_type :$

**assumes**  $l \in nat$

**shows**  $\varrho\_repl(l) : 2 +_{\omega} l \rightarrow 3 +_{\omega} l$

$\langle proof \rangle$

**definition**  $Lambda\_in\_M\_fm$  **where**  $[simp]: Lambda\_in\_M\_fm(\varphi, len) \equiv$

$\cdot (\exists \cdot pair\_fm(1, 0, 2) \wedge$

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

**lemma**  $Lambda\_in\_M\_fm\_type[TC]: \varphi \in formula \Longrightarrow len \in nat \Longrightarrow Lambda\_in\_M\_fm(\varphi, len) \in formula$

$\langle proof \rangle$

**definition**  $\varrho\_pair\_repl :: i \Rightarrow i$  **where**

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

**definition**  $LambdaPair\_in\_M\_fm$  **where**  $LambdaPair\_in\_M\_fm(\varphi, len) \equiv$

$\cdot (\exists \cdot pair\_fm(1, 0, 2) \wedge$

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

$len) \text{ ‘ } \varrho\_pair\_repl(len) \cdot) \cdot) \text{ ‘ } (2 +_{\omega} len) \text{ ‘ }$

$(3 +_{\omega} len) \text{ ‘ }$

$\varrho\_repl(len) \cdot) \wedge$

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

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

$\langle proof \rangle$

**lemma**  $ren\_type' :$

**assumes**  $l \in nat$

**shows**  $\varrho\_pair\_repl(l) : 3 +_{\omega} l \rightarrow 4 +_{\omega} l$

$\langle proof \rangle$

**lemma** *LambdaPair\_in\_M\_fm\_type*[TC]:  $\varphi \in \text{formula} \implies \text{len} \in \text{nat} \implies \text{LambdaPair\_in\_M\_fm}(\varphi, \text{len}) \in \text{formula}$   
 ⟨proof⟩

## 5.2 The relation *frecrel*

### definition

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

⟨ML⟩

### definition

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

⟨ML⟩

### definition

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

**lemma** *names\_belowsD*:

**assumes**  $x \in \text{names\_below}(P, z)$

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

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

⟨proof⟩

⟨ML⟩

**lemma** *number2\_iff* :

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

⟨proof⟩

⟨ML⟩

### definition

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

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

⟨ML⟩

### 5.3 Definition of Forces

#### 5.3.1 Definition of forces for equality and membership

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

**definition**

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

$\langle ML \rangle$

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

**definition**

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

$\langle ML \rangle$

**lemma**  $arity\_eq\_case\_fm[arity]$ :

**assumes**

$n1 \in nat \ n2 \in nat \ p \in nat \ P \in nat \ leq \in nat \ f \in nat$

**shows**

$arity(eq\_case\_fm(n1, n2, p, P, leq, f)) =$   
 $succ(n1) \cup succ(n2) \cup succ(p) \cup succ(P) \cup succ(leq) \cup succ(f)$   
 $\langle proof \rangle$

$\langle ML \rangle$

**lemma**  $arity\_mem\_case\_fm[arity]$  :

**assumes**

$n1 \in nat \ n2 \in nat \ p \in nat \ P \in nat \ leq \in nat \ f \in nat$

**shows**

$arity(mem\_case\_fm(n1, n2, p, P, leq, f)) =$   
 $succ(n1) \cup succ(n2) \cup succ(p) \cup succ(P) \cup succ(leq) \cup succ(f)$   
 $\langle proof \rangle$

**definition**

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

$\langle ML \rangle$

**definition**

$is\_Hfrc\_at :: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$  **where**

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

$\langle ML \rangle$

**lemma** *arity\_Hfrc\_fm*[arity] :

**assumes**

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

**shows**

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

$\langle proof \rangle$

$\langle ML \rangle$

### 5.3.2 The well-founded relation *forcere*<sub>l</sub>

**definition**

*forcere*<sub>l</sub> ::  $i \Rightarrow i \Rightarrow i$  **where**

$$\text{forcere}_l(P,x) \equiv \text{frec}_l(\text{names\_below}(P,x)) \hat{+}$$

**definition**

*is\_forcere*<sub>l</sub> ::  $[i \Rightarrow o, i, i, i] \Rightarrow o$  **where**

$$\begin{aligned} is\_forcere_l(M,P,x,z) &\equiv \exists r[M]. \exists nb[M]. \text{tran\_closure}(M,r,z) \wedge \\ &(\text{is\_names\_below}(M,P,x,nb) \wedge is\_frec_l(M,nb,r)) \end{aligned}$$

$\langle ML \rangle$

## 5.4 *frc\_at*, forcing for atomic formulas

**definition**

*frc\_at* ::  $[i, i, i] \Rightarrow i$  **where**

$$\begin{aligned} frc\_at(P,leq,fnnc) &\equiv wfrec(\text{frec}_l(\text{names\_below}(P,fnnc)),fnnc, \\ &\lambda x f. \text{bool\_of\_o}(Hfrc(P,leq,x,f))) \end{aligned}$$

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

**definition**

*is\_frc\_at* ::  $[i \Rightarrow o, i, i, i, i] \Rightarrow o$  **where**

$$\begin{aligned} is\_frc\_at(M,P,leq,x,z) &\equiv \exists r[M]. is\_forcere_l(M,P,x,r) \wedge \\ &is\_wfrec(M, is\_Hfrc\_at(M,P,leq), r, x, z) \end{aligned}$$

**definition**

*frc\_at\_fm* ::  $[i, i, i, i] \Rightarrow i$  **where**

$$\begin{aligned} frc\_at\_fm(p,l,x,z) &\equiv \text{Exists}(\text{And}(is\_forcere\_fm(\text{succ}(p),\text{succ}(x),0), \\ &is\_wfrec\_fm(Hfrc\_at\_fm(6+\omega p,6+\omega l,2,1,0),0,\text{succ}(x),\text{succ}(z)))))) \end{aligned}$$

**lemma** *frc\_at\_fm\_type* [TC] :

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

$\langle proof \rangle$

**lemma** *arity\_frc\_at\_fm*[arity] :

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

**lemma**  $\text{sats\_frc\_at\_fm}$  :

**assumes**  
 $p \in \text{nat } l \in \text{nat } i \in \text{nat } j \in \text{nat } \text{env} \in \text{list}(A) \ i < \text{length}(\text{env}) \ j < \text{length}(\text{env})$   
**shows**  
 $(A, \text{env} \models \text{frc\_at\_fm}(p,l,i,j)) \longleftrightarrow$   
 $\text{is\_frc\_at}(\#\#A, \text{nth}(p, \text{env}), \text{nth}(l, \text{env}), \text{nth}(i, \text{env}), \text{nth}(j, \text{env}))$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{frc\_at\_fm\_iff\_sats}$ :

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

**declare**  $\text{frc\_at\_fm\_iff\_sats}$  [ $\text{iff\_sats}$ ]

**definition**

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

**definition**

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

**definition**

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

**definition**

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

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

**definition**

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

$\text{is\_tuple}(M, z, t1, t2, p, t) \wedge \text{is\_frc\_at}(M, P, l, t, o)$

**definition**

$\text{is\_forces\_mem}' :: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$  **where**  
 $\text{is\_forces\_mem}'(M, P, l, p, t1, t2) \equiv \exists o[M]. \exists t[M]. \text{number1}(M, o) \wedge$   
 $\text{is\_tuple}(M, o, t1, t2, p, t) \wedge \text{is\_frc\_at}(M, P, l, t, o)$

**definition**

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

**definition**

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

$\langle ML \rangle$

**context**

**notes**  $Un\_assoc[simp]$   $Un\_trasposition\_aux2[simp]$

**begin**

$\langle ML \rangle$

**end**

## 5.5 Forcing for general formulas

**definition**

$ren\_forces\_nand :: i \Rightarrow i$  **where**  
 $ren\_forces\_nand(\varphi) \equiv Exists(And(Equal(0,1), iterates(\lambda p. incr\_bv(p)'1, 2, \varphi)))$

**lemma**  $ren\_forces\_nand\_type[TC]$  :

$\varphi \in formula \Longrightarrow ren\_forces\_nand(\varphi) \in formula$   
 $\langle proof \rangle$

**lemma**  $arity\_ren\_forces\_nand$  :

**assumes**  $\varphi \in formula$   
**shows**  $arity(ren\_forces\_nand(\varphi)) \leq succ(arity(\varphi))$   
 $\langle proof \rangle$

**lemma**  $sats\_ren\_forces\_nand$ :

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

**definition**

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

**lemma**  $arity\_ren\_forces\_all$  :

**assumes**  $\varphi \in formula$   
**shows**  $arity(ren\_forces\_forall(\varphi)) = 5 \cup arity(\varphi)$

$\langle proof \rangle$

**lemma** *ren\_forces\_forall\_type*[TC] :  
 $\varphi \in formula \implies ren\_forces\_forall(\varphi) \in formula$   
 $\langle proof \rangle$

**lemma** *sats\_ren\_forces\_forall* :  
 $[x,P,leg,o,p] @ env \in list(M) \implies \varphi \in formula \implies$   
 $(M, [x,p,P,leg,o] @ env \models ren\_forces\_forall(\varphi)) \longleftrightarrow (M, [p,P,leg,o,x] @ env$   
 $\models \varphi)$   
 $\langle proof \rangle$

### 5.5.1 The primitive recursion

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

**primrec**

$forces'(Member(x,y)) = forces\_mem\_fm(1,2,0,x+\omega 4,y+\omega 4)$   
 $forces'(Equal(x,y)) = forces\_eq\_fm(1,2,0,x+\omega 4,y+\omega 4)$   
 $forces'(Nand(p,q)) =$   
 $Neg(Exists(And(Member(0,2),And(is\_leq\_fm(3,0,1),And(ren\_forces\_nand(forces'(p)),$   
 $ren\_forces\_nand(forces'(q)))))))$   
 $forces'(Forall(p)) = Forall(ren\_forces\_forall(forces'(p)))$

**definition**

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

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

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

### 5.6 The arity of forces

**lemma** *arity\_forces\_at*:

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

**lemma** *arity\_forces'*:

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

**lemma** *arity\_forces* :

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

*<proof>*

**lemma** *arity\_forces\_le* :

**assumes**  $\varphi \in \text{formula}$   $n \in \text{nat}$   $\text{arity}(\varphi) \leq n$

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

*<proof>*

**definition** *rename\_split\_fm* **where**

$\text{rename\_split\_fm}(\varphi) \equiv (\cdot \exists (\cdot \exists (\cdot \exists (\cdot \exists (\cdot \exists (\cdot \exists (\cdot \text{snd}(9) \text{ is } 0 \cdot \wedge \cdot \text{fst}(9) \text{ is } 4 \cdot \wedge \cdot 1=11 \cdot$   
 $\wedge$   
 $\cdot 2=12 \cdot \wedge \cdot 3=13 \cdot \wedge \cdot 5=7 \cdot \wedge$   
 $(\lambda p. \text{incr\_bv}(p) '6) ^8(\text{forces}(\varphi)) \dots \dots \dots \cdot) \cdot) \cdot) \cdot) \cdot) \cdot)$

**lemma** *rename\_split\_fm\_type*[TC]:  $\varphi \in \text{formula} \implies \text{rename\_split\_fm}(\varphi) \in \text{formula}$

*<proof>*

**schematic\_goal** *arity\_rename\_split\_fm*:  $\varphi \in \text{formula} \implies \text{arity}(\text{rename\_split\_fm}(\varphi))$

$= ?m$

*<proof>*

**lemma** *arity\_rename\_split\_fm\_le*:

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

**shows**  $\text{arity}(\text{rename\_split\_fm}(\varphi)) \leq 8 \cup (6 +_{\omega} \text{arity}(\varphi))$

*<proof>*

**definition** *body\_ground\_repl\_fm* **where**

$\text{body\_ground\_repl\_fm}(\varphi) \equiv (\cdot \exists (\cdot \exists (\cdot \text{is\_Vset\_fm}(2, 0) \wedge \cdot 1 \in 0 \cdot \wedge \text{rename\_split\_fm}(\varphi)$   
 $\dots) \cdot) \cdot)$

**lemma** *body\_ground\_repl\_fm\_type*[TC]:  $\varphi \in \text{formula} \implies \text{body\_ground\_repl\_fm}(\varphi) \in \text{formula}$

*<proof>*

**lemma** *arity\_body\_ground\_repl\_fm\_le*:

**notes** *le\_trans*[trans]

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

**shows**  $\text{arity}(\text{body\_ground\_repl\_fm}(\varphi)) \leq 6 \cup (\text{arity}(\varphi) +_{\omega} 4)$

*<proof>*

**definition** *ground\_repl\_fm* **where**

$\text{ground\_repl\_fm}(\varphi) \equiv \text{least\_fm}(\text{body\_ground\_repl\_fm}(\varphi), 1)$

**lemma** *ground\_repl\_fm\_type*[TC]:

$\varphi \in \text{formula} \implies \text{ground\_repl\_fm}(\varphi) \in \text{formula}$

*<proof>*

**lemma** *arity\_ground\_repl\_fm*:

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

**shows**  $\text{arity}(\text{ground\_repl\_fm}(\varphi)) \leq 5 \cup (3 +_{\omega} \text{arity}(\varphi))$

*<proof>*

$\langle ML \rangle$

**definition** *omap\_wfrec\_body* **where**

$omap\_wfrec\_body(A,r) \equiv (\cdot \exists \cdot image\_fm(2, 0, 1) \wedge pred\_set\_fm(9+\omega A, 3, 9+\omega r, 0) \cdot \cdot)$

**lemma** *type\_omap\_wfrec\_body\_fm* :  $A \in nat \implies r \in nat \implies omap\_wfrec\_body(A,r) \in formula$   
 $\langle proof \rangle$

**lemma** *arity\_omap\_wfrec\_aux* :  $A \in nat \implies r \in nat \implies arity(omap\_wfrec\_body(A,r))$   
 $= (9+\omega A) \cup (9+\omega r)$   
 $\langle proof \rangle$

**lemma** *arity\_omap\_wfrec* :  $A \in nat \implies r \in nat \implies$   
 $arity(is\_wfrec\_fm(omap\_wfrec\_body(A,r), r+\omega 3, 1, 0)) = (4+\omega A) \cup (4+\omega r)$   
 $\langle proof \rangle$

**lemma** *arity\_isordermap* :  $A \in nat \implies r \in nat \implies d \in nat \implies$   
 $arity(is\_ordermap\_fm(A,r,d)) = succ(d) \cup (succ(A) \cup succ(r))$   
 $\langle proof \rangle$

**lemma** *arity\_is\_ordertype* :  $A \in nat \implies r \in nat \implies d \in nat \implies$   
 $arity(is\_ordertype\_fm(A,r,d)) = succ(d) \cup (succ(A) \cup succ(r))$   
 $\langle proof \rangle$

**lemma** *arity\_is\_order\_body* :  $arity(is\_order\_body\_fm(0,1)) = 2$   
 $\langle proof \rangle$

**definition** *H\_order\_pred* **where**

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

$\langle ML \rangle$

**definition** *order\_pred\_wfrec\_body* **where**

$order\_pred\_wfrec\_body(M,A,r,z,x) \equiv \exists y[M].$

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

$(\exists f[M].$

$(\forall z[M].$

$z \in f \longleftrightarrow$

$(\exists xa[M].$

$\exists y[M].$

$\exists xaa[M].$

$\exists sx[M].$

$\exists r\_sx[M].$

$\exists f\_r\_sx[M].$

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

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

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

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

$\langle ML \rangle$

**definition** *ordtype\_replacement\_fm* **where** *ordtype\_replacement\_fm*  $\equiv (\cdot \exists \cdot \text{is\_order\_body\_fm}(1, 0) \wedge \cdot \langle 1, 0 \rangle \text{ is } 2 \dots)$

**definition** *wfrec\_ordertype\_fm* **where** *wfrec\_ordertype\_fm*  $\equiv \text{order\_pred\_wfrec\_body\_fm}(3, 2, 1, 0)$

**definition** *replacement\_is\_aleph\_fm* **where** *replacement\_is\_aleph\_fm*  $\equiv \cdot 0 \text{ is ordinal} \cdot \wedge \cdot \aleph(0) \text{ is } 1 \cdot$

**definition**

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

$\langle ML \rangle$

**definition** *wfrec\_Hfrc\_at\_fm* **where** *wfrec\_Hfrc\_at\_fm*  $\equiv (\cdot \exists \cdot \text{pair\_fm}(1, 0, 2) \wedge \text{is\_wfrec\_fm}(\text{Hfrc\_at\_fm}(8, 9, 2, 1, 0), 5, 1, 0) \dots)$

**definition** *list\_repl1\_intf\_fm* **where** *list\_repl1\_intf\_fm*  $\equiv (\cdot \exists \cdot \text{pair\_fm}(1, 0, 2) \wedge \text{is\_wfrec\_fm}(\text{iterates\_MH\_fm}(\text{list\_functor\_fm}(13, 1, 0), 10, 2, 1, 0), 3, 1, 0) \dots)$

**definition** *list\_repl2\_intf\_fm* **where** *list\_repl2\_intf\_fm*  $\equiv \cdot 0 \in 4 \cdot \wedge \text{is\_iterates\_fm}(\text{list\_functor\_fm}(13, 1, 0), 3, 0, 1) \cdot$

**definition** *formula\_repl2\_intf\_fm* **where** *formula\_repl2\_intf\_fm*  $\equiv \cdot 0 \in 3 \cdot \wedge \text{is\_iterates\_fm}(\text{formula\_functor\_fm}(1, 0), 2, 0, 1) \cdot$

**definition** *eclose\_abs\_fm* **where** *eclose\_abs\_fm*  $\equiv \cdot 0 \in 3 \cdot \wedge \text{is\_iterates\_fm}(\cdot \cup 1 \text{ is } 0 \cdot, 2, 0, 1) \cdot$

**definition** *powapply\_repl\_fm* **where** *powapply\_repl\_fm*  $\equiv \text{is\_Powapply\_fm}(2, 0, 1)$

**definition** *wfrec\_rank\_fm* **where** *wfrec\_rank\_fm*  $\equiv (\cdot \exists \cdot \text{pair\_fm}(1, 0, 2) \wedge \text{is\_wfrec\_fm}(\text{is\_Hrank\_fm}(2, 1, 0), 3, 1, 0) \dots)$

**definition** *transrec\_VFrom\_fm* **where** *transrec\_VFrom\_fm*  $\equiv (\cdot \exists \cdot \text{pair\_fm}(1, 0, 2) \wedge \text{is\_wfrec\_fm}(\text{is\_HVfrom\_fm}(8, 2, 1, 0), 4, 1, 0) \dots)$

**definition** *wfrec\_Hcheck\_fm* **where** *wfrec\_Hcheck\_fm*  $\equiv (\cdot \exists \cdot \text{pair\_fm}(1, 0, 2) \wedge \text{is\_wfrec\_fm}(\text{is\_Hcheck\_fm}(8, 2, 1, 0), 4, 1, 0) \dots)$

**definition** *repl\_PHcheck\_fm* **where** *repl\_PHcheck\_fm*  $\equiv \text{PHcheck\_fm}(2, 3, 0, 1)$

**definition** *tl\_repl\_intf\_fm* **where** *tl\_repl\_intf\_fm*  $\equiv (\cdot \exists \cdot \text{pair\_fm}(1, 0, 2) \wedge \text{is\_wfrec\_fm}(\text{iterates\_MH\_fm}(\text{tl\_fm}(1, 0), 9, 2, 1, 0), 3, 1, 0) \dots)$

**definition** *formula\_repl1\_intf\_fm* **where** *formula\_repl1\_intf\_fm*  $\equiv (\cdot \exists \cdot \text{pair\_fm}(1, 0, 2) \wedge \text{is\_wfrec\_fm}(\text{iterates\_MH\_fm}(\text{formula\_functor\_fm}(1, 0), 9, 2, 1, 0), 3, 1, 0) \dots)$

**definition** *eclose\_closed\_fm* **where** *eclose\_closed\_fm*  $\equiv (\cdot \exists \cdot \text{pair\_fm}(1, 0, 2) \wedge \text{is\_wfrec\_fm}(\text{iterates\_MH\_fm}(\cdot \cup 1 \text{ is } 0 \cdot, 9, 2, 1, 0), 3, 1, 0) \dots)$

**definition** *replacement\_assm* **where**

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

**definition** *ground\_replacement\_assm* **where**

$ground\_replacement\_assm(M, env, \varphi) \equiv replacement\_assm(M, env, ground\_repl\_fm(\varphi))$

**end**

## 6 The ZFC axioms, internalized

**theory** *Internal\_ZFC\_Axioms*

**imports**

*Fm\_Definitions*

**begin**

**schematic\_goal** *ZF\_union\_auto*:

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

$\langle ML \rangle$

**notation** *ZF\_union\_fm* ( $\langle \cdot Union\ Ax \cdot \rangle$ )

**schematic\_goal** *ZF\_power\_auto*:

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

$\langle ML \rangle$

**notation** *ZF\_power\_fm* ( $\langle \cdot Powerset\ Ax \cdot \rangle$ )

**schematic\_goal** *ZF\_pairing\_auto*:

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

$\langle ML \rangle$

**notation** *ZF\_pairing\_fm* ( $\langle \cdot Pairing \cdot \rangle$ )

**schematic\_goal** *ZF\_foundation\_auto*:

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

$\langle ML \rangle$

**notation** *ZF\_foundation\_fm* ( $\langle \cdot Foundation \cdot \rangle$ )

**schematic\_goal** *ZF\_extensionality\_auto*:

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

$\langle ML \rangle$   
**notation**  $ZF\_extensionality\_fm$  ( $\langle \cdot Extensionality \cdot \rangle$ )

**schematic\_goal**  $ZF\_infinity\_auto$ :  
 $infinity\_ax(\#\#A) \longleftrightarrow (A, [] \models (? \varphi(i,j,h)))$   
 $\langle proof \rangle$

$\langle ML \rangle$   
**notation**  $ZF\_infinity\_fm$  ( $\langle \cdot Infinity \cdot \rangle$ )

**schematic\_goal**  $ZF\_choice\_auto$ :  
 $choice\_ax(\#\#A) \longleftrightarrow (A, [] \models (? \varphi(i,j,h)))$   
 $\langle proof \rangle$

$\langle ML \rangle$   
**notation**  $ZF\_choice\_fm$  ( $\langle \cdot AC \cdot \rangle$ )

**lemmas**  $ZFC\_fm\_defs = ZF\_extensionality\_fm\_def ZF\_foundation\_fm\_def ZF\_pairing\_fm\_def$   
 $ZF\_union\_fm\_def ZF\_infinity\_fm\_def ZF\_power\_fm\_def ZF\_choice\_fm\_def$

**lemmas**  $ZFC\_fm\_sats = ZF\_extensionality\_auto ZF\_foundation\_auto ZF\_pairing\_auto$   
 $ZF\_union\_auto ZF\_infinity\_auto ZF\_power\_auto ZF\_choice\_auto$

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

## 6.1 The Axiom of Separation, internalized

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

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

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

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

**lemma** *sats\_nForall*:

**assumes**

$\varphi \in \text{formula}$

**shows**

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

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

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

$\langle \text{proof} \rangle$

**definition**

*sep\_body\_fm* ::  $i \Rightarrow i$  **where**

$\text{sep\_body\_fm}(p) \equiv (\cdot \forall (\exists (\cdot \forall \cdot 0 \in 1 \leftrightarrow \cdot 0 \in 2 \wedge \text{incr\_bv1}^2(p) \dots) \cdot))$

**lemma** *sep\_body\_fm\_type* [TC]:  $p \in \text{formula} \implies \text{sep\_body\_fm}(p) \in \text{formula}$

$\langle \text{proof} \rangle$

**lemma** *sats\_sep\_body\_fm*:

**assumes**

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

**shows**

$(M, \text{rest} @ ms \models \text{sep\_body\_fm}(\varphi)) \longleftrightarrow$

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

$\langle \text{proof} \rangle$

**definition**

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

$\text{ZF\_separation\_fm}(p) \equiv \text{Forall}^{\wedge}(\text{pred}(\text{arity}(p)))(\text{sep\_body\_fm}(p))$

**lemma** *ZF\_separation\_fm\_type* [TC]:  $p \in \text{formula} \implies \text{ZF\_separation\_fm}(p) \in \text{formula}$

$\langle \text{proof} \rangle$

**lemma** *sats\_ZF\_separation\_fm\_iff*:

**assumes**

$\varphi \in \text{formula}$

**shows**

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

$\longleftrightarrow$

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

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

$\langle \text{proof} \rangle$

## 6.2 The Axiom of Replacement, internalized

**schematic\_goal** *sats\_univalent\_fm\_auto*:

**assumes**

$Q\_iff\_sats: \bigwedge x \ y \ z. x \in A \implies y \in A \implies z \in A \implies$

$Q(x, z) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, \text{env})))) \models Q1\_fm)$

$\bigwedge x y z. x \in A \implies y \in A \implies z \in A \implies$   
 $Q(x,y) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, \text{env})))) \models Q2\_fm$

**and**

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

**shows**

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

*<proof>*

*<ML>*

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

*<proof>*

**lemma** *sats\_univalent\_fm* :

**assumes**

$Q\_iff\_sats: \bigwedge x y z. x \in A \implies y \in A \implies z \in A \implies$   
 $Q(x,z) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, \text{env})))) \models Q1\_fm$

$\bigwedge x y z. x \in A \implies y \in A \implies z \in A \implies$   
 $Q(x,y) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, \text{env})))) \models Q2\_fm$

**and**

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

**shows**

$(A, \text{env} \models \text{univalent\_fm}(Q1\_fm, Q2\_fm, i)) \longleftrightarrow \text{univalent}(\#\#A, B, Q)$

*<proof>*

**definition**

*swap\_vars* ::  $i \Rightarrow i$  **where**

*swap\_vars*( $\varphi$ )  $\equiv$

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

**lemma** *swap\_vars\_type*[TC] :

$\varphi \in \text{formula} \implies \text{swap\_vars}(\varphi) \in \text{formula}$

*<proof>*

**lemma** *sats\_swap\_vars* :

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

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

*<proof>*

**definition**

*univalent\_Q1* ::  $i \Rightarrow i$  **where**

*univalent\_Q1*( $\varphi$ )  $\equiv \text{incr\_bv1}(\text{swap\_vars}(\varphi))$

**definition**

*univalent\_Q2* ::  $i \Rightarrow i$  **where**

*univalent\_Q2*( $\varphi$ )  $\equiv \text{incr\_bv}(\text{swap\_vars}(\varphi))'0$

**lemma** *univalent\_Qs\_type* [TC]:  
**assumes**  $\varphi \in \text{formula}$   
**shows**  $\text{univalent\_Q1}(\varphi) \in \text{formula}$   $\text{univalent\_Q2}(\varphi) \in \text{formula}$   
 $\langle \text{proof} \rangle$

**lemma** *sats\_univalent\_fm\_assm*:  
**assumes**  
 $x \in A$   $y \in A$   $z \in A$   $\text{env} \in \text{list}(A)$   $\varphi \in \text{formula}$   
**shows**  
 $(A, ([x,z] @ \text{env}) \models \varphi) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, \text{env})))) \models (\text{univalent\_Q1}(\varphi))$   
 $(A, ([x,y] @ \text{env}) \models \varphi) \longleftrightarrow (A, \text{Cons}(z, \text{Cons}(y, \text{Cons}(x, \text{env})))) \models (\text{univalent\_Q2}(\varphi))$   
 $\langle \text{proof} \rangle$

**definition**  
 $\text{rep\_body\_fm} :: i \Rightarrow i$  **where**  
 $\text{rep\_body\_fm}(p) \equiv \text{Forall}(\text{Implies}(\text{univalent\_fm}(\text{univalent\_Q1}(\text{incr\_bv}(p)^2), \text{univalent\_Q2}(\text{incr\_bv}(p)^2), 0), \text{Exists}(\text{Forall}(\text{Iff}(\text{Member}(0, 1), \text{Exists}(\text{And}(\text{Member}(0, 3), \text{incr\_bv}(\text{incr\_bv}(p)^2)^2))))))$

**lemma** *rep\_body\_fm\_type* [TC]:  $p \in \text{formula} \Longrightarrow \text{rep\_body\_fm}(p) \in \text{formula}$   
 $\langle \text{proof} \rangle$

**lemmas** *ZF\_replacement\_simps* = *formula\_add\_params1*[of  $\varphi$  2  $M$   $[\_, \_]$  ]  
*sats\_incr\_bv\_iff*[of  $\_ \_ M$   $[\_]$  ] — simplifies iterates of  $\lambda x. \text{incr\_bv}(x)^2$   
*sats\_incr\_bv\_iff*[of  $\_ \_ M$   $[\_, \_]$  ] — simplifies  $\lambda x. \text{incr\_bv}(x)^2$   
*sats\_incr\_bv1\_iff*[of  $\_ \_ M$  ] *sats\_swap\_vars* **for**  $\varphi$   $M$

**lemma** *sats\_rep\_body\_fm*:  
**assumes**  
 $\varphi \in \text{formula}$   $ms \in \text{list}(M)$   $rest \in \text{list}(M)$   
**shows**  
 $(M, rest @ ms \models \text{rep\_body\_fm}(\varphi)) \longleftrightarrow$   
 $\text{strong\_replacement}(\#\#M, \lambda x y. M, [x, y] @ rest @ ms \models \varphi)$   
 $\langle \text{proof} \rangle$

**definition**  
 $\text{ZF\_replacement\_fm} :: i \Rightarrow i$  ( $\cdot$  *Replacement'*  $(\_)$   $\cdot$ ) **where**  
 $\text{ZF\_replacement\_fm}(p) \equiv \text{Forall}(\text{pred}(\text{pred}(\text{arity}(p))))(\text{rep\_body\_fm}(p))$

**lemma** *ZF\_replacement\_fm\_type* [TC]:  $p \in \text{formula} \Longrightarrow \text{ZF\_replacement\_fm}(p) \in \text{formula}$   
 $\langle \text{proof} \rangle$

**lemma** *sats\_ZF\_replacement\_fm\_iff*:  
**assumes**  
 $\varphi \in \text{formula}$   
**shows**  
 $(M, [] \models \cdot \text{Replacement}(\varphi) \cdot) \longleftrightarrow (\forall \text{env}. \text{replacement\_assm}(M, \text{env}, \varphi))$

$\langle proof \rangle$

**definition**

$ZF\_schemes :: i$  **where**  
 $ZF\_schemes \equiv \{ \cdot Separation(p) \cdot \mid p \in formula \} \cup \{ \cdot Replacement(p) \cdot \mid p \in formula \}$

**lemma**  $Un\_subset\_formula$  [TC]:  $A \subseteq formula \wedge B \subseteq formula \implies A \cup B \subseteq formula$   
 $\langle proof \rangle$

**lemma**  $ZF\_schemes\_subset\_formula$  [TC]:  $ZF\_schemes \subseteq formula$   
 $\langle proof \rangle$

**lemma**  $ZF\_fin\_subset\_formula$  [TC]:  $ZF\_fin \subseteq formula$   
 $\langle proof \rangle$

**definition**

$ZF :: i$  **where**  
 $ZF \equiv ZF\_schemes \cup ZF\_fin$

**lemma**  $ZF\_subset\_formula$  [TC]:  $ZF \subseteq formula$   
 $\langle proof \rangle$

**definition**

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

**definition**

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

**definition**

$Zermelo\_fms :: i$  ( $\langle Z \cdot \rangle$ ) **where**  
 $Zermelo\_fms \equiv ZF\_fin \cup \{ \cdot Separation(p) \cdot \mid p \in formula \}$

**definition**

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

**lemma**  $ZFC\_subset\_formula$ :  $ZFC \subseteq formula$   
 $\langle proof \rangle$

Satisfaction of a set of sentences

**definition**

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

**lemma**  $satTI$  [intro!]:

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

**shows**  $A \models \Phi$   
*<proof>*

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

**lemma** *satT\_mono*:  $A \models \Phi \implies \Psi \subseteq \Phi \implies A \models \Psi$   
*<proof>*

**lemma** *satT\_Un\_iff*:  $M \models \Phi \cup \Psi \iff M \models \Phi \wedge M \models \Psi$  *<proof>*

**lemma** *sats\_ZFC\_iff\_sats\_ZF\_AC*:  
 $(N \models ZFC) \iff (N \models ZF) \wedge (N, [] \models \cdot AC \cdot)$   
*<proof>*

**lemma** *satT\_ZF\_imp\_satT\_Z*:  $M \models ZF \implies M \models \cdot Z \cdot$   
*<proof>*

**lemma** *satT\_ZFC\_imp\_satT\_ZC*:  $M \models ZFC \implies M \models ZC$   
*<proof>*

**lemma** *satT\_Z\_ZF\_replacement\_imp\_satT\_ZF*:  $N \models \cdot Z \cdot \implies N \models \{\cdot Replacement(x) \cdot$   
 $\cdot x \in formula\} \implies N \models ZF$   
*<proof>*

**lemma** *satT\_ZC\_ZF\_replacement\_imp\_satT\_ZFC*:  $N \models ZC \implies N \models \{\cdot Replacement(x) \cdot$   
 $\cdot x \in formula\} \implies N \models ZFC$   
*<proof>*

**lemma** *ground\_repl\_fm\_sub\_ZF*:  $\{\cdot Replacement(ground\_repl\_fm(\varphi)) \cdot \cdot \varphi \in formula\} \subseteq ZF$   
*<proof>*

**lemma** *ZF\_replacement\_fms\_sub\_ZFC*:  $\{\cdot Replacement(\varphi) \cdot \cdot \varphi \in formula\} \subseteq ZFC$   
*<proof>*

**lemma** *ground\_repl\_fm\_sub\_ZFC*:  $\{\cdot Replacement(ground\_repl\_fm(\varphi)) \cdot \cdot \varphi \in formula\} \subseteq ZFC$   
*<proof>*

**lemma** *ZF\_replacement\_ground\_repl\_fm\_type*:  $\{\cdot Replacement(ground\_repl\_fm(\varphi)) \cdot$   
 $\cdot \varphi \in formula\} \subseteq formula$   
*<proof>*

**end**

## 7 Interface between set models and Constructibility

This theory provides an interface between Paulson's relativization results and set models of ZFC. In particular, it is used to prove that the locale *forcing\_data* is a sublocale of all relevant locales in **ZF-Constructible** (*M\_trivial*, *M\_basic*, *M\_eclose*, etc).

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

**theory** *Interface*

**imports**

*Fm\_Definitions*

*Transitive\_Models.Cardinal\_AC\_Relative*

**begin**

**locale** *M\_Z\_basic* =

**fixes** *M*

**assumes**

*upair\_ax*: *upair\_ax*(##*M*) **and**

*Union\_ax*: *Union\_ax*(##*M*) **and**

*power\_ax*: *power\_ax*(##*M*) **and**

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

*foundation\_ax*: *foundation\_ax*(##*M*) **and**

*infinity\_ax*: *infinity\_ax*(##*M*) **and**

*separation\_ax*:  $\varphi \in \text{formula} \implies \text{env} \in \text{list}(M) \implies$

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

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

**locale** *M\_transset* =

**fixes** *M*

**assumes**

*trans\_M*: *Transset*(*M*)

**locale** *M\_Z\_trans* = *M\_Z\_basic* + *M\_transset*

**locale** *M\_ZF1* = *M\_Z\_basic* +

**assumes**

*replacement\_ax1*:

*replacement\_assm*(*M*, *env*, *eclose\_closed\_fm*)

*replacement\_assm*(*M*, *env*, *eclose\_abs\_fm*)

*replacement\_assm*(*M*, *env*, *wfrec\_rank\_fm*)

*replacement\_assm*(*M*, *env*, *transrec\_VFrom\_fm*)

**definition** *instances1\_fms* **where** *instances1\_fms*  $\equiv$

```

{ eclose_closed_fm,
  eclose_abs_fm,
  wfrec_rank_fm,
  transrec_VFrom_fm
}

```

This set has 4 internalized formulas.

```

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

```

```

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

```

```

declare (in M_ZF1) replacement_instances1_defs[simp]

```

```

locale M_ZF1_trans = M_ZF1 + M_Z_trans

```

```

context M_Z_trans
begin

```

```

lemmas transitivity = Transset_intf[OF trans_M]

```

## 7.1 Interface with $M_{trivial}$

```

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

```

```

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

```

```

end — M_Z_trans

```

```

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

```

```

locale M_ZFC1 = M_ZF1 + M_ZC_basic

```

```

locale M_ZFC1_trans = M_ZF1_trans + M_ZFC1

```

```

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

```

**sublocale**  $M\_Z\_trans \subseteq M\_trivial \#\#M$   
*<proof>*

## 7.2 Interface with $M\_basic$

**definition** *Intersection* **where**

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

*<ML>*

**definition** *CartProd* **where**

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

*<ML>*

**definition** *ImageSep* **where**

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

*<ML>*

**definition** *Converse* **where**

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

*<ML>*

**definition** *Restrict* **where**

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

*<ML>*

**definition** *Comp* **where**

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

*<ML>*

**definition** *Pred* **where**

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

*<ML>*

**definition** *is\_Memrel* **where**

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

*<ML>*

**definition** *RecFun* **where**

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

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

$\langle ML \rangle$

**context**  $M\_Z\_trans$

**begin**

**lemma** *inter\_sep\_intf* :

**assumes**  $A \in M$

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

$\langle proof \rangle$

**lemma** *diff\_sep\_intf* :

**assumes**  $B \in M$

**shows**  $\text{separation}(\#\#M, \lambda x. x \notin B)$

$\langle proof \rangle$

**lemma** *cartprod\_sep\_intf* :

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

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

$\langle proof \rangle$

**lemma** *image\_sep\_intf* :

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

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

$\langle proof \rangle$

**lemma** *converse\_sep\_intf* :

**assumes**  $R \in M$

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

$\langle proof \rangle$

**lemma** *restrict\_sep\_intf* :

**assumes**  $A \in M$

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

$\langle proof \rangle$

**lemma** *comp\_sep\_intf* :

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

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

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

$yz \in R)$

$\langle proof \rangle$

**lemma** *pred\_sep\_intf*:

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

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

**lemma** *memrel\_sep\_intf*:  
 $\text{separation}(\#\#M, \lambda z. \exists x \in M. \exists y \in M. \text{pair}(\#\#M, x, y, z) \wedge x \in y)$   
 <proof>

**lemma** *is\_recfun\_sep\_intf* :  
**assumes**  $r \in M \ f \in M \ g \in M \ a \in M \ b \in M$   
**shows**  $\text{separation}(\#\#M, \lambda x. \exists xa \in M. \exists xb \in M.$   
 $\text{pair}(\#\#M, x, a, xa) \wedge xa \in r \wedge \text{pair}(\#\#M, x, b, xb) \wedge xb \in r \wedge$   
 $(\exists fx \in M. \exists gx \in M. \text{fun\_apply}(\#\#M, f, x, fx) \wedge \text{fun\_apply}(\#\#M, g, x, gx)$   
 $\wedge$   
 $fx \neq gx))$   
 <proof>

**lemmas** *M\_basic\_sep\_instances* =  
*inter\_sep\_intf* *diff\_sep\_intf* *cartprod\_sep\_intf*  
*image\_sep\_intf* *converse\_sep\_intf* *restrict\_sep\_intf*  
*pred\_sep\_intf* *memrel\_sep\_intf* *comp\_sep\_intf* *is\_recfun\_sep\_intf*

**end** — *M\_Z\_trans*

**sublocale** *M\_Z\_trans*  $\subseteq$  *M\_basic\_no\_repl*  $\#\#M$   
 <proof>

**lemma** *Replace\_eq\_Collect*:  
**assumes**  $\bigwedge x \ y \ y'. x \in A \implies P(x, y) \implies P(x, y') \implies y = y' \ \{y . x \in A, P(x, y)\}$   
 $\subseteq B$   
**shows**  $\{y . x \in A, P(x, y)\} = \{y \in B . \exists x \in A. P(x, y)\}$   
 <proof>

**context** *M\_Z\_trans*  
**begin**

**lemma** *Pow\_inter\_M\_closed*: **assumes**  $A \in M$  **shows**  $\text{Pow}(A) \cap M \in M$   
 <proof>

**lemma** *Pow'\_inter\_M\_closed*: **assumes**  $A \in M$  **shows**  $\{a \in \text{Pow}(A) . a \in M\}$   
 $\in M$   
 <proof>

**end** — *M\_Z\_trans*

**context** *M\_basic\_no\_repl*  
**begin**

**lemma** *Replace\_funspace\_succ\_rep\_intf\_sub*:  
**assumes**

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

**lemma** *funspace\_succ\_rep\_intf\_uniq*:

**assumes**

$\text{funspace\_succ\_rep\_intf\_rel}(M,p,z,n) \text{ funspace\_succ\_rep\_intf\_rel}(M,p,z',n)$

**shows**

$z = z'$

$\langle \text{proof} \rangle$

**lemma** *Replace\_funspace\_succ\_rep\_intf\_eq*:

**assumes**

$M(A) M(n)$

**shows**

$\{z . p \in A, \text{funspace\_succ\_rep\_intf\_rel}(M,p,z,n)\} =$

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

.

$\exists p \in A. \text{funspace\_succ\_rep\_intf\_rel}(M,p,z,n)\}$

$\langle \text{proof} \rangle$

**end** — *M\_basic\_no\_repl*

**definition** *fsri* **where**

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

$\langle \text{ML} \rangle$

**context** *M\_Z\_trans*

**begin**

**lemma** *separation\_fsri*:

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

$\langle \text{proof} \rangle$

**lemma** *separation\_funspace\_succ\_rep\_intf\_rel*:

$(\#\#M)(A) \implies (\#\#M)(B) \implies \text{separation}(\#\#M, \lambda z. \exists p \in A. \text{funspace\_succ\_rep\_intf\_rel}(\#\#M,p,z,B))$

$\langle \text{proof} \rangle$

**lemma** *Replace\_funspace\_succ\_rep\_intf\_in\_M*:

**assumes**

$A \in M \ n \in M$

**shows**

$\{z . p \in A, \text{funspace\_succ\_rep\_intf\_rel}(\#\#M,p,z,n)\} \in M$

$\langle \text{proof} \rangle$

**lemma** *funspace\_succ\_rep\_intf*:  
**assumes**  $n \in M$   
**shows**  
*strong\_replacement*( $\#\#M$ ,  
 $\lambda p z. \exists f \in M. \exists b \in M. \exists nb \in M. \exists cnbf \in M.$   
 $pair(\#\#M, f, b, p) \wedge pair(\#\#M, n, b, nb) \wedge is\_cons(\#\#M, nb, f, cnbf) \wedge$   
 $upair(\#\#M, cnbf, cnbf, z)$ )  
 $\langle proof \rangle$

**end** — *M\_Z\_trans*

**sublocale**  $M\_Z\_trans \subseteq M\_basic \#\#M$   
 $\langle proof \rangle$

### 7.3 Interface with *M\_trancl*

**context** *M\_ZF1\_trans*  
**begin**

**lemma** *rtrancl\_separation\_intf*:  
**assumes**  $r \in M$   $A \in M$   
**shows** *separation* ( $\#\#M$ , *rtran\_closure\_mem*( $\#\#M, A, r$ ))  
 $\langle proof \rangle$

**lemma** *wftrancl\_separation\_intf*:  
**assumes**  $r \in M$  **and**  $Z \in M$   
**shows** *separation* ( $\#\#M$ , *wellfounded\_trancl*( $\#\#M, Z, r$ ))  
 $\langle proof \rangle$

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

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

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

**lemma** *nat\_in\_M*:  $nat \in M$   
 $\langle proof \rangle$

**end** — *M\_ZF1\_trans*

**sublocale**  $M\_ZF1\_trans \subseteq M\_trancl \#\#M$   
 $\langle proof \rangle$

### 7.4 Interface with *M\_eclose*

**lemma** *repl\_sats*:

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

$\langle ML \rangle$

**context**  $M\_ZF1\_trans$   
**begin**

This lemma obtains *iterates\_replacement* for predicates without parameters.

**lemma** *iterates\_repl\_intf* :

**assumes**  
 $v \in M$  **and**  
 $isfm: is\_F\_fm \in formula$  **and**  
 $arty: arity(is\_F\_fm) = 2$  **and**  
 $satsf: \bigwedge a b env'. \llbracket a \in M ; b \in M ; env' \in list(M) \rrbracket$   
 $\implies is\_F(a, b) \iff (M, [b, a] @ env' \models is\_F\_fm)$   
**and**  $is\_F\_fm\_replacement:$   
 $\bigwedge env. (\exists \cdot \langle 1, 0 \rangle is\ 2 \cdot \wedge is\_wfrec\_fm(iterates\_MH\_fm(is\_F\_fm, 9, 2, 1, 0), 3, 1, 0)$   
 $\cdot)) \in formula \implies env \in list(M) \implies$   
 $arity((\exists \cdot \langle 1, 0 \rangle is\ 2 \cdot \wedge is\_wfrec\_fm(iterates\_MH\_fm(is\_F\_fm, 9, 2, 1, 0), 3, 1, 0)$   
 $\cdot)) \leq 2 + \omega length(env) \implies$   
 $strong\_replacement(\#\#M, \lambda x y.$   
 $M, [x, y] @ env \models (\exists \cdot \langle 1, 0 \rangle is\ 2 \cdot \wedge is\_wfrec\_fm(iterates\_MH\_fm(is\_F\_fm, 9, 2, 1, 0), 3, 1, 0)$   
 $\cdot))$   
**shows**  
 $iterates\_replacement(\#\#M, is\_F, v)$   
 $\langle proof \rangle$

**lemma** *eclose\_repl1\_intf*:

**assumes**  $A \in M$   
**shows**  $iterates\_replacement(\#\#M, big\_union(\#\#M), A)$   
 $\langle proof \rangle$

**lemma** *eclose\_repl2\_intf*:

**assumes**  $A \in M$   
**shows**  $strong\_replacement(\#\#M, \lambda n y. n \in nat \wedge is\_iterates(\#\#M, big\_union(\#\#M),$   
 $A, n, y))$   
 $\langle proof \rangle$

**end** —  $M\_ZF1\_trans$

**sublocale**  $M\_ZF1\_trans \subseteq M\_eclose \#\#M$   
 $\langle proof \rangle$

Interface with  $M\_eclose$ .

```

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

```

<ML>

```

declare is_Hrank_fm_def[fm_definitions add]

```

```

context M_ZF1_trans
begin

```

```

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

```

```

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

```

```

end — M_ZF1_trans

```

## 7.5 Interface for proving Collects and Replace in M.

```

context M_ZF1_trans
begin

```

```

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

```

```

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

```

**end** —  $M\_ZF1\_trans$

**context**  $M\_Z\_trans$

**begin**

**lemma**  $strong\_replacement\_in\_ctm$ :

**assumes**

$f\_fm$ :  $\varphi \in formula$  **and**

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

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

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

$phi\_replacement$ :  $replacement\_assm(M, env, \varphi)$  **and**

$env \in list(M)$

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

$\langle proof \rangle$

**lemma**  $strong\_replacement\_rel\_in\_ctm$  :

**assumes**

$f\_fm$ :  $\varphi \in formula$  **and**

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

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

$phi\_replacement$ :  $replacement\_assm(M, env, \varphi)$  **and**

$env \in list(M)$

**shows**  $strong\_replacement(\#\#M, f)$

$\langle proof \rangle$

**lemma**  $Replace\_in\_M$  :

**assumes**

$f\_fm$ :  $\varphi \in formula$  **and**

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

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

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

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

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

)

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

$\langle proof \rangle$

**lemma**  $Replace\_relativized\_in\_M$  :

**assumes**

$f\_fm$ :  $\varphi \in formula$  **and**

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

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

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

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

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

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

)

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

$\langle \text{proof} \rangle$

**lemma** *ren\_action* :

**assumes**

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

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

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

$\langle \text{proof} \rangle$

**lemma** *Lambda\_in\_M* :

**assumes**

$f\_fm$ :  $\varphi \in \text{formula}$  **and**

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

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

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

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

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

$phi'\_replacement2$ :  $replacement\_assm(M, env @ [A], Lambda\_in\_M\_fm(\varphi, length(env)))$

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

$\langle \text{proof} \rangle$

**lemma** *ren\_action'* :

**assumes**

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

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

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

$\langle \text{proof} \rangle$

**lemma** *LambdaPair\_in\_M* :

**assumes**

$f\_fm$ :  $\varphi \in \text{formula}$  **and**

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

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

**and**

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

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

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

$phi'\_replacement3$ :  $replacement\_assm(M, env @ [A], LambdaPair\_in\_M\_fm(\varphi, length(env)))$

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

$\langle \text{proof} \rangle$

**lemma** (*in M\_ZF1\_trans*) *lam\_replacement2\_in\_ctm* :

**assumes**

$f\_fm$ :  $\varphi \in \text{formula}$  **and**

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

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

**and**

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

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

$env \in list(M)$  **and**  
 $phi\_replacement3: \bigwedge A. A \in M \implies replacement\_assm(M, env@[A], LambdaPair\_in\_M\_fm(\varphi, length(env)))$   
**shows**  $lam\_replacement(\#\#M, \lambda x. f(fst(x), snd(x)))$   
 $\langle proof \rangle$

$\langle ML \rangle$

**lemma**  $separation\_sat\_after\_function\_1:$

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

**lemma**  $separation\_sat\_after\_function3:$

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

**lemma**  $separation\_sat\_after\_function:$

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

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

**definition**  $\text{separation\_assm\_fm} :: [i, i, i] \Rightarrow i$

**where**

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

**lemma**  $\text{separation\_assm\_fm\_type}[TC]:$

$A \in \omega \implies y \in \omega \implies f\_fm \in \text{formula} \implies \text{separation\_assm\_fm}(A, y, f\_fm) \in$   
 $\text{formula}$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{arity\_separation\_assm\_fm} : A \in \omega \implies x \in \omega \implies f\_fm \in \text{formula} \implies$   
 $\text{arity}(\text{separation\_assm\_fm}(A, x, f\_fm)) = \text{succ}(A) \cup \text{succ}(x) \cup \text{pred}(\text{pred}(\text{arity}(f\_fm)))$   
 $\langle \text{proof} \rangle$

**definition**  $\text{separation\_assm\_bin\_fm}$  **where**

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

**lemma**  $\text{separation\_assm\_bin\_fm\_type}[TC]:$

$A \in \omega \implies y \in \omega \implies f\_fm \in \text{formula} \implies \text{separation\_assm\_bin\_fm}(A, y, f\_fm)$   
 $\in \text{formula}$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{arity\_separation\_assm\_bin\_fm} : A \in \omega \implies x \in \omega \implies f\_fm \in \text{formula}$   
 $\implies$

$\text{arity}(\text{separation\_assm\_bin\_fm}(A, x, f\_fm)) = \text{succ}(A) \cup \text{succ}(x) \cup (\text{pred}^4(\text{arity}(f\_fm)))$   
 $\langle \text{proof} \rangle$

**context**  $M\_Z\_trans$

**begin**

**lemma**  $\text{separation\_assm\_sats} :$

**assumes**

$f\_fm: \varphi \in \text{formula}$  **and**

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

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

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

**lemma** *separation\_assm\_bin\_sats* :  
**assumes**  
*f\_fm*:  $\varphi \in \text{formula}$  **and**  
*f\_ar*:  $\text{arity}(\varphi) = 3$  **and**  
*fats*:  $\bigwedge env\ x\ z\ y. env \in \text{list}(M) \implies x \in M \implies z \in M \implies y \in M \implies (M, [x,z,y]@env \models \varphi) \longleftrightarrow is\_f(x,z,y)$  **and**  
*fabs*:  $\bigwedge x\ z\ y. x \in M \implies z \in M \implies y \in M \implies is\_f(x,z,y) \longleftrightarrow y = f(x,z)$  **and**  
*fclosed*:  $\bigwedge x\ z . x \in M \implies z \in M \implies f(x,z) \in M$  **and**  
 $A \in M$   
**shows** *separation*( $\#\#M, \lambda y. \exists x \in M . x \in A \wedge y = \langle x, f(\text{fst}(x), \text{snd}(x)) \rangle$ )  
*<proof>*

**lemma** *separation\_Union*:  $A \in M \implies$   
*separation*( $\#\#M, \lambda y. \exists x \in M . x \in A \wedge y = \langle x, \text{Union}(x) \rangle$ )  
*<proof>*

**lemma** *lam\_replacement\_Union*: *lam\_replacement*( $\#\#M, \text{Union}$ )  
*<proof>*

**lemma** *separation\_fst*:  $A \in M \implies$   
*separation*( $\#\#M, \lambda y. \exists x \in M . x \in A \wedge y = \langle x, \text{fst}(x) \rangle$ )  
*<proof>*

**lemma** *lam\_replacement\_fst*: *lam\_replacement*( $\#\#M, \text{fst}$ )  
*<proof>*

**lemma** *separation\_snd*:  $A \in M \implies$   
*separation*( $\#\#M, \lambda y. \exists x \in M . x \in A \wedge y = \langle x, \text{snd}(x) \rangle$ )  
*<proof>*

**lemma** *lam\_replacement\_snd*: *lam\_replacement*( $\#\#M, \text{snd}$ )  
*<proof>*

Binary lambda-replacements

**lemma** *separation\_Image*:  $A \in M \implies$   
*separation*( $\#\#M, \lambda y. \exists x \in M . x \in A \wedge y = \langle x, \text{fst}(x) \text{ “ } \text{snd}(x) \rangle$ )  
*<proof>*

**lemma** *lam\_replacement\_Image*: *lam\_replacement*( $\#\#M, \lambda x . \text{fst}(x) \text{ “ } \text{snd}(x)$ )  
*<proof>*

**lemma** *separation\_middle\_del*:  $A \in M \implies$

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

**lemma**  $lam\_replacement\_middle\_del$ :  $lam\_replacement(\#\#M, \lambda r. middle\_del(fst(r), snd(r)))$   
 $\langle proof \rangle$

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

**lemma**  $lam\_replacement\_prodRepl$ :  $lam\_replacement(\#\#M, \lambda r. prodRepl(fst(r), snd(r)))$   
 $\langle proof \rangle$

**end** —  $M\_Z\_trans$

**context**  $M\_trivial$   
**begin**

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

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

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

**end** —  $M\_trivial$

**context**  $M\_Z\_trans$   
**begin**

**lemma** (**in**  $M\_basic$ )  $is\_minimum\_equivalence$  :  
 $M(R) \implies M(X) \implies M(u) \implies is\_minimum(M, R, X, u) \longleftrightarrow is\_minimum'(M, R, X, u)$   
 $\langle proof \rangle$

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

**lemma**  $lam\_replacement\_minimum$ :  $lam\_replacement(\#\#M, \lambda x. minimum(fst(x), snd(x)))$   
 $\langle proof \rangle$

**end** —  $M\_Z\_trans$

**end**

## 7.6 More Instances of Separation

**theory** *Separation\_Instances*

**imports**

*Interface*

**begin**

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

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

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

$\langle ML \rangle$

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

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

$\langle ML \rangle$

**lemma** (in *M\_replacement*) *separation\_well\_ord\_iso*:

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

**definition** *is\_obase\_body* ::  $[i \Rightarrow o, i, i, i] \Rightarrow o$  **where**

$$\begin{aligned} \text{is\_obase\_body}(N,A,r,x) \equiv & x \in A \longrightarrow \\ & \neg (\exists y[N]. \\ & \quad \exists g[N]. \\ & \quad \text{ordinal}(N, y) \wedge \\ & \quad (\exists my[N]. \\ & \quad \quad \exists pxr[N]. \\ & \quad \quad \text{membership}(N, y, my) \wedge \\ & \quad \quad \text{pred\_set}(N, A, x, r, pxr) \wedge \\ & \quad \quad \text{order\_isomorphism}(N, pxr, r, y, my, g))) \end{aligned}$$

$\langle ML \rangle$

**definition** *is\_obase\_equals* ::  $[i \Rightarrow o, i, i, i] \Rightarrow o$  **where**

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

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

$\langle ML \rangle$

**context**  $M\_ZF1\_trans$

**begin**

**lemma**  $\text{radd\_body\_abs}$ :

**assumes**  $(\#\#M)(R) (\#\#M)(S) (\#\#M)(x)$   
**shows**  $\text{is\_radd\_body}(\#\#M, R, S, x) \longleftrightarrow \text{radd\_body}(R, S, x)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{separation\_radd\_body}$ :

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

**lemma**  $\text{rmult\_body\_abs}$ :

**assumes**  $(\#\#M)(b) (\#\#M)(d) (\#\#M)(x)$   
**shows**  $\text{is\_rmult\_body}(\#\#M, b, d, x) \longleftrightarrow \text{rmult\_body}(b, d, x)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{separation\_rmult\_body}$ :

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

**lemma**  $\text{separation\_is\_obase}$ :

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

**lemma**  $\text{separation\_obase\_equals}$ :

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

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

**lemma separation\_PiP\_rel:**  
 $(\#\#M)(A) \implies separation(\#\#M, PiP\_rel(\#\#M, A))$   
 ⟨proof⟩

**lemma separation\_injP\_rel:**  
 $(\#\#M)(A) \implies separation(\#\#M, injP\_rel(\#\#M, A))$   
 ⟨proof⟩

**lemma separation\_surjP\_rel:**  
 $(\#\#M)(A) \implies (\#\#M)(B) \implies separation(\#\#M, surjP\_rel(\#\#M, A, B))$   
 ⟨proof⟩

**lemma separation\_is\_function:**  
 $separation(\#\#M, is\_function(\#\#M))$   
 ⟨proof⟩

**end** —  $M\_ZF1\_trans$

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

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

**context**  $M\_ZF1\_trans$   
**begin**

**lemma fstsnd\_in\_sndsnd\_abs:**  
**assumes**  $(\#\#M)(x)$   
**shows**  $is\_fstsnd\_in\_sndsnd(\#\#M, x) \longleftrightarrow fstsnd\_in\_sndsnd(x)$   
 ⟨proof⟩

**lemma separation\_fstsnd\_in\_sndsnd:**  
 $separation(\#\#M, \lambda x. fst(snd(x)) \in snd(snd(x)))$   
 ⟨proof⟩

**lemma sndfst\_eq\_fstsnd\_abs:**

```

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

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

end —  $M\_ZF1\_trans$ 

end

```

## 8 More Instances of Replacement

**theory**  $Replacement\_Instances$

**imports**

$Separation\_Instances$

$Transitive\_Models.Pointed\_DC\_Relative$

**begin**

**lemma**  $composition\_fm\_type[TC]$ :  $a0 \in \omega \implies a1 \in \omega \implies a2 \in \omega \implies$   
 $composition\_fm(a0,a1,a2) \in formula$   
 $\langle proof \rangle$

$\langle ML \rangle$

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

$\langle ML \rangle$

**definition**  $HAleph\_wfrec\_repl\_body$  **where**

$HAleph\_wfrec\_repl\_body(N, mesa, x, z) \equiv \exists y[N].$

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

$(\exists g[N].$

$(\forall u[N].$

$u \in g \longleftrightarrow$

$(\exists a[N].$

$\exists y[N].$

$\exists ax[N].$

$\exists sx[N].$

$\exists r\_sx[N].$

$\exists f\_r\_sx[N].$

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

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

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

$pre\_image(N, mesa, sx, r\_sx) \wedge$

$restriction(N, g, r\_sx, f\_r\_sx) \wedge ax \in mesa \wedge is\_HAleph(N, a, f\_r\_sx, y)))$



**lemma** *is\_nat\_case\_dcwit\_aux\_fm\_type*[TC]:  $A \in \omega \implies a \in \omega \implies s \in \omega \implies R \in \omega \implies \text{is\_nat\_case\_dcwit\_aux\_fm}(A, a, s, R) \in \text{formula}$

*<proof>*

*<ML>*

*<proof>*

*<ML>*

*<proof>*

**lemma** *arity\_dcwit\_repl\_body*:  $\text{arity}(\text{dcwit\_repl\_body\_fm}(6, 5, 4, 3, 2, 0, 1)) = 7$

*<proof>*

**definition** *fst2\_snd2*

**where**  $\text{fst2\_snd2}(x) \equiv \langle \text{fst}(\text{fst}(x)), \text{snd}(\text{snd}(x)) \rangle$

*<ML>*

**lemma** (**in** *M\_trivial*) *fst2\_snd2\_abs*:

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

**shows**  $\text{is\_fst2\_snd2}(M, x, \text{res}) \longleftrightarrow \text{res} = \text{fst2\_snd2}(x)$

*<proof>*

*<ML>*

**definition** *sndfst\_fst2\_snd2*

**where**  $\text{sndfst\_fst2\_snd2}(x) \equiv \langle \text{snd}(\text{fst}(x)), \text{fst}(\text{fst}(x)), \text{snd}(\text{snd}(x)) \rangle$

*<ML>*

**definition** *order\_eq\_map* **where**

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

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

$\text{pred\_set}(M, A, a, r, par) \ \& \ \text{order\_isomorphism}(M, par, r, x, mx, g)$

*<ML>*

**definition** *banach\_body\_iterates* **where**

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

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

$(\exists fa[M].$

$(\forall z[M].$

$z \in fa \longleftrightarrow$

$(\exists xa[M].$

$\exists y[M].$

$\exists xaa[M].$

$\exists sx[M].$

$\exists r\_sx[M].$

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

$\langle ML \rangle$

**definition** *banach\_is\_iterates\_body* where

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

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

$$\begin{aligned}
& \text{successor}(M, m, x) \longrightarrow \\
& (\exists gm[M]. \\
& \quad \text{fun\_apply}(M, f\_r\_sx, m, gm) \wedge \\
\text{is\_banach\_functor}(M, X, Y, f, g, gm, y))) \wedge \\
& \quad (\text{is\_quasimat}(M, x) \vee \text{empty}(M, y))) \wedge \\
& (\text{empty}(M, n) \longrightarrow y = W) \wedge \\
& (\forall m[M]. \\
& \quad \text{successor}(M, m, n) \longrightarrow \\
& \quad (\exists gm[M]. \text{fun\_apply}(M, fa, m, gm) \wedge \text{is\_banach\_functor}(M, \\
& X, Y, f, g, gm, y))) \wedge \\
& \quad (\text{is\_quasimat}(M, n) \vee \text{empty}(M, y)))
\end{aligned}$$

$\langle ML \rangle$

**definition** *trans\_apply\_image* **where**

$$\text{trans\_apply\_image}(f) \equiv \lambda a. g. f \text{ ' } (g \text{ ' } a)$$

$\langle ML \rangle$

**schematic\_goal** *arity\_is\_recfun\_fm*[arity]:

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

**schematic\_goal** *arity\_is\_wfrec\_fm*[arity]:

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

**schematic\_goal** *arity\_is\_transrec\_fm*[arity]:

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

$\langle ML \rangle$

**definition** *transrec\_apply\_image\_body* **where**

$$\begin{aligned}
\text{transrec\_apply\_image\_body}(M, f, mesa, x, z) \equiv & \exists y[M]. \text{pair}(M, x, y, z) \wedge \\
& (\exists fa[M]. \\
& \quad (\forall z[M]. \\
& \quad \quad z \in fa \longleftrightarrow \\
& \quad \quad (\exists xa[M]. \\
& \quad \quad \quad \exists y[M]. \\
& \quad \quad \quad \quad \exists xaa[M]. \\
& \quad \quad \quad \quad \quad \exists sx[M]. \\
& \quad \quad \quad \quad \quad \quad \exists r\_sx[M]. \\
& \quad \quad \quad \quad \quad \quad \quad \exists f\_r\_sx[M].
\end{aligned}$$

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

$\langle ML \rangle$

**definition** *is\_trans\_apply\_image\_body* **where**

$$\begin{aligned}
\text{is\_trans\_apply\_image\_body}(M, f, \beta, a, w) \equiv & \exists z[M]. \text{pair}(M, a, z, w) \wedge a \in \beta \wedge (\exists sa[M]. \\
& \exists esa[M]. \\
& \exists mesa[M]. \\
& \text{upair}(M, a, a, sa) \wedge \\
& \text{is\_eclose}(M, sa, esa) \wedge \\
& \text{membership}(M, esa, mesa) \wedge \\
& (\exists fa[M]. \\
& (\forall z[M]. \\
& z \in fa \longleftrightarrow \\
& (\exists x[M]. \\
& \exists y[M]. \\
& \exists xa[M]. \\
& \exists sx[M]. \\
& \exists r\_sx[M]. \\
& \exists f\_r\_sx[M]. \\
& \text{pair}(M, x, y, z) \wedge \\
& \text{pair}(M, x, a, xa) \wedge \\
& \text{upair}(M, x, x, sx) \wedge \\
& \text{pre\_image}(M, mesa, sx, r\_sx) \wedge \\
& \text{restriction}(M, fa, r\_sx, f\_r\_sx) \wedge \\
& xa \in mesa \wedge \text{is\_trans\_apply\_image}(M, f, \\
& x, f\_r\_sx, y))) \wedge \\
& \text{is\_trans\_apply\_image}(M, f, a, fa, z)))
\end{aligned}$$

$\langle ML \rangle$

**definition** *replacement\_is\_omega\_funspace\_fm* **where** *replacement\_is\_omega\_funspace\_fm*  
 $\equiv \text{omega\_funspace\_fm}(2, 0, 1)$

**definition** *wfrec\_Aleph\_fm* **where** *wfrec\_Aleph\_fm*  $\equiv \text{HAleph\_wfrec\_repl\_body\_fm}(2, 0, 1)$

**definition** *replacement\_is\_fst2\_snd2\_fm* **where** *replacement\_is\_fst2\_snd2\_fm*  
 $\equiv \text{is\_fst2\_snd2\_fm}(0, 1)$

**definition** *replacement\_is\_sndfst\_fst2\_snd2\_fm* **where** *replacement\_is\_sndfst\_fst2\_snd2\_fm*  
 $\equiv \text{is\_sndfst\_fst2\_snd2\_fm}(0, 1)$

**definition** *omap\_replacement\_fm* **where** *omap\_replacement\_fm*  $\equiv \text{order\_eq\_map\_fm}(2, 3, 0, 1)$

**definition** *rec\_constr\_abs\_fm* **where** *rec\_constr\_abs\_fm*  $\equiv \text{transrec\_apply\_image\_body\_fm}(3, 2, 0, 1)$

**definition** *banach\_replacement\_iterates\_fm* **where** *banach\_replacement\_iterates\_fm*

$\equiv$  *banach\_is\_iterates\_body\_fm*(6,5,4,3,2,0,1)

**definition** *rec\_constr\_fm* **where** *rec\_constr\_fm*  $\equiv$  *is\_trans\_apply\_image\_body\_fm*(3,2,0,1)

**definition** *dc\_abs\_fm* **where** *dc\_abs\_fm*  $\equiv$  *dcwit\_repl\_body\_fm*(6,5,4,3,2,0,1)

**definition** *lam\_replacement\_check\_fm* **where** *lam\_replacement\_check\_fm*  $\equiv$  *Lambda\_in\_M\_fm*(*check\_fm*)

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

**locale** *M\_ZF\_ground* = *M\_ZF1* +  
**assumes**

*ZF\_ground\_replacements*:  
*replacement\_assm*(*M*,*env*,*wfrec\_Hfrc\_at\_fm*)  
*replacement\_assm*(*M*,*env*,*wfrec\_Hcheck\_fm*)  
*replacement\_assm*(*M*,*env*,*lam\_replacement\_check\_fm*)

**locale** *M\_ZF\_ground\_trans* = *M\_ZF1\_trans* + *M\_ZF\_ground*

**definition** *instances\_ground\_fms* **where** *instances\_ground\_fms*  $\equiv$   
{ *wfrec\_Hfrc\_at\_fm*,  
*wfrec\_Hcheck\_fm*,  
*lam\_replacement\_check\_fm* }

**lemmas** *replacement\_instances\_ground\_defs* =  
*wfrec\_Hfrc\_at\_fm\_def* *wfrec\_Hcheck\_fm\_def* *lam\_replacement\_check\_fm\_def*

**declare** (**in** *M\_ZF\_ground*) *replacement\_instances\_ground\_defs* [*simp*]

**lemma** *instances\_ground\_fms\_type*[*TC*]: *instances\_ground\_fms*  $\subseteq$  *formula*  
{*proof*}

**locale** *M\_ZF\_ground\_notCH* = *M\_ZF\_ground* +  
**assumes**

*ZF\_ground\_notCH\_replacements*:  
*replacement\_assm*(*M*,*env*,*rec\_constr\_abs\_fm*)  
*replacement\_assm*(*M*,*env*,*rec\_constr\_fm*)

**definition** *instances\_ground\_notCH\_fms* **where** *instances\_ground\_notCH\_fms*  
 $\equiv$   
{ *rec\_constr\_abs\_fm*,  
*rec\_constr\_fm* }

**lemma** *instances\_ground\_notCH\_fms\_type*[*TC*]: *instances\_ground\_notCH\_fms*  
 $\subseteq$  *formula*  
{*proof*}

**declare** (**in** *M\_ZF\_ground\_notCH*) *rec\_constr\_abs\_fm\_def* [*simp*]  
*rec\_constr\_fm\_def* [*simp*]

```

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

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

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

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

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

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

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

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

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

end —  $M\_ZF\_ground\_CH\_trans$ 

```

**context**  $M\_ZF1\_trans$   
**begin**

**lemmas**  $M\_replacement\_ZF\_instances = lam\_replacement\_fst lam\_replacement\_snd$   
 $lam\_replacement\_Union lam\_replacement\_Image$   
 $lam\_replacement\_middle\_del lam\_replacement\_prodRepl$

**lemmas**  $M\_separation\_ZF\_instances = separation\_fstsnd\_in\_sndsnd separation\_sndfst\_eq\_fstsnd$

**lemma**  $separation\_is\_dcwit\_body$ :  
**assumes**  $(\#\#M)(A) (\#\#M)(a) (\#\#M)(g) (\#\#M)(R)$   
**shows**  $separation(\#\#M, is\_dcwit\_body(\#\#M, A, a, g, R))$   
 $\langle proof \rangle$

**end** —  $M\_ZF1\_trans$

**sublocale**  $M\_ZF1\_trans \subseteq M\_replacement \#\#M$   
 $\langle proof \rangle$

**context**  $M\_ZF1\_trans$   
**begin**

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

**lemma**  $strong\_replacement\_Powapply\_rel$ :  
 $f \in M \implies strong\_replacement(\#\#M, \lambda x y. y = Powapply^{\#\#M}(f, x))$   
 $\langle proof \rangle$

**end** —  $M\_ZF1\_trans$

**sublocale**  $M\_ZF1\_trans \subseteq M\_Vfrom \#\#M$   
 $\langle proof \rangle$

**sublocale**  $M\_ZF1\_trans \subseteq M\_Perm \#\#M$   
 $\langle proof \rangle$

**sublocale**  $M\_ZF1\_trans \subseteq M\_pre\_seqspace \#\#M$   
 $\langle proof \rangle$

**context**  $M\_ZF1\_trans$   
**begin**

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

**lemma**  $lam\_replacement\_inj\_rel$ :  $lam\_replacement(\#\#M, \lambda x. inj\_rel(\#\#M, fst(x), snd(x)))$

*<proof>*

**end** — *M\_ZF1\_trans*

**lemma** (in *M\_basic*) *rel2\_trans\_apply*:

$M(f) \implies \text{relation2}(M, \text{is\_trans\_apply\_image}(M, f), \text{trans\_apply\_image}(f))$   
*<proof>*

**lemma** (in *M\_basic*) *apply\_image\_closed*:

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

**context** *M\_ZF\_ground\_notCH\_trans*

**begin**

**lemma** *replacement\_transrec\_apply\_image\_body* :

$(\#\#M)(f) \implies (\#\#M)(\text{mesa}) \implies \text{strong\_replacement}(\#\#M, \text{transrec\_apply\_image\_body}(\#\#M, f, \text{mesa}))$   
*<proof>*

**lemma** *transrec\_replacement\_apply\_image*:

**assumes**  $(\#\#M)(f) (\#\#M)(\alpha)$   
**shows**  $\text{transrec\_replacement}(\#\#M, \text{is\_trans\_apply\_image}(\#\#M, f), \alpha)$   
*<proof>*

**lemma** *rec\_trans\_apply\_image\_abs*:

**assumes**  $(\#\#M)(f) (\#\#M)(x) (\#\#M)(y) \text{Ord}(x)$   
**shows**  $\text{is\_transrec}(\#\#M, \text{is\_trans\_apply\_image}(\#\#M, f), x, y) \longleftrightarrow y = \text{transrec}(x, \text{trans\_apply\_image}(f))$   
*<proof>*

**lemma** *replacement\_is\_trans\_apply\_image*:

$(\#\#M)(f) \implies (\#\#M)(\beta) \implies \text{strong\_replacement}(\#\#M, \lambda x z . \exists y[\#\#M]. \text{pair}(\#\#M, x, y, z) \wedge x \in \beta \wedge (\text{is\_transrec}(\#\#M, \text{is\_trans\_apply\_image}(\#\#M, f), x, y)))$   
*<proof>*

**lemma** *trans\_apply\_abs*:

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

**lemma** *replacement\_trans\_apply\_image*:

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

**end** —  $M\_ZF\_ground\_notCH\_trans$

**definition**  $ifrFb\_body$  **where**

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

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

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

$\langle ML \rangle$

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

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

$\langle ML \rangle$

**lemma** (in  $M\_Z\_trans$ )  $separation\_is\_ifrangeF\_body$ :

$(\#\#M)(A) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(s) \Longrightarrow separation(\#\#M, is\_ifrangeF\_body(\#\#M,A,r,s))$

$\langle proof \rangle$

**lemma** (in  $M\_basic$ )  $is\_ifrFb\_body\_closed: M(r) \Longrightarrow M(s) \Longrightarrow is\_ifrFb\_body(M, r, s, x, i) \Longrightarrow M(i)$

$\langle proof \rangle$

**lemma** (in  $M\_ZF1\_trans$ )  $ifrangeF\_body\_abs$ :

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

**shows**  $is\_ifrangeF\_body(\#\#M,A,r,s,x) \longleftrightarrow ifrangeF\_body(\#\#M,A,r,s,x)$

$\langle proof \rangle$

**lemma** (in  $M\_ZF1\_trans$ )  $separation\_ifrangeF\_body$ :

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

$(\#\#M, \lambda y. \exists x \in A. y = \langle x, \mu i. x \in if\_range\_F\_else\_F(\lambda x. if (\#\#M)(x)$   
then  $x$  else 0,  $b, f, i \rangle)$

$\langle proof \rangle$

**definition**  $ifrFb\_body2$  **where**

$ifrFb\_body2(M,G,b,f,x,i) \equiv x \in$

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

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

$\langle ML \rangle$

**definition**  $ifrangeF\_body2 :: [i \Rightarrow o, i, i, i, i, i] \Rightarrow o$  **where**

$ifrangeF\_body2(M,A,G,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb\_body2(M,G,b,f,x,i) \rangle$

$\langle ML \rangle$

**lemma** (in  $M\_Z\_trans$ ) *separation\_is\_ifrangeF\_body2*:

$(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(r) \implies (\#\#M)(s) \implies separation(\#\#M, is\_ifrangeF\_body2(\#\#M,A,G,r,s))$   
 ⟨proof⟩

**lemma** (in  $M\_basic$ ) *is\_ifrFb\_body2\_closed*:  $M(G) \implies M(r) \implies M(s) \implies is\_ifrFb\_body2(M, G, r, s, x, i) \implies M(i)$

⟨proof⟩

**lemma** (in  $M\_ZF1\_trans$ ) *ifrangeF\_body2\_abs*:

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

**shows**  $is\_ifrangeF\_body2(\#\#M,A,G,r,s,x) \longleftrightarrow ifrangeF\_body2(\#\#M,A,G,r,s,x)$   
 ⟨proof⟩

**lemma** (in  $M\_ZF1\_trans$ ) *separation\_ifrangeF\_body2*:

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

*separation*

$(\#\#M,$

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

$y =$

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

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

$i \rangle \rangle)$

⟨proof⟩

**definition** *ifrFb\_body3* **where**

$ifrFb\_body3(M, G, b, f, x, i) \equiv x \in$

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

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

⟨ML⟩

**definition** *ifrangeF\_body3* ::  $[i \Rightarrow o, i, i, i, i, i] \Rightarrow o$  **where**

$ifrangeF\_body3(M, A, G, b, f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb\_body3(M, G, b, f, x, i) \rangle$

⟨ML⟩

**lemma** (in  $M\_Z\_trans$ ) *separation\_is\_ifrangeF\_body3*:

$(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(r) \implies (\#\#M)(s) \implies separation(\#\#M, is\_ifrangeF\_body3(\#\#M,A,G,r,s))$

⟨proof⟩

**lemma** (in  $M\_basic$ ) *is\_ifrFb\_body3\_closed*:  $M(G) \implies M(r) \implies M(s) \implies is\_ifrFb\_body3(M, G, r, s, x, i) \implies M(i)$

⟨proof⟩

**lemma** (in  $M\_ZF1\_trans$ ) *ifrangeF\_body3\_abs*:  
**assumes**  $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$   
**shows**  $is\_ifrangeF\_body3(\#\#M,A,G,r,s,x) \longleftrightarrow ifrangeF\_body3(\#\#M,A,G,r,s,x)$   
 $\langle proof \rangle$

**lemma** (in  $M\_ZF1\_trans$ ) *separation\_ifrangeF\_body3*:  
 $(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(b) \Longrightarrow (\#\#M)(f) \Longrightarrow$   
*separation*  
 $(\#\#M,$   
 $\lambda y. \exists x \in A.$   
 $y =$   
 $\langle x, \mu i. x \in$   
 $if\_range\_F\_else\_F(\lambda a. if (\#\#M)(a) then G^{-}\{a\} else 0, b,$   
 $f, i) \rangle)$   
 $\langle proof \rangle$

**definition** *ifrFb\_body4* **where**  
 $ifrFb\_body4(G,b,f,x,i) \equiv x \in$   
 $(if b = 0 then if i \in range(f) then G^{-}(converse(f) \text{ } i) else 0 else G^{-}i)$

$\langle ML \rangle$

**definition** *ifrangeF\_body4*  $:: [i \Rightarrow o, i, i, i, i] \Rightarrow o$  **where**  
 $ifrangeF\_body4(M,A,G,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb\_body4(G,b,f,x,i) \rangle$

$\langle ML \rangle$

**lemma** (in  $M\_Z\_trans$ ) *separation\_is\_ifrangeF\_body4*:  
 $(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(s) \Longrightarrow separation(\#\#M,$   
 $is\_ifrangeF\_body4(\#\#M,A,G,r,s))$   
 $\langle proof \rangle$

**lemma** (in  $M\_basic$ ) *is\_ifrFb\_body4\_closed*:  $M(G) \Longrightarrow M(r) \Longrightarrow M(s) \Longrightarrow$   
 $is\_ifrFb\_body4(M, G, r, s, x, i) \Longrightarrow M(i)$   
 $\langle proof \rangle$

**lemma** (in  $M\_ZF1\_trans$ ) *ifrangeF\_body4\_abs*:  
**assumes**  $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$   
**shows**  $is\_ifrangeF\_body4(\#\#M,A,G,r,s,x) \longleftrightarrow ifrangeF\_body4(\#\#M,A,G,r,s,x)$   
 $\langle proof \rangle$

**lemma** (in  $M\_ZF1\_trans$ ) *separation\_ifrangeF\_body4*:  
 $(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(b) \Longrightarrow (\#\#M)(f) \Longrightarrow$   
 $separation(\#\#M, \lambda y. \exists x \in A. y = \langle x, \mu i. x \in if\_range\_F\_else\_F((\text{ })(G),$   
 $b, f, i) \rangle)$   
 $\langle proof \rangle$

**definition** *ifrFb\_body5* **where**

$ifrFb\_body5(G,b,f,x,i) \equiv x \in$   
 $(if\ b = 0\ then\ if\ i \in\ range(f)\ then\ \{xa \in G .\ converse(f)\ 'i \in xa\}\ else\ 0\ else\ \{xa$   
 $\in G .\ i \in xa\})$

$\langle ML \rangle$

**definition** *ifrangeF\_body5*  $:: [i \Rightarrow o, i, i, i, i] \Rightarrow o$  **where**

$ifrangeF\_body5(M,A,G,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb\_body5(G,b,f,x,i) \rangle$

$\langle ML \rangle$

**lemma** (in *M\_Z\_trans*) *separation\_is\_ifrangeF\_body5*:

$(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(r) \Longrightarrow (\#\#M)(s) \Longrightarrow separation(\#\#M,$   
 $is\_ifrangeF\_body5(\#\#M,A,G,r,s))$   
 $\langle proof \rangle$

**lemma** (in *M\_basic*) *is\_ifrFb\_body5\_closed*:  $M(G) \Longrightarrow M(r) \Longrightarrow M(s) \Longrightarrow$   
 $is\_ifrFb\_body5(M, G, r, s, x, i) \Longrightarrow M(i)$

$\langle proof \rangle$

**lemma** (in *M\_ZF1\_trans*) *ifrangeF\_body5\_abs*:

**assumes**  $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$   
**shows**  $is\_ifrangeF\_body5(\#\#M,A,G,r,s,x) \longleftrightarrow ifrangeF\_body5(\#\#M,A,G,r,s,x)$   
 $\langle proof \rangle$

**lemma** (in *M\_ZF1\_trans*) *separation\_ifrangeF\_body5*:

$(\#\#M)(A) \Longrightarrow (\#\#M)(G) \Longrightarrow (\#\#M)(b) \Longrightarrow (\#\#M)(f) \Longrightarrow$   
 $separation(\#\#M, \lambda y. \exists x \in A. y = \langle x, \mu i. x \in if\_range\_F\_else\_F(\lambda x. \{xa$   
 $\in G .\ x \in xa\}, b, f, i) \rangle)$   
 $\langle proof \rangle$

**definition** *ifrFb\_body6* **where**

$ifrFb\_body6(G,b,f,x,i) \equiv x \in$   
 $(if\ b = 0\ then\ if\ i \in\ range(f)\ then\ \{p \in G .\ domain(p) = converse(f)\ 'i\}\ else\ 0$   
 $else\ \{p \in G .\ domain(p) = i\})$

$\langle ML \rangle$

**definition** *ifrangeF\_body6*  $:: [i \Rightarrow o, i, i, i, i] \Rightarrow o$  **where**

$ifrangeF\_body6(M,A,G,b,f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. ifrFb\_body6(G,b,f,x,i) \rangle$

$\langle ML \rangle$

**lemma** (in *M\_Z\_trans*) *separation\_is\_ifrangeF\_body6*:

$(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(r) \implies (\#\#M)(s) \implies \text{separation}(\#\#M, \text{is\_ifrangeF\_body6}(\#\#M, A, G, r, s))$   
 ⟨proof⟩

**lemma** (in  $M\_basic$ )  $\text{ifrFb\_body6\_closed}: M(G) \implies M(r) \implies M(s) \implies \text{ifrFb\_body6}(G, r, s, x, i) \longleftrightarrow M(i) \wedge \text{ifrFb\_body6}(G, r, s, x, i)$   
 ⟨proof⟩

**lemma** (in  $M\_basic$ )  $\text{is\_ifrFb\_body6\_closed}: M(G) \implies M(r) \implies M(s) \implies \text{is\_ifrFb\_body6}(M, G, r, s, x, i) \implies M(i)$   
 ⟨proof⟩

**lemma** (in  $M\_ZF1\_trans$ )  $\text{ifrangeF\_body6\_abs}$ :  
 assumes  $(\#\#M)(A) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s) (\#\#M)(x)$   
 shows  $\text{is\_ifrangeF\_body6}(\#\#M, A, G, r, s, x) \longleftrightarrow \text{ifrangeF\_body6}(\#\#M, A, G, r, s, x)$   
 ⟨proof⟩

**lemma** (in  $M\_ZF1\_trans$ )  $\text{separation\_ifrangeF\_body6}$ :  
 $(\#\#M)(A) \implies (\#\#M)(G) \implies (\#\#M)(b) \implies (\#\#M)(f) \implies \text{separation}(\#\#M, \lambda y. \exists x \in A. y = \langle x, \mu i. x \in \text{if\_range\_F\_else\_F}(\lambda a. \{p \in G . \text{domain}(p) = a\}, b, f, i)))$   
 ⟨proof⟩

**definition**  $\text{ifrFb\_body7}$  **where**

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

⟨ML⟩

**definition**  $\text{ifrangeF\_body7} :: [i \Rightarrow o, i, i, i, i, i, i] \Rightarrow o$  **where**

$\text{ifrangeF\_body7}(M, A, B, D, G, b, f) \equiv \lambda y. \exists x \in A. y = \langle x, \mu i. \text{ifrFb\_body7}(B, D, G, b, f, x, i) \rangle$

⟨ML⟩

**lemma** (in  $M\_Z\_trans$ )  $\text{separation\_is\_ifrangeF\_body7}$ :

$(\#\#M)(A) \implies (\#\#M)(B) \implies (\#\#M)(D) \implies (\#\#M)(G) \implies (\#\#M)(r) \implies (\#\#M)(s) \implies \text{separation}(\#\#M, \text{is\_ifrangeF\_body7}(\#\#M, A, B, D, G, r, s))$   
 ⟨proof⟩

**lemma** (in  $M\_basic$ )  $\text{ifrFb\_body7\_closed}: M(B) \implies M(D) \implies M(G) \implies M(r) \implies M(s) \implies \text{ifrFb\_body7}(B, D, G, r, s, x, i) \longleftrightarrow M(i) \wedge \text{ifrFb\_body7}(B, D, G, r, s, x, i)$

*<proof>*

**lemma** (in *M\_basic*) *is\_ifrFb\_body7\_closed*:  $M(B) \implies M(D) \implies M(G) \implies M(r) \implies M(s) \implies$   
 $is\_ifrFb\_body7(M, B, D, G, r, s, x, i) \implies M(i)$   
*<proof>*

**lemma** (in *M\_ZF1\_trans*) *ifrangeF\_body7\_abs*:  
**assumes**  $(\#\#M)(A) (\#\#M)(B) (\#\#M)(D) (\#\#M)(G) (\#\#M)(r) (\#\#M)(s)$   
 $(\#\#M)(x)$   
**shows**  $is\_ifrangeF\_body7(\#\#M, A, B, D, G, r, s, x) \longleftrightarrow ifrangeF\_body7(\#\#M, A, B, D, G, r, s, x)$   
*<proof>*

**lemma** (in *M\_ZF1\_trans*) *separation\_ifrangeF\_body7*:  
 $(\#\#M)(A) \implies (\#\#M)(B) \implies (\#\#M)(D) \implies (\#\#M)(G) \implies (\#\#M)(b) \implies$   
 $(\#\#M)(f) \implies$   
 $separation(\#\#M,$   
 $\lambda y. \exists x \in A. y = \langle x, \mu i. x \in if\_range\_F\_else\_F(drSR\_Y(B, D, G), b, f, i) \rangle)$   
*<proof>*

**definition** *omfunspace* ::  $[i, i] \Rightarrow o$  **where**  
 $omfunspace(B) \equiv \lambda z. \exists x. \exists n \in \omega. z \in x \wedge x = n \rightarrow B$   
*<ML>*

**context** *M\_pre\_seqspace*  
**begin**

*<ML>*  
*<proof>*

**end** — *M\_pre\_seqspace*

**context** *M\_ZF1\_trans*  
**begin**

**lemma** *separation\_omfunspace*:  
**assumes**  $(\#\#M)(B)$   
**shows**  $separation(\#\#M, \lambda z. \exists x[\#\#M]. \exists n[\#\#M]. n \in \omega \wedge z \in x \wedge x = n \rightarrow^M B)$   
*<proof>*

**end** — *M\_ZF1\_trans*

**sublocale**  $M\_ZF1\_trans \subseteq M\_seqspace \#\#M$   
*<proof>*

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

```

definition cdeggamma :: [i] ⇒ o where
  cdeggamma ≡ λZ . |fst(Z)| = snd(Z)
⟨ML⟩

context M_Perm
begin

⟨ML⟩
  ⟨proof⟩

⟨ML⟩
  ⟨proof⟩

lemma is_cdeggamma_iff_split: M(Z) ⇒ cdeggamma_rel(M, Z) ↔ (λ⟨x,y⟩.
|x|M = y)(Z)
  ⟨proof⟩

end

context M_ZF1_trans
begin

lemma separation_cdltgamma:
  assumes (##M)(γ)
  shows separation(##M, λZ . cardinal_rel(##M,Z) < γ)
  ⟨proof⟩

lemma separation_cdeggamma:
  shows separation(##M, λZ. (λ⟨x,y⟩ . cardinal_rel(##M,x) = y)(Z))
  ⟨proof⟩

end — M_ZF1_trans

end

```

## 9 Further instances of axiom-schemes

```

theory ZF_Trans_Interpretations
  imports
    Internal_ZFC_Axioms
    Replacement_Instances

begin

locale M_ZF2 = M_ZF1 +
  assumes
    replacement_ax2:
    replacement_assm(M,env,ordtype_replacement_fm)

```

```

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

```

**definition** *instances2\_fms* **where** *instances2\_fms*  $\equiv$   
{ *ordtype\_replacement\_fm*,  
*wfrec\_ordertype\_fm*,  
*wfrec\_Aleph\_fm*,  
*omap\_replacement\_fm* }

**lemmas** *replacement\_instances2\_defs* =  
*ordtype\_replacement\_fm\_def wfrec\_ordertype\_fm\_def*  
*wfrec\_Aleph\_fm\_def omap\_replacement\_fm\_def*

**declare** (in *M\_ZF2*) *replacement\_instances2\_defs* [*simp*]

**locale** *M\_ZF2\_trans* = *M\_ZF1\_trans* + *M\_ZF2*

**locale** *M\_ZFC2* = *M\_ZFC1* + *M\_ZF2*

**locale** *M\_ZFC2\_trans* = *M\_ZFC1\_trans* + *M\_ZF2\_trans* + *M\_ZFC2*

**locale** *M\_ZF2\_ground\_notCH* = *M\_ZF2* + *M\_ZF\_ground\_notCH*

**locale** *M\_ZF2\_ground\_notCH\_trans* = *M\_ZF2\_trans* + *M\_ZF2\_ground\_notCH*  
+ *M\_ZF\_ground\_notCH\_trans*

**locale** *M\_ZFC2\_ground\_notCH* = *M\_ZFC2* + *M\_ZF2\_ground\_notCH*

**locale** *M\_ZFC2\_ground\_notCH\_trans* = *M\_ZFC2\_trans* + *M\_ZFC2\_ground\_notCH*  
+ *M\_ZF2\_ground\_notCH\_trans*

**locale** *M\_ZFC2\_ground\_CH\_trans* = *M\_ZFC2\_ground\_notCH\_trans* + *M\_ZF\_ground\_CH\_trans*

**locale** *M\_ctm2* = *M\_ctm1* + *M\_ZF2\_ground\_notCH\_trans*

**locale** *M\_ctm2\_AC* = *M\_ctm2* + *M\_ctm1\_AC* + *M\_ZFC2\_ground\_notCH\_trans*

**locale** *M\_ctm2\_AC\_CH* = *M\_ctm2\_AC* + *M\_ZFC2\_ground\_CH\_trans*

**lemmas** (in *M\_ZF1\_trans*) *separation\_instances* =  
*separation\_well\_ord\_iso*  
*separation\_obase\_equals separation\_is\_obase*  
*separation\_PiP\_rel separation\_surjP\_rel*  
*separation\_radd\_body separation\_rmult\_body*

**context** *M\_ZF2\_trans*  
**begin**

**lemma** *replacement\_HAleph\_wfrec\_repl\_body*:

$B \in M \implies \text{strong\_replacement}(\#\#M, \text{HAleph\_wfrec\_repl\_body}(\#\#M, B))$

*<proof>*

**lemma** *HAleph\_wfrec\_repl*:

$(\#\#M)(sa) \implies$

$(\#\#M)(esa) \implies$

$(\#\#M)(mesa) \implies$

*strong\_replacement*

$(\#\#M,$

$\lambda x z. \exists y[\#\#M].$

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

$(\exists f[\#\#M].$

$(\forall z[\#\#M].$

$z \in f \longleftrightarrow$

$(\exists xa[\#\#M].$

$\exists y[\#\#M].$

$\exists xaa[\#\#M].$

$\exists sx[\#\#M].$

$\exists r\_sx[\#\#M].$

$\exists f\_r\_sx[\#\#M].$

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

$\text{pair}(\#\#M, xa, x, xaa) \wedge$

$\text{upair}(\#\#M, xa, xa, sx) \wedge$

$\text{pre\_image}(\#\#M, mesa, sx, r\_sx) \wedge$

$\text{restriction}(\#\#M, f, r\_sx, f\_r\_sx) \wedge xaa \in mesa \wedge \text{is\_HAleph}(\#\#M, xa, f\_r\_sx, y)) \wedge$

$\text{is\_HAleph}(\#\#M, x, f, y))$

*<proof>*

**lemma** *replacement\_is\_order\_eq\_map*:

$A \in M \implies r \in M \implies \text{strong\_replacement}(\#\#M, \text{order\_eq\_map}(\#\#M, A, r))$

*<proof>*

**end** — *M\_ZF2\_trans*

**definition** *omap\_wfrec\_body* **where**

$\text{omap\_wfrec\_body}(A, r) \equiv (\cdot \exists \cdot \text{image\_fm}(2, 0, 1) \wedge \text{pred\_set\_fm}(A \# + 9, 3, r \# + 9, 0) \cdot \cdot)$

**lemma** *type\_omap\_wfrec\_body\_fm* :  $A \in \text{nat} \implies r \in \text{nat} \implies \text{omap\_wfrec\_body}(A, r) \in \text{formula}$

*<proof>*

**lemma** *arity\_aux* :  $A \in \text{nat} \implies r \in \text{nat} \implies \text{arity}(\text{omap\_wfrec\_body}(A, r)) = (9 +_\omega A)$

$\cup (9 +_\omega r)$

*<proof>*

**lemma** *arity\_omap\_wfrec* :  $A \in \text{nat} \implies r \in \text{nat} \implies$

$\text{arity}(\text{is\_wfrec\_fm}(\text{omap\_wfrec\_body}(A, r), \text{succ}(\text{succ}(\text{succ}(r))), 1, 0)) =$

$(4+\omega A) \cup (4+\omega r)$   
 $\langle proof \rangle$

**lemma** *arity\_isordermap*:  $A \in nat \implies r \in nat \implies d \in nat \implies$   
 $arity(is\_ordermap\_fm(A,r,d)) = succ(d) \cup (succ(A) \cup succ(r))$   
 $\langle proof \rangle$

**lemma** *arity\_is\_ordertype*:  $A \in nat \implies r \in nat \implies d \in nat \implies$   
 $arity(is\_ordertype\_fm(A,r,d)) = succ(d) \cup (succ(A) \cup succ(r))$   
 $\langle proof \rangle$

**lemma** *arity\_is\_order\_body*:  $arity(is\_order\_body\_fm(1,0)) = 2$   
 $\langle proof \rangle$

**lemma** (in *M\_ZF2\_trans*) *replacement\_is\_order\_body*:  
 $strong\_replacement(\#\#M, \lambda x z . \exists y[\#\#M]. is\_order\_body(\#\#M,x,y) \wedge z =$   
 $\langle x,y \rangle)$   
 $\langle proof \rangle$

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

$\langle ML \rangle$

**lemma** (in *M\_basic*) *H\_order\_pred\_abs* :  
 $M(A) \implies M(r) \implies M(x) \implies M(f) \implies M(z) \implies$   
 $is\_H\_order\_pred(M,A,r,x,f,z) \longleftrightarrow z = H\_order\_pred(A,r,x,f)$   
 $\langle proof \rangle$

$\langle ML \rangle$

**lemma** (in *M\_ZF2\_trans*) *wfrec\_replacement\_order\_pred*:  
 $A \in M \implies r \in M \implies wfrec\_replacement(\#\#M, \lambda x g z . is\_H\_order\_pred(\#\#M,A,r,x,g,z)$   
 $, r)$   
 $\langle proof \rangle$

**lemma** (in *M\_ZF2\_trans*) *wfrec\_replacement\_order\_pred'*:  
 $A \in M \implies r \in M \implies wfrec\_replacement(\#\#M, \lambda x g z . z = H\_order\_pred(A,r,x,g)$   
 $, r)$   
 $\langle proof \rangle$

**sublocale** *M\_ZF2\_trans*  $\subseteq$  *M\_pre\_cardinal\_arith*  $\#\#M$   
 $\langle proof \rangle$

**definition** *is\_well\_ord\_fst\_snd* **where**  
 $is\_well\_ord\_fst\_snd(A,x) \equiv (\exists a[A]. \exists b[A]. is\_well\_ord(A,a,b) \wedge is\_snd(A, x,$   
 $b) \wedge is\_fst(A, x, a))$

$\langle ML \rangle$

**lemma** (in  $M\_ZF2\_trans$ ) *separation\_well\_ord*:  $separation(\#\#M, \lambda x. is\_well\_ord(\#\#M, fst(x), snd(x)))$   
 $\langle proof \rangle$

**sublocale**  $M\_ZF2\_trans \subseteq M\_pre\_aleph \#\#M$   
 $\langle proof \rangle$

$\langle ML \rangle$

**lemma** *arity\_is\_HAleph\_fm*:  $arity(is\_HAleph\_fm(2, 1, 0)) = 3$   
 $\langle proof \rangle$

**lemma** *arity\_is\_Aleph*[*arity*]:  $arity(is\_Aleph\_fm(0, 1)) = 2$   
 $\langle proof \rangle$

**definition** *bex\_Aleph\_rel* ::  $[i \Rightarrow o, i, i] \Rightarrow o$  **where**  
 $bex\_Aleph\_rel(M, x) \equiv \lambda y. \exists z \in x. y = \aleph_z^M$

$\langle ML \rangle$

**schematic\_goal** *sats\_is\_bex\_Aleph\_fm\_auto*:  
 $a \in nat \Longrightarrow c \in nat \Longrightarrow env \in list(A) \Longrightarrow$   
 $a < length(env) \Longrightarrow c < length(env) \Longrightarrow 0 \in A \Longrightarrow$   
 $is\_bex\_Aleph(\#\#A, nth(a, env), nth(c, env)) \longleftrightarrow A, env \models ?fm(a, c)$   
 $\langle proof \rangle$

$\langle ML \rangle$

**lemma** *is\_bex\_Aleph\_fm\_type* [TC]:  
 $x \in \omega \Longrightarrow z \in \omega \Longrightarrow is\_bex\_Aleph\_fm(x, z) \in formula$   
 $\langle proof \rangle$

**lemma** *sats\_is\_bex\_Aleph\_fm*:  
 $x \in \omega \Longrightarrow$   
 $z \in \omega \Longrightarrow x < length(env) \Longrightarrow z < length(env) \Longrightarrow$   
 $env \in list(Aa) \Longrightarrow$   
 $0 \in Aa \Longrightarrow$   
 $(Aa, env \models is\_bex\_Aleph\_fm(x, z)) \longleftrightarrow$   
 $is\_bex\_Aleph(\#\#Aa, nth(x, env), nth(z, env))$   
 $\langle proof \rangle$

**lemma** *is\_bex\_Aleph\_iff\_sats* [*iff\_sats*]:  
 $nth(x, env) = xa \Longrightarrow$   
 $nth(z, env) = za \Longrightarrow$   
 $x \in \omega \Longrightarrow$   
 $z \in \omega \Longrightarrow x < length(env) \Longrightarrow z < length(env) \Longrightarrow$

$env \in list(Aa) \implies$   
 $0 \in Aa \implies$   
 $is\_bex\_Aleph(\#\#Aa, xa, za) \longleftrightarrow$   
 $Aa, env \models is\_bex\_Aleph\_fm(x, z)$   
 <proof>

<ML>

**lemma** (in  $M\_ZF1\_trans$ ) *separation\_is\_bex\_Aleph*:  
**assumes**  $(\#\#M)(A)$   
**shows**  $separation(\#\#M, is\_bex\_Aleph(\#\#M, A))$   
 <proof>

**lemma** (in  $M\_pre\_aleph$ ) *bex\_Aleph\_rel\_abs*:  
**assumes**  $Ord(u) M(u) M(v)$   
**shows**  $is\_bex\_Aleph(M, u, v) \longleftrightarrow bex\_Aleph\_rel(M, u, v)$   
 <proof>

**lemma** (in  $M\_ZF2\_trans$ ) *separation\_bex\_Aleph\_rel*:  
 $Ord(x) \implies (\#\#M)(x) \implies separation(\#\#M, bex\_Aleph\_rel(\#\#M, x))$   
 <proof>

**sublocale**  $M\_ZF2\_trans \subseteq M\_aleph \#\#M$   
 <proof>

**sublocale**  $M\_ZF1\_trans \subseteq M\_FiniteFun \#\#M$   
 <proof>

**sublocale**  $M\_ZFC2\_trans \subseteq M\_cardinal\_AC \#\#M$   
 <proof>

**lemma** (in  $M\_ZF1\_trans$ ) *separation\_cardinal\_rel\_lespoll\_rel*:  
 $(\#\#M)(\kappa) \implies separation(\#\#M, \lambda x. x \prec^M \kappa)$   
 <proof>

**sublocale**  $M\_ZFC2\_trans \subseteq M\_library \#\#M$   
 <proof>

**locale**  $M\_ZF3 = M\_ZF2 +$   
**assumes**  
 $ground\_replacements3:$   
 $ground\_replacement\_assm(M, env, ordtype\_replacement\_fm)$   
 $ground\_replacement\_assm(M, env, wfrec\_ordertype\_fm)$   
 $ground\_replacement\_assm(M, env, eclose\_abs\_fm)$   
 $ground\_replacement\_assm(M, env, wfrec\_rank\_fm)$   
 $ground\_replacement\_assm(M, env, transrec\_VFrom\_fm)$   
 $ground\_replacement\_assm(M, env, eclose\_closed\_fm)$

$ground\_replacement\_assm(M, env, wfrec\_Aleph\_fm)$   
 $ground\_replacement\_assm(M, env, omap\_replacement\_fm)$

**definition**  $instances3\_fms$  **where**  $instances3\_fms \equiv$   
 $\{$   $ground\_repl\_fm(ordtype\_replacement\_fm),$   
 $ground\_repl\_fm(wfrec\_ordertype\_fm),$   
 $ground\_repl\_fm(eclose\_abs\_fm),$   
 $ground\_repl\_fm(wfrec\_rank\_fm),$   
 $ground\_repl\_fm(transrec\_VFrom\_fm),$   
 $ground\_repl\_fm(eclose\_closed\_fm),$   
 $ground\_repl\_fm(wfrec\_Aleph\_fm),$   
 $ground\_repl\_fm(omap\_replacement\_fm)$   $\}$

This set has 8 internalized formulas, corresponding to the total count of previous replacement instances (apart from those 5 in  $instances\_ground\_fms$  and  $instances\_ground\_notCH\_fms$ , and  $dc\_abs\_fm$ ).

**definition**  $overhead$  **where**  
 $overhead \equiv instances1\_fms \cup instances\_ground\_fms$

**definition**  $overhead\_notCH$  **where**  
 $overhead\_notCH \equiv overhead \cup instances2\_fms \cup$   
 $instances3\_fms \cup instances\_ground\_notCH\_fms$

**definition**  $overhead\_CH$  **where**  
 $overhead\_CH \equiv overhead\_notCH \cup \{ dc\_abs\_fm \}$

Hence, the “overhead” to create a proper extension of a ctm by forcing consists of 7 replacement instances. To force  $\neg CH$ , 21 instances are need, and one further instance is required to force  $CH$ .

**lemma**  $instances2\_fms\_type[TC] : instances2\_fms \subseteq formula$   
 $\langle proof \rangle$

**lemma**  $overhead\_type : overhead \subseteq formula$   
 $\langle proof \rangle$

**lemma**  $overhead\_notCH\_type : overhead\_notCH \subseteq formula$   
 $\langle proof \rangle$

**lemma**  $overhead\_CH\_type : overhead\_CH \subseteq formula$   
 $\langle proof \rangle$

**locale**  $M\_ZF3\_trans = M\_ZF2\_trans + M\_ZF3$

**locale**  $M\_ZFC3 = M\_ZFC2 + M\_ZF3$

**locale**  $M\_ZFC3\_trans = M\_ZFC2\_trans + M\_ZF3\_trans + M\_ZFC3$

**locale**  $M\_ctm3 = M\_ctm2 + M\_ZF3\_trans$

**locale**  $M\_ctm3\_AC = M\_ctm3 + M\_ctm1\_AC + M\_ZFC3\_trans$

**lemma**  $M\_satT\_imp\_M\_ZF2$ :  $(M \models ZF) \implies M\_ZF1(M)$   
*<proof>*

**lemma**  $M\_satT\_imp\_M\_ZFC1$ :  
**shows**  $(M \models ZFC) \longrightarrow M\_ZFC1(M)$   
*<proof>*

**lemma**  $M\_satT\_instances1\_imp\_M\_ZF1$ :  
**assumes**  $(M \models \cdot Z \cdot \cup \{\cdot Replacement(p) \cdot \cdot p \in instances1\_fms \})$   
**shows**  $M\_ZF1(M)$   
*<proof>*

**theorem**  $M\_satT\_imp\_M\_ZF\_ground\_trans$ :  
**assumes**  $Transset(M) \ M \models \cdot Z \cdot \cup \{\cdot Replacement(p) \cdot \cdot p \in overhead\}$   
**shows**  $M\_ZF\_ground\_trans(M)$   
*<proof>*

**theorem**  $M\_satT\_imp\_M\_ZF\_ground\_notCH\_trans$ :  
**assumes**  
     $Transset(M)$   
     $M \models \cdot Z \cdot \cup \{\cdot Replacement(p) \cdot \cdot p \in overhead\_notCH\}$   
**shows**  $M\_ZF\_ground\_notCH\_trans(M)$   
*<proof>*

**theorem**  $M\_satT\_imp\_M\_ZF\_ground\_CH\_trans$ :  
**assumes**  
     $Transset(M)$   
     $M \models \cdot Z \cdot \cup \{\cdot Replacement(p) \cdot \cdot p \in overhead\_CH \}$   
**shows**  $M\_ZF\_ground\_CH\_trans(M)$   
*<proof>*

**lemma** (in  $M\_Z\_basic$ )  $M\_satT\_Zermelo\_fms$ :  $M \models \cdot Z \cdot$   
*<proof>*

**lemma** (in  $M\_ZFC1$ )  $M\_satT\_ZC$ :  $M \models ZC$   
*<proof>*

**locale**  $M\_ZF = M\_Z\_basic +$   
**assumes**  
     $replacement\_ax: replacement\_assm(M, env, \varphi)$

**sublocale**  $M\_ZF \subseteq M\_ZF3$   
*<proof>*

**lemma**  $M\_satT\_imp\_M\_ZF$ :  $M \models ZF \implies M\_ZF(M)$   
*<proof>*

```

lemma (in  $M\_ZF$ )  $M\_satT\_ZF: M \models ZF$ 
  <proof>

lemma  $M\_ZF\_iff\_M\_satT: M\_ZF(M) \longleftrightarrow (M \models ZF)$ 
  <proof>

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

sublocale  $M\_ZFC \subseteq M\_ZFC3$ 
  <proof>

lemma  $M\_ZFC\_iff\_M\_satT:$ 
  notes  $iff\_trans[trans]$ 
  shows  $M\_ZFC(M) \longleftrightarrow (M \models ZFC)$ 
  <proof>

lemma  $M\_satT\_imp\_M\_ZF3: (M \models ZF) \longrightarrow M\_ZF3(M)$ 
  <proof>

lemma  $M\_satT\_imp\_M\_ZFC3:$ 
  shows  $(M \models ZFC) \longrightarrow M\_ZFC3(M)$ 
  <proof>

lemma  $M\_satT\_overhead\_imp\_M\_ZF3:$ 
   $(M \models ZC \cup \{\cdot Replacement(p) \cdot \mid p \in overhead\_notCH\}) \longrightarrow M\_ZFC3(M)$ 
  <proof>

end

```

## 10 Transitive set models of ZF

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

```

theory Forcing_Data
  imports
    Forcing_Notions
    Cohen_Posets_Relative
    ZF_Trans_Interpretations
begin

```

```

no_notation Aleph ( $\aleph_{\_}$ ) [90] 90)

```

### 10.1 A forcing locale and generic filters

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

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

```

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

```

```

locale forcing_data2 = forcing_data1 + M_ctm2_AC

```

```

locale forcing_data3 = forcing_data2 + M_ctm3_AC

```

```

context forcing_data1
begin

```

```

lemma P_sub_M :  $\mathbb{P} \subseteq M$ 
  <proof>

```

**definition**

```

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

```

```

declare iff_trans [trans]

```

```

lemma M_generic_imp_filter[dest]:  $M\_generic(G) \Longrightarrow filter(G)$ 
  <proof>

```

```

lemma generic_filter_existence:
   $p \in \mathbb{P} \Longrightarrow \exists G. p \in G \wedge M\_generic(G)$ 
  <proof>

```

```

lemma one_in_M:  $\mathbf{1} \in M$ 
  <proof>

```

```

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

```

```

end — forcing_data1

```

```

locale G_generic1 = forcing_data1 +
  fixes G ::  $i$ 
  assumes generic : M_generic(G)
begin

```

```

lemma G_nonempty:  $G \neq 0$ 
  <proof>

```

```

lemma M_genericD [dest]:  $x \in G \Longrightarrow x \in \mathbb{P}$ 
  <proof>

```

**lemma** *M\_generic\_leqD* [dest]:  $p \in G \implies q \in \mathbb{P} \implies p \preceq q \implies q \in G$   
 ⟨proof⟩

**lemma** *M\_generic\_compatD* [dest]:  $p \in G \implies r \in G \implies \exists q \in G. q \preceq p \wedge q \preceq r$   
 ⟨proof⟩

**lemma** *M\_generic\_denseD* [dest]:  $\text{dense}(D) \implies D \subseteq \mathbb{P} \implies D \in M \implies \exists q \in G. q \in D$   
 ⟨proof⟩

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

**lemma** *one\_in\_G* :  $\mathbf{1} \in G$   
 ⟨proof⟩

**lemma** *G\_subset\_M*:  $G \subseteq M$   
 ⟨proof⟩

**end** — *G\_generic1*

**locale** *G\_generic1\_AC* = *G\_generic1* + *M\_ctm1\_AC*

**end**

## 11 The definition of forces

**theory** *Forces\_Definition*

**imports**

*Forcing\_Data*

**begin**

This is the core of our development.

### 11.1 The relation *frecrel*

**lemma** *names\_belowsD*:

**assumes**  $x \in \text{names\_below}(P, z)$

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

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

⟨proof⟩

**context** *forcing\_data1*

**begin**

**lemma** *f\_type\_abs*:

$\llbracket x \in M; y \in M \rrbracket \implies \text{is\_f\_type}(\#\#M, x, y) \longleftrightarrow y = \text{f\_type}(x)$

⟨proof⟩

**lemma** *name1\_abs*:

$\llbracket x \in M; y \in M \rrbracket \implies is\_name1(\#\#M, x, y) \longleftrightarrow y = name1(x)$   
*<proof>*

**lemma** *snd\_snd\_abs*:

$\llbracket x \in M; y \in M \rrbracket \implies is\_snd\_snd(\#\#M, x, y) \longleftrightarrow y = snd(snd(x))$   
*<proof>*

**lemma** *name2\_abs*:

$\llbracket x \in M; y \in M \rrbracket \implies is\_name2(\#\#M, x, y) \longleftrightarrow y = name2(x)$   
*<proof>*

**lemma** *cond\_of\_abs*:

$\llbracket x \in M; y \in M \rrbracket \implies is\_cond\_of(\#\#M, x, y) \longleftrightarrow y = cond\_of(x)$   
*<proof>*

**lemma** *tuple\_abs*:

$\llbracket z \in M; t1 \in M; t2 \in M; p \in M; t \in M \rrbracket \implies$   
 $is\_tuple(\#\#M, z, t1, t2, p, t) \longleftrightarrow t = \langle z, t1, t2, p \rangle$   
*<proof>*

**lemmas** *components\_abs = ftype\_abs name1\_abs name2\_abs cond\_of\_abs tuple\_abs*

**lemma** *comp\_in\_M*:

$p \preceq q \implies p \in M$   
 $p \preceq q \implies q \in M$   
*<proof>*

**lemma** *eq\_case\_abs [simp]*:

**assumes**  $t1 \in M$   $t2 \in M$   $p \in M$   $f \in M$   
**shows**  $is\_eq\_case(\#\#M, t1, t2, p, \mathbb{P}, leq, f) \longleftrightarrow eq\_case(t1, t2, p, \mathbb{P}, leq, f)$   
*<proof>*

**lemma** *mem\_case\_abs [simp]*:

**assumes**  $t1 \in M$   $t2 \in M$   $p \in M$   $f \in M$   
**shows**  $is\_mem\_case(\#\#M, t1, t2, p, \mathbb{P}, leq, f) \longleftrightarrow mem\_case(t1, t2, p, \mathbb{P}, leq, f)$   
*<proof>*

**lemma** *Hfrc\_abs*:

$\llbracket fnnc \in M; f \in M \rrbracket \implies$   
 $is\_Hfrc(\#\#M, \mathbb{P}, leq, fnnc, f) \longleftrightarrow Hfrc(\mathbb{P}, leq, fnnc, f)$   
*<proof>*

**lemma** *Hfrc\_at\_abs*:

$\llbracket fnnc \in M; f \in M ; z \in M \rrbracket \implies$

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

**lemma** *components\_closed* :

$x \in M \implies (\#\#M)(ftype(x))$   
 $x \in M \implies (\#\#M)(name1(x))$   
 $x \in M \implies (\#\#M)(name2(x))$   
 $x \in M \implies (\#\#M)(cond\_of(x))$   
 ⟨proof⟩

**lemma** *ecloseN\_closed*:

$(\#\#M)(A) \implies (\#\#M)(ecloseN(A))$   
 $(\#\#M)(A) \implies (\#\#M)(eclose\_n(name1, A))$   
 $(\#\#M)(A) \implies (\#\#M)(eclose\_n(name2, A))$   
 ⟨proof⟩

**lemma** *eclose\_n\_abs* :

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

**lemma** *ecloseN\_abs* :

$\llbracket x \in M; ec \in M \rrbracket \implies is\_ecloseN(\#\#M, x, ec) \longleftrightarrow ec = ecloseN(x)$   
 ⟨proof⟩

**lemma** *frecR\_abs* :

$x \in M \implies y \in M \implies frecR(x, y) \longleftrightarrow is\_frecR(\#\#M, x, y)$   
 ⟨proof⟩

**lemma** *frecrelP\_abs* :

$z \in M \implies frecrelP(\#\#M, z) \longleftrightarrow (\exists x y. z = \langle x, y \rangle \wedge frecR(x, y))$   
 ⟨proof⟩

**lemma** *frecrel\_abs*:

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

**lemma** *frecrel\_closed*:

**assumes**  $x \in M$   
**shows**  $frecrel(x) \in M$   
 ⟨proof⟩

**lemma** *field\_frecrel* :  $field(frecrel(names\_below(\mathbb{P}, x))) \subseteq names\_below(\mathbb{P}, x)$

⟨proof⟩

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

*<proof>*

**lemma** *wf\_forcerel* :

*wf*(*forcerel*( $\mathbb{P}, x$ ))

*<proof>*

**lemma** *restrict\_trancl\_forcerel*:

**assumes** *frecR*( $w, y$ )

**shows**  $\text{restrict}(f, \text{frecrel}(\text{names\_below}(\mathbb{P}, x)) - \{\!-\{y\}\}) \text{ 'w}$   
 $= \text{restrict}(f, \text{forcerel}(\mathbb{P}, x) - \{\!-\{y\}\}) \text{ 'w}$

*<proof>*

**lemma** *names\_belowI* :

**assumes** *frecR*( $\langle ft, n1, n2, p \rangle, \langle a, b, c, d \rangle$ )  $p \in \mathbb{P}$

**shows**  $\langle ft, n1, n2, p \rangle \in \text{names\_below}(\mathbb{P}, \langle a, b, c, d \rangle)$  (**is**  $?x \in \text{names\_below}(\_, ?y)$ )

*<proof>*

**lemma** *names\_below\_tr* :

**assumes**  $x \in \text{names\_below}(\mathbb{P}, y)$   $y \in \text{names\_below}(\mathbb{P}, z)$

**shows**  $x \in \text{names\_below}(\mathbb{P}, z)$

*<proof>*

**lemma** *arg\_into\_names\_below2* :

**assumes**  $\langle x, y \rangle \in \text{frecrel}(\text{names\_below}(\mathbb{P}, z))$

**shows**  $x \in \text{names\_below}(\mathbb{P}, y)$

*<proof>*

**lemma** *arg\_into\_names\_below* :

**assumes**  $\langle x, y \rangle \in \text{frecrel}(\text{names\_below}(\mathbb{P}, z))$

**shows**  $x \in \text{names\_below}(\mathbb{P}, x)$

*<proof>*

**lemma** *forcerel\_arg\_into\_names\_below* :

**assumes**  $\langle x, y \rangle \in \text{forcerel}(\mathbb{P}, z)$

**shows**  $x \in \text{names\_below}(\mathbb{P}, x)$

*<proof>*

**lemma** *names\_below\_mono* :

**assumes**  $\langle x, y \rangle \in \text{frecrel}(\text{names\_below}(\mathbb{P}, z))$

**shows**  $\text{names\_below}(\mathbb{P}, x) \subseteq \text{names\_below}(\mathbb{P}, y)$

*<proof>*

**lemma** *frecrel\_mono* :

**assumes**  $\langle x, y \rangle \in \text{frecrel}(\text{names\_below}(\mathbb{P}, z))$

**shows**  $\text{frecrel}(\text{names\_below}(\mathbb{P}, x)) \subseteq \text{frecrel}(\text{names\_below}(\mathbb{P}, y))$

*<proof>*

**lemma** *forcerel\_mono2* :

**assumes**  $\langle x, y \rangle \in \text{frecrel}(\text{names\_below}(\mathbb{P}, z))$

**shows**  $\text{forcere}(\mathbb{P},x) \subseteq \text{forcere}(\mathbb{P},y)$   
 <proof>

**lemma forcere\_mono\_aux :**  
**assumes**  $\langle x,y \rangle \in \text{frecr}(\text{names\_below}(\mathbb{P}, w)) \hat{+}$   
**shows**  $\text{forcere}(\mathbb{P},x) \subseteq \text{forcere}(\mathbb{P},y)$   
 <proof>

**lemma forcere\_mono :**  
**assumes**  $\langle x,y \rangle \in \text{forcere}(\mathbb{P},z)$   
**shows**  $\text{forcere}(\mathbb{P},x) \subseteq \text{forcere}(\mathbb{P},y)$   
 <proof>

**lemma forcere\_eq\_aux:**  $x \in \text{names\_below}(\mathbb{P}, w) \implies \langle x,y \rangle \in \text{forcere}(\mathbb{P},z) \implies$   
 $(y \in \text{names\_below}(\mathbb{P}, w) \longrightarrow \langle x,y \rangle \in \text{forcere}(\mathbb{P},w))$   
 <proof>

**lemma forcere\_eq :**  
**assumes**  $\langle z,x \rangle \in \text{forcere}(\mathbb{P},x)$   
**shows**  $\text{forcere}(\mathbb{P},z) = \text{forcere}(\mathbb{P},x) \cap \text{names\_below}(\mathbb{P},z) \times \text{names\_below}(\mathbb{P},z)$   
 <proof>

**lemma forcere\_below\_aux :**  
**assumes**  $\langle z,x \rangle \in \text{forcere}(\mathbb{P},x)$   $\langle u,z \rangle \in \text{forcere}(\mathbb{P},x)$   
**shows**  $u \in \text{names\_below}(\mathbb{P},z)$   
 <proof>

**lemma forcere\_below :**  
**assumes**  $\langle z,x \rangle \in \text{forcere}(\mathbb{P},x)$   
**shows**  $\text{forcere}(\mathbb{P},x) - \{z\} \subseteq \text{names\_below}(\mathbb{P},z)$   
 <proof>

**lemma relation\_forcere :**  
**shows**  $\text{relation}(\text{forcere}(\mathbb{P},z)) \text{ trans}(\text{forcere}(\mathbb{P},z))$   
 <proof>

**lemma Hfrc\_restrict\_trancl:**  $\text{bool\_of\_o}(Hfrc(\mathbb{P}, \text{leq}, y, \text{restrict}(f, \text{frecr}(\text{names\_below}(\mathbb{P},x) - \{y\})))$   
 $= \text{bool\_of\_o}(Hfrc(\mathbb{P}, \text{leq}, y, \text{restrict}(f, (\text{frecr}(\text{names\_below}(\mathbb{P},x)) \hat{+} - \{y\})))$   
 <proof>

**lemma frc\_at\_trancl:**  $\text{frc\_at}(\mathbb{P}, \text{leq}, z) = \text{wfrec}(\text{forcere}(\mathbb{P},z), z, \lambda x f. \text{bool\_of\_o}(Hfrc(\mathbb{P}, \text{leq}, x, f)))$   
 <proof>

**lemma forcereI1 :**  
**assumes**  $n1 \in \text{domain}(b) \vee n1 \in \text{domain}(c)$   $p \in \mathbb{P}$   $d \in \mathbb{P}$   
**shows**  $\langle \langle 1, n1, b, p \rangle, \langle 0, b, c, d \rangle \rangle \in \text{forcere}(\mathbb{P}, \langle 0, b, c, d \rangle)$   
 <proof>

**lemma** *forcerelI2* :

**assumes**  $n1 \in \text{domain}(b) \vee n1 \in \text{domain}(c) \ p \in \mathbb{P} \ d \in \mathbb{P}$

**shows**  $\langle \langle 1, n1, c, p \rangle, \langle 0, b, c, d \rangle \rangle \in \text{forcerel}(\mathbb{P}, \langle 0, b, c, d \rangle)$

*<proof>*

**lemma** *forcerelI3* :

**assumes**  $\langle n2, r \rangle \in c \ p \in \mathbb{P} \ d \in \mathbb{P} \ r \in \mathbb{P}$

**shows**  $\langle \langle 0, b, n2, p \rangle, \langle 1, b, c, d \rangle \rangle \in \text{forcerel}(\mathbb{P}, \langle 1, b, c, d \rangle)$

*<proof>*

**lemmas** *forcerelI* = *forcerelI1*[*THEN vimage\_singleton\_iff*[*THEN iffD2*]]

*forcerelI2*[*THEN vimage\_singleton\_iff*[*THEN iffD2*]]

*forcerelI3*[*THEN vimage\_singleton\_iff*[*THEN iffD2*]]

**lemma** *aux\_def\_frc\_at*:

**assumes**  $z \in \text{forcerel}(\mathbb{P}, x) \ -'' \{x\}$

**shows**  $\text{wfrec}(\text{forcerel}(\mathbb{P}, x), z, H) = \text{wfrec}(\text{forcerel}(\mathbb{P}, z), z, H)$

*<proof>*

## 11.2 Recursive expression of *frc\_at*

**lemma** *def\_frc\_at* :

**assumes**  $p \in \mathbb{P}$

**shows**

$\text{frc\_at}(\mathbb{P}, \text{leq}, \langle \text{ft}, n1, n2, p \rangle) =$

$\text{bool\_of\_o}(p \in \mathbb{P} \wedge$

$(\text{ft} = 0 \wedge (\forall s. s \in \text{domain}(n1) \cup \text{domain}(n2) \longrightarrow$

$(\forall q. q \in \mathbb{P} \wedge q \preceq p \longrightarrow (\text{frc\_at}(\mathbb{P}, \text{leq}, \langle 1, s, n1, q \rangle) = 1 \longleftrightarrow \text{frc\_at}(\mathbb{P}, \text{leq}, \langle 1, s, n2, q \rangle)$

$= 1)))$

$\vee \text{ft} = 1 \wedge (\forall v \in \mathbb{P}. v \preceq p \longrightarrow$

$(\exists q. \exists s. \exists r. r \in \mathbb{P} \wedge q \in \mathbb{P} \wedge q \preceq v \wedge \langle s, r \rangle \in n2 \wedge q \preceq r \wedge \text{frc\_at}(\mathbb{P}, \text{leq}, \langle 0, n1, s, q \rangle)$

$= 1))))$

*<proof>*

## 11.3 Absoluteness of *frc\_at*

**lemma** *forcerel\_in\_M* :

**assumes**  $x \in M$

**shows**  $\text{forcerel}(\mathbb{P}, x) \in M$

*<proof>*

**lemma** *relation2\_Hfrc\_at\_abs*:

$\text{relation2}(\#\#M, \text{is\_Hfrc\_at}(\#\#M, \mathbb{P}, \text{leq}), \lambda x f. \text{bool\_of\_o}(\text{Hfrc}(\mathbb{P}, \text{leq}, x, f)))$

*<proof>*

**lemma** *Hfrc\_at\_closed* :

$\forall x \in M. \forall g \in M. \text{function}(g) \longrightarrow \text{bool\_of\_o}(\text{Hfrc}(\mathbb{P}, \text{leq}, x, g)) \in M$

*<proof>*

**lemma** *wfrec\_Hfrc\_at* :

**assumes**  $X \in M$   
**shows**  $wfrec\_replacement(\#\#M, is\_Hfrc\_at(\#\#M, \mathbb{P}, leq), forcereI(\mathbb{P}, X))$   
 $\langle proof \rangle$

**lemma**  $names\_below\_abs :$   
 $\llbracket Q \in M ; x \in M ; nb \in M \rrbracket \implies is\_names\_below(\#\#M, Q, x, nb) \longleftrightarrow nb = names\_below(Q, x)$   
 $\langle proof \rangle$

**lemma**  $names\_below\_closed :$   
 $\llbracket Q \in M ; x \in M \rrbracket \implies names\_below(Q, x) \in M$   
 $\langle proof \rangle$

**lemma**  $names\_below\_productE :$   
**assumes**  $Q \in M \ x \in M$   
 $\bigwedge A1 \ A2 \ A3 \ A4. A1 \in M \implies A2 \in M \implies A3 \in M \implies A4 \in M \implies R(A1$   
 $\times A2 \times A3 \times A4)$   
**shows**  $R(names\_below(Q, x))$   
 $\langle proof \rangle$

**lemma**  $forcereI\_abs :$   
 $\llbracket x \in M ; z \in M \rrbracket \implies is\_forcereI(\#\#M, \mathbb{P}, x, z) \longleftrightarrow z = forcereI(\mathbb{P}, x)$   
 $\langle proof \rangle$

**lemma**  $frc\_at\_abs :$   
**assumes**  $fnc \in M \ z \in M$   
**shows**  $is\_frc\_at(\#\#M, \mathbb{P}, leq, fnc, z) \longleftrightarrow z = frc\_at(\mathbb{P}, leq, fnc)$   
 $\langle proof \rangle$

**lemma**  $forces\_eq'\_abs :$   
 $\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \implies is\_forces\_eq'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces\_eq'(\mathbb{P}, leq, p, t1, t2)$   
 $\langle proof \rangle$

**lemma**  $forces\_mem'\_abs :$   
 $\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \implies is\_forces\_mem'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces\_mem'(\mathbb{P}, leq, p, t1, t2)$   
 $\langle proof \rangle$

**lemma**  $forces\_neq'\_abs :$   
**assumes**  $p \in M \ t1 \in M \ t2 \in M$   
**shows**  $is\_forces\_neq'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces\_neq'(\mathbb{P}, leq, p, t1, t2)$   
 $\langle proof \rangle$

**lemma**  $forces\_nmem'\_abs :$   
**assumes**  $p \in M \ t1 \in M \ t2 \in M$   
**shows**  $is\_forces\_nmem'(\#\#M, \mathbb{P}, leq, p, t1, t2) \longleftrightarrow forces\_nmem'(\mathbb{P}, leq, p, t1, t2)$   
 $\langle proof \rangle$

**lemma**  $leq\_abs :$   
 $\llbracket l \in M ; q \in M ; p \in M \rrbracket \implies is\_leq(\#\#M, l, q, p) \longleftrightarrow \langle q, p \rangle \in l$   
 $\langle proof \rangle$

## 11.4 Forcing for atomic formulas in context

### definition

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

### definition

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

### abbreviation $is\_forces\_eq$

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

### abbreviation

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

**lemma**  $def\_forces\_eq: p \in \mathbb{P} \Longrightarrow p forces_a (t1 = t2) \longleftrightarrow$   
 $(\forall s \in domain(t1) \cup domain(t2). \forall q. q \in \mathbb{P} \wedge q \preceq p \longrightarrow$   
 $(q forces_a (s \in t1) \longleftrightarrow q forces_a (s \in t2)))$   
 $\langle proof \rangle$

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

### lemma $forces\_eq\_abs :$

$\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \Longrightarrow is\_forces\_eq(p,t1,t2) \longleftrightarrow p forces_a (t1 = t2)$   
 $\langle proof \rangle$

### lemma $forces\_mem\_abs :$

$\llbracket p \in M ; t1 \in M ; t2 \in M \rrbracket \Longrightarrow is\_forces\_mem(p,t1,t2) \longleftrightarrow p forces_a (t1 \in t2)$   
 $\langle proof \rangle$

### definition

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

### definition

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

### lemma $forces\_neq :$

$p forces_a (t1 \neq t2) \longleftrightarrow forces\_neq'(\mathbb{P},leq,p,t1,t2)$   
 $\langle proof \rangle$

### lemma $forces\_nmem :$

$p \text{ forces}_a (t1 \notin t2) \longleftrightarrow \text{forces\_nmem}'(\mathbb{P}, \text{leq}, p, t1, t2)$   
 $\langle \text{proof} \rangle$

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

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

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

**lemma**  $\text{sats\_forces\_Nand} :$   
**assumes**  $\varphi \in \text{formula } \psi \in \text{formula } \text{env} \in \text{list}(M) \ p \in M$   
**shows**  $p \Vdash \cdot \neg(\varphi \wedge \psi) \cdot \text{env} \longleftrightarrow$   
 $p \in \mathbb{P} \wedge \neg(\exists q \in M. q \in \mathbb{P} \wedge \text{is\_leq}(\#\#M, \text{leq}, q, p) \wedge (q \Vdash \varphi \text{ env}) \wedge (q \Vdash \psi \text{ env}))$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{sats\_forces\_Neg} :$   
**assumes**  $\varphi \in \text{formula } \text{env} \in \text{list}(M) \ p \in M$   
**shows**  $p \Vdash \cdot \neg \varphi \cdot \text{env} \longleftrightarrow$   
 $(p \in \mathbb{P} \wedge \neg(\exists q \in M. q \in \mathbb{P} \wedge \text{is\_leq}(\#\#M, \text{leq}, q, p) \wedge (q \Vdash \varphi \text{ env})))$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{sats\_forces\_Forall} :$   
**assumes**  $\varphi \in \text{formula } \text{env} \in \text{list}(M) \ p \in M$   
**shows**  $p \Vdash \cdot (\forall \varphi) \cdot \text{env} \longleftrightarrow p \in \mathbb{P} \wedge (\forall x \in M. p \Vdash \varphi ([x] @ \text{env}))$   
 $\langle \text{proof} \rangle$

**end** — *forcing\_data1*

**end**

## 12 Names and generic extensions

**theory** *Names*  
**imports**  
 $\text{Forcing\_Data}$   
 $\text{FrecR\_Arities}$   
 $\text{ZF\_Trans\_Interpretations}$   
**begin**

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

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

The function *val* interprets a name in *M* according to a (generic) filter *G*. Note the definition in terms of the well-founded recursor.

**definition**

$$\begin{aligned} \text{val} &:: [i,i] \Rightarrow i \text{ where} \\ \text{val}(G,\tau) &\equiv \text{wfrec}(\text{edrel}(\text{eclose}(\{\tau\})), \tau, Hv(G)) \end{aligned}$$

**definition**

$$\begin{aligned} \text{GenExt} &:: [i,i] \Rightarrow i \quad (\langle \_ \_ \rangle [71,1]) \\ \text{where } M[G] &\equiv \{ \text{val}(G,\tau) . \tau \in M \} \end{aligned}$$

**lemma** *map\_val\_in\_MG*:

$$\begin{aligned} \text{assumes} & \\ \text{env} &\in \text{list}(M) \\ \text{shows} & \\ \text{map}(\text{val}(G), \text{env}) &\in \text{list}(M[G]) \\ \langle \text{proof} \rangle & \end{aligned}$$

## 12.1 Values and check-names

**context** *forcing\_data1*

**begin**

**lemma** *name\_components\_in\_M*:

$$\begin{aligned} \text{assumes} & \langle \sigma,p \rangle \in \vartheta \quad \vartheta \in M \\ \text{shows} & \sigma \in M \quad p \in M \\ \langle \text{proof} \rangle & \end{aligned}$$

**definition**

$$\begin{aligned} \text{Hcheck} &:: [i,i] \Rightarrow i \text{ where} \\ \text{Hcheck}(z,f) &\equiv \{ \langle f'y,1 \rangle . y \in z \} \end{aligned}$$

**definition**

$$\begin{aligned} \text{check} &:: i \Rightarrow i \text{ where} \\ \text{check}(x) &\equiv \text{transrec}(x, \text{Hcheck}) \end{aligned}$$

**lemma** *checkD*:

$$\begin{aligned} \text{check}(x) &= \text{wfrec}(\text{Memrel}(\text{eclose}(\{x\})), x, \text{Hcheck}) \\ \langle \text{proof} \rangle & \end{aligned}$$

**lemma** *Hcheck\_trancl*:  $\text{Hcheck}(y, \text{restrict}(f, \text{Memrel}(\text{eclose}(\{x\}))-'\{y\}))$

$$= \text{Hcheck}(y, \text{restrict}(f, (\text{Memrel}(\text{eclose}(\{x\}))^{\wedge+})-'\{y\}))$$

$\langle \text{proof} \rangle$

**lemma** *check\_trancl*:  $\text{check}(x) = \text{wfrec}(\text{rcheck}(x), x, \text{Hcheck})$

$\langle \text{proof} \rangle$

**lemma** *rcheck\_in\_M* :  $x \in M \implies \text{rcheck}(x) \in M$

*<proof>*

**lemma** *rcheck\_subset\_M* :  $x \in M \implies \text{field}(\text{rcheck}(x)) \subseteq \text{eclose}(\{x\})$   
*<proof>*

**lemma** *aux\_def\_check*:  $x \in y \implies$   
 $\text{wfrec}(\text{Memrel}(\text{eclose}(\{y\})), x, H\text{check}) =$   
 $\text{wfrec}(\text{Memrel}(\text{eclose}(\{x\})), x, H\text{check})$   
*<proof>*

**lemma** *def\_check* :  $\text{check}(y) = \{ \langle \text{check}(w), \mathbf{1} \rangle . w \in y \}$   
*<proof>*

**lemma** *def\_checkS* :  
  **fixes**  $n$   
  **assumes**  $n \in \text{nat}$   
  **shows**  $\text{check}(\text{succ}(n)) = \text{check}(n) \cup \{ \langle \text{check}(n), \mathbf{1} \rangle \}$   
*<proof>*

**lemma** *field\_Memrel2* :  
  **assumes**  $x \in M$   
  **shows**  $\text{field}(\text{Memrel}(\text{eclose}(\{x\}))) \subseteq M$   
*<proof>*

**lemma** *aux\_def\_val*:  
  **assumes**  $z \in \text{domain}(x)$   
  **shows**  $\text{wfrec}(\text{edrel}(\text{eclose}(\{x\})), z, Hv(G)) = \text{wfrec}(\text{edrel}(\text{eclose}(\{z\})), z, Hv(G))$   
*<proof>*

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

**lemma** *def\_val*:  $\text{val}(G, x) = \{ z . t \in \text{domain}(x) , (\exists p \in G . \langle t, p \rangle \in x) \wedge z = \text{val}(G, t) \}$   
*<proof>*

**lemma** *val\_mono* :  $x \subseteq y \implies \text{val}(G, x) \subseteq \text{val}(G, y)$   
*<proof>*

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

**lemma** *val\_check* :  $\mathbf{1} \in G \implies \mathbf{1} \in \mathbb{P} \implies \text{val}(G, \text{check}(y)) = y$   
*<proof>*

**lemma** *val\_of\_name* :  
 $\text{val}(G, \{ x \in A \times \mathbb{P} . Q(x) \}) = \{ z . t \in A , (\exists p \in \mathbb{P} . Q(\langle t, p \rangle)) \wedge p \in G \wedge z = \text{val}(G, t) \}$   
*<proof>*

**lemma** *val\_of\_name\_alt* :  
 $\text{val}(G, \{ x \in A \times \mathbb{P} . Q(x) \}) = \{ z . t \in A , (\exists p \in \mathbb{P} \cap G . Q(\langle t, p \rangle)) \wedge z = \text{val}(G, t) \}$   
*<proof>*

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

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

**lemma** *val\_subset\_domain\_times\_range*:  $val(F,\tau) \subseteq val(F, domain(\tau) \times range(\tau))$   
 $\langle proof \rangle$

**lemma** *val\_of\_elem*:  $\langle \vartheta,p \rangle \in \pi \implies p \in G \implies val(G,\vartheta) \in val(G,\pi)$   
 $\langle proof \rangle$

**lemma** *elem\_of\_val*:  $x \in val(G,\pi) \implies \exists \vartheta \in domain(\pi). val(G,\vartheta) = x$   
 $\langle proof \rangle$

**lemma** *elem\_of\_val\_pair*:  $x \in val(G,\pi) \implies \exists \vartheta. \exists p \in G. \langle \vartheta,p \rangle \in \pi \wedge val(G,\vartheta) = x$   
 $\langle proof \rangle$

**lemma** *elem\_of\_val\_pair'*:  
**assumes**  $\pi \in M \ x \in val(G,\pi)$   
**shows**  $\exists \vartheta \in M. \exists p \in G. \langle \vartheta,p \rangle \in \pi \wedge val(G,\vartheta) = x$   
 $\langle proof \rangle$

**lemma** *GenExtD*:  $x \in M[G] \implies \exists \tau \in M. x = val(G,\tau)$   
 $\langle proof \rangle$

**lemma** *GenExtI*:  $x \in M \implies val(G,x) \in M[G]$   
 $\langle proof \rangle$

**lemma** *Transset\_MG* :  $Transset(M[G])$   
 $\langle proof \rangle$

**lemmas** *transitivity\_MG* =  $Transset\_intf[OF\ Transset\_MG]$

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

**lemma** *check\_nat\_M* :  
**assumes**  $n \in nat$   
**shows**  $check(n) \in M$   
 $\langle proof \rangle$

**lemma** *def\_PHcheck*:  
**assumes**  
 $z \in M \ f \in M$   
**shows**  
 $Hcheck(z,f) = Replace(z, PHcheck(\#\#M, \mathbf{1}, f))$   
 $\langle proof \rangle$

**lemma** *wfrec\_Hcheck* :  
**assumes**  $X \in M$   
**shows**  $wfrec\_replacement(\#\#M, is\_Hcheck(\#\#M, \mathbf{1}), rcheck(X))$   
*<proof>*

**lemma** *Hcheck\_closed'* :  $f \in M \implies z \in M \implies \{f \cdot x \mid x \in z\} \in M$   
*<proof>*

**lemma** *repl\_PHcheck* :  
**assumes**  $f \in M$   
**shows**  $lam\_replacement(\#\#M, \lambda x. Hcheck(x, f))$   
*<proof>*

**lemma** *univ\_PHcheck* :  $\llbracket z \in M ; f \in M \rrbracket \implies univalent(\#\#M, z, PHcheck(\#\#M, \mathbf{1}, f))$   
*<proof>*

**lemma** *PHcheck\_closed* :  $\llbracket z \in M ; f \in M ; x \in z ; PHcheck(\#\#M, \mathbf{1}, f, x, y) \rrbracket \implies (\#\#M)(y)$   
*<proof>*

**lemma** *relation2\_Hcheck* :  $relation2(\#\#M, is\_Hcheck(\#\#M, \mathbf{1}), Hcheck)$   
*<proof>*

**lemma** *Hcheck\_closed* :  $\forall y \in M. \forall g \in M. Hcheck(y, g) \in M$   
*<proof>*

**lemma** *wf\_rcheck* :  $x \in M \implies wf(rcheck(x))$   
*<proof>*

**lemma** *trans\_rcheck* :  $x \in M \implies trans(rcheck(x))$   
*<proof>*

**lemma** *relation\_rcheck* :  $x \in M \implies relation(rcheck(x))$   
*<proof>*

**lemma** *check\_in\_M* :  $x \in M \implies check(x) \in M$   
*<proof>*

**lemma** *rcheck\_abs[Rel]* :  $\llbracket x \in M ; r \in M \rrbracket \implies is\_rcheck(\#\#M, x, r) \longleftrightarrow r = rcheck(x)$   
*<proof>*

**lemma** *check\_abs[Rel]* :  
**assumes**  $x \in M \ z \in M$   
**shows**  $is\_check(\#\#M, \mathbf{1}, x, z) \longleftrightarrow z = check(x)$   
*<proof>*

**lemma** *check\_lam\_replacement*: *lam\_replacement*(##*M*,*check*)  
⟨*proof*⟩

**lemma** *check\_replacement*: {*check*(*x*). *x* ∈  $\mathbb{P}$ } ∈ *M*  
⟨*proof*⟩

**lemma** *M\_subset\_MG* :  $\mathbf{1} \in G \implies M \subseteq M[G]$   
⟨*proof*⟩

The name for the generic filter

**definition**

*G\_dot* :: *i* **where**  
*G\_dot* ≡ {⟨*check*(*p*),*p*⟩ . *p* ∈  $\mathbb{P}$ }

**lemma** *G\_dot\_in\_M* : *G\_dot* ∈ *M*  
⟨*proof*⟩

**lemma** *zero\_in\_MG* :  $0 \in M[G]$   
⟨*proof*⟩

**declare** *check\_in\_M* [*simp*,*intro*]

**end** — *forcing\_data1*

**context** *G\_generic1*  
**begin**

**lemma** *val\_G\_dot* : *val*(*G*,*G\_dot*) = *G*  
⟨*proof*⟩

**lemma** *G\_in\_Gen\_Ext* : *G* ∈ *M*[*G*]  
⟨*proof*⟩

**lemmas** *generic\_simps* = *val\_check*[*OF one\_in\_G one\_in\_P*]  
*M\_subset\_MG*[*OF one\_in\_G, THEN subsetD*]  
*GenExtI P\_in\_M*

**lemmas** *generic\_dests* = *M\_genericD M\_generic\_compatD*

**bundle** *G\_generic1\_lemmas* = *generic\_simps*[*simp*] *generic\_dests*[*dest*]

**end** — *G\_generic1*

**sublocale** *G\_generic1* ⊆ *ext*: *M\_trans* ##*M*[*G*]  
⟨*proof*⟩

**end**

## 13 The Forcing Theorems

```
theory Forcing_Theorems
  imports
    Cohen_Posets_Relative
    Forces_Definition
    Names
```

```
begin
```

```
context forcing_data1
```

```
begin
```

### 13.1 The forcing relation in context

```
lemma separation_forces :
  assumes
    fty:  $\varphi \in \text{formula}$  and
    far:  $\text{arity}(\varphi) \leq \text{length}(\text{env})$  and
    envty:  $\text{env} \in \text{list}(M)$ 
  shows
    separation( $\#\#M, \lambda p. (p \Vdash \varphi \text{ env})$ )
  <proof>
```

```
lemma Collect_forces :
  assumes
     $\varphi \in \text{formula}$  and
     $\text{arity}(\varphi) \leq \text{length}(\text{env})$  and
     $\text{env} \in \text{list}(M)$ 
  shows
     $\{p \in \mathbb{P} . p \Vdash \varphi \text{ env}\} \in M$ 
  <proof>
```

```
lemma forces_mem_iff_dense_below:  $p \in \mathbb{P} \implies p \text{ forces}_a (t1 \in t2) \iff \text{dense\_below}(\{q \in \mathbb{P} . \exists s. \exists r. r \in \mathbb{P} \wedge \langle s, r \rangle \in t2 \wedge q \preceq r \wedge q \text{ forces}_a (t1 = s)\}, p)$ 
  <proof>
```

### 13.2 Kunen 2013, Lemma IV.2.37(a)

```
lemma strengthening_eq:
  assumes  $p \in \mathbb{P} \ r \in \mathbb{P} \ r \preceq p \ p \text{ forces}_a (t1 = t2)$ 
  shows  $r \text{ forces}_a (t1 = t2)$ 
  <proof>
```

### 13.3 Kunen 2013, Lemma IV.2.37(a)

```
lemma strengthening_mem:
  assumes  $p \in \mathbb{P} \ r \in \mathbb{P} \ r \preceq p \ p \text{ forces}_a (t1 \in t2)$ 
  shows  $r \text{ forces}_a (t1 \in t2)$ 
```

*<proof>*

### 13.4 Kunen 2013, Lemma IV.2.37(b)

**lemma** *density\_mem*:

**assumes**  $p \in \mathbb{P}$

**shows**  $p \text{ forces}_a (t1 \in t2) \longleftrightarrow \text{dense\_below}(\{q \in \mathbb{P}. q \text{ forces}_a (t1 \in t2)\}, p)$

*<proof>*

**lemma** *aux\_density\_eq*:

**assumes**

$\text{dense\_below}(\{q' \in \mathbb{P}. \forall q. q \in \mathbb{P} \wedge q \preceq q' \longrightarrow q \text{ forces}_a (s \in t1) \longleftrightarrow q \text{ forces}_a (s \in t2)\}, p)$

$q \text{ forces}_a (s \in t1) \quad q \in \mathbb{P} \quad p \in \mathbb{P} \quad q \preceq p$

**shows**

$\text{dense\_below}(\{r \in \mathbb{P}. r \text{ forces}_a (s \in t2)\}, q)$

*<proof>*

**lemma** *density\_eq*:

**assumes**  $p \in \mathbb{P}$

**shows**  $p \text{ forces}_a (t1 = t2) \longleftrightarrow \text{dense\_below}(\{q \in \mathbb{P}. q \text{ forces}_a (t1 = t2)\}, p)$

*<proof>*

### 13.5 Kunen 2013, Lemma IV.2.38

**lemma** *not\_forces\_neq*:

**assumes**  $p \in \mathbb{P}$

**shows**  $p \text{ forces}_a (t1 = t2) \longleftrightarrow \neg (\exists q \in \mathbb{P}. q \preceq p \wedge q \text{ forces}_a (t1 \neq t2))$

*<proof>*

**lemma** *not\_forces\_nmem*:

**assumes**  $p \in \mathbb{P}$

**shows**  $p \text{ forces}_a (t1 \in t2) \longleftrightarrow \neg (\exists q \in \mathbb{P}. q \preceq p \wedge q \text{ forces}_a (t1 \notin t2))$

*<proof>*

### 13.6 The relation of forcing and atomic formulas

**lemma** *Forces\_Equal*:

**assumes**

$p \in \mathbb{P} \quad t1 \in M \quad t2 \in M \quad env \in \text{list}(M) \quad nth(n, env) = t1 \quad nth(m, env) = t2 \quad n \in \text{nat} \quad m \in \text{nat}$

**shows**

$(p \Vdash \text{Equal}(n, m) \text{ env}) \longleftrightarrow p \text{ forces}_a (t1 = t2)$

*<proof>*

**lemma** *Forces\_Member*:

**assumes**

$p \in \mathbb{P} \quad t1 \in M \quad t2 \in M \quad env \in \text{list}(M) \quad nth(n, env) = t1 \quad nth(m, env) = t2 \quad n \in \text{nat} \quad m \in \text{nat}$

**shows**

$(p \Vdash \text{Member}(n,m) \text{ env}) \longleftrightarrow p \text{ forces}_a (t1 \in t2)$   
 $\langle \text{proof} \rangle$

**lemma** *Forces\_Neg*:

**assumes**

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

**shows**

$(p \Vdash \text{Neg}(\varphi) \text{ env}) \longleftrightarrow \neg(\exists q \in M. q \in \mathbb{P} \wedge q \preceq p \wedge (q \Vdash \varphi \text{ env}))$

$\langle \text{proof} \rangle$

### 13.7 The relation of forcing and connectives

**lemma** *Forces\_Nand*:

**assumes**

$p \in \mathbb{P} \text{ env} \in \text{list}(M) \varphi \in \text{formula} \psi \in \text{formula}$

**shows**

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

$\langle \text{proof} \rangle$

**lemma** *Forces\_And\_aux*:

**assumes**

$p \in \mathbb{P} \text{ env} \in \text{list}(M) \varphi \in \text{formula} \psi \in \text{formula}$

**shows**

$p \Vdash \text{And}(\varphi, \psi) \text{ env} \longleftrightarrow$

$(\forall q \in M. q \in \mathbb{P} \wedge q \preceq p \longrightarrow (\exists r \in M. r \in \mathbb{P} \wedge r \preceq q \wedge (r \Vdash \varphi \text{ env}) \wedge (r \Vdash \psi \text{ env})))$

$\langle \text{proof} \rangle$

**lemma** *Forces\_And\_iff\_dense\_below*:

**assumes**

$p \in \mathbb{P} \text{ env} \in \text{list}(M) \varphi \in \text{formula} \psi \in \text{formula}$

**shows**

$(p \Vdash \text{And}(\varphi, \psi) \text{ env}) \longleftrightarrow \text{dense\_below}(\{r \in \mathbb{P}. (r \Vdash \varphi \text{ env}) \wedge (r \Vdash \psi \text{ env})\}, p)$

$\langle \text{proof} \rangle$

**lemma** *Forces\_Forall*:

**assumes**

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

**shows**

$(p \Vdash \text{Forall}(\varphi) \text{ env}) \longleftrightarrow (\forall x \in M. (p \Vdash \varphi ([x] @ \text{env})))$

$\langle \text{proof} \rangle$

**bundle** *some\_rules* = *elem\_of\_val\_pair* [dest]

**context**

**includes** *some\_rules*

**begin**

**lemma** *elem\_of\_valI*:  $\exists \vartheta. \exists p \in \mathbb{P}. p \in G \wedge \langle \vartheta, p \rangle \in \pi \wedge \text{val}(G, \vartheta) = x \implies x \in \text{val}(G, \pi)$

*<proof>*

**lemma** *GenExt\_iff*:  $x \in M[G] \longleftrightarrow (\exists \tau \in M. x = \text{val}(G, \tau))$   
*<proof>*  
**end**

**end**

**context** *G\_generic1*

**begin**

### 13.8 Kunen 2013, Lemma IV.2.29

**lemma** *generic\_inter\_dense\_below*:  
**assumes**  $D \in M$  *dense\_below*( $D, p$ )  $p \in G$   
**shows**  $D \cap G \neq \emptyset$   
*<proof>*

### 13.9 Auxiliary results for Lemma IV.2.40(a)

**lemma** (*in forcing\_data1*) *IV240a\_mem\_Collect*:  
**assumes**  
 $\pi \in M$   $\tau \in M$   
**shows**  
 $\{q \in \mathbb{P}. \exists \sigma. \exists r. r \in \mathbb{P} \wedge \langle \sigma, r \rangle \in \tau \wedge q \leq r \wedge q \text{ forces}_a (\pi = \sigma)\} \in M$   
*<proof>*

**lemma** *IV240a\_mem*:  
**assumes**  
 $p \in G$   $\pi \in M$   $\tau \in M$   $p \text{ forces}_a (\pi \in \tau)$   
 $\bigwedge q \sigma. q \in \mathbb{P} \implies q \in G \implies \sigma \in \text{domain}(\tau) \implies q \text{ forces}_a (\pi = \sigma) \implies$   
 $\text{val}(G, \pi) = \text{val}(G, \sigma)$   
**shows**  
 $\text{val}(G, \pi) \in \text{val}(G, \tau)$   
*<proof>*

**lemma** *refl\_forces\_eq*:  $p \in \mathbb{P} \implies p \text{ forces}_a (x = x)$   
*<proof>*

**lemma** *forces\_memI*:  $\langle \sigma, r \rangle \in \tau \implies p \in \mathbb{P} \implies r \in \mathbb{P} \implies p \leq r \implies p \text{ forces}_a (\sigma \in \tau)$   
*<proof>*

**lemma** *IV240a\_eq\_1st\_incl*:  
**includes** *some\_rules*  
**assumes**  
 $p \in G$   $p \text{ forces}_a (\tau = \emptyset)$   
**and**

*IH*:  $\bigwedge q \sigma. q \in \mathbb{P} \implies q \in G \implies \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies$   
 $(q \text{ forces}_a (\sigma \in \tau) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \tau)) \wedge$   
 $(q \text{ forces}_a (\sigma \in \vartheta) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \vartheta))$

**shows**

$\text{val}(G, \tau) \subseteq \text{val}(G, \vartheta)$

*<proof>*

**lemma** *IV240a\_eq\_2nd\_incl*:

**includes** *some\_rules*

**assumes**

$p \in G \ p \text{ forces}_a (\tau = \vartheta)$

**and**

*IH*:  $\bigwedge q \sigma. q \in \mathbb{P} \implies q \in G \implies \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies$   
 $(q \text{ forces}_a (\sigma \in \tau) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \tau)) \wedge$   
 $(q \text{ forces}_a (\sigma \in \vartheta) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \vartheta))$

**shows**

$\text{val}(G, \vartheta) \subseteq \text{val}(G, \tau)$

*<proof>*

**lemma** *IV240a\_eq*:

**includes** *some\_rules*

**assumes**

$p \in G \ p \text{ forces}_a (\tau = \vartheta)$

**and**

*IH*:  $\bigwedge q \sigma. q \in \mathbb{P} \implies q \in G \implies \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies$   
 $(q \text{ forces}_a (\sigma \in \tau) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \tau)) \wedge$   
 $(q \text{ forces}_a (\sigma \in \vartheta) \longrightarrow \text{val}(G, \sigma) \in \text{val}(G, \vartheta))$

**shows**

$\text{val}(G, \tau) = \text{val}(G, \vartheta)$

*<proof>*

### 13.10 Induction on names

**lemma** (*in forcing\_data1*) *core\_induction*:

**assumes**

$\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in \text{domain}(\vartheta) \rrbracket \implies Q(0, \tau, \sigma, q) \rrbracket \implies Q(1, \tau, \vartheta, p)$   
 $\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \rrbracket \implies Q(1, \sigma, \tau, q)$   
 $\wedge Q(1, \sigma, \vartheta, q) \rrbracket \implies Q(0, \tau, \vartheta, p)$   
 $ft \in 2 \ p \in \mathbb{P}$

**shows**

$Q(ft, \tau, \vartheta, p)$

*<proof>*

**lemma** (*in forcing\_data1*) *forces\_induction\_with\_conds*:

**assumes**

$\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in \text{domain}(\vartheta) \rrbracket \implies Q(q, \tau, \sigma) \rrbracket \implies R(p, \tau, \vartheta)$   
 $\bigwedge \tau \vartheta p. p \in \mathbb{P} \implies \llbracket \bigwedge q \sigma. \llbracket q \in \mathbb{P} ; \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \rrbracket \implies R(q, \sigma, \tau) \wedge R(q, \sigma, \vartheta) \rrbracket \implies Q(p, \tau, \vartheta)$   
 $p \in \mathbb{P}$   
**shows**  
 $Q(p, \tau, \vartheta) \wedge R(p, \tau, \vartheta)$   
 $\langle \text{proof} \rangle$

**lemma** (in *forcing\_data1*) *forces\_induction*:

**assumes**  
 $\bigwedge \tau \vartheta. \llbracket \bigwedge \sigma. \sigma \in \text{domain}(\vartheta) \implies Q(\tau, \sigma) \rrbracket \implies R(\tau, \vartheta)$   
 $\bigwedge \tau \vartheta. \llbracket \bigwedge \sigma. \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies R(\sigma, \tau) \wedge R(\sigma, \vartheta) \rrbracket \implies Q(\tau, \vartheta)$   
**shows**  
 $Q(\tau, \vartheta) \wedge R(\tau, \vartheta)$   
 $\langle \text{proof} \rangle$

### 13.11 Lemma IV.2.40(a), in full

**lemma** *IV240a*:

**shows**  
 $(\tau \in M \longrightarrow \vartheta \in M \longrightarrow (\forall p \in G. p \text{ forces}_a (\tau = \vartheta) \longrightarrow \text{val}(G, \tau) = \text{val}(G, \vartheta))) \wedge$   
 $(\tau \in M \longrightarrow \vartheta \in M \longrightarrow (\forall p \in G. p \text{ forces}_a (\tau \in \vartheta) \longrightarrow \text{val}(G, \tau) \in \text{val}(G, \vartheta)))$   
**(is**  $?Q(\tau, \vartheta) \wedge ?R(\tau, \vartheta)$   
 $\langle \text{proof} \rangle$

### 13.12 Lemma IV.2.40(b)

**lemma** *IV240b\_mem*:

**includes** *some\_rules*  
**assumes**  
 $\text{val}(G, \pi) \in \text{val}(G, \tau) \quad \pi \in M \quad \tau \in M$   
**and**  
 $IH: \bigwedge \sigma. \sigma \in \text{domain}(\tau) \implies \text{val}(G, \pi) = \text{val}(G, \sigma) \implies$   
 $\exists p \in G. p \text{ forces}_a (\pi = \sigma)$   
**shows**  
 $\exists p \in G. p \text{ forces}_a (\pi \in \tau)$   
 $\langle \text{proof} \rangle$

**end** — *G\_generic1*

**context** *forcing\_data1*

**begin**

**lemma** *Collect\_forces\_eq\_in\_M*:

**assumes**  $\tau \in M \quad \vartheta \in M$   
**shows**  $\{p \in \mathbb{P}. p \text{ forces}_a (\tau = \vartheta)\} \in M$   
 $\langle \text{proof} \rangle$

**lemma** *IV240b\_eq\_Collects*:

**assumes**  $\tau \in M \quad \vartheta \in M$

**shows**  $\{p \in \mathbb{P}. \exists \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta). p \text{ forces}_a (\sigma \in \tau) \wedge p \text{ forces}_a (\sigma \notin \vartheta)\} \in M$  **and**  
 $\{p \in \mathbb{P}. \exists \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta). p \text{ forces}_a (\sigma \notin \tau) \wedge p \text{ forces}_a (\sigma \in \vartheta)\} \in M$   
 $\langle \text{proof} \rangle$

**end** — *forcing\_data1*

**context** *G\_generic1*  
**begin**

**lemma** *IV240b\_eq*:

**includes** *some\_rules*

**assumes**

$\text{val}(G, \tau) = \text{val}(G, \vartheta) \quad \tau \in M \quad \vartheta \in M$

**and**

$IH: \bigwedge \sigma. \sigma \in \text{domain}(\tau) \cup \text{domain}(\vartheta) \implies$

$(\text{val}(G, \sigma) \in \text{val}(G, \tau) \longrightarrow (\exists q \in G. q \text{ forces}_a (\sigma \in \tau))) \wedge$

$(\text{val}(G, \sigma) \in \text{val}(G, \vartheta) \longrightarrow (\exists q \in G. q \text{ forces}_a (\sigma \in \vartheta)))$

**shows**

$\exists p \in G. p \text{ forces}_a (\tau = \vartheta)$

$\langle \text{proof} \rangle$

**lemma** *IV240b*:

$(\tau \in M \longrightarrow \vartheta \in M \longrightarrow \text{val}(G, \tau) = \text{val}(G, \vartheta) \longrightarrow (\exists p \in G. p \text{ forces}_a (\tau = \vartheta))) \wedge$

$(\tau \in M \longrightarrow \vartheta \in M \longrightarrow \text{val}(G, \tau) \in \text{val}(G, \vartheta) \longrightarrow (\exists p \in G. p \text{ forces}_a (\tau \in \vartheta)))$

**(is**  $?Q(\tau, \vartheta) \wedge ?R(\tau, \vartheta)$

$\langle \text{proof} \rangle$

**lemma** *truth\_lemma\_mem*:

**assumes**

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

$n \in \text{nat} \quad m \in \text{nat} \quad n < \text{length}(\text{env}) \quad m < \text{length}(\text{env})$

**shows**

$(\exists p \in G. p \Vdash \text{Member}(n, m) \text{ env}) \longleftrightarrow M[G], \text{map}(\text{val}(G), \text{env}) \models \text{Member}(n, m)$

$\langle \text{proof} \rangle$

**lemma** *truth\_lemma\_eq*:

**assumes**

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

$n \in \text{nat} \quad m \in \text{nat} \quad n < \text{length}(\text{env}) \quad m < \text{length}(\text{env})$

**shows**

$(\exists p \in G. p \Vdash \text{Equal}(n, m) \text{ env}) \longleftrightarrow M[G], \text{map}(\text{val}(G), \text{env}) \models \text{Equal}(n, m)$

$\langle \text{proof} \rangle$

**end** — *G\_generic1*

**lemma** *arities\_at\_aux*:

**assumes**

$n \in \text{nat } m \in \text{nat } \text{env} \in \text{list}(M) \text{ succ}(n) \cup \text{succ}(m) \leq \text{length}(\text{env})$

**shows**

$n < \text{length}(\text{env}) \ m < \text{length}(\text{env})$

$\langle \text{proof} \rangle$

### 13.13 The Strengthening Lemma

**context** *forcing\_data1*

**begin**

**lemma** *strengthening\_lemma*:

**assumes**

$p \in \mathbb{P} \ \varphi \in \text{formula} \ r \in \mathbb{P} \ r \preceq p$

$\text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{length}(\text{env})$

**shows**

$p \Vdash \varphi \ \text{env} \implies r \Vdash \varphi \ \text{env}$

$\langle \text{proof} \rangle$

### 13.14 The Density Lemma

**lemma** *arity\_Nand\_le*:

**assumes**  $\varphi \in \text{formula} \ \psi \in \text{formula} \ \text{arity}(\text{Nand}(\varphi, \psi)) \leq \text{length}(\text{env}) \ \text{env} \in \text{list}(A)$

**shows**  $\text{arity}(\varphi) \leq \text{length}(\text{env}) \ \text{arity}(\psi) \leq \text{length}(\text{env})$

$\langle \text{proof} \rangle$

**lemma** *dense\_below\_imp\_forces*:

**assumes**

$p \in \mathbb{P} \ \varphi \in \text{formula}$

$\text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{length}(\text{env})$

**shows**

$\text{dense\_below}(\{q \in \mathbb{P}. (q \Vdash \varphi \ \text{env})\}, p) \implies (p \Vdash \varphi \ \text{env})$

$\langle \text{proof} \rangle$

**lemma** *density\_lemma*:

**assumes**

$p \in \mathbb{P} \ \varphi \in \text{formula} \ \text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{length}(\text{env})$

**shows**

$p \Vdash \varphi \ \text{env} \iff \text{dense\_below}(\{q \in \mathbb{P}. (q \Vdash \varphi \ \text{env})\}, p)$

$\langle \text{proof} \rangle$

### 13.15 The Truth Lemma

**lemma** *Forces\_And*:

**assumes**

$p \in \mathbb{P} \ \text{env} \in \text{list}(M) \ \varphi \in \text{formula} \ \psi \in \text{formula}$

$\text{arity}(\varphi) \leq \text{length}(\text{env}) \ \text{arity}(\psi) \leq \text{length}(\text{env})$

**shows**

$p \Vdash \text{And}(\varphi, \psi) \ \text{env} \iff (p \Vdash \varphi \ \text{env}) \wedge (p \Vdash \psi \ \text{env})$

$\langle proof \rangle$

**lemma** *Forces\_Nand\_alt*:

**assumes**

$p \in \mathbb{P}$   $env \in list(M)$   $\varphi \in formula$   $\psi \in formula$   
 $arity(\varphi) \leq length(env)$   $arity(\psi) \leq length(env)$

**shows**

$(p \Vdash Nand(\varphi, \psi) env) \longleftrightarrow (p \Vdash Neg(And(\varphi, \psi)) env)$

$\langle proof \rangle$

**end**

**context** *G\_generic1*

**begin**

**lemma** *truth\_lemma\_Neg*:

**assumes**

$\varphi \in formula$   $env \in list(M)$   $arity(\varphi) \leq length(env)$  **and**  
 $IH: (\exists p \in G. p \Vdash \varphi env) \longleftrightarrow M[G], map(val(G), env) \models \varphi$

**shows**

$(\exists p \in G. p \Vdash Neg(\varphi) env) \longleftrightarrow M[G], map(val(G), env) \models Neg(\varphi)$

$\langle proof \rangle$

**lemma** *truth\_lemma\_And*:

**assumes**

$env \in list(M)$   $\varphi \in formula$   $\psi \in formula$   
 $arity(\varphi) \leq length(env)$   $arity(\psi) \leq length(env)$

**and**

$IH: (\exists p \in G. p \Vdash \varphi env) \longleftrightarrow M[G], map(val(G), env) \models \varphi$   
 $(\exists p \in G. p \Vdash \psi env) \longleftrightarrow M[G], map(val(G), env) \models \psi$

**shows**

$(\exists p \in G. (p \Vdash And(\varphi, \psi) env)) \longleftrightarrow M[G], map(val(G), env) \models And(\varphi, \psi)$

$\langle proof \rangle$

**end**

**definition**

*ren\_truth\_lemma* ::  $i \Rightarrow i$  **where**

*ren\_truth\_lemma*( $\varphi$ )  $\equiv$

$Exists(Exists(Exists(Exists(Exists($   
 $And(Equal(0, 5), And(Equal(1, 8), And(Equal(2, 9), And(Equal(3, 10), And(Equal(4, 6),$   
 $iterates(\lambda p. incr\_bv(p) '5 , 6, \varphi))))))))))$

**lemma** *ren\_truth\_lemma\_type*[*TC*] :

$\varphi \in formula \implies ren\_truth\_lemma(\varphi) \in formula$

$\langle proof \rangle$

**lemma** *arity\_ren\_truth* :

**assumes**  $\varphi \in formula$

**shows**  $\text{arity}(\text{ren\_truth\_lemma}(\varphi)) \leq 6 \cup \text{succ}(\text{arity}(\varphi))$   
 $\langle \text{proof} \rangle$

**lemma** *sats\_ren\_truth\_lemma*:  
 $[q,b,d,a1,a2,a3] @ \text{env} \in \text{list}(M) \implies \varphi \in \text{formula} \implies$   
 $(M, [q,b,d,a1,a2,a3] @ \text{env} \models \text{ren\_truth\_lemma}(\varphi)) \longleftrightarrow$   
 $(M, [q,a1,a2,a3,b] @ \text{env} \models \varphi)$   
 $\langle \text{proof} \rangle$

**context** *forcing\_data1*  
**begin**

**lemma** *truth\_lemma'* :  
**assumes**  
 $\varphi \in \text{formula} \ \text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{succ}(\text{length}(\text{env}))$   
**shows**  
 $\text{separation}(\#\#M, \lambda d. \exists b \in M. \forall q \in \mathbb{P}. q \preceq d \longrightarrow \neg(q \Vdash \varphi ([b] @ \text{env})))$   
 $\langle \text{proof} \rangle$

**end**

**context** *G\_generic1*  
**begin**

**lemma** *truth\_lemma*:  
**assumes**  
 $\varphi \in \text{formula}$   
 $\text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{length}(\text{env})$   
**shows**  
 $(\exists p \in G. p \Vdash \varphi \ \text{env}) \longleftrightarrow M[G], \text{map}(\text{val}(G), \text{env}) \models \varphi$   
 $\langle \text{proof} \rangle$

**end**

**context** *forcing\_data1*  
**begin**

### 13.16 The “Definition of forcing”

**lemma** *definition\_of\_forcing*:  
**assumes**  
 $p \in \mathbb{P} \ \varphi \in \text{formula} \ \text{env} \in \text{list}(M) \ \text{arity}(\varphi) \leq \text{length}(\text{env})$   
**shows**  
 $(p \Vdash \varphi \ \text{env}) \longleftrightarrow$   
 $(\forall G. M\_generic(G) \wedge p \in G \longrightarrow M[G], \text{map}(\text{val}(G), \text{env}) \models \varphi)$   
 $\langle \text{proof} \rangle$

**lemmas** *definability = forces\_type*

**end** — *forcing\_data1*

**end**

## 14 Ordinals in generic extensions

**theory** *Ordinals\_In\_MG*

**imports**

*Forcing\_Theorems*

**begin**

**context** *G\_generic1*

**begin**

**lemma** *rank\_val*:  $\text{rank}(\text{val}(G,x)) \leq \text{rank}(x)$  (**is**  $?Q(x)$ )

*<proof>*

**lemma** *Ord\_MG\_iff*:

**assumes**  $\text{Ord}(\alpha)$

**shows**  $\alpha \in M \longleftrightarrow \alpha \in M[G]$

*<proof>*

**end** — *G\_generic1*

**end**

## 15 Auxiliary renamings for Separation

**theory** *Separation\_Rename*

**imports**

*Interface*

**begin**

**no\_notation** *Aleph* ( $\aleph_{[90]}$  *90*)

**lemmas** *apply\_fun* = *apply\_iff*[*THEN iffD1*]

**lemma** *nth\_concat* :  $[p,t] \in \text{list}(A) \implies \text{env} \in \text{list}(A) \implies \text{nth}(1 +_{\omega} \text{length}(\text{env}), [p]@$

$\text{env} @ [t]) = t$

*<proof>*

**lemma** *nth\_concat2* :  $\text{env} \in \text{list}(A) \implies \text{nth}(\text{length}(\text{env}), \text{env} @ [p,t]) = p$

*<proof>*

**lemma** *nth\_concat3* :  $\text{env} \in \text{list}(A) \implies u = \text{nth}(\text{succ}(\text{length}(\text{env})), \text{env} @ [pi, u])$

*<proof>*

**definition**

$sep\_var :: i \Rightarrow i$  **where**  
 $sep\_var(n) \equiv \{\langle 0,1 \rangle, \langle 1,3 \rangle, \langle 2,4 \rangle, \langle 3,5 \rangle, \langle 4,0 \rangle, \langle 5+\omega n,6 \rangle, \langle 6+\omega n,2 \rangle\}$

**definition**

$sep\_env :: i \Rightarrow i$  **where**  
 $sep\_env(n) \equiv \lambda i \in (5+\omega n)-5 . i+\omega 2$

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

$weak(n,m) \equiv \{i+\omega m . i \in n\}$

**lemma**  $weakD$  :

**assumes**  $n \in nat$   $k \in nat$   $x \in weak(n,k)$   
**shows**  $\exists i \in n . x = i+\omega k$   
 $\langle proof \rangle$

**lemma**  $weak\_equal$  :

**assumes**  $n \in nat$   $m \in nat$   
**shows**  $weak(n,m) = (m+\omega n) - m$   
 $\langle proof \rangle$

**lemma**  $weak\_zero$ :

**shows**  $weak(0,n) = 0$   
 $\langle proof \rangle$

**lemma**  $weakening\_diff$  :

**assumes**  $n \in nat$   
**shows**  $weak(n,7) - weak(n,5) \subseteq \{5+\omega n, 6+\omega n\}$   
 $\langle proof \rangle$

**lemma**  $in\_add\_del$  :

**assumes**  $x \in j+\omega n$   $n \in nat$   $j \in nat$   
**shows**  $x < j \vee x \in weak(n,j)$   
 $\langle proof \rangle$

**lemma**  $sep\_env\_action$ :

**assumes**  
 $[t,p,u,P,leg,o,pi] \in list(M)$   
 $env \in list(M)$   
**shows**  $\forall i . i \in weak(length(env),5) \longrightarrow$   
 $nth(sep\_env(length(env))) 'i,[t,p,u,P,leg,o,pi]@env = nth(i,[p,P,leg,o,t] @ env$   
 $@ [pi,u])$   
 $\langle proof \rangle$

**lemma**  $sep\_env\_type$  :

**assumes**  $n \in nat$   
**shows**  $sep\_env(n) : (5+\omega n)-5 \rightarrow (7+\omega n)-7$   
 $\langle proof \rangle$

**lemma** *sep\_var\_fin\_type* :

**assumes**  $n \in \text{nat}$

**shows**  $\text{sep\_var}(n) : \gamma_{+\omega} n - || > \gamma_{+\omega} n$

$\langle \text{proof} \rangle$

**lemma** *sep\_var\_domain* :

**assumes**  $n \in \text{nat}$

**shows**  $\text{domain}(\text{sep\_var}(n)) = \gamma_{+\omega} n - \text{weak}(n, 5)$

$\langle \text{proof} \rangle$

**lemma** *sep\_var\_type* :

**assumes**  $n \in \text{nat}$

**shows**  $\text{sep\_var}(n) : (\gamma_{+\omega} n) - \text{weak}(n, 5) \rightarrow \gamma_{+\omega} n$

$\langle \text{proof} \rangle$

**lemma** *sep\_var\_action* :

**assumes**

$[t, p, u, P, \text{leq}, o, pi] \in \text{list}(M)$

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

**shows**  $\forall i . i \in (\gamma_{+\omega} \text{length}(\text{env})) - \text{weak}(\text{length}(\text{env}), 5) \longrightarrow$

$\text{nth}(\text{sep\_var}(\text{length}(\text{env})) 'i, [t, p, u, P, \text{leq}, o, pi] @ \text{env}) = \text{nth}(i, [p, P, \text{leq}, o, t] @ \text{env}$

$@ [pi, u]$

$\langle \text{proof} \rangle$

**definition**

$\text{rensep} :: i \Rightarrow i$  **where**

$\text{rensep}(n) \equiv \text{union\_fun}(\text{sep\_var}(n), \text{sep\_env}(n), \gamma_{+\omega} n - \text{weak}(n, 5), \text{weak}(n, 5))$

**lemma** *rensep\_aux* :

**assumes**  $n \in \text{nat}$

**shows**  $(\gamma_{+\omega} n - \text{weak}(n, 5)) \cup \text{weak}(n, 5) = \gamma_{+\omega} n \cap \gamma_{+\omega} n \cup (\gamma_{+\omega} n - \gamma) = \gamma_{+\omega} n$

$\langle \text{proof} \rangle$

**lemma** *rensep\_type* :

**assumes**  $n \in \text{nat}$

**shows**  $\text{rensep}(n) \in \gamma_{+\omega} n \rightarrow \gamma_{+\omega} n$

$\langle \text{proof} \rangle$

**lemma** *rensep\_action* :

**assumes**  $[t, p, u, P, \text{leq}, o, pi] @ \text{env} \in \text{list}(M)$

**shows**  $\forall i . i < \gamma_{+\omega} \text{length}(\text{env}) \longrightarrow \text{nth}(\text{rensep}(\text{length}(\text{env})) 'i, [t, p, u, P, \text{leq}, o, pi] @ \text{env})$

$= \text{nth}(i, [p, P, \text{leq}, o, t] @ \text{env} @ [pi, u])$

$\langle \text{proof} \rangle$

**definition** *sep\_ren* ::  $[i, i] \Rightarrow i$  **where**

$\text{sep\_ren}(n, \varphi) \equiv \text{ren}(\varphi) '( \gamma_{+\omega} n ) '( \gamma_{+\omega} n ) \text{rensep}(n)$

**lemma** *arity\_rensep*: **assumes**  $\varphi \in \text{formula}$   $\text{env} \in \text{list}(M)$

$\text{arity}(\varphi) \leq \gamma_{+\omega} \text{length}(\text{env})$

**shows**  $\text{arity}(\text{sep\_ren}(\text{length}(\text{env}),\varphi)) \leq 7 +_{\omega} \text{length}(\text{env})$   
 ⟨proof⟩

**lemma** *type\_rensep* [TC]:  
**assumes**  $\varphi \in \text{formula}$   $\text{env} \in \text{list}(M)$   
**shows**  $\text{sep\_ren}(\text{length}(\text{env}),\varphi) \in \text{formula}$   
 ⟨proof⟩

**lemma** *sepren\_action*:  
**assumes**  $\text{arity}(\varphi) \leq 7 +_{\omega} \text{length}(\text{env})$   
 $[t,p,u,P,\text{leq},o,\text{pi}] \in \text{list}(M)$   
 $\text{env} \in \text{list}(M)$   
 $\varphi \in \text{formula}$   
**shows**  $\text{sats}(M, \text{sep\_ren}(\text{length}(\text{env}),\varphi), [t,p,u,P,\text{leq},o,\text{pi}] @ \text{env}) \longleftrightarrow \text{sats}(M,$   
 $\varphi, [p,P,\text{leq},o,t] @ \text{env} @ [pi,u])$   
 ⟨proof⟩

**end**

## 16 The Axiom of Separation in $M[G]$

**theory** *Separation\_Axiom*  
**imports** *Forcing\_Theorems Separation\_Rename*  
**begin**

**context** *G\_generic1*  
**begin**

**lemma** *map\_val* :  
**assumes**  $\text{env} \in \text{list}(M[G])$   
**shows**  $\exists \text{nenv} \in \text{list}(M). \text{env} = \text{map}(\text{val}(G), \text{nenv})$   
 ⟨proof⟩

**lemma** *Collect\_sats\_in\_MG* :  
**assumes**  
 $A \in M[G]$   
 $\varphi \in \text{formula}$   $\text{env} \in \text{list}(M[G])$   $\text{arity}(\varphi) \leq 1 +_{\omega} \text{length}(\text{env})$   
**shows**  
 $\{x \in A . (M[G], [x] @ \text{env} \models \varphi)\} \in M[G]$   
 ⟨proof⟩

**theorem** *separation\_in\_MG*:  
**assumes**  
 $\varphi \in \text{formula}$  **and**  $\text{arity}(\varphi) \leq 1 +_{\omega} \text{length}(\text{env})$  **and**  $\text{env} \in \text{list}(M[G])$   
**shows**  
 $\text{separation}(\#\#M[G], \lambda x. (M[G], [x] @ \text{env} \models \varphi))$   
 ⟨proof⟩

**end** — *G\_generic1*

end

## 17 The Axiom of Pairing in $M[G]$

**theory** *Pairing\_Axiom*

**imports**

*Names*

**begin**

**context** *G\_generic1*

**begin**

**lemma** *val\_Upair* :

$\mathbf{1} \in G \implies \text{val}(G, \{\langle \tau, \mathbf{1} \rangle, \langle \rho, \mathbf{1} \rangle\}) = \{\text{val}(G, \tau), \text{val}(G, \rho)\}$

*<proof>*

**lemma** *pairing\_in\_MG* : *upair\_ax*( $\#\#M[G]$ )

*<proof>*

**end** — *G\_generic1*

**end**

## 18 The Axiom of Unions in $M[G]$

**theory** *Union\_Axiom*

**imports** *Names*

**begin**

**definition** *Union\_name\_body* ::  $[i, i, i, i] \Rightarrow o$  **where**

$\text{Union\_name\_body}(P, \text{leq}, \tau, x) \equiv \exists \sigma \in \text{domain}(\tau) . \exists q \in P . \exists r \in P .$

$\langle \sigma, q \rangle \in \tau \wedge \langle \text{fst}(x), r \rangle \in \sigma \wedge \langle \text{snd}(x), r \rangle \in \text{leq} \wedge \langle \text{snd}(x), q \rangle \in \text{leq}$

**definition** *Union\_name* ::  $[i, i, i] \Rightarrow i$  **where**

$\text{Union\_name}(P, \text{leq}, \tau) \equiv \{u \in \text{domain}(\bigcup (\text{domain}(\tau))) \times P . \text{Union\_name\_body}(P, \text{leq}, \tau, u)\}$

**context** *forcing\_data1*

**begin**

**lemma** *Union\_name\_closed* :

**assumes**  $\tau \in M$

**shows**  $\text{Union\_name}(\mathbb{P}, \text{leq}, \tau) \in M$

*<proof>*

**lemma** *Union\_MG\_Eq* :

**assumes**  $a \in M[G]$  **and**  $a = \text{val}(G, \tau)$  **and** *filter*( $G$ ) **and**  $\tau \in M$

**shows**  $\bigcup a = \text{val}(G, \text{Union\_name}(\mathbb{P}, \text{leq}, \tau))$

*<proof>*

**lemma** *union\_in\_MG* :  
  **assumes** *filter(G)*  
  **shows** *Union\_ax(##M[G])*  
  *<proof>*

**theorem** *Union\_MG* : *M\_generic(G)  $\implies$  Union\_ax(##M[G])*  
  *<proof>*

**end** — *forcing\_data1*

**end**

## 19 The Powerset Axiom in $M[G]$

**theory** *Powerset\_Axiom*  
  **imports**  
    *Separation\_Axiom Pairing\_Axiom Union\_Axiom*  
**begin**

*<ML>*

**context** *G\_generic1*  
**begin**

**lemma** *sats\_fst\_snd\_in\_M*:  
  **assumes**  
    *A ∈ M B ∈ M  $\varphi$  ∈ formula p ∈ M l ∈ M o ∈ M  $\chi ∈ M$  arity( $\varphi$ ) ≤ 6*  
  **shows**  $\{ \langle s, q \rangle \in A \times B . M, [q, p, l, o, s, \chi] \models \varphi \} \in M$  (**is**  $\vartheta \in M$ )  
  *<proof>*

**declare** *nat\_into\_M* [*rule del, simplified setclass\_iff, intro*]  
**lemmas** *ssimps = domain\_closed cartprod\_closed cons\_closed Pow\_rel\_closed*  
**declare** *ssimps* [*simp del, simplified setclass\_iff, simp, intro*]

— We keep  $Pow(a) \cap M[G]$  to be consistent with Kunen.

**lemma** *Pow\_inter\_MG*:  
  **assumes** *a ∈ M[G]*  
  **shows**  $Pow(a) \cap M[G] \in M[G]$   
  *<proof>*

**end** — *G\_generic1*

**sublocale** *G\_generic1*  $\subseteq$  *ext: M\_trivial ##M[G]*  
  *<proof>*

**context** *G\_generic1* **begin**

```

theorem power_in_MG : power_ax(##( $M[G]$ ))
  <proof>

end — G_generic1

end

```

## 20 The Axiom of Extensionality in $M[G]$

```

theory Extensionality_Axiom
  imports
    Names
  begin

  context forcing_data1
  begin

  lemma extensionality_in_MG : extensionality(##( $M[G]$ ))
    <proof>

  end — forcing_data1

end

```

## 21 The Axiom of Foundation in $M[G]$

```

theory Foundation_Axiom
  imports
    Names
  begin

  context forcing_data1
  begin

  lemma foundation_in_MG : foundation_ax(##( $M[G]$ ))
    <proof>

  lemma foundation_ax(##( $M[G]$ ))
    <proof>

  end — forcing_data1

end

```

## 22 The Axiom of Replacement in $M[G]$

**theory** *Replacement\_Axiom*

**imports**

*Separation\_Axiom*

**begin**

**context** *forcing\_data1*

**begin**

**bundle** *sharp\_simps1* = *snd\_abs[simp]* *fst\_abs[simp]* *fst\_closed[simp del, simplified, simp]*

*snd\_closed[simp del, simplified, simp]* *M\_inhabited[simplified, simp]*

*pair\_in\_M\_iff[simp del, simplified, simp]*

**lemma** *sats\_body\_ground\_repl\_fm*:

**includes** *sharp\_simps1*

**assumes**

$\exists t p. x = \langle t, p \rangle$   $[x, \alpha, m, \mathbb{P}, \text{leq}, \mathbf{1}] @ nenv \in \text{list}(M)$

$\varphi \in \text{formula}$

**shows**

$(\exists \tau \in M. \exists V \in M. \text{is\_Vset}(\lambda a. (\#\#M)(a), \alpha, V) \wedge \tau \in V \wedge (\text{snd}(x) \Vdash \varphi$   
 $([\text{fst}(x), \tau] @ nenv)))$

$\longleftrightarrow M, [\alpha, x, m, \mathbb{P}, \text{leq}, \mathbf{1}] @ nenv \models \text{body\_ground\_repl\_fm}(\varphi)$

$\langle \text{proof} \rangle$

**end** — *forcing\_data1*

**context** *G\_generic1*

**begin**

**lemma** *Replace\_sats\_in\_MG*:

**assumes**

$c \in M[G]$   $env \in \text{list}(M[G])$

$\varphi \in \text{formula}$   $\text{arity}(\varphi) \leq 2 + \omega \text{ length}(env)$

$\text{univalent}(\#\#M[G], c, \lambda x v. (M[G], [x, v] @ env \models \varphi))$

**and**

*ground\_replacement*:

$\bigwedge nenv. \text{ground\_replacement\_assm}(M, [\mathbb{P}, \text{leq}, \mathbf{1}] @ nenv, \varphi)$

**shows**

$\{v. x \in c, v \in M[G] \wedge (M[G], [x, v] @ env \models \varphi)\} \in M[G]$

$\langle \text{proof} \rangle$

**theorem** *strong\_replacement\_in\_MG*:

**assumes**

$\varphi \in \text{formula}$  **and**  $\text{arity}(\varphi) \leq 2 + \omega \text{ length}(env)$   $env \in \text{list}(M[G])$

**and**

*ground\_replacement*:

$\bigwedge nenv. \text{ground\_replacement\_assm}(M, [\mathbb{P}, \text{leq}, \mathbf{1}] @ nenv, \varphi)$

**shows**  
*strong\_replacement*(## $M[G]$ ,  $\lambda x v. M[G], [x, v]$  @  $env \models \varphi$ )  
 <proof>

**lemma** *replacement\_assm\_MG*:

**assumes**

*ground\_replacement*:

$\bigwedge nenv. \text{ground\_replacement\_assm}(M, [\mathbb{P}, \text{leq}, \mathbf{1}] \text{ @ } nenv, \varphi)$

**shows**

*replacement\_assm*( $M[G]$ ,  $env, \varphi$ )

<proof>

**end** — *G\_generic1*

**end**

## 23 The Axiom of Infinity in $M[G]$

**theory** *Infinity\_Axiom*

**imports** *Union\_Axiom Pairing\_Axiom*

**begin**

**context** *G\_generic1* **begin**

**interpretation** *mg\_triv*:  $M\_trivial$ ## $M[G]$

<proof>

**lemma** *infinity\_in\_MG* : *infinity\_ax*(## $M[G]$ )

<proof>

**end** — *G\_generic1*

**end**

## 24 The Axiom of Choice in $M[G]$

**theory** *Choice\_Axiom*

**imports**

*Powerset\_Axiom*

*Extensionality\_Axiom*

*Foundation\_Axiom*

*Replacement\_Axiom*

*Infinity\_Axiom*

**begin**

**definition**

*upair\_name* ::  $i \Rightarrow i \Rightarrow i \Rightarrow i$  **where**

*upair\_name*( $\tau, \rho, on$ )  $\equiv$  *Upair*( $\langle \tau, on \rangle, \langle \rho, on \rangle$ )

**definition**

$opair\_name :: i \Rightarrow i \Rightarrow i \Rightarrow i$  **where**  
 $opair\_name(\tau, \varrho, on) \equiv upair\_name(upair\_name(\tau, \tau, on), upair\_name(\tau, \varrho, on), on)$

**definition**

$induced\_surj :: i \Rightarrow i \Rightarrow i \Rightarrow i$  **where**  
 $induced\_surj(f, a, e) \equiv f^{-1}((range(f)-a) \times \{e\} \cup restrict(f, f^{-1}a))$

**lemma**  $domain\_induced\_surj$ :  $domain(induced\_surj(f, a, e)) = domain(f)$   
 $\langle proof \rangle$

**lemma**  $range\_restrict\_vimage$ :  
**assumes**  $function(f)$   
**shows**  $range(restrict(f, f^{-1}a)) \subseteq a$   
 $\langle proof \rangle$

**lemma**  $induced\_surj\_type$ :  
**assumes**  $function(f)$   
**shows**  
 $induced\_surj(f, a, e): domain(f) \rightarrow \{e\} \cup a$   
**and**  
 $x \in f^{-1}a \implies induced\_surj(f, a, e) x = f x$   
 $\langle proof \rangle$

**lemma**  $induced\_surj\_is\_surj$  :  
**assumes**  
 $e \in a \ function(f) \ domain(f) = \alpha \ \wedge \ y. y \in a \implies \exists x \in \alpha. f x = y$   
**shows**  $induced\_surj(f, a, e) \in surj(\alpha, a)$   
 $\langle proof \rangle$

**lemma** (**in**  $M\_ZF1\_trans$ )  $upair\_name\_closed$  :  
 $\llbracket x \in M; y \in M; o \in M \rrbracket \implies upair\_name(x, y, o) \in M$   
 $\langle proof \rangle$

**context**  $G\_generic1$   
**begin**

**lemma**  $val\_upair\_name$  :  $val(G, upair\_name(\tau, \varrho, 1)) = \{val(G, \tau), val(G, \varrho)\}$   
 $\langle proof \rangle$

**lemma**  $val\_opair\_name$  :  $val(G, opair\_name(\tau, \varrho, 1)) = \langle val(G, \tau), val(G, \varrho) \rangle$   
 $\langle proof \rangle$

**lemma**  $val\_RepFun\_one$ :  $val(G, \{\langle f(x), 1 \rangle . x \in a\}) = \{val(G, f(x)) . x \in a\}$   
 $\langle proof \rangle$

**end**—  $G\_generic1$

## 24.1 $M[G]$ is a transitive model of ZF

**sublocale**  $G\_generic1 \subseteq ext:M\_Z\_trans M[G]$   
 $\langle proof \rangle$

**lemma** (in  $M\_replacement$ )  $upair\_name\_lam\_replacement$  :  
 $M(z) \implies lam\_replacement(M, \lambda x . upair\_name(fst(x), snd(x), z))$   
 $\langle proof \rangle$

**lemma** (in  $forcing\_data1$ )  $repl\_opname\_check$  :  
**assumes**  $A \in M \ f \in M$   
**shows**  $\{opair\_name(check(x), f'x, \mathbf{1}). x \in A\} \in M$   
 $\langle proof \rangle$

**theorem** (in  $G\_generic1$ )  $choice\_in\_MG$ :  
**assumes**  $choice\_ax(\#\#M)$   
**shows**  $choice\_ax(\#\#M[G])$   
 $\langle proof \rangle$

**sublocale**  $G\_generic1\_AC \subseteq ext:M\_ZC\_basic M[G]$   
 $\langle proof \rangle$

**end**

## 25 Separative notions and proper extensions

**theory**  $Proper\_Extension$   
**imports**  
 $Names$

**begin**

The key ingredient to obtain a proper extension is to have a *separative preorder*:

**locale**  $separative\_notion = forcing\_notion +$   
**assumes**  $separative: p \in \mathbb{P} \implies \exists q \in \mathbb{P}. \exists r \in \mathbb{P}. q \preceq p \wedge r \preceq p \wedge q \perp r$   
**begin**

For separative preorders, the complement of every filter is dense. Hence an  $M$ -generic filter cannot belong to the ground model.

**lemma**  $filter\_complement\_dense$ :  
**assumes**  $filter(G)$   
**shows**  $dense(\mathbb{P} - G)$   
 $\langle proof \rangle$

**end** —  $separative\_notion$

**locale**  $ctm\_separative = forcing\_data1 + separative\_notion$   
**begin**

```

context
  fixes  $G$ 
  assumes  $generic: M\_generic(G)$ 
begin

interpretation  $G\_generic1 \mathbb{P} leq 1 M enum G$ 
   $\langle proof \rangle$ 

lemma  $generic\_not\_in\_M:$ 
  shows  $G \notin M$ 
   $\langle proof \rangle$ 

theorem  $proper\_extension: M \neq M[G]$ 
   $\langle proof \rangle$ 
end
end —  $ctm\_separative$ 

end

```

## 26 A poset of successions

```

theory  $Succession\_Poset$ 
  imports
     $ZF\_Trans\_Interpretations$ 
     $Proper\_Extension$ 
begin

```

In this theory we define a separative poset. Its underlying set is the set of finite binary sequences (that is, with codomain  $2 = \{0, 1\}$ ); of course, one can see that set as the set  $\omega \text{-} || > 2$  or equivalently as the set of partial functions  $Fn(\omega, \omega, 2)$ , i.e. the set of partial functions bounded by  $\omega$ .

The order relation of the poset is that of being less defined as functions (cf.  $Fnlerel(A^{<\omega})$ ), so it could be surprising that we have not used  $Fn(\omega, \omega, 2)$  for the set. The only reason why we keep this alternative definition is because we can prove  $A^{<\omega} \in M$  (and therefore  $Fnlerel(A^{<\omega}) \in M$ ) using only one instance of separation.

```

definition  $seq\_upd :: i \Rightarrow i \Rightarrow i$  where
   $seq\_upd(f,a) \equiv \lambda j \in succ(domain(f)) . if j < domain(f) then f`j else a$ 

```

```

lemma  $seq\_upd\_succ\_type :$ 
  assumes  $n \in nat \ f \in n \rightarrow A \ a \in A$ 
  shows  $seq\_upd(f,a) \in succ(n) \rightarrow A$ 
   $\langle proof \rangle$ 

```

```

lemma  $seq\_upd\_type :$ 
  assumes  $f \in A^{<\omega} \ a \in A$ 

```

**shows**  $seq\_upd(f,a) \in A^{<\omega}$   
*<proof>*

**lemma**  $seq\_upd\_apply\_domain$  [*simp*]:  
**assumes**  $f:n \rightarrow A$   $n \in nat$   
**shows**  $seq\_upd(f,a) \cdot n = a$   
*<proof>*

**lemma**  $zero\_in\_seqspace$  :  
**shows**  $0 \in A^{<\omega}$   
*<proof>*

**definition**  
 $seqlerel :: i \Rightarrow i$  **where**  
 $seqlerel(A) \equiv Fnlerel(A^{<\omega})$

**definition**  
 $seqle :: i$  **where**  
 $seqle \equiv seqlerel(2)$

**lemma**  $seqleI$ [*intro!*]:  
 $\langle f,g \rangle \in 2^{<\omega} \times 2^{<\omega} \implies g \subseteq f \implies \langle f,g \rangle \in seqle$   
*<proof>*

**lemma**  $seqleD$ [*dest!*]:  
 $z \in seqle \implies \exists x y. \langle x,y \rangle \in 2^{<\omega} \times 2^{<\omega} \wedge y \subseteq x \wedge z = \langle x,y \rangle$   
*<proof>*

**lemma**  $upd\_leI$  :  
**assumes**  $f \in 2^{<\omega}$   $a \in 2$   
**shows**  $\langle seq\_upd(f,a), f \rangle \in seqle$  (**is**  $\langle ?f, \_ \rangle \in \_$ )  
*<proof>*

**lemma**  $preorder\_on\_seqle$ :  $preorder\_on(2^{<\omega}, seqle)$   
*<proof>*

**lemma**  $zero\_seqle\_max$ :  $x \in 2^{<\omega} \implies \langle x, 0 \rangle \in seqle$   
*<proof>*

**interpretation**  $sp$ :  $forcing\_notion$   $2^{<\omega}$   $seqle$   $0$   
*<proof>*

**notation**  $sp$ . $Leq$  (**infixl**  $\langle \preceq_s \rangle$  50)  
**notation**  $sp$ . $Incompatible$  (**infixl**  $\langle \perp_s \rangle$  50)

**lemma**  $seqspace\_separative$ :  
**assumes**  $f \in 2^{<\omega}$   
**shows**  $seq\_upd(f,0) \perp_s seq\_upd(f,1)$  (**is**  $?f \perp_s ?g$ )  
*<proof>*

**definition**  $seqleR\_fm :: i \Rightarrow i$  **where**  
 $seqleR\_fm(fg) \equiv Exists(Exists(And(pair\_fm(0,1,fg+\omega 2),subset\_fm(1,0))))$

**lemma**  $type\_seqleR\_fm : fg \in nat \implies seqleR\_fm(fg) \in formula$   
 $\langle proof \rangle$

$\langle ML \rangle$

**lemma** (in  $M\_ctm1$ )  $seqleR\_fm\_sats :$   
**assumes**  $fg \in nat \ env \in list(M)$   
**shows**  $(M, env \models seqleR\_fm(fg)) \longleftrightarrow (\exists f[\#\#M]. \exists g[\#\#M]. pair(\#\#M, f, g, nth(fg, env))$   
 $\wedge f \supseteq g)$   
 $\langle proof \rangle$

**context**  $M\_ctm1$   
**begin**

**lemma**  $seqle\_in\_M: seqle \in M$   
 $\langle proof \rangle$

## 26.1 Cohen extension is proper

**interpretation**  $ctm\_separative \ 2^{<\omega} \ seqle \ 0$   
 $\langle proof \rangle$

**lemma**  $cohen\_extension\_is\_proper: \exists G. M\_generic(G) \wedge M \neq M[G]$   
 $\langle proof \rangle$

**end** —  $M\_ctm1$

**end**

## 27 The existence of generic extensions

**theory**  $Forcing\_Main$

**imports**

$Ordinals\_In\_MG$

$Choice\_Axiom$

$Succession\_Poset$

**begin**

### 27.1 The generic extension is countable

**lemma** (in  $forcing\_data1$ )  $surj\_nat\_MG : \exists f. f \in surj(\omega, M[G])$   
 $\langle proof \rangle$

**lemma** (in  $G\_generic1$ )  $MG\_eqpoll\_nat: M[G] \approx \omega$

*<proof>*

## 27.2 Extensions of ctms of fragments of ZFC

**context**  $G\_generic1$

**begin**

**lemma**  $sats\_ground\_repl\_fm\_imp\_sats\_ZF\_replacement\_fm$ :

**assumes**

$\varphi \in formula\ M, \Box \models \cdot Replacement(ground\_repl\_fm(\varphi)) \cdot$

**shows**

$M[G], \Box \models \cdot Replacement(\varphi) \cdot$

*<proof>*

**lemma**  $satT\_ground\_repl\_fm\_imp\_satT\_ZF\_replacement\_fm$ :

**assumes**

$\Phi \subseteq formula\ M \models \{ \cdot Replacement(ground\_repl\_fm(\varphi)) \cdot \ . \varphi \in \Phi \}$

**shows**

$M[G] \models \{ \cdot Replacement(\varphi) \cdot \ . \varphi \in \Phi \}$

*<proof>*

**end** —  $G\_generic1$

**theorem**  $extensions\_of\_ctms$ :

**assumes**

$M \approx \omega\ Transset(M)$

$M \models \cdot Z \cup \{ \cdot Replacement(p) \cdot \ . p \in overhead \}$

$\Phi \subseteq formula\ M \models \{ \cdot Replacement(ground\_repl\_fm(\varphi)) \cdot \ . \varphi \in \Phi \}$

**shows**

$\exists N.$

$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge M \neq N \wedge$

$(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N)) \wedge$

$((M, \Box \models \cdot AC \cdot) \longrightarrow N, \Box \models \cdot AC \cdot) \wedge N \models \cdot Z \cup \{ \cdot Replacement(\varphi) \cdot \ . \varphi \in \Phi \}$

*<proof>*

**lemma**  $ZF\_replacement\_overhead\_sub\_ZF$ :  $\{ \cdot Replacement(p) \cdot \ . p \in overhead \} \subseteq ZF$

*<proof>*

**theorem**  $extensions\_of\_ctms\_ZF$ :

**assumes**

$M \approx \omega\ Transset(M)\ M \models ZF$

**shows**

$\exists N.$

$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZF \wedge M \neq N \wedge$

$(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N)) \wedge$

$((M, \Box \models \cdot AC \cdot) \longrightarrow N \models ZFC)$

*<proof>*

end

## 28 Preservation of cardinals in generic extensions

theory *Cardinal\_Preservation*

imports

*Forcing\_Main*

begin

context *forcing\_data1*

begin

lemma *antichain\_abs'* [*absolut*]:

$\llbracket A \in M \rrbracket \implies \text{antichain}^M(\mathbb{P}, \text{leq}, A) \longleftrightarrow \text{antichain}(\mathbb{P}, \text{leq}, A)$   
*<proof>*

lemma *inconsistent\_imp\_incompatible*:

assumes  $p \Vdash \varphi$  env  $q \Vdash \text{Neg}(\varphi)$  env  $p \in \mathbb{P}$   $q \in \mathbb{P}$   
 $\text{arity}(\varphi) \leq \text{length}(\text{env})$   $\varphi \in \text{formula}$  env  $\in \text{list}(M)$

shows  $p \perp q$

*<proof>*

notation *check* ( $\langle \_ \rangle$  [101] 100)

end — *forcing\_data1*

locale *G\_generic2* = *G\_generic1* + *forcing\_data2*

locale *G\_generic2\_AC* = *G\_generic1\_AC* + *G\_generic2*

locale *G\_generic3* = *G\_generic2* + *forcing\_data3*

locale *G\_generic3\_AC* = *G\_generic2\_AC* + *G\_generic3*

locale *G\_generic3\_AC\_CH* = *G\_generic3\_AC* + *M\_ZFC2\_ground\_CH\_trans*

sublocale *G\_generic3\_AC*  $\subseteq$  ext:*M\_ZFC2\_trans*  $M[G]$

*<proof>*

lemma (in *forcing\_data1*) *forces\_neq\_apply\_imp\_incompatible*:

assumes

$p \Vdash \cdot 0'1$  is 2.  $[f, a, b^v]$

$q \Vdash \cdot 0'1$  is 2.  $[f, a, b^w]$

$b \neq b'$

— More general version: taking general names  $b^v$  and  $b^w$ , satisfying  $p \Vdash \cdot \neg \cdot 0 = 1 \cdot [b^v, b^w]$  and  $q \Vdash \cdot \neg \cdot 0 = 1 \cdot [b^v, b^w]$ .

and

types:  $f \in M$   $a \in M$   $b \in M$   $b' \in M$   $p \in \mathbb{P}$   $q \in \mathbb{P}$

shows

$p \perp q$   
 <proof>  
 include G\_generic1\_lemmas  
 <proof>

**context** M\_ctm2\_AC  
**begin**

— Simplifying simp rules (because of the occurrence of *setclass*)

**lemmas** sharp\_simps = Card\_rel\_Union Card\_rel\_cardinal\_rel Collect\_abs  
 Cons\_abs Cons\_in\_M\_iff Diff\_closed Equal\_abs Equal\_in\_M\_iff Finite\_abs  
 Forall\_abs Forall\_in\_M\_iff Inl\_abs Inl\_in\_M\_iff Inr\_abs Inr\_in\_M\_iff  
 Int\_closed Inter\_abs Inter\_closed M\_nat Member\_abs Member\_in\_M\_iff  
 Memrel\_closed Nand\_abs Nand\_in\_M\_iff Nil\_abs Nil\_in\_M Ord\_cardinal\_rel  
 Pow\_rel\_closed Un\_closed Union\_abs Union\_closed and\_abs and\_closed  
 apply\_abs apply\_closed bij\_rel\_closed bijection\_abs bool\_of\_o\_abs  
 bool\_of\_o\_closed cadd\_rel\_0 cadd\_rel\_closed cardinal\_rel\_0\_iff\_0  
 cardinal\_rel\_closed cardinal\_rel\_idem cartprod\_abs cartprod\_closed  
 cmult\_rel\_0 cmult\_rel\_1 cmult\_rel\_closed comp\_closed composition\_abs  
 cons\_abs cons\_closed converse\_abs converse\_closed csquare\_lam\_closed  
 csquare\_rel\_closed depth\_closed domain\_abs domain\_closed eclose\_abs  
 eclose\_closed empty\_abs field\_abs field\_closed finite\_funspace\_closed  
 finite\_ordinal\_abs fst\_closed function\_abs function\_space\_rel\_closed  
 hd\_abs image\_abs image\_closed inj\_rel\_closed injection\_abs inter\_abs  
 irreflexive\_abs is\_eclose\_n\_abs is\_funspace\_abs  
 iterates\_closed length\_closed lepoll\_rel\_refl  
 limit\_ordinal\_abs linear\_rel\_abs  
 mem\_bij\_abs mem\_eclose\_abs mem\_inj\_abs membership\_abs  
 minimum\_closed nat\_case\_abs nat\_case\_closed nonempty\_not\_abs  
 not\_closed number1\_abs number2\_abs number3\_abs omega\_abs  
 or\_abs or\_closed order\_isomorphism\_abs ordermap\_closed  
 ordertype\_closed ordinal\_abs pair\_abs pair\_in\_M\_iff powerset\_abs  
 pred\_closed pred\_set\_abs quaselist\_abs quasinat\_abs radd\_closed  
 rall\_abs range\_abs range\_closed relation\_abs restrict\_closed  
 restriction\_abs rex\_abs rmult\_closed rtrancl\_abs rtrancl\_closed  
 rvimage\_closed separation\_closed setdiff\_abs singleton\_abs  
 singleton\_in\_M\_iff snd\_closed strong\_replacement\_closed subset\_abs  
 succ\_in\_M\_iff successor\_abs successor\_ordinal\_abs sum\_abs sum\_closed  
 surj\_rel\_closed surjection\_abs tl\_abs trancl\_abs trancl\_closed  
 transitive\_rel\_abs transitive\_set\_abs typed\_function\_abs union\_abs  
 upair\_abs upair\_in\_M\_iff vimage\_abs vimage\_closed well\_ord\_abs  
 nth\_closed Aleph\_rel\_closed csucc\_rel\_closed  
 Card\_rel\_Aleph\_rel

**declare** sharp\_simps[simp del, simplified setclass\_iff, simp]

**lemmas** sharp\_intros = nat\_into\_M Aleph\_rel\_closed Card\_rel\_Aleph\_rel

**declare** sharp\_intros[rule del, simplified setclass\_iff, intro]

```

end — M_ctm2_AC

context G_generic3_AC begin

context
  includes G_generic1_lemmas
begin

lemmas mg_sharp_simps = ext.Card_rel_Union ext.Card_rel_cardinal_rel
  ext.Collect_abs ext.Cons_abs ext.Cons_in_M_iff ext.Diff_closed
  ext.Equal_abs ext.Equal_in_M_iff ext.Finite_abs ext.Forall_abs
  ext.Forall_in_M_iff ext.Inl_abs ext.Inl_in_M_iff ext.Inr_abs
  ext.Inr_in_M_iff ext.Int_closed ext.Inter_abs ext.Inter_closed
  ext.M_nat ext.Member_abs ext.Member_in_M_iff ext.Memrel_closed
  ext.Nand_abs ext.Nand_in_M_iff ext.Nil_abs ext.Nil_in_M
  ext.Ord_cardinal_rel ext.Pow_rel_closed ext.Un_closed
  ext.Union_abs ext.Union_closed ext.and_abs ext.and_closed
  ext.apply_abs ext.apply_closed ext.bij_rel_closed
  ext.bijection_abs ext.bool_of_o_abs ext.bool_of_o_closed
  ext.cadd_rel_0 ext.cadd_rel_closed ext.cardinal_rel_0_iff_0
  ext.cardinal_rel_closed ext.cardinal_rel_idem ext.cartprod_abs
  ext.cartprod_closed ext.cmult_rel_0 ext.cmult_rel_1
  ext.cmult_rel_closed ext.comp_closed ext.composition_abs
  ext.cons_abs ext.cons_closed ext.converse_abs ext.converse_closed
  ext.csquare_lam_closed ext.csquare_rel_closed ext.depth_closed
  ext.domain_abs ext.domain_closed ext.eclose_abs ext.eclose_closed
  ext.empty_abs ext.field_abs ext.field_closed
  ext.finite_funspace_closed ext.finite_ordinal_abs
  ext.fst_closed ext.function_abs ext.function_space_rel_closed
  ext.hd_abs ext.image_abs ext.image_closed ext.inj_rel_closed
  ext.injection_abs ext.inter_abs ext.irreflexive_abs
  ext.is_eclose_n_abs ext.is_funspace_abs
  ext.iterates_closed ext.length_closed
  ext.lepoll_rel_refl ext.limit_ordinal_abs ext.linear_rel_abs
  ext.mem_bij_abs ext.mem_eclose_abs
  ext.mem_inj_abs ext.membership_abs
  ext.nat_case_abs ext.nat_case_closed
  ext.nonempty ext.not_abs ext.not_closed
  ext.number1_abs ext.number2_abs ext.number3_abs ext.omega_abs
  ext.or_abs ext.or_closed ext.order_isomorphism_abs
  ext.ordermap_closed ext.ordertype_closed ext.ordinal_abs
  ext.pair_abs ext.pair_in_M_iff ext.powerset_abs ext.pred_closed
  ext.pred_set_abs ext.quaselist_abs ext.quasinat_abs
  ext.radd_closed ext.rall_abs ext.range_abs ext.range_closed
  ext.relation_abs ext.restrict_closed ext.restriction_abs
  ext.rex_abs ext.rmult_closed ext.rtrancl_abs ext.rtrancl_closed
  ext.rvimage_closed ext.separation_closed ext.setdiff_abs
  ext.singleton_abs ext.singleton_in_M_iff ext.snd_closed

```

*ext.strong\_replacement\_closed ext.subset\_abs ext.succ\_in\_M\_iff*  
*ext.successor\_abs ext.successor\_ordinal\_abs ext.sum\_abs*  
*ext.sum\_closed ext.surj\_rel\_closed ext.surjection\_abs ext.tl\_abs*  
*ext.trancl\_abs ext.trancl\_closed ext.transitive\_rel\_abs*  
*ext.transitive\_set\_abs ext.typed\_function\_abs ext.union\_abs*  
*ext.upair\_abs ext.upair\_in\_M\_iff ext.vimage\_abs ext.vimage\_closed*  
*ext.well\_ord\_abs ext.nth\_closed ext.Aleph\_rel\_closed*  
*ext.csucc\_rel\_closed ext.Card\_rel\_Aleph\_rel*

— The following was motivated by the fact that *ext.apply\_closed* did not simplify appropriately.

**declare** *mg\_sharp\_simps*[*simp del, simplified setclass\_iff, simp*]

**lemmas** *mg\_sharp\_intros* = *ext.nat\_into\_M ext.Aleph\_rel\_closed*  
*ext.Card\_rel\_Aleph\_rel*

**declare** *mg\_sharp\_intros*[*rule del, simplified setclass\_iff, intro*]

— Kunen IV.2.31

**lemma** *forces\_below\_filter*:

**assumes**  $M[G]$ ,  $\text{map}(\text{val}(G), \text{env}) \models \varphi$   $p \in G$   
 $\text{arity}(\varphi) \leq \text{length}(\text{env})$   $\varphi \in \text{formula}$   $\text{env} \in \text{list}(M)$   
**shows**  $\exists q \in G. q \preceq p \wedge q \Vdash \varphi$   $\text{env}$

*<proof>*

## 28.1 Preservation by ccc forcing notions

**lemma** *ccc\_fun\_closed\_lemma\_aux*:

**assumes**  $f \cdot \text{dot} \in M$   $p \in M$   $a \in M$   $b \in M$

**shows**  $\{q \in \mathbb{P} . q \preceq p \wedge (M, [q, \mathbb{P}, \text{leq}, \mathbf{1}, f \cdot \text{dot}, a^v, b^v] \models \text{forces}(\cdot 0'1 \text{ is } 2 \cdot))\} \in M$

*<proof>*

**lemma** *ccc\_fun\_closed\_lemma\_aux2*:

**assumes**  $B \in M$   $f \cdot \text{dot} \in M$   $p \in M$   $a \in M$

**shows**  $(\#\#M)(\lambda b \in B. \{q \in \mathbb{P} . q \preceq p \wedge (M, [q, \mathbb{P}, \text{leq}, \mathbf{1}, f \cdot \text{dot}, a^v, b^v] \models \text{forces}(\cdot 0'1 \text{ is } 2 \cdot))\})$

*<proof>*

**lemma** *ccc\_fun\_closed\_lemma*:

**assumes**  $A \in M$   $B \in M$   $f \cdot \text{dot} \in M$   $p \in M$

**shows**  $(\lambda a \in A. \{b \in B. \exists q \in \mathbb{P}. q \preceq p \wedge (q \Vdash \cdot 0'1 \text{ is } 2 \cdot [f \cdot \text{dot}, a^v, b^v])\}) \in M$

*<proof>*

**lemma** *ccc\_fun\_approximation\_lemma*:

**notes** *le\_trans*[*trans*]

**assumes**  $\text{ccc}^M(\mathbb{P}, \text{leq})$   $A \in M$   $B \in M$   $f \in M[G]$   $f : A \rightarrow B$

**shows**

$\exists F \in M. F : A \rightarrow \text{Pow}^M(B) \wedge (\forall a \in A. f'a \in F'a \wedge |F'a|^M \leq \omega)$

*<proof>*

**end** — *G\_generic1\_lemmas* bundle

**end** — *G\_generic3\_AC*

**end**

## 29 Model of the negation of the Continuum Hypothesis

**theory** *Not\_CH*

**imports**

*Cardinal\_Preservation*

**begin**

We are taking advantage that the poset of finite functions is absolute, and thus we work with the unrelativized *Fn*. But it would have been more appropriate to do the following using the relative *Fn\_rel*. As it turns out, the present theory was developed prior to having *Fn* relativized!

We also note that  $Fn(\omega, \kappa \times \omega, 2)$  is separative, i.e. each  $X \in Fn(\omega, \kappa \times \omega, 2)$  has two incompatible extensions; therefore we may recover part of our previous theorem *extensions\_of\_ctms\_ZF*. But that result also included the possibility of not having *AC* in the ground model, which would not be sensible in a context where the cardinality of the continuum is under discussion. It is also the case that *extensions\_of\_ctms\_ZF* was historically our first formalized result (with a different proof) that showed the forcing machinery had all of its elements in place.

**abbreviation**

*Add\_subs* ::  $i \Rightarrow i$  **where**

$Add\_subs(\kappa) \equiv Fn(\omega, \kappa \times \omega, 2)$

**abbreviation**

*Add\_le* ::  $i \Rightarrow i$  **where**

$Add\_le(\kappa) \equiv Fnle(\omega, \kappa \times \omega, 2)$

**lemma** (**in** *M\_aleph*) *Aleph\_rel2\_closed*[*intro,simp*]:  $M(\aleph_2^M)$

*<proof>*

**locale** *M\_master* = *M\_cohen* + *M\_library* +

**assumes**

*UN\_lepoll\_assumptions*:

$M(A) \Longrightarrow M(b) \Longrightarrow M(f) \Longrightarrow M(A') \Longrightarrow separation(M, \lambda y. \exists x \in A'. y = \langle x, \mu \ i. x \in if\_range\_F\_else\_F((\cdot)(A), b, f, i))$ )

### 29.1 Non-absolute concepts between extensions

**sublocale** *M\_master*  $\subseteq$  *M\_Pi\_replacement*

*<proof>*

**locale**  $M\_master\_sub = M\_master + N:M\_aleph\ N$  **for**  $N +$   
**assumes**  
   $M\_imp\_N: M(x) \implies N(x)$  **and**  
   $Ord\_iff: Ord(x) \implies M(x) \longleftrightarrow N(x)$

**sublocale**  $M\_master\_sub \subseteq M\_N\_Perm$   
*<proof>*

**context**  $M\_master\_sub$   
**begin**

**lemma**  $cardinal\_rel\_le\_cardinal\_rel: M(X) \implies |X|^N \leq |X|^M$   
*<proof>*

**lemma**  $Aleph\_rel\_sub\_closed: Ord(\alpha) \implies M(\alpha) \implies N(\aleph_\alpha^M)$   
*<proof>*

**lemma**  $Card\_rel\_imp\_Card\_rel: Card^N(\kappa) \implies M(\kappa) \implies Card^M(\kappa)$   
*<proof>*

**lemma**  $csucc\_rel\_le\_csucc\_rel:$   
  **assumes**  $Ord(\kappa) M(\kappa)$   
  **shows**  $(\kappa^+)^M \leq (\kappa^+)^N$   
*<proof>*

**lemma**  $Aleph\_rel\_le\_Aleph\_rel: Ord(\alpha) \implies M(\alpha) \implies \aleph_\alpha^M \leq \aleph_\alpha^N$   
*<proof>*

**end** —  $M\_master\_sub$

**lemmas** (**in**  $M\_ZF2\_trans$ )  $sep\_instances =$   
   $separation\_ifrangeF\_body\ separation\_ifrangeF\_body2\ separation\_ifrangeF\_body3$   
   $separation\_ifrangeF\_body4\ separation\_ifrangeF\_body5\ separation\_ifrangeF\_body6$   
   $separation\_ifrangeF\_body7\ separation\_cardinal\_rel\_lesspoll\_rel$   
   $separation\_is\_dcwit\_body\ separation\_cdltgamma\ separation\_cdeggamma$

**lemmas** (**in**  $M\_ZF2\_trans$ )  $repl\_instances = lam\_replacement\_inj\_rel$

**sublocale**  $M\_ZFC2\_ground\_notCH\_trans \subseteq M\_master\ \#\#M$   
*<proof>*

**sublocale**  $M\_ZFC2\_trans \subseteq M\_Pi\_replacement\ \#\#M$   
*<proof>*

## 29.2 Cohen forcing is ccc

**context**  $M\_ctm2\_AC$

```

begin

lemma ccc_Add_subs_Aleph_2: cccM(Add_subs( $\aleph_2^M$ ), Add_le( $\aleph_2^M$ ))
⟨proof⟩

end — M_ctm2_AC

sublocale G_generic3_AC ⊆ M_master_sub ##M ##(M[G])
⟨proof⟩

lemma (in M_trans) mem_F_bound4:
  fixes F A
  defines F ≡ (·)
  shows  $x \in F(A, c) \implies c \in (\text{range}(f) \cup \text{domain}(A))$ 
  ⟨proof⟩

lemma (in M_trans) mem_F_bound5:
  fixes F A
  defines F ≡  $\lambda x. A \text{ ` } x$ 
  shows  $x \in F(A, c) \implies c \in (\text{range}(f) \cup \text{domain}(A))$ 
  ⟨proof⟩

sublocale M_ctm2_AC ⊆ M_replacement_lepoll ##M (·)
⟨proof⟩

context G_generic3_AC begin

context
  includes G_generic1_lemmas
begin

lemma G_in_MG:  $G \in M[G]$ 
⟨proof⟩

lemma ccc_preserves_Aleph_succ:
  assumes cccM( $\mathbb{P}$ , leq) Ord(z)  $z \in M$ 
  shows  $\text{Card}^{M[G]}(\aleph_{\text{succ}(z)}^M)$ 
  ⟨proof⟩

end — bundle G_generic1_lemmas

end — G_generic3_AC

context M_ctm1
begin

abbreviation
  Add :: i where
  Add ≡ Fn( $\omega$ ,  $\aleph_2^M \times \omega$ , 2)

```

```

end — M_ctm1

locale add_generic3 = G_generic3_AC Fn( $\omega$ ,  $\aleph_2^{\#\#M} \times \omega$ , 2) Fnle( $\omega$ ,  $\aleph_2^{\#\#M} \times \omega$ , 2) 0

sublocale add_generic3  $\subseteq$  cohen_data  $\omega$   $\aleph_2^M \times \omega$  2 <proof>

context add_generic3
begin

notation Leq (infixl  $\langle \preceq \rangle$  50)
notation Incompatible (infixl  $\langle \perp \rangle$  50)

lemma Add_subs_preserves_Aleph_succ:  $\text{Ord}(z) \implies z \in M \implies \text{Card}^{M[G]}(\aleph_{\text{succ}(z)}^M)$ 
<proof>

lemma Aleph_rel_nats_MG_eq_Aleph_rel_nats_M:
includes G_generic1_lemmas
assumes  $z \in \omega$ 
shows  $\aleph_z^{M[G]} = \aleph_z^M$ 
<proof>

abbreviation
f_G ::  $i$  (f_G) where
f_G  $\equiv \bigcup G$ 

abbreviation
dom_dense ::  $i \Rightarrow i$  where
dom_dense( $x$ )  $\equiv \{p \in \text{Add} . x \in \text{domain}(p)\}$ 

declare (in M_ctm2_AC) Fnat_closed[simplified setclass_iff, simp, intro]
declare (in M_ctm2_AC) Fnle_nat_closed[simp del, rule del,
simplified setclass_iff, simp, intro]
declare (in M_ctm2_AC) cexp_rel_closed[simplified setclass_iff, simp, intro]
declare (in G_generic3_AC) ext.cexp_rel_closed[simplified setclass_iff, simp,
intro]

lemma dom_dense_closed[intro, simp]:  $x \in \aleph_2^M \times \omega \implies \text{dom\_dense}(x) \in M$ 
<proof>

lemma domain_f_G: assumes  $x \in \aleph_2^M$   $y \in \omega$ 
shows  $\langle x, y \rangle \in \text{domain}(f_G)$ 
<proof>

lemma f_G_funtype:
includes G_generic1_lemmas
shows  $f_G : \aleph_2^M \times \omega \rightarrow 2$ 
<proof>

```

**lemma** *inj\_dense\_closed*[*intro,simp*]:  
 $w \in \aleph_2^M \implies x \in \aleph_2^M \implies \text{inj\_dense}(\aleph_2^M, 2, w, x) \in M$   
 ⟨*proof*⟩

**lemma** *Aleph\_rel2\_new\_reals*:  
**assumes**  $w \in \aleph_2^M$   $x \in \aleph_2^M$   $w \neq x$   
**shows**  $(\lambda n \in \omega. f_G \text{ ` } \langle w, n \rangle) \neq (\lambda n \in \omega. f_G \text{ ` } \langle x, n \rangle)$   
 ⟨*proof*⟩

**definition**  
 $h_G :: i \text{ ` } \langle h_G \rangle$  **where**  
 $h_G \equiv \lambda \alpha \in \aleph_2^M. \lambda n \in \omega. f_G \text{ ` } \langle \alpha, n \rangle$

**lemma** *h\_G\_in\_MG*[*simp*]:  
**includes** *G\_generic1\_lemmas*  
**shows**  $h_G \in M[G]$   
 ⟨*proof*⟩

**lemma** *h\_G\_inj\_Aleph\_rel2\_reals*:  $h_G \in \text{inj}^{M[G]}(\aleph_2^M, \omega \rightarrow^{M[G]} 2)$   
 ⟨*proof*⟩

**lemma** *Aleph2\_extension\_le\_continuum\_rel*:  
**includes** *G\_generic1\_lemmas*  
**shows**  $\aleph_2^{M[G]} \leq 2^{\uparrow \aleph_0^{M[G]}, M[G]}$   
 ⟨*proof*⟩

**lemma** *Aleph\_rel\_lt\_continuum\_rel*:  $\aleph_1^{M[G]} < 2^{\uparrow \aleph_0^{M[G]}, M[G]}$   
 ⟨*proof*⟩

**corollary** *not\_CH*:  $\aleph_1^{M[G]} \neq 2^{\uparrow \aleph_0^{M[G]}, M[G]}$   
 ⟨*proof*⟩

**end** — *add\_generic3*

### 29.3 Models of fragments of $ZFC + \neg CH$

**definition**  
 $ContHyp :: o$  **where**  
 $ContHyp \equiv \aleph_1 = 2^{\uparrow \aleph_0}$

⟨*ML*⟩  
**notation** *ContHyp\_rel* ( $\langle CH \rightarrow \rangle$ )  
 ⟨*ML*⟩

**context** *M\_ZF\_library*  
**begin**

⟨*ML*⟩

*<proof>*

**end** — *M\_ZF\_library*

*<ML>*

**notation** *is\_ContHyp\_fm* ( $\langle \cdot CH \cdot \rangle$ )

**theorem** *ctm\_of\_not\_CH*:

**assumes**

$M \approx \omega$  *Transset*(*M*)  $M \models ZC \cup \{ \cdot Replacement(p) \cdot \mid p \in overhead\_notCH \}$

$\Phi \subseteq formula$   $M \models \{ \cdot Replacement(ground\_repl\_fm(\varphi)) \cdot \mid \varphi \in \Phi \}$

**shows**

$\exists N.$

$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZC \cup \{ \cdot \neg CH \cdot \} \cup \{ \cdot Replacement(\varphi) \cdot \mid \varphi \in \Phi \} \wedge$

$(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$

*<proof>*

**lemma** *ZF\_replacement\_overhead\_sub\_ZFC*:  $\{ \cdot Replacement(p) \cdot \mid p \in overhead \} \subseteq ZFC$

*<proof>*

**lemma** *ZF\_replacement\_overhead\_notCH\_sub\_ZFC*:  $\{ \cdot Replacement(p) \cdot \mid p \in overhead\_notCH \} \subseteq ZFC$

*<proof>*

**lemma** *ZF\_replacement\_overhead\_CH\_sub\_ZFC*:  $\{ \cdot Replacement(p) \cdot \mid p \in overhead\_CH \} \subseteq ZFC$

*<proof>*

**corollary** *ctm\_ZFC\_imp\_ctm\_not\_CH*:

**assumes**

$M \approx \omega$  *Transset*(*M*)  $M \models ZFC$

**shows**

$\exists N.$

$M \subseteq N \wedge N \approx \omega \wedge Transset(N) \wedge N \models ZFC \cup \{ \cdot \neg CH \cdot \} \wedge$

$(\forall \alpha. Ord(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$

*<proof>*

**end**

## 30 Preservation results for $\kappa$ -closed forcing notions

**theory** *Kappa\_Closed\_Notions*

**imports**

*Not\_CH*

**begin**

**definition**

$lerel :: i \Rightarrow i$  **where**  
 $lerel(\alpha) \equiv Memrel(\alpha) \cup id(\alpha)$

**lemma**  $lerelI[intro!]$ :  $x \leq y \Rightarrow y \in \alpha \Rightarrow Ord(\alpha) \Rightarrow \langle x, y \rangle \in lerel(\alpha)$   
 $\langle proof \rangle$

**lemma**  $lerelD[dest]$ :  $\langle x, y \rangle \in lerel(\alpha) \Rightarrow Ord(\alpha) \Rightarrow x \leq y$   
 $\langle proof \rangle$

**definition**

$mono\_seqspace :: [i, i, i] \Rightarrow i (\langle \_ \rangle \langle \_ \rangle \rightarrow '(\_, \_)' [61] 60)$  **where**  
 $\alpha \langle \_ \rangle (P, leq) \equiv mono\_map(\alpha, Memrel(\alpha), P, leq)$

$\langle ML \rangle$

**context**  $M\_ZF\_library$

**begin**

$\langle ML \rangle$

$\langle proof \rangle$

**end** —  $M\_ZF\_library$

**abbreviation**

$mono\_seqspace\_r (\langle \_ \rangle \langle \_ \rangle \rightarrow '(\_, \_)' [61] 60)$  **where**  
 $\alpha \langle \_ \rangle^M (P, leq) \equiv mono\_seqspace\_rel(M, \alpha, P, leq)$

**abbreviation**

$mono\_seqspace\_r\_set (\langle \_ \rangle \langle \_ \rangle \rightarrow '(\_, \_)' [61] 60)$  **where**  
 $\alpha \langle \_ \rangle^M (P, leq) \equiv mono\_seqspace\_rel(\#\#M, \alpha, P, leq)$

**lemma**  $mono\_seqspaceI[intro!]$ :

**includes**  $mono\_map\_rules$

**assumes**  $f: A \rightarrow P \wedge x y. x \in A \Rightarrow y \in A \Rightarrow x < y \Rightarrow \langle f'x, f'y \rangle \in leq$   $Ord(A)$

**shows**  $f: A \langle \_ \rangle (P, leq)$

$\langle proof \rangle$

**lemma** (**in**  $M\_ZF\_library$ )  $mono\_seqspace\_rel\_char$ :

**assumes**  $M(A) M(P) M(leq)$

**shows**  $A \langle \_ \rangle^M (P, leq) = \{f \in A \langle \_ \rangle (P, leq). M(f)\}$

$\langle proof \rangle$

**lemma** (**in**  $M\_ZF\_library$ )  $mono\_seqspace\_relI[intro!]$ :

**assumes**  $f: A \rightarrow^M P \wedge x y. x \in A \Rightarrow y \in A \Rightarrow x < y \Rightarrow \langle f'x, f'y \rangle \in leq$

$Ord(A) M(A) M(P) M(leq)$

**shows**  $f: A \langle \_ \rangle^M (P, leq)$

$\langle proof \rangle$

**lemma** *mono\_seqspace\_is\_fun*[*dest*]:  
**includes** *mono\_map\_rules*  
**shows**  $j: A \lt \rightarrow (P, leq) \implies j: A \rightarrow P$   
 $\langle proof \rangle$

**lemma** *mono\_map\_lt\_le\_is\_mono*[*dest*]:  
**includes** *mono\_map\_rules*  
**assumes**  $j: A \lt \rightarrow (P, leq) \ a \in A \ c \in A \ a \leq c \ Ord(A) \ refl(P, leq)$   
**shows**  $\langle j'a, j'c \rangle \in leq$   
 $\langle proof \rangle$

**lemma** (**in** *M\_ZF\_library*) *mem\_mono\_seqspace\_abs*[*absolut*]:  
**assumes**  $M(f) \ M(A) \ M(P) \ M(leq)$   
**shows**  $f: A \lt \rightarrow^M (P, leq) \longleftrightarrow f: A \lt \rightarrow (P, leq)$   
 $\langle proof \rangle$

**definition**  
 $mono\_map\_lt\_le :: [i, i] \Rightarrow i \ (\mathbf{infixr} \ \lt \rightarrow \leq) \ 60$  **where**  
 $\alpha \lt \rightarrow \leq \beta \equiv \alpha \lt \rightarrow (\beta, lerel(\beta))$

**lemma** *mono\_map\_lt\_leI*[*intro!*]:  
**includes** *mono\_map\_rules*  
**assumes**  $f: A \rightarrow B \ \bigwedge x \ y. \ x \in A \implies y \in A \implies x < y \implies f'x \leq f'y \ Ord(A) \ Ord(B)$   
**shows**  $f: A \lt \rightarrow \leq B$   
 $\langle proof \rangle$

**definition**  
 $kappa\_closed :: [i, i, i] \Rightarrow o \ (\lt\_closed'(\_, \_))$  **where**  
 $\kappa\_closed(P, leq) \equiv \forall \delta. \ \delta < \kappa \longrightarrow (\forall f \in \delta \ \lt \rightarrow (P, converse(leq))). \ \exists q \in P. \ \forall \alpha \in \delta. \ \langle q, f'\alpha \rangle \in leq$

$\langle ML \rangle$

**abbreviation**  
 $kappa\_closed\_r \ (\lt\_closed'(\_, \_))$  [61] 60 **where**  
 $\kappa\_closed^M(P, leq) \equiv kappa\_closed\_rel(M, \kappa, P, leq)$

**abbreviation**  
 $kappa\_closed\_r\_set \ (\lt\_closed'(\_, \_))$  [61] 60 **where**  
 $\kappa\_closed^M(P, leq) \equiv kappa\_closed\_rel(\#\#\ M, \kappa, P, leq)$

**lemma** (**in** *forcing\_data3*) *forcing\_a\_value*:  
**assumes**  $p \Vdash \cdot 0: 1 \rightarrow 2 \cdot [f\_dot, A^v, B^v] \ a \in A$   
 $q \preceq p \ q \in \mathbb{P} \ p \in \mathbb{P} \ f\_dot \in M \ A \in M \ B \in M$   
**shows**  $\exists d \in \mathbb{P}. \ \exists b \in B. \ d \preceq q \wedge d \Vdash \cdot 0' 1 \text{ is } 2 \cdot [f\_dot, a^v, b^v]$

$\langle proof \rangle$   
**include** *G\_generic1\_lemmas*  
 $\langle proof \rangle$

**locale**  $M\_master\_CH = M\_master + M\_library\_DC$

**sublocale**  $M\_ZFC2\_ground\_CH\_trans \subseteq M\_master\_CH \#\#M$   
 $\langle proof \rangle$

**context**  $G\_generic3\_AC\_CH$  **begin**

**context**  
**includes**  $G\_generic1\_lemmas$   
**begin**

**lemma**  $separation\_check\_snd\_aux$ :  
**assumes**  $f\_dot \in M \ \tau \in M \ \chi \in formula \ arity(\chi) \leq 7$   
**shows**  $separation(\#\#M, \lambda r. M, [fst(r), \mathbb{P}, leq, \mathbf{1}, f\_dot, \tau, snd(r)^v] \models \chi)$   
 $\langle proof \rangle$

**lemma**  $separation\_check\_fst\_snd\_aux$  :  
**assumes**  $f\_dot \in M \ r \in M \ \chi \in formula \ arity(\chi) \leq 7$   
**shows**  $separation(\#\#M, \lambda p. M, [r, \mathbb{P}, leq, \mathbf{1}, f\_dot, fst(p)^v, snd(p)^v] \models \chi)$   
 $\langle proof \rangle$

**lemma**  $separation\_leq\_and\_forces\_apply\_aux$ :  
**assumes**  $f\_dot \in M \ B \in M$   
**shows**  $\forall n \in M. separation(\#\#M, \lambda x. snd(x) \preceq fst(x) \wedge$   
 $(\exists b \in B. M, [snd(x), \mathbb{P}, leq, \mathbf{1}, f\_dot, (\bigcup (n))^v, b^v] \models forces(\cdot 0'1 \text{ is } 2. )))$   
 $\langle proof \rangle$

**lemma**  $separation\_leq\_and\_forces\_apply\_aux'$ :  
**assumes**  $f\_dot \in M \ p \in M \ B \in M$   
**shows**  $separation$   
 $(\#\#M, \lambda p. snd(snd(p)) \preceq fst(snd(p)) \wedge$   
 $(\exists b \in B. M, [snd(snd(p)), \mathbb{P}, leq, \mathbf{1}, f\_dot, (\bigcup fst(p))^v, b^v] \models forces(\cdot 0'1 \text{ is } 2. )))$   
 $\langle proof \rangle$

**lemma**  $separation\_closed\_leq\_and\_forces\_eq\_check\_aux$  :  
**assumes**  $A \in M \ r \in G \ \tau \in M$   
**shows**  $(\#\#M)(\{q \in \mathbb{P}. \exists h \in A. q \preceq r \wedge q \Vdash \cdot 0 = 1. [\tau, h^v]\})$   
 $\langle proof \rangle$

**lemma**  $separation\_closed\_forces\_apply\_aux$ :  
**assumes**  $B \in M \ f\_dot \in M \ r \in M$   
**shows**  $(\#\#M)(\{(n, b) \in \omega \times B. r \Vdash \cdot 0'1 \text{ is } 2. [f\_dot, n^v, b^v]\})$   
 $\langle proof \rangle$

**lemma**  $kunen\_IV\_6\_9\_function\_space\_rel\_eq$ :  
**assumes**  $\bigwedge p \ \tau. p \Vdash \cdot 0:1 \rightarrow 2. [\tau, A^v, B^v] \implies p \in \mathbb{P} \implies \tau \in M \implies$   
 $\exists q \in \mathbb{P}. \exists h \in A \rightarrow^M B. q \preceq p \wedge q \Vdash \cdot 0 = 1. [\tau, h^v] \ A \in M \ B \in M$   
**shows**  
 $A \rightarrow^M B = A \rightarrow^{M[G]} B$   
 $\langle proof \rangle$

### 30.1 $(\omega + 1)$ -Closed notions preserve countable sequences

**lemma** *succ\_omega\_closed\_imp\_no\_new\_nat\_sequences*:  
**assumes** *succ( $\omega$ )-closed $^M(\mathbb{P}, leq)$*   $f : \omega \rightarrow B$   $f \in M[G]$   $B \in M$   
**shows**  $f \in M$   
*<proof>*

**declare** *mono\_seqspace\_rel\_closed*[*rule del*]  
— Mysteriously breaks the end of the next proof

**lemma** *succ\_omega\_closed\_imp\_no\_new\_reals*:  
**assumes** *succ( $\omega$ )-closed $^M(\mathbb{P}, leq)$*   
**shows**  $\omega \rightarrow^M 2 = \omega \rightarrow^{M[G]} 2$   
*<proof>*

**lemma** *succ\_omega\_closed\_imp\_Aleph\_1\_preserved*:  
**assumes** *succ( $\omega$ )-closed $^M(\mathbb{P}, leq)$*   
**shows**  $\aleph_1^M = \aleph_1^{M[G]}$   
*<proof>*

**end** — bundle *G\_generic1\_lemmas*

**end** — *G\_generic3\_AC*

**end**

## 31 Forcing extension satisfying the Continuum Hypothesis

**theory** *CH*  
**imports**  
*Kappa\_Closed\_Notions*  
*Cohen\_Posets\_Relative*  
**begin**

**context** *M\_ctm2\_AC*  
**begin**

**declare** *F $n$ \_rel\_closed*[*simp del, rule del, simplified setclass\_iff, simp, intro*]  
**declare** *F $n$ le\_rel\_closed*[*simp del, rule del, simplified setclass\_iff, simp, intro*]

**abbreviation**  
*Coll* :: *i* **where**  
*Coll*  $\equiv$   $F $n$ ^M(\aleph_1^M, \aleph_1^M, \omega \rightarrow^M 2)$

**abbreviation**  
*Colleq* :: *i* **where**  
*Colleq*  $\equiv$   $F $n$ le^M(\aleph_1^M, \aleph_1^M, \omega \rightarrow^M 2)$

**lemma** *Coll\_in\_M*[*intro,simp*]:  $Coll \in M$  *<proof>*

**lemma** *Colleq\_refl* :  $refl(Coll, Colleq)$   
*<proof>*

### 31.1 Collapse forcing is sufficiently closed

**lemma** *succ\_omega\_closed\_Coll*:  $succ(\omega)$ - $closed^M(Coll, Colleq)$   
*<proof>*

**end** — *M\_ctm2\_AC*

**locale** *collapse\_CH* = *G\_generic3\_AC\_CH*  $Fn^M(\aleph_1^{##M}, \aleph_1^M, \omega \rightarrow^M 2)$   $Fnle^M(\aleph_1^{##M}, \aleph_1^M, \omega \rightarrow^M 2)$  0

**sublocale** *collapse\_CH*  $\subseteq$  *forcing\_notion* *Coll* *Colleq* 0  
*<proof>*

**context** *collapse\_CH*  
**begin**

**notation** *Leq* (**infixl**  $\langle \preceq \rangle$  50)

**notation** *Incompatible* (**infixl**  $\langle \perp \rangle$  50)

**abbreviation**

$f_G :: i \langle f_G \rangle$  **where**  
 $f_G \equiv \bigcup G$

**lemma** *f\_G\_in\_MG*[*simp*]:  
**shows**  $f_G \in M[G]$   
*<proof>*

**abbreviation**

$dom\_dense :: i \Rightarrow i$  **where**  
 $dom\_dense(x) \equiv \{ p \in Coll . x \in domain(p) \}$

**lemma** *dom\_dense\_closed*[*intro,simp*]:  $x \in M \implies dom\_dense(x) \in M$   
*<proof>*

**lemma** *domain\_f\_G*: **assumes**  $x \in \aleph_1^M$   
**shows**  $x \in domain(f_G)$   
*<proof>*

**lemma** *Un\_filter\_is\_function*:  
**assumes**  $filter(G)$   
**shows**  $function(\bigcup G)$   
*<proof>*

**lemma** *f\_G\_funtype*:  
**shows**  $f_G : \aleph_1^M \rightarrow \omega \rightarrow^{M[G]} \mathcal{P}$   
 $\langle \text{proof} \rangle$

**abbreviation**  
*surj\_dense* ::  $i \Rightarrow i$  **where**  
*surj\_dense*( $x$ )  $\equiv \{ p \in \text{Coll} . x \in \text{range}(p) \}$

**lemma** *surj\_dense\_closed*[*intro,simp*]:  
 $x \in \omega \rightarrow^M \mathcal{P} \implies \text{surj\_dense}(x) \in M$   
 $\langle \text{proof} \rangle$

**lemma** *reals\_sub\_image\_f\_G*:  
**assumes**  $x \in \omega \rightarrow^M \mathcal{P}$   
**shows**  $\exists \alpha \in \aleph_1^M . f_G \upharpoonright \alpha = x$   
 $\langle \text{proof} \rangle$

**lemma** *f\_G\_surj\_Aleph\_rel1\_reals*:  $f_G \in \text{surj}^{M[G]}(\aleph_1^M, \omega \rightarrow^{M[G]} \mathcal{P})$   
 $\langle \text{proof} \rangle$

**lemma** *continuum\_rel\_le\_Aleph1\_extension*:  
**includes** *G\_generic1\_lemmas*  
**shows**  $\mathcal{P}^{\uparrow \aleph_0^{M[G], M[G]}} \leq \aleph_1^{M[G]}$   
 $\langle \text{proof} \rangle$

**theorem** *CH*:  $\aleph_1^{M[G]} = \mathcal{P}^{\uparrow \aleph_0^{M[G], M[G]}}$   
 $\langle \text{proof} \rangle$

**end** — *collapse\_CH*

## 31.2 Models of fragments of ZFC + CH

**theorem** *ctm\_of\_CH*:  
**assumes**  
 $M \approx \omega \text{ Transset}(M)$   
 $M \models \text{ZC} \cup \{ \cdot \text{Replacement}(p) . p \in \text{overhead\_CH} \}$   
 $\Phi \subseteq \text{formula } M \models \{ \cdot \text{Replacement}(\text{ground\_repl\_fm}(\varphi)) . \varphi \in \Phi \}$   
**shows**  
 $\exists N .$   
 $M \subseteq N \wedge N \approx \omega \wedge \text{Transset}(N) \wedge N \models \text{ZC} \cup \{ \cdot \text{CH} \} \cup \{ \cdot \text{Replacement}(\varphi) .$   
 $\varphi \in \Phi \} \wedge$   
 $(\forall \alpha . \text{Ord}(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$   
 $\langle \text{proof} \rangle$

**corollary** *ctm\_ZFC\_imp\_ctm\_CH*:  
**assumes**  
 $M \approx \omega \text{ Transset}(M) \wedge M \models \text{ZFC}$   
**shows**  
 $\exists N .$

$$M \subseteq N \wedge N \approx \omega \wedge \text{Transset}(N) \wedge N \models \text{ZFC} \cup \{\cdot\text{CH}\cdot\} \wedge$$

$$(\forall \alpha. \text{Ord}(\alpha) \longrightarrow (\alpha \in M \longleftrightarrow \alpha \in N))$$
 <proof>

end

## 32 From $M$ to $\mathcal{V}$

**theory** *Absolute\_Versions*

**imports**

*CH*

*ZF.Cardinal\_AC*

**begin**

**hide\_const** (open) *Order.pred*

### 32.1 Locales of a class $M$ hold in $\mathcal{V}$

**interpretation**  $V: M\_trivial \mathcal{V}$

<proof>

**lemmas** *bad\_simps* =  $V.nonempty$   $V.Forall\_in\_M\_iff$   $V.Inl\_in\_M\_iff$   $V.Inr\_in\_M\_iff$   
 $V.succ\_in\_M\_iff$   $V.singleton\_in\_M\_iff$   $V.Equal\_in\_M\_iff$   $V.Member\_in\_M\_iff$   
 $V.Nand\_in\_M\_iff$   
 $V.Cons\_in\_M\_iff$   $V.pair\_in\_M\_iff$   $V.upair\_in\_M\_iff$

**lemmas** *bad\_M\_trivial\_simps[simp del]* =  $V.Forall\_in\_M\_iff$   $V.Equal\_in\_M\_iff$   
 $V.nonempty$

**lemmas** *bad\_M\_trivial\_rules[rule del]* =  $V.pair\_in\_MI$   $V.singleton\_in\_MI$   
 $V.pair\_in\_MD$   $V.nat\_into\_M$   
 $V.depth\_closed$   $V.length\_closed$   $V.nat\_case\_closed$   $V.separation\_closed$   
 $V.Un\_closed$   $V.strong\_replacement\_closed$   $V.nonempty$

**interpretation**  $V: M\_basic \mathcal{V}$

<proof>

**interpretation**  $V: M\_eclose \mathcal{V}$

<proof>

**lemmas** *bad\_M\_basic\_rules[simp del, rule del]* =  
 $V.cartprod\_closed$   $V.finite\_funspace\_closed$   $V.converse\_closed$   
 $V.list\_case'\_closed$   $V.pred\_closed$

**interpretation**  $V: M\_cardinal\_arith \mathcal{V}$

<proof>

**lemmas** *bad\_M\_cardinals\_rules[simp del, rule del]* =  
 $V.iterates\_closed$   $V.M\_nat$   $V.trancl\_closed$   $V.rvimage\_closed$

**interpretation**  $V:M\_cardinal\_arith\_jump \mathcal{V}$   
*<proof>*

**lemma**  $choice\_ax\_Universe: choice\_ax(\mathcal{V})$   
*<proof>*

**interpretation**  $V:M\_master \mathcal{V}$   
*<proof>*

**named\_theorems**  $V\_simps$

— To work systematically, ASCII versions of ”\_absolute” theorems as those below are preferable.

**lemma**  $eqpoll\_rel\_absolute[V\_simps]: x \approx^{\mathcal{V}} y \longleftrightarrow x \approx y$   
*<proof>*

**lemma**  $cardinal\_rel\_absolute[V\_simps]: |x|^{\mathcal{V}} = |x|$   
*<proof>*

**lemma**  $Card\_rel\_absolute[V\_simps]: Card^{\mathcal{V}}(a) \longleftrightarrow Card(a)$   
*<proof>*

**lemma**  $csucc\_rel\_absolute[V\_simps]: (a^+)^{\mathcal{V}} = a^+$   
*<proof>*

**lemma**  $function\_space\_rel\_absolute[V\_simps]: x \rightarrow^{\mathcal{V}} y = x \rightarrow y$   
*<proof>*

**lemma**  $cexp\_rel\_absolute[V\_simps]: x^{\uparrow y, \mathcal{V}} = x^{\uparrow y}$   
*<proof>*

**lemma**  $HAleph\_rel\_absolute[V\_simps]: HAleph\_rel(\mathcal{V}, a, b) = HAleph(a, b)$   
*<proof>*

**lemma**  $Aleph\_rel\_absolute[V\_simps]: Ord(x) \implies \aleph_x^{\mathcal{V}} = \aleph_x$   
*<proof>*

Example of absolute lemmas obtained from the relative versions. Note the *only* declarations

**lemma**  $Ord\_cardinal\_idem': Ord(A) \implies ||A|| = |A|$   
*<proof>*

**lemma**  $Aleph\_succ': Ord(\alpha) \implies \aleph_{succ(\alpha)} = \aleph_{\alpha}^+$   
*<proof>*

These two results are new, first obtained in relative form (not ported).

**lemma**  $csucc\_cardinal:$   
**assumes**  $Ord(\kappa)$  **shows**  $|\kappa|^+ = \kappa^+$

*<proof>*

**lemma** *csucc\_le\_mono*:  
 **assumes**  $\kappa \leq \nu$  **shows**  $\kappa^+ \leq \nu^+$   
 *<proof>*

Example of transferring results from a transitive model to  $\mathcal{V}$

**lemma** (in *M\_Perm*) *eqpoll\_rel\_transfer\_absolute*:  
 **assumes**  $M(A)$   $M(B)$   $A \approx^M B$   
 **shows**  $A \approx B$   
 *<proof>*

The “relationalized” *CH* with respect to  $\mathcal{V}$  corresponds to the real *CH*.

**lemma** *is\_ContHyp\_iff\_CH*:  $is\_ContHyp(\mathcal{V}) \longleftrightarrow ContHyp$   
 *<proof>*

**end**

## 33 Main definitions of the development

**theory** *Definitions\_Main*  
 **imports**  
 *Absolute\_Versions*  
**begin**

This theory gathers the main definitions of the *Transitive\_Models* session and the present one.

It might be considered as the bare minimum reading requisite to trust that our development indeed formalizes the theory of forcing. This should be mathematically clear since this is the only known method for obtaining proper extensions of ctms while preserving the ordinals.

The main theorem of this session and all of its relevant definitions appear in Section 33.4. The reader trusting all the libraries on which our development is based, might jump directly to Section 33.3, which treats relative cardinal arithmetic as implemented in *Transitive\_Models*. But in case one wants to dive deeper, the following sections treat some basic concepts of the ZF logic (Section 33.1) and in the ZF-Constructible library (Section 33.2) on which our definitions are built.

**declare** *[[show\_question\_marks=false]]*

### 33.1 ZF

For the basic logic ZF we restrict ourselves to just a few concepts.

**thm** *bij\_def[unfolded inj\_def surj\_def]*

$$\begin{aligned} \text{bij}(A, B) &\equiv \\ &\{f \in A \rightarrow B . \forall w \in A. \forall x \in A. f \text{ ` } w = f \text{ ` } x \longrightarrow w = x\} \cap \\ &\{f \in A \rightarrow B . \forall y \in B. \exists x \in A. f \text{ ` } x = y\} \end{aligned}$$

**thm** *eqpoll\_def*

$$A \approx B \equiv \exists f. f \in \text{bij}(A, B)$$

**thm** *Transset\_def*

$$\text{Transset}(i) \equiv \forall x \in i. x \subseteq i$$

**thm** *Ord\_def*

$$\text{Ord}(i) \equiv \text{Transset}(i) \wedge (\forall x \in i. \text{Transset}(x))$$

**thm** *lt\_def le\_iff*

$$\begin{aligned} i < j &\equiv i \in j \wedge \text{Ord}(j) \\ i \leq j &\longleftrightarrow i < j \vee i = j \wedge \text{Ord}(j) \end{aligned}$$

With the concepts of empty set and successor in place,

**lemma** *empty\_def'*:  $\forall x. x \notin 0$  *<proof>*

**lemma** *succ\_def'*:  $\text{succ}(i) = i \cup \{i\}$  *<proof>*

we can define the set of natural numbers  $\omega$ . In the sources, it is defined as a fixpoint, but here we just write its characterization as the first limit ordinal.

**thm** *Limit\_nat[unfolded Limit\_def] nat\_le\_Limit[unfolded Limit\_def]*

$$\begin{aligned} \text{Ord}(\omega) \wedge 0 < \omega \wedge (\forall y. y < \omega \longrightarrow \text{succ}(y) < \omega) \\ \text{Ord}(i) \wedge 0 < i \wedge (\forall y. y < i \longrightarrow \text{succ}(y) < i) \implies \omega \leq i \end{aligned}$$

Then, addition and predecessor on  $\omega$  are inductively characterized as follows:

**thm** *add\_0\_right add\_succ\_right pred\_0 pred\_succ\_eq*

$$\begin{aligned} m +_{\omega} \text{succ}(n) &= \text{succ}(m +_{\omega} n) \\ m \in \omega \implies m +_{\omega} 0 &= m \\ \text{pred}(0) &= 0 \\ \text{pred}(\text{succ}(y)) &= y \end{aligned}$$

Lists on a set  $A$  can be characterized by being recursively generated from the empty list  $[]$  and the operation  $Cons$  that adds a new element to the left end; the induction theorem for them shows that the characterization is “complete”.

**thm** *Nil Cons list.induct*

$$\begin{aligned} & [] \in list(A) \\ & \llbracket a \in A; l \in list(A) \rrbracket \implies Cons(a, l) \in list(A) \\ & \llbracket x \in list(A); P([]); \bigwedge a l. \llbracket a \in A; l \in list(A); P(l) \rrbracket \implies P(Cons(a, l)) \rrbracket \\ & \implies P(x) \end{aligned}$$

Length, concatenation, and  $n$ th element of lists are recursively characterized as follows.

**thm** *length.simps app.simps nth\_0 nth\_Cons*

$$\begin{aligned} length([]) &= 0 \\ length(Cons(a, l)) &= succ(length(l)) \\ [] @ ys &= ys \\ Cons(a, l) @ ys &= Cons(a, l @ ys) \\ nth(0, Cons(a, l)) &= a \\ n \in \omega \implies nth(succ(n), Cons(a, l)) &= nth(n, l) \end{aligned}$$

We have the usual Haskell-like notation for iterated applications of  $Cons$ :

**lemma** *Cons\_app*:  $[a,b,c] = Cons(a, Cons(b, Cons(c, [])))$  *<proof>*

Relative quantifiers restrict the range of the bound variable to a class  $M$  of type  $i \Rightarrow o$ ; that is, a truth-valued function with set arguments.

**lemma**  $\forall x[M]. P(x) \equiv \forall x. M(x) \longrightarrow P(x)$   
 $\exists x[M]. P(x) \equiv \exists x. M(x) \wedge P(x)$   
*<proof>*

Finally, a set can be viewed (“cast”) as a class using the following function of type  $i \Rightarrow i \Rightarrow o$ .

**thm** *setclass\_iff*

$$(\#\#A)(x) \longleftrightarrow x \in A$$

### 33.2 Relative concepts

A list of relative concepts (mostly from the ZF-Constructible library) follows next.

**thm** *big\_union\_def*

$$\text{big\_union}(M, A, z) \equiv \forall x[M]. x \in z \longleftrightarrow (\exists y[M]. y \in A \wedge x \in y)$$

**thm** *upair\_def*

$$\text{upair}(M, a, b, z) \equiv a \in z \wedge b \in z \wedge (\forall x[M]. x \in z \longrightarrow x = a \vee x = b)$$

**thm** *pair\_def*

$$\begin{aligned} \text{pair}(M, a, b, z) &\equiv \\ &\exists x[M]. \text{upair}(M, a, a, x) \wedge (\exists y[M]. \text{upair}(M, a, b, y) \wedge \text{upair}(M, x, y, z)) \end{aligned}$$

**thm** *successor\_def[unfolded is\_cons\_def union\_def]*

$$\begin{aligned} \text{successor}(M, a, z) &\equiv \\ &\exists x[M]. \text{upair}(M, a, a, x) \wedge (\forall xa[M]. xa \in z \longleftrightarrow xa \in x \vee xa \in a) \end{aligned}$$

**thm** *empty\_def*

$$\text{empty}(M, z) \equiv \forall x[M]. x \notin z$$

**thm** *transitive\_set\_def[unfolded subset\_def]*

$$\text{transitive\_set}(M, a) \equiv \forall x[M]. x \in a \longrightarrow (\forall xa[M]. xa \in x \longrightarrow xa \in a)$$

**thm** *ordinal\_def*

$$\begin{aligned} \text{ordinal}(M, a) &\equiv \\ &\text{transitive\_set}(M, a) \wedge (\forall x[M]. x \in a \longrightarrow \text{transitive\_set}(M, x)) \end{aligned}$$

**thm** *image\_def*

$$\begin{aligned} \text{image}(M, r, A, z) &\equiv \\ &\forall y[M]. y \in z \longleftrightarrow (\exists w[M]. w \in r \wedge (\exists x[M]. x \in A \wedge \text{pair}(M, x, y, w))) \end{aligned}$$

**thm** *fun\_apply\_def*

$is\_apply(M, f, x, y) \equiv$   
 $\exists xs[M].$   
 $\quad \exists fxs[M]. \text{upair}(M, x, x, xs) \wedge \text{image}(M, f, xs, fxs) \wedge \text{big\_union}(M, fxs, y)$

**thm** *is\_function\_def*

$is\_function(M, r) \equiv$   
 $\forall x[M].$   
 $\quad \forall y[M].$   
 $\quad \quad \forall y'[M].$   
 $\quad \quad \quad \forall p[M].$   
 $\quad \quad \quad \quad \forall p'[M].$   
 $\quad \quad \quad \quad \quad \text{pair}(M, x, y, p) \longrightarrow$   
 $\quad \quad \quad \quad \quad \text{pair}(M, x, y', p') \longrightarrow p \in r \longrightarrow p' \in r \longrightarrow y = y'$

**thm** *is\_relation\_def*

$is\_relation(M, r) \equiv \forall z[M]. z \in r \longrightarrow (\exists x[M]. \exists y[M]. \text{pair}(M, x, y, z))$

**thm** *is\_domain\_def*

$is\_domain(M, r, z) \equiv$   
 $\forall x[M]. x \in z \longleftrightarrow (\exists w[M]. w \in r \wedge (\exists y[M]. \text{pair}(M, x, y, w)))$

**thm** *typed\_function\_def*

$typed\_function(M, A, B, r) \equiv$   
 $is\_function(M, r) \wedge$   
 $is\_relation(M, r) \wedge$   
 $is\_domain(M, r, A) \wedge$   
 $(\forall u[M]. u \in r \longrightarrow (\forall x[M]. \forall y[M]. \text{pair}(M, x, y, u) \longrightarrow y \in B))$

**thm** *is\_function\_space\_def*[*unfolded is\_funspace\_def*]  
*function\_space\_rel\_def* *surjection\_def*

$is\_function\_space(M, A, B, fs) \equiv$   
 $M(fs) \wedge (\forall f[M]. f \in fs \longleftrightarrow typed\_function(M, A, B, f))$   
 $A \rightarrow^M B \equiv \text{THE } d. is\_function\_space(M, A, B, d)$   
 $surjection(M, A, B, f) \equiv$   
 $typed\_function(M, A, B, f) \wedge$   
 $(\forall y[M]. y \in B \longrightarrow (\exists x[M]. x \in A \wedge is\_apply(M, f, x, y)))$

Relative version of the *ZFC* axioms

**thm** *extensionality\_def*

$$\text{extensionality}(M) \equiv \forall x[M]. \forall y[M]. (\forall z[M]. z \in x \longleftrightarrow z \in y) \longrightarrow x = y$$

**thm** *foundation\_ax\_def*

$$\begin{aligned} \text{foundation\_ax}(M) &\equiv \\ \forall x[M]. (\exists y[M]. y \in x) &\longrightarrow (\exists y[M]. y \in x \wedge \neg (\exists z[M]. z \in x \wedge z \in y)) \end{aligned}$$

**thm** *upair\_ax\_def*

$$\text{upair\_ax}(M) \equiv \forall x[M]. \forall y[M]. \exists z[M]. \text{upair}(M, x, y, z)$$

**thm** *Union\_ax\_def*

$$\text{Union\_ax}(M) \equiv \forall x[M]. \exists z[M]. \text{big\_union}(M, x, z)$$

**thm** *power\_ax\_def*[*unfolded powerset\_def subset\_def*]

$$\text{power\_ax}(M) \equiv \forall x[M]. \exists z[M]. \forall xa[M]. xa \in z \longleftrightarrow (\forall xb[M]. xb \in xa \longrightarrow xb \in x)$$

**thm** *infinity\_ax\_def*

$$\begin{aligned} \text{infinity\_ax}(M) &\equiv \\ \exists I[M]. & \\ (\exists z[M]. \text{empty}(M, z) \wedge z \in I) \wedge & \\ (\forall y[M]. y \in I \longrightarrow (\exists sy[M]. \text{successor}(M, y, sy) \wedge sy \in I)) & \end{aligned}$$

**thm** *choice\_ax\_def*

$$\text{choice\_ax}(M) \equiv \forall x[M]. \exists a[M]. \exists f[M]. \text{ordinal}(M, a) \wedge \text{surjection}(M, a, x, f)$$

**thm** *separation\_def*

$$\text{separation}(M, P) \equiv \forall z[M]. \exists y[M]. \forall x[M]. x \in y \longleftrightarrow x \in z \wedge P(x)$$

**thm** *univalent\_def*

$univalent(M, A, P) \equiv$   
 $\forall x[M]. x \in A \longrightarrow (\forall y[M]. \forall z[M]. P(x, y) \wedge P(x, z) \longrightarrow y = z)$

**thm** *strong\_replacement\_def*

$strong\_replacement(M, P) \equiv$   
 $\forall A[M].$   
 $univalent(M, A, P) \longrightarrow (\exists Y[M]. \forall b[M]. b \in Y \longleftrightarrow (\exists x[M]. x \in A \wedge P(x, b)))$

Internalized formulas

“Codes” for formulas (as sets) are constructed from natural numbers using *Member*, *Equal*, *Nand*, and *Forall*.

**thm** *Member Equal Nand Forall formula.induct*

$\llbracket x \in \omega; y \in \omega \rrbracket \Longrightarrow \cdot x \in y \cdot \in formula$   
 $\llbracket x \in \omega; y \in \omega \rrbracket \Longrightarrow \cdot x = y \cdot \in formula$   
 $\llbracket p \in formula; q \in formula \rrbracket \Longrightarrow \cdot \neg(p \wedge q) \cdot \in formula$   
 $p \in formula \Longrightarrow (\cdot \forall p \cdot) \in formula$   
 $\llbracket x \in formula; \wedge x y. \llbracket x \in \omega; y \in \omega \rrbracket \Longrightarrow P(\cdot x \in y \cdot);$   
 $\wedge x y. \llbracket x \in \omega; y \in \omega \rrbracket \Longrightarrow P(\cdot x = y \cdot);$   
 $\wedge p q. \llbracket p \in formula; P(p); q \in formula; P(q) \rrbracket \Longrightarrow P(\cdot \neg(p \wedge q) \cdot);$   
 $\wedge p. \llbracket p \in formula; P(p) \rrbracket \Longrightarrow P(\cdot (\forall p) \cdot)$   
 $\Longrightarrow P(x)$

Definitions for the other connectives and the internal existential quantifier are also provided. For instance, negation:

**thm** *Neg\_def*

$\cdot \neg p \cdot \equiv \cdot \neg(p \wedge p) \cdot$

**thm** *arity.simps*

$arity(\cdot x \in y \cdot) = succ(x) \cup succ(y)$   
 $arity(\cdot x = y \cdot) = succ(x) \cup succ(y)$   
 $arity(\cdot \neg(p \wedge q) \cdot) = arity(p) \cup arity(q)$   
 $arity(\cdot (\forall p) \cdot) = pred(arity(p))$

We have the satisfaction relation between  $\in$ -models and first order formulas (given a “environment” list representing the assignment of free variables),

**thm** *mem\_iff\_sats equal\_iff\_sats sats\_Nand\_iff sats\_Forall\_iff*

$$\begin{aligned}
& \llbracket nth(i, env) = x; nth(j, env) = y; env \in list(A) \rrbracket \\
& \implies x \in y \longleftrightarrow A, env \models \cdot i \in j \cdot \\
& \llbracket nth(i, env) = x; nth(j, env) = y; env \in list(A) \rrbracket \\
& \implies x = y \longleftrightarrow A, env \models \cdot i = j \cdot \\
& env \in list(A) \implies (A, env \models \cdot \neg(p \wedge q) \cdot) \longleftrightarrow \neg((A, env \models p) \wedge (A, env \models q)) \\
& env \in list(A) \implies (A, env \models (\cdot \forall p \cdot)) \longleftrightarrow (\forall x \in A. A, Cons(x, env) \models p)
\end{aligned}$$

as well as the satisfaction of an arbitrary set of sentences.

**thm** *satT\_def*

$$A \models \Phi \equiv \forall \varphi \in \Phi. A, [] \models \varphi$$

The internalized (viz. as elements of the set *formula*) version of the axioms follow next.

**thm** *ZF\_union\_iff\_sats ZF\_power\_iff\_sats ZF\_pairing\_iff\_sats*  
*ZF\_foundation\_iff\_sats ZF\_extensionality\_iff\_sats*  
*ZF\_infinity\_iff\_sats sats\_ZF\_separation\_fm\_iff*  
*sats\_ZF\_replacement\_fm\_iff ZF\_choice\_iff\_sats*

$$\begin{aligned}
& Union\_ax(\#\#A) \longleftrightarrow A, [] \models \cdot Union\ Ax \cdot \\
& power\_ax(\#\#A) \longleftrightarrow A, [] \models \cdot Powerset\ Ax \cdot \\
& upair\_ax(\#\#A) \longleftrightarrow A, [] \models \cdot Pairing \cdot \\
& foundation\_ax(\#\#A) \longleftrightarrow A, [] \models \cdot Foundation \cdot \\
& extensionality(\#\#A) \longleftrightarrow A, [] \models \cdot Extensionality \cdot \\
& infinity\_ax(\#\#A) \longleftrightarrow A, [] \models \cdot Infinity \cdot \\
& \varphi \in formula \implies \\
& (M, [] \models \cdot Separation(\varphi) \cdot) \longleftrightarrow \\
& (\forall env \in list(M). \\
& \quad arity(\varphi) \leq 1 +_{\omega} length(env) \longrightarrow separation(\#\#M, \lambda x. M, [x] @ env \models \varphi)) \\
& \varphi \in formula \implies \\
& (M, [] \models \cdot Replacement(\varphi) \cdot) \longleftrightarrow (\forall env. replacement\_assm(M, env, \varphi)) \\
& choice\_ax(\#\#A) \longleftrightarrow A, [] \models \cdot AC \cdot
\end{aligned}$$

Above, we use the following:

**thm** *replacement\_assm\_def*

$$\begin{aligned}
& replacement\_assm(M, env, \varphi) \equiv \\
& \varphi \in formula \longrightarrow \\
& env \in list(M) \longrightarrow \\
& arity(\varphi) \leq 2 +_{\omega} length(env) \longrightarrow \\
& strong\_replacement(\#\#M, \lambda x y. M, [x, y] @ env \models \varphi)
\end{aligned}$$

Finally, the axiom sets are defined as follows.

**thm** *ZF\_fin\_def ZF\_schemes\_def Zermelo\_fms\_def ZC\_def ZF\_def ZFC\_def*

$ZF\_fin \equiv$   
 $\{\cdot Extensionality \cdot, \cdot Foundation \cdot, \cdot Pairing \cdot, \cdot Union Ax \cdot, \cdot Infinity \cdot,$   
 $\cdot Powerset Ax \cdot\}$   
 $ZF\_schemes \equiv$   
 $\{\cdot Separation(p) \cdot \ . \ p \in formula\} \cup \{\cdot Replacement(p) \cdot \ . \ p \in formula\}$   
 $\cdot Z \cdot \equiv ZF\_fin \cup \{\cdot Separation(p) \cdot \ . \ p \in formula\}$   
 $ZC \equiv \cdot Z \cdot \cup \{\cdot AC \cdot\}$   
 $ZF \equiv ZF\_schemes \cup ZF\_fin$   
 $ZFC \equiv ZF \cup \{\cdot AC \cdot\}$

### 33.3 Relativization of infinitary arithmetic

In order to state the defining property of the relative equipotence relation, we work under the assumptions of the locale  $M\_cardinals$ . They comprise a finite set of instances of Separation and Replacement to prove closure properties of the transitive class  $M$ .

**lemma** (in  $M\_cardinals$ ) *eqpoll\_def'*:  
**assumes**  $M(A) \ M(B)$  **shows**  $A \approx^M B \longleftrightarrow (\exists f[M]. f \in bij(A,B))$   
*<proof>*

Below,  $\mu$  denotes the minimum operator on the ordinals.

**lemma** *cardinalities\_defs*:  
**fixes**  $M::i \Rightarrow o$   
**shows**  
 $|A|^M \equiv \mu i. M(i) \wedge i \approx^M A$   
 $Card^M(\alpha) \equiv \alpha = |\alpha|^M$   
 $\kappa^{\uparrow \nu, M} \equiv |\nu \rightarrow^M \kappa|^M$   
 $(\kappa^+)^M \equiv \mu x. M(x) \wedge Card^M(x) \wedge \kappa < x$   
*<proof>*

**context**  $M\_aleph$   
**begin**

Analogous to the previous Lemma *eqpoll\_def'*, we are now under the assumptions of the locale  $M\_aleph$ . The axiom instances included are sufficient to state and prove the defining properties of the relativized *Aleph* function (in particular, the required ability to perform transfinite recursions).

**thm** *Aleph\_rel\_zero Aleph\_rel\_succ Aleph\_rel\_limit*

$\aleph_0^M = \omega$   
 $\llbracket Ord(\alpha); M(\alpha) \rrbracket \implies \aleph_{succ(\alpha)}^M = (\aleph_\alpha^{M+})^M$   
 $\llbracket Limit(\alpha); M(\alpha) \rrbracket \implies \aleph_\alpha^M = (\bigcup_{j \in \alpha} \aleph_j^M)$

**end** —  $M\_aleph$

**lemma** *ContHyp\_rel\_def'*:  
**fixes**  $N::i\Rightarrow o$   
**shows**  
 $CH^N \equiv \aleph_1^N = 2^{\uparrow\aleph_0^{N,N}}$   
*<proof>*

Under appropriate hypotheses (this time, from the locale *M\_ZF\_library*),  $CH^M$  is equivalent to its fully relational version *is\_ContHyp*. As a sanity check, we see that if the transitive class is indeed  $\mathcal{V}$ , we recover the original *CH*.

**thm** *M\_ZF\_library.is\_ContHyp\_iff\_is\_ContHyp\_iff\_CH[unfolded ContHyp\_def]*

$$\begin{aligned} M\_ZF\_library(M) &\implies is\_ContHyp(M) \longleftrightarrow CH^M \\ is\_ContHyp(\mathcal{V}) &\longleftrightarrow \aleph_1 = 2^{\uparrow\aleph_0} \end{aligned}$$

In turn, the fully relational version evaluated on a nonempty transitive  $A$  is equivalent to the satisfaction of the first-order formula  $\cdot CH \cdot$ .

**thm** *is\_ContHyp\_iff\_sats*

$$\llbracket env \in list(A); 0 \in A \rrbracket \implies is\_ContHyp(\#\#A) \longleftrightarrow A, env \models \cdot CH \cdot$$

### 33.4 Forcing

Our first milestone was to obtain a proper extension using forcing. Its original proof didn't required the previous developments involving the relativization of material on cardinal arithmetic. Now it is derived from a stronger result, namely *extensions\_of\_ctms* below.

**thm** *extensions\_of\_ctms\_ZF*

$$\begin{aligned} \llbracket M \approx \omega; Transset(M); M \models ZF \rrbracket \\ \implies \exists N. M \subseteq N \wedge \\ N \approx \omega \wedge \\ Transset(N) \wedge \\ N \models ZF \wedge \\ M \neq N \wedge (\forall \alpha. Ord(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N) \wedge ((M, [] \models \cdot AC \cdot) \longrightarrow \\ N \models ZFC) \end{aligned}$$

We can finally state our main results, namely, the existence of models for  $ZFC + CH$  and  $ZFC + \neg CH$  under the assumption of a ctm of  $ZFC$ .

**thm** *ctm\_ZFC\_imp\_ctm\_not\_CH*

$$\begin{aligned} & \llbracket M \approx \omega; \text{Transset}(M); M \models \text{ZFC} \rrbracket \\ \implies & \exists N. M \subseteq N \wedge \\ & N \approx \omega \wedge \\ & \text{Transset}(N) \wedge N \models \text{ZFC} \cup \{\neg \text{CH}\} \wedge (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \\ & \alpha \in N) \end{aligned}$$

**thm** *ctm\_ZFC\_imp\_ctm\_CH*

$$\begin{aligned} & \llbracket M \approx \omega; \text{Transset}(M); M \models \text{ZFC} \rrbracket \\ \implies & \exists N. M \subseteq N \wedge \\ & N \approx \omega \wedge \\ & \text{Transset}(N) \wedge N \models \text{ZFC} \cup \{\text{CH}\} \wedge (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \\ & \in N) \end{aligned}$$

These results can be strengthened by enumerating six finite sets of replacement instances which are sufficient to develop forcing and for the construction of the aforementioned models: *instances1\_fms* through *instances3\_fms*, *instances\_ground\_fms*, and *instances\_ground\_notCH\_fms*, which are then collected into the 31-element set *overhead\_notCH*. For example, we have:

**thm** *instances1\_fms\_def*

$$\begin{aligned} \text{instances1\_fms} & \equiv \\ & \{\text{eclose\_closed\_fm}, \text{eclose\_abs\_fm}, \text{wfrec\_rank\_fm}, \text{transrec\_VFrom\_fm}\} \end{aligned}$$

**thm** *overhead\_def overhead\_notCH\_def*

$$\begin{aligned} \text{overhead} & \equiv \text{instances1\_fms} \cup \text{instances\_ground\_fms} \\ \text{overhead\_notCH} & \equiv \\ & \text{overhead} \cup \text{instances2\_fms} \cup \text{instances3\_fms} \cup \text{instances\_ground\_notCH\_fms} \\ \text{overhead\_CH} & \equiv \text{overhead\_notCH} \cup \{\text{dc\_abs\_fm}\} \end{aligned}$$

One further instance is needed to force *CH*, with a total count of 32 instances:

**thm** *overhead\_CH\_def*

$$\text{overhead\_CH} \equiv \text{overhead\_notCH} \cup \{\text{dc\_abs\_fm}\}$$

**thm** *extensions\_of\_ctms*

$$\begin{aligned}
& \llbracket M \approx \omega; \text{Transset}(M); M \models \cdot Z \cdot \cup \{\cdot \text{Replacement}(p) \cdot \mid p \in \text{overhead}\}; \\
& \Phi \subseteq \text{formula}; M \models \{\cdot \text{Replacement}(\text{ground\_repl\_fm}(\varphi)) \cdot \mid \varphi \in \Phi\} \rrbracket \\
\implies & \exists N. M \subseteq N \wedge \\
& N \approx \omega \wedge \\
& \text{Transset}(N) \wedge \\
& M \neq N \wedge \\
& (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N) \wedge \\
& ((M, [] \models \cdot AC \cdot) \longrightarrow N, [] \models \cdot AC \cdot) \wedge \\
& N \models \cdot Z \cdot \cup \{\cdot \text{Replacement}(\varphi) \cdot \mid \varphi \in \Phi\}
\end{aligned}$$

**thm** *ctm\_of\_not\_CH*

$$\begin{aligned}
& \llbracket M \approx \omega; \text{Transset}(M); M \models ZC \cup \{\cdot \text{Replacement}(p) \cdot \mid p \in \text{overhead\_notCH}\}; \\
& \Phi \subseteq \text{formula}; M \models \{\cdot \text{Replacement}(\text{ground\_repl\_fm}(\varphi)) \cdot \mid \varphi \in \Phi\} \rrbracket \\
\implies & \exists N. M \subseteq N \wedge \\
& N \approx \omega \wedge \\
& \text{Transset}(N) \wedge \\
& N \models ZC \cup \{\cdot \neg \cdot CH \cdot\} \cup \{\cdot \text{Replacement}(\varphi) \cdot \mid \varphi \in \Phi\} \wedge \\
& (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N)
\end{aligned}$$

**thm** *ctm\_of\_CH*

$$\begin{aligned}
& \llbracket M \approx \omega; \text{Transset}(M); M \models ZC \cup \{\cdot \text{Replacement}(p) \cdot \mid p \in \text{overhead\_CH}\}; \\
& \Phi \subseteq \text{formula}; M \models \{\cdot \text{Replacement}(\text{ground\_repl\_fm}(\varphi)) \cdot \mid \varphi \in \Phi\} \rrbracket \\
\implies & \exists N. M \subseteq N \wedge \\
& N \approx \omega \wedge \\
& \text{Transset}(N) \wedge \\
& N \models ZC \cup \{\cdot CH \cdot\} \cup \{\cdot \text{Replacement}(\varphi) \cdot \mid \varphi \in \Phi\} \wedge \\
& (\forall \alpha. \text{Ord}(\alpha) \longrightarrow \alpha \in M \longleftrightarrow \alpha \in N)
\end{aligned}$$

In the above three statements, the function *ground\_repl\_fm* takes an element  $\varphi$  of *formula* and returns the replacement instance in the ground model that produces the  $\varphi$ -replacement instance in the generic extension. The next result is stated in the context *G\_generic1*, which assumes the existence of a generic filter.

**context** *G\_generic1*  
**begin**

**thm** *sats\_ground\_repl\_fm\_imp\_sats\_ZF\_replacement\_fm*

$$\begin{aligned}
& \llbracket \varphi \in \text{formula}; M, [] \models \cdot \text{Replacement}(\text{ground\_repl\_fm}(\varphi)) \cdot \rrbracket \\
\implies & M[G], [] \models \cdot \text{Replacement}(\varphi) \cdot
\end{aligned}$$

**end** — *G\_generic1*

**end**

## 34 Some demonstrations

```

theory Demonstrations
  imports
    Definitions_Main
begin

```

The following theory is only intended to explore some details of the formalization and to show the appearance of relevant internalized formulas. It is **not** intended as the entry point of the session. For that purpose, consult *Independence\_CH.Definitions\_Main*

The snippet (by M. Pagano) commented out below outputs a directed graph picturing the locale structure.

```

locale Demo = M_trivial + M_AC +
  fixes  $t_1 t_2$ 
  assumes
     $ts\_in\_nat[simp]: t_1 \in \omega \ t_2 \in \omega$ 
  and
     $power\_infty: power\_ax(M) \ M(\omega)$ 
begin

```

The next fake lemma is intended to explore the instances of the axiom schemes that are needed to build our forcing models. They are categorized as plain replacements (using *strong\_replacement*), “lambda-replacements” using a higher order function, replacements to perform transfinite and general well-founded recursion (using *transrec\_replacement* and *wfrec\_replacement* respectively) and for the construction of fixpoints (using *iterates\_replacement*). Lastly, separations instances.

```

lemma
  assumes
    sorried_replacements:
       $\bigwedge P. strong\_replacement(M,P)$ 
       $\bigwedge F. lam\_replacement(M,F)$ 
       $\bigwedge Q \ S. iterates\_replacement(M,Q,S)$ 
       $\bigwedge Q \ S. wfrec\_replacement(M,Q,S)$ 
       $\bigwedge Q \ S. transrec\_replacement(M,Q,S)$ 
  and
    sorried_separations:  $\bigwedge Q. separation(M,Q)$ 
  shows
     $M\_master(M)$ 
    <proof>
no_notation mem (infixl  $\langle \in \rangle$  50)
no_notation conj (infixr  $\langle \wedge \rangle$  35)
no_notation disj (infixr  $\langle \vee \rangle$  30)
no_notation iff (infixr  $\langle \longleftrightarrow \rangle$  25)
no_notation imp (infixr  $\langle \longrightarrow \rangle$  25)
no_notation not ( $\langle \neg \_ \rangle$  [40] 40)

```

**no\_notation** *All* ( $\langle'(\forall \_)\rangle$ )  
**no\_notation** *Ex* ( $\langle'(\exists \_)\rangle$ )  
  
**no\_notation** *Member* ( $\langle \_ \in / \_ \rangle$ )  
**no\_notation** *Equal* ( $\langle \_ = / \_ \rangle$ )  
**no\_notation** *Nand* ( $\langle \_ \neg'(\_ \wedge / \_ \wedge) \rangle$ )  
**no\_notation** *And* ( $\langle \_ \wedge / \_ \rangle$ )  
**no\_notation** *Or* ( $\langle \_ \vee / \_ \rangle$ )  
**no\_notation** *Iff* ( $\langle \_ \leftrightarrow / \_ \rangle$ )  
**no\_notation** *Implies* ( $\langle \_ \rightarrow / \_ \rangle$ )  
**no\_notation** *Neg* ( $\langle \_ \neg \_ \rangle$ )  
**no\_notation** *Forall* ( $\langle'(\forall (\_))\rangle$ )  
**no\_notation** *Exists* ( $\langle'(\exists (\_))\rangle$ )

**notation** *Member* (**infixl**  $\langle \in \rangle$  50)  
**notation** *Equal* (**infixl**  $\langle \equiv \rangle$  50)  
**notation** *Nand* ( $\langle \_ \neg'(\_ \wedge / \_ \wedge) \rangle$ )  
**notation** *And* (**infixr**  $\langle \wedge \rangle$  35)  
**notation** *Or* (**infixr**  $\langle \vee \rangle$  30)  
**notation** *Iff* (**infixr**  $\langle \longleftrightarrow \rangle$  25)  
**notation** *Implies* (**infixr**  $\langle \longrightarrow \rangle$  25)  
**notation** *Neg* ( $\langle \_ \neg \_ \rangle$  [40] 40)  
**notation** *Forall* ( $\langle'(\forall \_)\rangle$ )  
**notation** *Exists* ( $\langle'(\exists \_)\rangle$ )

**lemma** *forces*( $t_1 \in t_2$ ) = ( $0 \in 1 \wedge \text{forces\_mem\_fm}(1, 2, 0, t_1 + \omega 4, t_2 + \omega 4)$ )  
*<proof>*

**definition** *forces\_0\_mem\_1* **where** *forces\_0\_mem\_1*  $\equiv$  *forces\_mem\_fm*(1,2,0, $t_1 + \omega 4$ , $t_2 + \omega 4$ )

**thm** *forces\_0\_mem\_1\_def* [  
*unfolded frc\_at\_fm\_def ftype\_fm\_def*  
*name1\_fm\_def name2\_fm\_def snd\_snd\_fm\_def hcomp\_fm\_def*  
*ecloseN\_fm\_def eclose\_n1\_fm\_def eclose\_n2\_fm\_def*  
*is\_eclose\_fm\_def mem\_eclose\_fm\_def eclose\_n\_fm\_def*  
*is\_If\_fm\_def least\_fm\_def Replace\_fm\_def Collect\_fm\_def*  
*fm\_definitions,simplified*]

**named\_theorems** *incr\_bv\_new\_simps*

**schematic\_goal** *incr\_bv\_Neg*:  
 $\text{mem}(n, \omega) \implies \text{mem}(\varphi, \text{formula}) \implies \text{incr\_bv}(\text{Neg}(\varphi))'n = ?x$   
*<proof>*

**schematic\_goal** *incr\_bv\_Exists* [*incr\_bv\_new\_simps*]:

$mem(n,\omega) \implies mem(\varphi,formula) \implies incr\_bv(Exists(\varphi)) 'n = ?x$   
*<proof>*

**end** — *Demo*

**end**

## References

- [1] E. GUNTHER, M. PAGANO, P. SÁNCHEZ TERRAF, First steps towards a formalization of forcing, in: Proceedings of the 13th Workshop on Logical and Semantic Frameworks with Applications, LSFA 2018, Fortaleza, Brazil, September 26-28, 2018, pp. 119–136 (2018).
- [2] E. GUNTHER, M. PAGANO, P. SÁNCHEZ TERRAF, Mechanization of Separation in Generic Extensions, *arXiv e-prints* **1901.03313** (2019).
- [3] E. GUNTHER, M. PAGANO, P. SÁNCHEZ TERRAF, Formalization of Forcing in Isabelle/ZF, arXiv e-prints, in: N. Peltier, V. Sofronie-Stokkermans (Eds.), Automated Reasoning. 10th International Joint Conference, IJCAR 2020, Paris, France, July 1–4, 2020, Proceedings, Part II, Lecture Notes in Artificial Intelligence **12167**, Springer International Publishing: 221–235 (2020).
- [4] L.C. PAULSON, K. GRABCZEWSKI, Mechanizing set theory, *J. Autom. Reasoning* **17**: 291–323 (1996).