

# Equivalents of the Axiom of Choice

Krzysztof Grabczewski

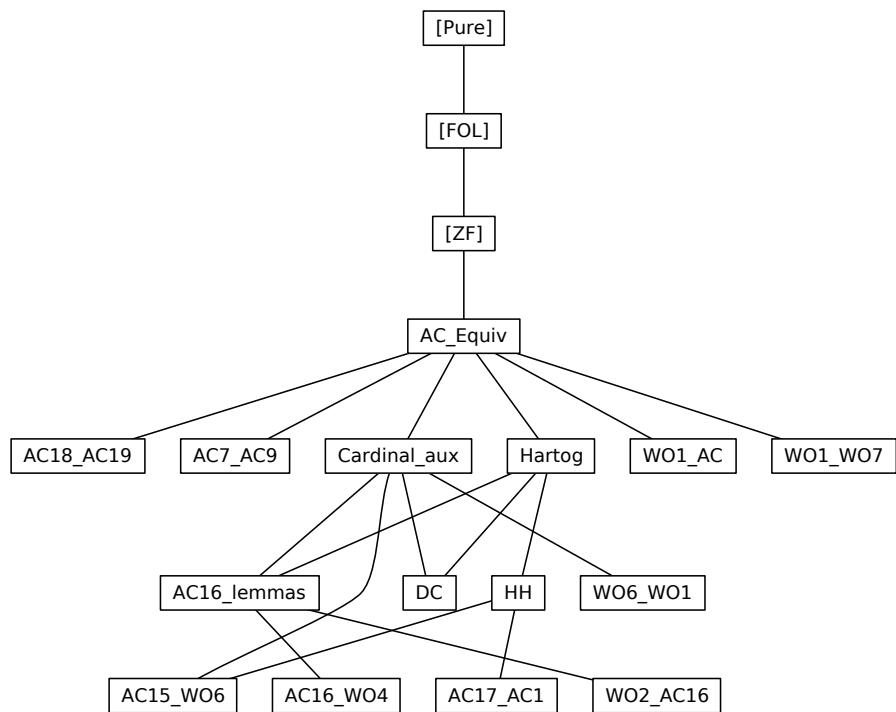
February 20, 2021

## Abstract

This development [1] proves the equivalence of seven formulations of the well-ordering theorem and twenty formulations of the axiom of choice. It formalizes the first two chapters of the monograph *Equivalents of the Axiom of Choice* by Rubin and Rubin [2]. Some of this material involves extremely complex techniques.

## Contents

0.1	Lemmas useful in each of the three proofs . . . . .	23
0.2	Lemmas used in the proofs of AC1 == $\_$ WO2 and AC17 == $\_$ AC1 . . . . .	24
0.3	The proof of AC1 == $\_$ WO2 . . . . .	25



```

theory AC_Equiv
imports ZF
begin

definition
"W01 == ∀ A. ∃ R. well_ord(A,R)"

definition
"W02 == ∀ A. ∃ a. Ord(a) & A≈a"

definition
"W03 == ∀ A. ∃ a. Ord(a) & (∃ b. b ⊆ a & A≈b)"

definition
"W04(m) == ∀ A. ∃ a f. Ord(a) & domain(f)=a &
                           (⋃ b<a. f`b) = A & (∀ b<a. f`b ⪻ m)"

definition
"W05 == ∃ m ∈ nat. 1≤m & W04(m)"

definition
"W06 == ∀ A. ∃ m ∈ nat. 1≤m & (∃ a f. Ord(a) & domain(f)=a
                           & (⋃ b<a. f`b) = A & (∀ b<a. f`b ⪻ m))"

definition
"W07 == ∀ A. Finite(A) ↔ (∀ R. well_ord(A,R) → well_ord(A,converse(R)))"

definition
"W08 == ∀ A. (∃ f. f ∈ (Π X ∈ A. X)) → (∃ R. well_ord(A,R))"

definition
pairwise_disjoint :: "i => o" where
  "pairwise_disjoint(A) == ∀ A1 ∈ A. ∀ A2 ∈ A. A1 ∩ A2 ≠ 0 → A1=A2"

definition
sets_of_size_between :: "[i, i, i] => o" where
  "sets_of_size_between(A,m,n) == ∀ B ∈ A. m ⪻ B & B ⪻ n"

definition
"AC0 == ∀ A. ∃ f. f ∈ (Π X ∈ Pow(A)-{0}. X)"

```

**definition**  
 $"AC1 == \forall A. 0 \notin A \longrightarrow (\exists f. f \in (\prod X \in A. X))"$

**definition**  
 $"AC2 == \forall A. 0 \notin A \& pairwise\_disjoint(A) \longrightarrow (\exists C. \forall B \in A. \exists y. B \cap C = \{y\})"$

**definition**  
 $"AC3 == \forall A B. \forall f \in A \rightarrow B. \exists g. g \in (\prod x \in \{a \in A. f'a \neq 0\}. f'x)"$

**definition**  
 $"AC4 == \forall R A B. (R \subseteq A * B \longrightarrow (\exists f. f \in (\prod x \in domain(R). R'^{\{x\}})))"$

**definition**  
 $"AC5 == \forall A B. \forall f \in A \rightarrow B. \exists g \in range(f) \rightarrow A. \forall x \in domain(g). f'(g'x) = x"$

**definition**  
 $"AC6 == \forall A. 0 \notin A \longrightarrow (\prod B \in A. B) \neq 0"$

**definition**  
 $"AC7 == \forall A. 0 \notin A \& (\forall B1 \in A. \forall B2 \in A. B1 \approx B2) \longrightarrow (\prod B \in A. B) \neq 0"$

**definition**  
 $"AC8 == \forall A. (\forall B \in A. \exists B1 B2. B = \langle B1, B2 \rangle \& B1 \approx B2) \longrightarrow (\exists f. \forall B \in A. f'B \in bij(fst(B), snd(B)))"$

**definition**  
 $"AC9 == \forall A. (\forall B1 \in A. \forall B2 \in A. B1 \approx B2) \longrightarrow (\exists f. \forall B1 \in A. \forall B2 \in A. f' \langle B1, B2 \rangle \in bij(B1, B2))"$

**definition**  
 $"AC10(n) == \forall A. (\forall B \in A. \neg Finite(B)) \longrightarrow (\exists f. \forall B \in A. (pairwise\_disjoint(f'B) \& sets\_of\_size\_between(f'B, 2, succ(n)) \& \bigcup (f'B) = B))"$

**definition**  
 $"AC11 == \exists n \in nat. 1 \leq n \& AC10(n)"$

**definition**  
 $"AC12 == \forall A. (\forall B \in A. \neg Finite(B)) \longrightarrow (\exists n \in nat. 1 \leq n \& (\exists f. \forall B \in A. (pairwise\_disjoint(f'B) \& sets\_of\_size\_between(f'B, 2, succ(n)) \& \bigcup (f'B) = B)))"$

**definition**  
 $"AC13(m) == \forall A. 0 \notin A \longrightarrow (\exists f. \forall B \in A. f'B \neq 0 \& f'B \subseteq B \& f'B \lesssim m)"$

**definition**

```

"AC14 == ∃ m ∈ nat. 1 ≤ m & AC13(m)"

definition
"AC15 == ∀ A. 0 ∉ A —>
(∃ m ∈ nat. 1 ≤ m & (∃ f. ∀ B ∈ A. f'B ≠ 0 & f'B ⊆ B & f'B
≤ m))"

definition
"AC16(n, k) ==
∀ A. ~Finite(A) —>
(∃ T. T ⊆ {X ∈ Pow(A). X ≈ succ(n)} &
(∀ X ∈ {X ∈ Pow(A). X ≈ succ(k)}. ∃! Y. Y ∈ T & X ⊆ Y))"

definition
"AC17 == ∀ A. ∀ g ∈ (Pow(A)-{0} → A) → Pow(A)-{0}.
∃ f ∈ Pow(A)-{0} → A. f'(g'f) ∈ g'f"

locale AC18 =
assumes AC18: "A ≠ 0 & (∀ a ∈ A. B(a) ≠ 0) —>
((∩ a ∈ A. ∪ b ∈ B(a). X(a, b)) =
(∪ f ∈ (Π a ∈ A. B(a). ∩ a ∈ A. X(a, f'a)))"
— AC18 cannot be expressed within the object-logic

definition
"AC19 == ∀ A. A ≠ 0 & 0 ∉ A —> ((∩ a ∈ A. ∪ b ∈ a. b) =
(∪ f ∈ (Π B ∈ A. B). ∩ a ∈ A. f'a))"

lemma rvimage_id: "rvimage(A, id(A), r) = r ∩ A * A"
⟨proof⟩

lemma ordertype_Int:
"well_ord(A, r) ==> ordertype(A, r ∩ A * A) = ordertype(A, r)"
⟨proof⟩

lemma lam_sing_bij: "(λx ∈ A. {x}) ∈ bij(A, {{x}. x ∈ A})"
⟨proof⟩

lemma inj_strengthen_type:
"[| f ∈ inj(A, B); !!a. a ∈ A ==> f'a ∈ C |] ==> f ∈ inj(A, C)"
⟨proof⟩

```

```

lemma ex1_two_eq: "[| ∃ ! x. P(x); P(x); P(y) |] ==> x=y"
⟨proof⟩

lemma first_in_B:
  "[| well_ord(∪(A), r); 0 ∉ A; B ∈ A |] ==> (THE b. first(b, B, r))
  ∈ B"
⟨proof⟩

lemma ex_choice_fun: "[| well_ord(∪(A), R); 0 ∉ A |] ==> ∃ f. f ∈ (∏ X
  ∈ A. X)"
⟨proof⟩

lemma ex_choice_fun_Pow: "well_ord(A, R) ==> ∃ f. f ∈ (∏ X ∈ Pow(A)-{0}.
  X)"
⟨proof⟩

lemma lepoll_m_imp_domain_lepoll_m:
  "[| m ∈ nat; u ⪻ m |] ==> domain(u) ⪻ m"
⟨proof⟩

lemma rel_domain_ex1:
  "[| succ(m) ⪻ domain(r); r ⪻ succ(m); m ∈ nat |] ==> function(r)"
⟨proof⟩

lemma rel_is_fun:
  "[| succ(m) ⪻ domain(r); r ⪻ succ(m); m ∈ nat;
  r ⊆ A*B; A=domain(r) |] ==> r ∈ A->B"
⟨proof⟩

end

theory Cardinal_aux imports AC_Equiv begin

```

```
lemma Diff_lepoll: "[| A ≲ succ(m); B ⊆ A; B ≠ 0 |] ==> A - B ≲ m"
⟨proof⟩
```

```
lemma lepoll_imp_ex_le_eqpoll:
  "[| A ≲ i; Ord(i) |] ==> ∃ j. j ≤ i & A ≈ j"
⟨proof⟩
```

```
lemma lesspoll_imp_ex_lt_eqpoll:
  "[| A < i; Ord(i) |] ==> ∃ j. j < i & A ≈ j"
⟨proof⟩
```

```
lemma Un_eqpoll_Inf_Ord:
  assumes A: "A ≈ i" and B: "B ≈ i" and NFI: "¬ Finite(i)" and i:
  "Ord(i)"
  shows "A ∪ B ≈ i"
⟨proof⟩
```

```
schematic_goal paired_bij: "?f ∈ bij({{y,z}. y ∈ x}, x)"
⟨proof⟩
```

```
lemma paired_eqpoll: "{{y,z}. y ∈ x} ≈ x"
⟨proof⟩
```

```
lemma ex_eqpoll_disjoint: "∃ B. B ≈ A & B ∩ C = 0"
⟨proof⟩
```

```
lemma Un_lepoll_Inf_Ord:
  "[| A ≲ i; B ≲ i; ¬ Finite(i); Ord(i) |] ==> A ∪ B ≲ i"
⟨proof⟩
```

```
lemma Least_in_Ord: "[| P(i); i ∈ j; Ord(j) |] ==> (μ i. P(i)) ∈ j"
⟨proof⟩
```

```
lemma Diff_first_lepoll:
  "[| well_ord(x,r); y ⊆ x; y ≲ succ(n); n ∈ nat |]
  ==> y - {THE b. first(b,y,r)} ≲ n"
⟨proof⟩
```

```

lemma UN_subset_split:
  " $(\bigcup_{x \in X} P(x)) \subseteq (\bigcup_{x \in X} P(x) - Q(x)) \cup (\bigcup_{x \in X} Q(x))$ "
⟨proof⟩

lemma UN_sing_lepoll: " $\text{Ord}(a) \Rightarrow (\bigcup_{x \in a. \{P(x)\}} \lesssim a)$ "
⟨proof⟩

lemma UN_fun_lepoll_lemma [rule_format]:
  " $\forall T R. \forall a. \forall n. \text{well\_ord}(T, R); \text{Finite}(a); \text{Ord}(a); n \in \text{nat} \Rightarrow$ 
    $\Rightarrow \forall f. (\forall b \in a. f'b \lesssim n \& f'b \subseteq T) \rightarrow (\bigcup_{b \in a. f'b} \lesssim a)$ "
⟨proof⟩

lemma UN_fun_lepoll:
  " $\forall T R. \forall a. \forall n. \text{well\_ord}(T, R); \text{Finite}(a); \text{Ord}(a); n \in \text{nat} \Rightarrow$ 
    $\Rightarrow (\bigcup_{b \in a. f'b} \lesssim a)$ "
⟨proof⟩

lemma UN_lepoll:
  " $\forall T R. \forall a. \forall n. \text{well\_ord}(T, R); \text{Finite}(a); \text{Ord}(a); n \in \text{nat} \Rightarrow$ 
    $\Rightarrow (\bigcup_{b \in a. F(b)} \lesssim a)$ "
⟨proof⟩

lemma UN_eq_UN_Diffs:
  " $\text{Ord}(a) \Rightarrow (\bigcup_{b \in a. F(b)} = (\bigcup_{b \in a. F(b)} - (\bigcup_{c \in b. F(c))))$ "
⟨proof⟩

lemma lepoll_imp_eqpoll_subset:
  " $a \lesssim X \Rightarrow \exists Y. Y \subseteq X \& a \approx Y$ "
⟨proof⟩

lemma Diff_lesspoll_eqpoll_Card_lemma:
  " $\forall A a. \text{Finite}(a); \text{Card}(a); B \prec a; A-B \prec a \Rightarrow P$ "
⟨proof⟩

lemma Diff_lesspoll_eqpoll_Card:
  " $\forall A a. \text{Finite}(a); \text{Card}(a); B \prec a \Rightarrow A - B \approx a$ "
⟨proof⟩

end

theory W06_W01
imports Cardinal_aux
begin

```

**definition**

```
NN :: "i => i" where
"NN(y) == {m ∈ nat. ∃a. ∃f. Ord(a) & domain(f)=a &
(⋃ b<a. f'b) = y & (∀ b<a. f'b ⪯ m)}"
```

**definition**

```
uu :: "[i, i, i] => i" where
"uu(f, beta, gamma, delta) == (f'beta * f'gamma) ∩ f'delta"
```

**definition**

```
vv1 :: "[i, i, i] => i" where
"vv1(f, m, b) ==
let g = μ g. (∃d. Ord(d) & (domain(uu(f, b, g, d)) ≠ 0 &
domain(uu(f, b, g, d)) ⪯ m));
d = μ d. domain(uu(f, b, g, d)) ≠ 0 &
domain(uu(f, b, g, d)) ⪯ m
in if f'b ≠ 0 then domain(uu(f, b, g, d)) else 0"
```

**definition**

```
ww1 :: "[i, i, i] => i" where
"ww1(f, m, b) == f'b - vv1(f, m, b)"
```

**definition**

```
gg1 :: "[i, i, i] => i" where
"gg1(f, a, m) == λb ∈ a++a. if b<a then vv1(f, m, b) else ww1(f, m, b--a)"
```

**definition**

```
vv2 :: "[i, i, i, i] => i" where
"vv2(f, b, g, s) ==
if f'g ≠ 0 then {uu(f, b, g, μ d. uu(f, b, g, d) ≠ 0)'s} else
0"
```

**definition**

```
ww2 :: "[i, i, i, i] => i" where
"ww2(f, b, g, s) == f'g - vv2(f, b, g, s)"
```

**definition**

```
gg2 :: "[i, i, i, i] => i" where
"gg2(f, a, b, s) ==
λg ∈ a++a. if g<a then vv2(f, b, g, s) else ww2(f, b, g--a, s)"
```

**lemma** W02\_W03: "W02 ==> W03"

*(proof)*

**lemma** W03\_W01: "W03 ==> W01"  
*(proof)*

**lemma** W01\_W02: "W01 ==> W02"  
*(proof)*

**lemma** lam\_sets: "f ∈ A → B ==> (λx ∈ A. {f‘x}): A → {{b}. b ∈ B}"  
*(proof)*

**lemma** surj\_imp\_eq': "f ∈ surj(A, B) ==> (⋃ a ∈ A. {f‘a}) = B"  
*(proof)*

**lemma** surj\_imp\_eq: "[| f ∈ surj(A, B); Ord(A) |] ==> (⋃ a<A. {f‘a}) = B"  
*(proof)*

**lemma** W01\_W04: "W01 ==> W04(1)"  
*(proof)*

**lemma** W04\_mono: "[| m ≤ n; W04(m) |] ==> W04(n)"  
*(proof)*

**lemma** W04\_W05: "[| m ∈ nat; 1 ≤ m; W04(m) |] ==> W05"  
*(proof)*

**lemma** W05\_W06: "W05 ==> W06"  
*(proof)*

**lemma** lt\_oadd\_odiff\_disj:  
" [| k < i+j; Ord(i); Ord(j) |]  
==> k < i | (~ k < i & k = i ++ (k--i) & (k--i) < j)"  
*(proof)*

```

lemma domain_uu_subset: "domain(uu(f,b,g,d)) ⊆ f'b"
⟨proof⟩

lemma quant_domain_uu_lepoll_m:
  " $\forall b < a. f'b \lesssim m \implies \forall b < a. \forall g < a. \forall d < a. domain(uu(f,b,g,d)) \lesssim m$ "
⟨proof⟩

lemma uu_subset1: "uu(f,b,g,d) ⊆ f'b * f'g"
⟨proof⟩

lemma uu_subset2: "uu(f,b,g,d) ⊆ f'd"
⟨proof⟩

lemma uu_lepoll_m: "[| \forall b < a. f'b \lesssim m; d < a |] \implies uu(f,b,g,d) \lesssim m"
⟨proof⟩

```

```

lemma cases:
  " $\forall b < a. \forall g < a. \forall d < a. u(f,b,g,d) \lesssim m$ 
   \implies (\forall b < a. f'b \neq 0 \longrightarrow
             (\exists g < a. \exists d < a. u(f,b,g,d) \neq 0 \& u(f,b,g,d) \prec m))
   \mid (\exists b < a. f'b \neq 0 \& (\forall g < a. \forall d < a. u(f,b,g,d) \neq 0 \longrightarrow
               u(f,b,g,d) \approx m))"
⟨proof⟩

```

```

lemma UN_oadd: "Ord(a) \implies (\bigcup_{b < a++a} C(b)) = (\bigcup_{b < a} C(b) \cup C(a++b))"
⟨proof⟩

```

```

lemma vv1_subset: "vv1(f,m,b) ⊆ f'b"

```

$\langle proof \rangle$

```
lemma UN_gg1_eq:  
  "[| Ord(a); m ∈ nat |] ==> (∪ b<a++a. gg1(f,a,m) ` b) = (∪ b<a. f ` b)"  
 $\langle proof \rangle$ 
```

```
lemma domain_gg1: "domain(gg1(f,a,m)) = a++a"  
 $\langle proof \rangle$ 
```

```
lemma nested_LeastI:  
  "[| P(a, b); Ord(a); Ord(b);  
     Least_a = (μ a. ∃ x. Ord(x) & P(a, x)) |]  
 ==> P(Least_a, μ b. P(Least_a, b))"  
 $\langle proof \rangle$ 
```

```
lemmas nested_Least_instance =  
  nested_LeastI [of "%g d. domain(uu(f,b,g,d)) ≠ 0 &  
                 domain(uu(f,b,g,d)) ⪻ m"] for f b m
```

```
lemma gg1_lepoll_m:  
  "[| Ord(a); m ∈ nat;  
     ∀ b<a. f ` b ≠ 0 →  
     (∃ g<a. ∃ d<a. domain(uu(f,b,g,d)) ≠ 0 &  
      domain(uu(f,b,g,d)) ⪻ m);  
     ∀ b<a. f ` b ⪻ succ(m); b<a++a |]  
 ==> gg1(f,a,m) ` b ⪻ m"  
 $\langle proof \rangle$ 
```

```
lemma ex_d_uu_not_empty:  
  "[| b<a; g<a; f ` b ≠ 0; f ` g ≠ 0;  
     y*y ⊆ y; (∪ b<a. f ` b) = y |]  
 ==> ∃ d<a. uu(f,b,g,d) ≠ 0"  
 $\langle proof \rangle$ 
```

```
lemma uu_not_empty:
```

```

" [ | b < a; g < a; f' b ≠ 0; f' g ≠ 0; y * y ⊆ y; ( ∪ b < a. f' b ) = y | ]
  ==> uu(f, b, g, μ d. (uu(f, b, g, d) ≠ 0)) ≠ 0"
⟨proof⟩

lemma not_empty_rel_imp_domain: "[ | r ⊆ A * B; r ≠ 0 | ] ==> domain(r) ≠ 0"
⟨proof⟩

lemma Least_uu_not_empty_lt_a:
" [ | b < a; g < a; f' b ≠ 0; f' g ≠ 0; y * y ⊆ y; ( ∪ b < a. f' b ) = y | ]
  ==> (μ d. uu(f, b, g, d) ≠ 0) < a"
⟨proof⟩

lemma subset_Diff_sing: "[ | B ⊆ A; a ∉ B | ] ==> B ⊆ A - {a}"
⟨proof⟩

lemma supset_lepoll_imp_eq:
" [ | A ⪻ m; m ⪻ B; B ⊆ A; m ∈ nat | ] ==> A = B"
⟨proof⟩

lemma uu_Least_is_fun:
" [ | ∀ g < a. ∀ d < a. domain(uu(f, b, g, d)) ≠ 0 →
    domain(uu(f, b, g, d)) ≈ succ(m);
    ∀ b < a. f' b ⪻ succ(m); y * y ⊆ y;
    ( ∪ b < a. f' b ) = y; b < a; g < a; d < a;
    f' b ≠ 0; f' g ≠ 0; m ∈ nat; s ∈ f' b | ]
  ==> uu(f, b, g, μ d. uu(f, b, g, d) ≠ 0) ∈ f' b -> f' g"
⟨proof⟩

lemma vv2_subset:
" [ | ∀ g < a. ∀ d < a. domain(uu(f, b, g, d)) ≠ 0 →
    domain(uu(f, b, g, d)) ≈ succ(m);
    ∀ b < a. f' b ⪻ succ(m); y * y ⊆ y;
    ( ∪ b < a. f' b ) = y; b < a; g < a; m ∈ nat; s ∈ f' b | ]
  ==> vv2(f, b, g, s) ⊆ f' g"
⟨proof⟩

lemma UN_gg2_eq:
" [ | ∀ g < a. ∀ d < a. domain(uu(f, b, g, d)) ≠ 0 →
    domain(uu(f, b, g, d)) ≈ succ(m);
    ∀ b < a. f' b ⪻ succ(m); y * y ⊆ y;
    ( ∪ b < a. f' b ) = y; Ord(a); m ∈ nat; s ∈ f' b; b < a | ]
  ==> ( ∪ g < a + a. gg2(f, a, b, s) ' g ) = y"
⟨proof⟩

lemma domain_gg2: "domain(gg2(f, a, b, s)) = a + a"

```

$\langle proof \rangle$

**lemma** *vv2\_lepoll*: " $[| m \in \text{nat}; m \neq 0 |] \implies \text{vv2}(f, b, g, s) \lesssim m$ "  
 $\langle proof \rangle$

**lemma** *ww2\_lepoll*:  
" $[| \forall b < a. f'b \lesssim \text{succ}(m); g < a; m \in \text{nat}; \text{vv2}(f, b, g, d) \subseteq f'g |]$   
 $\implies \text{ww2}(f, b, g, d) \lesssim m$ "  
 $\langle proof \rangle$

**lemma** *gg2\_lepoll\_m*:  
" $[| \forall g < a. \forall d < a. \text{domain}(\text{uu}(f, b, g, d)) \neq 0 \longrightarrow$   
 $\text{domain}(\text{uu}(f, b, g, d)) \approx \text{succ}(m);$   
 $\forall b < a. f'b \lesssim \text{succ}(m); y * y \subseteq y;$   
 $(\bigcup b < a. f'b) = y; b < a; s \in f'b; m \in \text{nat}; m \neq 0; g < a + a |]$   
 $\implies \text{gg2}(f, a, b, s) \cdot g \lesssim m$ "  
 $\langle proof \rangle$

**lemma** *lemma\_ii*: " $[| \text{succ}(m) \in \text{NN}(y); y * y \subseteq y; m \in \text{nat}; m \neq 0 |] \implies m \in \text{NN}(y)$ "  
 $\langle proof \rangle$

**lemma** *z\_n\_subset\_z\_succ\_n*:  
" $\forall n \in \text{nat}. \text{rec}(n, x, \%k r. r \cup r * r) \subseteq \text{rec}(\text{succ}(n), x, \%k r. r \cup r * r)"$   
 $\langle proof \rangle$

**lemma** *le\_subsets*:  
" $[| \forall n \in \text{nat}. f(n) \leq f(\text{succ}(n)); n \leq m; n \in \text{nat}; m \in \text{nat} |]$   
 $\implies f(n) \leq f(m)$ "

$\langle proof \rangle$

```
lemma le_imp_rec_subset:
  "[| n ≤ m; m ∈ nat |]
   ==> rec(n, x, %k r. r ∪ r*r) ⊆ rec(m, x, %k r. r ∪ r*r)"
```

$\langle proof \rangle$

```
lemma lemma_iv: "∃ y. x ∪ y*y ⊆ y"
```

$\langle proof \rangle$

```
lemma W06_imp_NN_not_empty: "W06 ==> NN(y) ≠ 0"
```

$\langle proof \rangle$

```
lemma lemma1:
  "[| (∪ b < a. f‘b) = y; x ∈ y; ∀ b < a. f‘b ⪻ 1; Ord(a) |] ==> ∃ c < a. f‘c
  = {x}"
```

$\langle proof \rangle$

```
lemma lemma2:
  "[| (∪ b < a. f‘b) = y; x ∈ y; ∀ b < a. f‘b ⪻ 1; Ord(a) |]
   ==> f‘(μ i. f‘i = {x}) = {x}"
```

$\langle proof \rangle$

```
lemma NN_imp_ex_inj: "1 ∈ NN(y) ==> ∃ a f. Ord(a) & f ∈ inj(y, a)"
```

$\langle proof \rangle$

```
lemma y_well_ord: "[| y*y ⊆ y; 1 ∈ NN(y) |] ==> ∃ r. well_ord(y, r)"
```

$\langle proof \rangle$

```

lemma rev_induct_lemma [rule_format]:
  "[| n ∈ nat; !!m. [| m ∈ nat; m ≠ 0; P(succ(m)) |] ==> P(m) |]
  ==> n ≠ 0 —> P(n) —> P(1)"
⟨proof⟩

lemma rev_induct:
  "[| n ∈ nat; P(n); n ≠ 0;
    !!m. [| m ∈ nat; m ≠ 0; P(succ(m)) |] ==> P(m) |]
  ==> P(1)"
⟨proof⟩

lemma NN_into_nat: "n ∈ NN(y) ==> n ∈ nat"
⟨proof⟩

lemma lemma3: "[| n ∈ NN(y); y*y ⊆ y; n ≠ 0 |] ==> 1 ∈ NN(y)"
⟨proof⟩

lemma NN_y_0: "0 ∈ NN(y) ==> y=0"
⟨proof⟩

lemma W06_imp_W01: "W06 ==> W01"
⟨proof⟩

end

theory W01_W07
imports AC_Equiv
begin

definition
"LEMMA ==
  ∀X. ~Finite(X) —> (∃R. well_ord(X,R) & ~well_ord(X,converse(R)))"

lemma W07_iff_LEMMA: "W07 ↔ LEMMA"
⟨proof⟩

```

```

lemma LEMMA_imp_W01: "LEMMA ==> W01"
⟨proof⟩

lemma converse_Memrel_not_wf_on:
  "[| Ord(a); ~Finite(a) |] ==> ~wf[a](converse(Memrel(a)))"
⟨proof⟩

lemma converse_Memrel_not_well_ord:
  "[| Ord(a); ~Finite(a) |] ==> ~well_ord(a,converse(Memrel(a)))"
⟨proof⟩

lemma well_ord_rvimage_ordertype:
  "well_ord(A,r) ==>
   rvimage (ordertype(A,r), converse(ordermap(A,r)),r) =
   Memrel(ordertype(A,r))"
⟨proof⟩

lemma well_ord_converse_Memrel:
  "[| well_ord(A,r); well_ord(A,converse(r)) |]
  ==> well_ord(ordertype(A,r), converse(Memrel(ordertype(A,r))))"
⟨proof⟩

lemma W01_imp_LEMMA: "W01 ==> LEMMA"
⟨proof⟩

lemma W01_iff_W07: "W01 <→ W07"
⟨proof⟩

```

```

lemma W01_W08: "W01 ==> W08"
<proof>

lemma W08_W01: "W08 ==> W01"
<proof>

end

theory AC7_AC9
imports AC_Equiv
begin

lemma Sigma_fun_space_not0: "[| 0notin A; B ∈ A |] ==> (nat->J(A)) * B ≠ 0"
<proof>

lemma inj_lemma:
  "C ∈ A ==> (λg ∈ (nat->J(A))*C.
    (λn ∈ nat. if(n=0, snd(g), fst(g) ` (n #- 1)))
    ∈ inj((nat->J(A))*C, (nat->J(A))) ) "
<proof>

lemma Sigma_fun_space_eqpoll:
  "[| C ∈ A; 0notin A |] ==> (nat->J(A)) * C ≈ (nat->J(A))"
<proof>

lemma AC6_AC7: "AC6 ==> AC7"
<proof>

```

**lemma lemma1\_1:** " $y \in (\prod B \in A. Y*B) \implies (\lambda B \in A. \text{snd}(y^B)) \in (\prod B \in A. B)$ "  
*(proof)*

**lemma lemma1\_2:**  
 " $y \in (\prod B \in \{Y*C. C \in A\}. B) \implies (\lambda B \in A. y^C(Y*B)) \in (\prod B \in A. Y*B)$ "  
*(proof)*

**lemma AC7\_AC6\_lemma1:**  
 " $(\prod B \in \{(\text{nat} \rightarrow \bigcup(A)) * C. C \in A\}. B) \neq 0 \implies (\prod B \in A. B) \neq 0$ "  
*(proof)*

**lemma AC7\_AC6\_lemma2:** " $0 \notin A \implies 0 \notin \{(\text{nat} \rightarrow \bigcup(A)) * C. C \in A\}$ "  
*(proof)*

**lemma AC7\_AC6:** "AC7  $\implies$  AC6"  
*(proof)*

**lemma AC1\_AC8\_lemma1:**  
 " $\forall B \in A. \exists B1 B2. B = \langle B1, B2 \rangle \& B1 \approx B2 \implies 0 \notin \{ \text{bij}(\text{fst}(B), \text{snd}(B)). B \in A \}$ "  
*(proof)*

**lemma AC1\_AC8\_lemma2:**  
 "[| f \in (\prod X \in \text{RepFun}(A, p). X); D \in A |] \implies (\lambda x \in A. f^x p(x))^D \in p(D)"  
*(proof)*

**lemma AC1\_AC8:** "AC1  $\implies$  AC8"  
*(proof)*

**lemma AC8\_AC9\_lemma:**  
 " $\forall B1 \in A. \forall B2 \in A. B1 \approx B2$

$\Rightarrow \forall B \in A * A. \exists B1 B2. B = \langle B1, B2 \rangle \& B1 \approx B2$ "  
*(proof)*

**lemma** AC8\_AC9: "AC8 ==> AC9"  
*(proof)*

**lemma** snd\_lepoll\_SigmaI: "b ∈ B  $\implies$  X  $\lesssim$  B × X"  
*(proof)*

**lemma** nat\_lepoll\_lemma:  
" [| 0  $\notin$  A; B ∈ A |] ==> nat  $\lesssim$  ((nat → ∪(A)) × B) × nat"  
*(proof)*

**lemma** AC9\_AC1\_lemma1:  
" [| 0  $\notin$  A; A  $\neq$  0;  
C = {((nat → ∪(A)) \* B) \* nat. B ∈ A}  $\cup$   
{cons(0, ((nat → ∪(A)) \* B) \* nat). B ∈ A};  
B1 ∈ C; B2 ∈ C |]  
==> B1 ≈ B2"  
*(proof)*

**lemma** AC9\_AC1\_lemma2:  
"  $\forall B1 \in \{(F * B) * N. B \in A\} \cup \{\text{cons}(0, (F * B) * N). B \in A\}$ .  
 $\forall B2 \in \{(F * B) * N. B \in A\} \cup \{\text{cons}(0, (F * B) * N). B \in A\}$ .  
f ' $\langle B1, B2 \rangle$  ∈ bij(B1, B2)  
==> ( $\lambda B \in A. \text{snd}(\text{fst}((f ' $\langle \text{cons}(0, (F * B) * N), (F * B) * N \rangle$ , 0))) \in (\prod_{x \in A. X)$ "  
*(proof)*

**lemma** AC9\_AC1: "AC9 ==> AC1"  
*(proof)*

**end**

**theory** W01\_AC  
**imports** AC\_Equiv  
**begin**

```

theorem W01_AC1: "W01 ==> AC1"
⟨proof⟩

lemma lemma1: "[| W01; ∀B ∈ A. ∃C ∈ D(B). P(C,B) |] ==> ∃f. ∀B ∈ A.
P(f‘B,B)"
⟨proof⟩

lemma lemma2_1: "[| ~Finite(B); W01 |] ==> |B| + |B| ≈ B"
⟨proof⟩

lemma lemma2_2:
  "f ∈ bij(D+D, B) ==> {{f‘Inl(i), f‘Inr(i)}. i ∈ D} ∈ Pow(Pow(B))"
⟨proof⟩

lemma lemma2_3:
  "f ∈ bij(D+D, B) ==> pairwise_disjoint({{f‘Inl(i), f‘Inr(i)}.
i ∈ D})"
⟨proof⟩

lemma lemma2_4:
  "[| f ∈ bij(D+D, B); 1 ≤ n |]
  ==> sets_of_size_between({{f‘Inl(i), f‘Inr(i)}. i ∈ D}, 2, succ(n))"
⟨proof⟩

lemma lemma2_5:
  "f ∈ bij(D+D, B) ==> ∪({{f‘Inl(i), f‘Inr(i)}. i ∈ D}) = B"
⟨proof⟩

lemma lemma2:
  "[| W01; ~Finite(B); 1 ≤ n |]
  ==> ∃C ∈ Pow(Pow(B)). pairwise_disjoint(C) &
  sets_of_size_between(C, 2, succ(n)) &
  ∪(C) = B"
⟨proof⟩

theorem W01_AC10: "[| W01; 1 ≤ n |] ==> AC10(n)"
⟨proof⟩

end

```

```

theory Hartog
imports AC_Equiv
begin

definition
Hartog :: "i => i" where
  "Hartog(X) == μ i. ~ i ⪻ X"

lemma Ords_in_set: "∀ a. Ord(a) → a ∈ X ==> P"
⟨proof⟩

lemma Ord_lepoll_imp_ex_well_ord:
  "[| Ord(a); a ⪻ X |]
  ==> ∃ Y. Y ⊆ X & (∃ R. well_ord(Y,R) & ordertype(Y,R)=a)"
⟨proof⟩

lemma Ord_lepoll_imp_eq_ordertype:
  "[| Ord(a); a ⪻ X |] ==> ∃ Y. Y ⊆ X & (∃ R. R ⊆ X*X & ordertype(Y,R)=a)"
⟨proof⟩

lemma Ords_lepoll_set_lemma:
  "(∀ a. Ord(a) → a ⪻ X) ==>
   ∀ a. Ord(a) →
   a ∈ {b. Z ∈ Pow(X)*Pow(X*X), ∃ Y R. Z=<Y,R> & ordertype(Y,R)=b}"
⟨proof⟩

lemma Ords_lepoll_set: "∀ a. Ord(a) → a ⪻ X ==> P"
⟨proof⟩

lemma ex_Ord_not_lepoll: "∃ a. Ord(a) & ~a ⪻ X"
⟨proof⟩

lemma not_Hartog_lepoll_self: "~ Hartog(A) ⪻ A"
⟨proof⟩

lemmas Hartog_lepoll_selfE = not_Hartog_lepoll_self [THEN note]

lemma Ord_Hartog: "Ord(Hartog(A))"
⟨proof⟩

lemma less_HartogE1: "[| i < Hartog(A); ~ i ⪻ A |] ==> P"
⟨proof⟩

lemma less_HartogE: "[| i < Hartog(A); i ≈ Hartog(A) |] ==> P"
⟨proof⟩

lemma Card_Hartog: "Card(Hartog(A))"
⟨proof⟩

```

end

```
theory HH
imports AC_Equiv Hartog
begin
```

**definition**

```
HH :: "[i, i, i] => i" where
"HH(f, x, a) == transrec(a, %b r. let z = x - (\bigcup c ∈ b. r'c)
                                in if f'z ∈ Pow(z)-{0} then f'z else
                                {x})"
```

## 0.1 Lemmas useful in each of the three proofs

**lemma HH\_def\_satisfies\_eq:**

```
"HH(f, x, a) = (let z = x - (\bigcup b ∈ a. HH(f, x, b))
                  in if f'z ∈ Pow(z)-{0} then f'z else {x})"
```

*(proof)*

**lemma HH\_values:** "HH(f, x, a) ∈ Pow(x)-{0} ∣ HH(f, x, a)={x}"
*(proof)*

**lemma subset\_imp\_Diff\_eq:**

```
"B ⊆ A ==> X - (\bigcup a ∈ A. P(a)) = X - (\bigcup a ∈ A-B. P(a)) - (\bigcup b ∈ B. P(b))"
```

*(proof)*

**lemma Ord\_DiffE:** "[| c ∈ a-b; b < a |] ==> c=b ∣ b < c & c < a"
*(proof)*

**lemma Diff\_UN\_eq\_self:** "(!!y. y ∈ A ==> P(y) = {x}) ==> x - (\bigcup y ∈ A.
P(y)) = x"
*(proof)*

**lemma HH\_eq:** "x - (\bigcup b ∈ a. HH(f, x, b)) = x - (\bigcup b ∈ a1. HH(f, x, b))
==> HH(f, x, a) = HH(f, x, a1)"
*(proof)*

**lemma HH\_is\_x\_gt\_too:** "[| HH(f, x, b)={x}; b < a |] ==> HH(f, x, a)={x}"
*(proof)*

**lemma HH\_subset\_x\_lt\_too:**

```
"[| HH(f, x, a) ∈ Pow(x)-{0}; b < a |] ==> HH(f, x, b) ∈ Pow(x)-{0}"
```

*(proof)*

**lemma HH\_subset\_x\_imp\_subset\_Diff\_UN:**

```
"HH(f, x, a) ∈ Pow(x)-{0} ==> HH(f, x, a) ∈ Pow(x - (\bigcup b ∈ a. HH(f, x, b))) - {0}"
```

*(proof)*

```

lemma HH_eq_arg_lt:
  "[| HH(f,x,v)=HH(f,x,w); HH(f,x,v) ∈ Pow(x)-{0}; v ∈ w |] ==> P"
⟨proof⟩

lemma HH_eq_imp_arg_eq:
  "[| HH(f,x,v)=HH(f,x,w); HH(f,x,w) ∈ Pow(x)-{0}; Ord(v); Ord(w) |] ==>
   v=w"
⟨proof⟩

lemma HH_subset_x_imp_lepoll:
  "[| HH(f, x, i) ∈ Pow(x)-{0}; Ord(i) |] ==> i ⪻ Pow(x)-{0}"
⟨proof⟩

lemma HH_Hartog_is_x: "HH(f, x, Hartog(Pow(x)-{0})) = {x}"
⟨proof⟩

lemma HH_Least_eq_x: "HH(f, x, μ i. HH(f, x, i) = {x}) = {x}"
⟨proof⟩

lemma less_Least_subset_x:
  "a ∈ (μ i. HH(f,x,i)={x}) ==> HH(f,x,a) ∈ Pow(x)-{0}"
⟨proof⟩

```

**0.2 Lemmas used in the proofs of AC1 ==<sub>i</sub> WO2 and AC17  
==<sub>i</sub> AC1**

```

lemma lam_Least_HH_inj_Pow:
  "(λa ∈ (μ i. HH(f,x,i)={x}). HH(f,x,a))
   ∈ inj(μ i. HH(f,x,i)={x}, Pow(x)-{0})"
⟨proof⟩

lemma lam_Least_HH_inj:
  "∀a ∈ (μ i. HH(f,x,i)={x}). ∃z ∈ x. HH(f,x,a) = {z}
   ==> (λa ∈ (μ i. HH(f,x,i)={x}). HH(f,x,a))
   ∈ inj(μ i. HH(f,x,i)={x}, {{y}. y ∈ x})"
⟨proof⟩

lemma lam_surj_sing:
  "[| x - (⋃a ∈ A. F(a)) = 0; ∀a ∈ A. ∃z ∈ x. F(a) = {z} |]
   ==> (λa ∈ A. F(a)) ∈ surj(A, {{y}. y ∈ x})"
⟨proof⟩

lemma not_emptyI2: "y ∈ Pow(x)-{0} ==> x ≠ 0"
⟨proof⟩

lemma f_subset_imp_HH_subset:
  "f‘(x - (⋃j ∈ i. HH(f,x,j))) ∈ Pow(x - (⋃j ∈ i. HH(f,x,j)))-{0}"

```

```

==> HH(f, x, i) ∈ Pow(x) - {0}"
⟨proof⟩

lemma f_subsets_imp_UN_HH_eq_x:
  "∀ z ∈ Pow(x)-{0}. f‘z ∈ Pow(z)-{0}
   ==> x - (⋃ j ∈ (μ i. HH(f,x,i)={x}). HH(f,x,j)) = 0"
⟨proof⟩

lemma HH_values2: "HH(f,x,i) = f‘(x - (⋃ j ∈ i. HH(f,x,j))) ∣ HH(f,x,i)={x}"
⟨proof⟩

lemma HH_subset_imp_eq:
  "HH(f,x,i): Pow(x)-{0} ==> HH(f,x,i)=f‘(x - (⋃ j ∈ i. HH(f,x,j)))"
⟨proof⟩

lemma f_sing_imp_HH_sing:
  "[| f ∈ (Pow(x)-{0}) -> {{z}}. z ∈ x;
     a ∈ (μ i. HH(f,x,i)={x}) |] ==> ∃ z ∈ x. HH(f,x,a) = {z}"
⟨proof⟩

lemma f_sing_lam_bij:
  "[| x - (⋃ j ∈ (μ i. HH(f,x,i)={x}). HH(f,x,j)) = 0;
     f ∈ (Pow(x)-{0}) -> {{z}}. z ∈ x |]
   ==> (λa ∈ (μ i. HH(f,x,i)={x}). HH(f,x,a))
     ∈ bij(μ i. HH(f,x,i)={x}, {{y}}. y ∈ x)"
⟨proof⟩

lemma lam_singI:
  "f ∈ (Π X ∈ Pow(x)-{0}. F(X))
   ==> (λX ∈ Pow(x)-{0}. {f‘X}) ∈ (Π X ∈ Pow(x)-{0}. {{z}}. z ∈ F(X))"
⟨proof⟩

lemmas bij_Least_HH_x =
  comp_bij [OF f_sing_lam_bij [OF _ lam_singI]
            lam_sing_bij [THEN bij_converse_bij]]
```

### 0.3 The proof of AC1 ==> WO2

```

lemma bijection:
  "f ∈ (Π X ∈ Pow(x) - {0}. X)
   ==> ∃ g. g ∈ bij(x, μ i. HH(λX ∈ Pow(x)-{0}. {f‘X}, x, i) = {x})"
⟨proof⟩

lemma AC1_WO2: "AC1 ==> WO2"
⟨proof⟩

end
```

```

theory AC15_W06
imports HH Cardinal_aux
begin

lemma lepoll_Sigma: "A ≠ 0 ==> B ≲ A*B"
⟨proof⟩

lemma cons_times_nat_not_Finite:
  "0 ∉ A ==> ∀ B ∈ {cons(0, x*nat). x ∈ A}. ~Finite(B)"
⟨proof⟩

lemma lemma1: "[| ∪ (C)=A; a ∈ A |] ==> ∃ B ∈ C. a ∈ B & B ⊆ A"
⟨proof⟩

lemma lemma2:
  "[| pairwise_disjoint(A); B ∈ A; C ∈ A; a ∈ B; a ∈ C |] ==>
B=C"
⟨proof⟩

lemma lemma3:
  "∀ B ∈ {cons(0, x*nat). x ∈ A}. pairwise_disjoint(f`B) &
sets_of_size_between(f`B, 2, n) & ∪(f`B)=B
==> ∀ B ∈ A. ∃! u. u ∈ f`cons(0, B*nat) & u ⊆ cons(0, B*nat) &
0 ∈ u & 2 ≲ u & u ≲ n"
⟨proof⟩

lemma lemma4: "[| A ≲ i; Ord(i) |] ==> {P(a). a ∈ A} ≲ i"
⟨proof⟩

lemma lemma5_1:
  "[| B ∈ A; 2 ≲ u(B) |] ==> (λx ∈ A. {fst(x). x ∈ u(x)-{0}})`B ≠
0"
⟨proof⟩

lemma lemma5_2:
  "[| B ∈ A; u(B) ⊆ cons(0, B*nat) |]
==> (λx ∈ A. {fst(x). x ∈ u(x)-{0}})`B ⊆ B"
⟨proof⟩

lemma lemma5_3:

```

$"[| n \in \text{nat}; B \in A; 0 \in u(B); u(B) \lesssim \text{succ}(n) |]$   
 $\implies (\lambda x \in A. \{fst(x). x \in u(x)-\{0\}\})'B \lesssim n"$   
*(proof)*

**lemma** *ex\_fun\_AC13\_AC15*:  
 $"[| \forall B \in \{\text{cons}(0, x*\text{nat}). x \in A\}.$   
 $\quad \text{pairwise\_disjoint}(f'B) \ \&$   
 $\quad \text{sets\_of\_size\_between}(f'B, 2, \text{succ}(n)) \ \& \bigcup(f'B)=B;$   
 $\quad n \in \text{nat} |]$   
 $\implies \exists f. \forall B \in A. f'B \neq 0 \ \& \ f'B \subseteq B \ \& \ f'B \lesssim n"$   
*(proof)*

**theorem** *AC10\_AC11*: " $[| n \in \text{nat}; 1 \leq n; AC10(n) |] \implies AC11$ "  
*(proof)*

**theorem** *AC11\_AC12*: "AC11  $\implies$  AC12"  
*(proof)*

**theorem** *AC12\_AC15*: "AC12  $\implies$  AC15"  
*(proof)*

**lemma** *OUN\_eq\_UN*: " $\text{Ord}(x) \implies (\bigcup_{a < x. F(a)}) = (\bigcup_{a \in x. F(a)})$ "  
*(proof)*

**lemma** *AC15\_W06\_aux1*:  
 $"\forall x \in \text{Pow}(A)-\{0\}. f'x \neq 0 \ \& \ f'x \subseteq x \ \& \ f'x \lesssim m$   
 $\implies (\bigcup_{i < \mu x. HH(f, A, x)=\{A\}. HH(f, A, i)} = A)"$   
*(proof)*

**lemma** *AC15\_W06\_aux2*:

" $\forall x \in Pow(A) - \{0\}. f'x \neq 0 \ \& \ f'x \subseteq x \ \& \ f'x \lesssim m$   
 $\implies \forall x < (\mu x. HH(f, A, x) = \{A\}). HH(f, A, x) \lesssim m$ "

*(proof)*

**theorem** *AC15\_W06*: "AC15 ==> W06"

*(proof)*

**theorem** *AC10\_AC13*: "[| n ∈ nat; 1 ≤ n; AC10(n) |] ==> AC13(n)"

*(proof)*

**lemma** *AC1\_AC13*: "AC1 ==> AC13(1)"

*(proof)*

**lemma** *AC13\_mono*: "[| m ≤ n; AC13(m) |] ==> AC13(n)"

*(proof)*

**theorem** *AC13\_AC14*: "[| n ∈ nat; 1 ≤ n; AC13(n) |] ==> AC14"  
*(proof)*

**theorem** *AC14\_AC15*: "AC14 ==> AC15"  
*(proof)*

**lemma** *lemma\_aux*: "[| A ≠ 0; A ≈ 1 |] ==> ∃ a. A = {a}"  
*(proof)*

**lemma** *AC13\_AC1\_lemma*:  
 "∀ B ∈ A. f(B) ≠ 0 & f(B) ≤ B & f(B) ≈ 1  
 ==> (∀ x ∈ A. THE y. f(x) = {y}) ∈ (Π X ∈ A. X)"  
*(proof)*

**theorem** *AC13\_AC1*: "AC13(1) ==> AC1"  
*(proof)*

**theorem** *AC11\_AC14*: "AC11 ==> AC14"  
*(proof)*

end

**theory** *AC16\_lemmas*  
**imports** *AC\_Equiv Hartog Cardinal\_aux*  
**begin**

**lemma** *cons\_Diff\_eq*: "a ∉ A ==> cons(a, A) - {a} = A"  
*(proof)*

**lemma** *nat\_1\_lepoll\_iff*: "1 ≈ X ↔ (∃ x. x ∈ X)"

```

⟨proof⟩

lemma eqpoll_1_iff_singleton: "X≈1  $\longleftrightarrow$  ( $\exists x$ . X={x})"
⟨proof⟩

lemma cons_eqpoll_succ: "[| x≈n; ynotin x |] ==> cons(y,x)≈succ(n)"
⟨proof⟩

lemma subsets_eqpoll_1_eq: "{Y ∈ Pow(X). Y≈1} = {{x}. x ∈ X}"
⟨proof⟩

lemma eqpoll_RepFun_sing: "X≈{{x}. x ∈ X}"
⟨proof⟩

lemma subsets_eqpoll_1_eqpoll: "{Y ∈ Pow(X). Y≈1}≈X"
⟨proof⟩

lemma InfCard_Least_in:
  "[| InfCard(x); y ⊆ x; y ≈ succ(z) |] ==> (μ i. i ∈ y) ∈ y"
⟨proof⟩

lemma subsets_lepoll_lemma1:
  "[| InfCard(x); n ∈ nat |]
  ==> {y ∈ Pow(x). y≈succ(succ(n))} ⪻ x*{y ∈ Pow(x). y≈succ(n)}"
⟨proof⟩

lemma set_of_Ord_succ_Union: "(∀y ∈ z. Ord(y)) ==> z ⊆ succ(∪(z))"
⟨proof⟩

lemma subset_not_mem: "j ⊆ i ==> i ∉ j"
⟨proof⟩

lemma succ_Union_not_mem:
  "(!!y. y ∈ z ==> Ord(y)) ==> succ(∪(z)) ∉ z"
⟨proof⟩

lemma Union_cons_eq_succ_Union:
  " $\bigcup(\text{cons}(\text{succ}(\bigcup(z)), z)) = \text{succ}(\bigcup(z))$ "
⟨proof⟩

lemma Un_Ord_disj: "[| Ord(i); Ord(j) |] ==> i ∪ j = i ∣ i ∪ j = j"
⟨proof⟩

lemma Union_eq_Un: "x ∈ X ==> ∪(X) = x ∪ ∪(X-{x})"
⟨proof⟩

lemma Union_in_lemma [rule_format]:
  " $n ∈ \text{nat} ==> \forall z. (\forall y ∈ z. Ord(y)) \& z≈n \& z\neq0 \longrightarrow \bigcup(z) ∈ z$ "
⟨proof⟩

```

```

lemma Union_in: "[| ∀ x ∈ z. Ord(x); z≈n; z≠0; n ∈ nat |] ==> ∪(z)
∈ z"
⟨proof⟩

lemma succ_Union_in_x:
" [| InfCard(x); z ∈ Pow(x); z≈n; n ∈ nat |] ==> succ(∪(z)) ∈ x"
⟨proof⟩

lemma succ_lepoll_succ_succ:
" [| InfCard(x); n ∈ nat |]
==> {y ∈ Pow(x). y≈succ(n)} ≤ {y ∈ Pow(x). y≈succ(succ(n))}"
⟨proof⟩

lemma subsets_eqpoll_X:
" [| InfCard(X); n ∈ nat |] ==> {Y ∈ Pow(X). Y≈succ(n)} ≈ X"
⟨proof⟩

lemma image_vimage_eq:
" [| f ∈ surj(A,B); y ⊆ B |] ==> f `` (converse(f) `` y) = y"
⟨proof⟩

lemma vimage_image_eq: " [| f ∈ inj(A,B); y ⊆ A |] ==> converse(f) `` (f `` y) = y"
= y"
⟨proof⟩

lemma subsets_eqpoll:
"A≈B ==> {Y ∈ Pow(A). Y≈n}≈{Y ∈ Pow(B). Y≈n}"
⟨proof⟩

lemma W02_imp_ex_Card: "W02 ==> ∃ a. Card(a) & X≈a"
⟨proof⟩

lemma lepoll_infinite: "[| X≤Y; ~Finite(X) |] ==> ~Finite(Y)"
⟨proof⟩

lemma infinite_Card_is_InfCard: "[| ~Finite(X); Card(X) |] ==> InfCard(X)"
⟨proof⟩

lemma W02_infinite_subsets_eqpoll_X: "[| W02; n ∈ nat; ~Finite(X) |]
==> {Y ∈ Pow(X). Y≈succ(n)}≈X"
⟨proof⟩

lemma well_ord_imp_ex_Card: "well_ord(X,R) ==> ∃ a. Card(a) & X≈a"
⟨proof⟩

lemma well_ord_infinite_subsets_eqpoll_X:
" [| well_ord(X,R); n ∈ nat; ~Finite(X) |] ==> {Y ∈ Pow(X). Y≈succ(n)}≈X"

```

```

⟨proof⟩

end

theory W02_AC16 imports AC_Equiv AC16_lemmas Cardinal_aux begin

definition
  recfunAC16 :: "[i,i,i,i] => i" where
    "recfunAC16(f,h,i,a) ==
      transrec2(i, 0,
        %g r. if (∃y ∈ r. h'g ⊆ y) then r
        else r ∪ {f'(μ i. h'g ⊆ f'i &
          (∀b<a. (h'b ⊆ f'i → (∀t ∈ r. ~ h'b ⊆ t))))})"

```

```

lemma recfunAC16_0: "recfunAC16(f,h,0,a) = 0"
⟨proof⟩

lemma recfunAC16_succ:
  "recfunAC16(f,h,succ(i),a) =
    (if (∃y ∈ recfunAC16(f,h,i,a). h' i ⊆ y) then recfunAC16(f,h,i,a)

    else recfunAC16(f,h,i,a) ∪
      {f' (μ j. h' i ⊆ f' j &
        (∀b<a. (h'b ⊆ f'j
          → (∀t ∈ recfunAC16(f,h,i,a). ~ h'b ⊆ t))))})"
⟨proof⟩

lemma recfunAC16_Limit: "Limit(i)
  ==> recfunAC16(f,h,i,a) = (⋃j<i. recfunAC16(f,h,j,a))"
⟨proof⟩

lemma transrec2_mono_lemma [rule_format]:
  "[| !!g r. r ⊆ B(g,r); Ord(i) |]
  ==> j<i → transrec2(j, 0, B) ⊆ transrec2(i, 0, B)"
⟨proof⟩

lemma transrec2_mono:
  "[| !!g r. r ⊆ B(g,r); j ≤ i |]"

```

$\Rightarrow \text{transrec2}(j, 0, B) \subseteq \text{transrec2}(i, 0, B)$ "  
*(proof)*

**lemma recfunAC16\_mono:**  
 $i \leq j \Rightarrow \text{recfunAC16}(f, g, i, a) \subseteq \text{recfunAC16}(f, g, j, a)$ "  
*(proof)*

**lemma lemma3\_1:**  
 $[\forall y < x. \forall z < a. z < y \mid (\exists Y \in F(y). f(z) \leq Y) \rightarrow (\exists ! Y. Y \in F(y) \& f(z) \leq Y);$   
 $\forall i j. i \leq j \rightarrow F(i) \subseteq F(j); j \leq i; i < x; z < a;$   
 $V \in F(i); f(z) \leq V; W \in F(j); f(z) \leq W]$   
 $\Rightarrow V = W$ "  
*(proof)*

**lemma lemma3:**  
 $[\forall y < x. \forall z < a. z < y \mid (\exists Y \in F(y). f(z) \leq Y) \rightarrow (\exists ! Y. Y \in F(y) \& f(z) \leq Y);$   
 $\forall i j. i \leq j \rightarrow F(i) \subseteq F(j); i < x; j < x; z < a;$   
 $V \in F(i); f(z) \leq V; W \in F(j); f(z) \leq W]$   
 $\Rightarrow V = W$ "  
*(proof)*

**lemma lemma4:**  
 $[\forall y < x. F(y) \subseteq X \&$   
 $(\forall x < a. x < y \mid (\exists Y \in F(y). h(x) \subseteq Y) \rightarrow$   
 $(\exists ! Y. Y \in F(y) \& h(x) \subseteq Y));$   
 $x < a]$   
 $\Rightarrow \forall y < x. \forall z < a. z < y \mid (\exists Y \in F(y). h(z) \subseteq Y) \rightarrow$   
 $(\exists ! Y. Y \in F(y) \& h(z) \subseteq Y)"$   
*(proof)*

**lemma lemma5:**  
 $[\forall y < x. F(y) \subseteq X \&$   
 $(\forall x < a. x < y \mid (\exists Y \in F(y). h(x) \subseteq Y) \rightarrow$   
 $(\exists ! Y. Y \in F(y) \& h(x) \subseteq Y));$   
 $x < a; \text{Limit}(x); \forall i j. i \leq j \rightarrow F(i) \subseteq F(j)]$   
 $\Rightarrow (\bigcup_{x < x} F(x)) \subseteq X \&$   
 $(\forall x a < a. x a < x \mid (\exists x \in \bigcup_{x < x} F(x). h(xa) \subseteq x))$

$\longrightarrow (\exists ! Y. Y \in (\bigcup_{x < x} F(x)) \& h(xa) \subseteq Y)"$

*(proof)*

**lemma dbl\_Diff\_eqpoll\_Card:**

"[| A ≈ a; Card(a); ~Finite(a); B ≺ a; C ≺ a |] ==> A - B - C ≈ a"

*(proof)*

**lemma Finite\_lesspoll\_infinite\_Ord:**

"[| Finite(X); ~Finite(a); Ord(a) |] ==> X ≺ a"

*(proof)*

**lemma Union\_lesspoll:**

"[| \forall x \in X. x \lesssim n \& x \subseteq T; well\_ord(T, R); X \lesssim b; b < a; ~Finite(a); Card(a); n \in nat |]  
==> \bigcup X ≺ a"

*(proof)*

**lemma Un\_sing\_eq\_cons:** "A ∪ {a} = cons(a, A)"

*(proof)*

**lemma Un\_lepoll\_succ:** "A \lesssim B ==> A ∪ {a} \lesssim succ(B)"

**lemma Diff\_UN\_succ\_empty:** "Ord(a) ==> F(a) - (\bigcup\_{b < succ(a)} F(b)) = 0"

**lemma Diff\_UN\_succ\_subset:** "Ord(a) ==> F(a) ∪ X - (\bigcup\_{b < succ(a)} F(b)) ⊆ X"

*(proof)*

```

lemma recfunAC16_Diff_lepoll_1:
  "Ord(x)
   ==> recfunAC16(f, g, x, a) - (\bigcup_{i < x} recfunAC16(f, g, i, a)) \leq 1"
⟨proof⟩

lemma in_Least_Diff:
  "[| z ∈ F(x); Ord(x) |]
   ==> z ∈ F(μ i. z ∈ F(i)) - (\bigcup_{j < (μ i. z ∈ F(i))} F(j))"
⟨proof⟩

lemma Least_eq_imp_ex:
  "[| (μ i. w ∈ F(i)) = (μ i. z ∈ F(i));
     w ∈ (\bigcup_{i < a} F(i)); z ∈ (\bigcup_{i < a} F(i)) |]
   ==> ∃ b < a. w ∈ (F(b) - (\bigcup_{c < b} F(c))) & z ∈ (F(b) - (\bigcup_{c < b} F(c)))"
⟨proof⟩

lemma two_in_lepoll_1: "[| A \leq 1; a ∈ A; b ∈ A |] ==> a = b"
⟨proof⟩

lemma UN_lepoll_index:
  "[| ∀ i < a. F(i) - (\bigcup_{j < i} F(j)) \leq 1; Limit(a) |]
   ==> (\bigcup_{x < a} F(x)) \leq a"
⟨proof⟩

lemma recfunAC16_lepoll_index: "Ord(y) ==> recfunAC16(f, h, y, a) \leq y"
⟨proof⟩

lemma Union_recfunAC16_lesspoll:
  "[| recfunAC16(f, g, y, a) ⊆ {X ∈ Pow(A). X ≈ n};
     A ≈ a; y < a; ~Finite(a); Card(a); n ∈ nat |]
   ==> ∪ (recfunAC16(f, g, y, a)) ≈ a"
⟨proof⟩

lemma dbl_Diff_eqpoll:
  "[| recfunAC16(f, h, y, a) ⊆ {X ∈ Pow(A). X ≈ succ(k #+ m)};
     Card(a); ~Finite(a); A ≈ a;
     k ∈ nat; y < a;
     h ∈ bij(a, {Y ∈ Pow(A). Y ≈ succ(k)}) |]
   ==> A - ∪ (recfunAC16(f, h, y, a)) - h' y ≈ a"
⟨proof⟩

```

```

lemmas disj_Un_eqpoll_nat_sum =
eqpoll_trans [THEN eqpoll_trans,
OF disj_Un_eqpoll_sum sum_eqpoll_cong nat_sum_eqpoll_sum]

lemma Un_in_Collect: "[| x ∈ Pow(A - B - h`i); x≈m;
h ∈ bij(a, {x ∈ Pow(A) . x≈k}); i<a; k ∈ nat; m ∈ nat |]
==> h ` i ∪ x ∈ {x ∈ Pow(A) . x≈k #+ m}"
⟨proof⟩

lemma lemma6:
"[| ∀ y<succ(j). F(y)≤X & (∀ x<a. x<y | P(x,y) → Q(x,y)); succ(j)<a |]
==> F(j)≤X & (∀ x<a. x<j | P(x,j) → Q(x,j))"
⟨proof⟩

lemma lemma7:
"[| ∀ x<a. x<j | P(x,j) → Q(x,j); succ(j)<a |]
==> P(j,j) → (∀ x<a. x≤j | P(x,j) → Q(x,j))"
⟨proof⟩

lemma ex_subset_eqpoll:
"[| A≈a; ~ Finite(a); Ord(a); m ∈ nat |] ==> ∃ X ∈ Pow(A). X≈m"
⟨proof⟩

lemma subset_Un_disjoint: "[| A ⊆ B ∪ C; A ∩ C = 0 |] ==> A ⊆ B"
⟨proof⟩

lemma Int_empty:
"[| X ∈ Pow(A - ∪(B) -C); T ∈ B; F ⊆ T |] ==> F ∩ X = 0"
⟨proof⟩

```

```

lemma subset_imp_eq_lemma:
  " $m \in \text{nat} \implies \forall A B. A \subseteq B \& m \lesssim A \& B \lesssim m \longrightarrow A=B"$ 
(proof)

lemma subset_imp_eq: "[| A ⊆ B; m ≈ A; B ≈ m; m ∈ nat |] ==> A=B"
(proof)

lemma bij_imp_arg_eq:
  "[| f ∈ bij(a, {Y ∈ X. Y≈succ(k)}); k ∈ nat; f'b ⊆ f'y; b<a; y<a
  |]
  ==> b=y"
(proof)

lemma ex_next_set:
  "[| \text{recfunAC16}(f, h, y, a) ⊆ \{X ∈ Pow(A) . X≈succ(k #+ m)\};
  Card(a); ~ Finite(a); A≈a;
  k ∈ nat; m ∈ nat; y<a;
  h ∈ bij(a, {Y ∈ Pow(A). Y≈succ(k)});
  ~ (\exists Y ∈ \text{recfunAC16}(f, h, y, a). h'y ⊆ Y) |]
  ==> \exists X ∈ \{Y ∈ Pow(A). Y≈succ(k #+ m)\}. h'y ⊆ X &
  (\forall b<a. h'b ⊆ X \longrightarrow
  (\forall T ∈ \text{recfunAC16}(f, h, y, a). ~ h'b ⊆ T))"
(proof)

lemma ex_next_0rd:
  "[| \text{recfunAC16}(f, h, y, a) ⊆ \{X ∈ Pow(A) . X≈succ(k #+ m)\},
  Card(a); ~ Finite(a); A≈a;
  k ∈ nat; m ∈ nat; y<a;
  h ∈ bij(a, {Y ∈ Pow(A). Y≈succ(k)});
  f ∈ bij(a, {Y ∈ Pow(A). Y≈succ(k #+ m)});
  ~ (\exists Y ∈ \text{recfunAC16}(f, h, y, a). h'y ⊆ Y) |]
  ==> \exists c<a. h'y ⊆ f'c &
  (\forall b<a. h'b ⊆ f'c \longrightarrow
  (\forall T ∈ \text{recfunAC16}(f, h, y, a). ~ h'b ⊆ T))"
(proof)

```

```

lemma lemma8:
  "[| ∀ x<a. x<j | (exists xa ∈ F(j). P(x, xa))
   → (exists ! Y. Y ∈ F(j) & P(x, Y)); F(j) ⊆ X;
   L ∈ X; P(j, L) & (forall x<a. P(x, L) → (forall xa ∈ F(j). ~P(x, xa)))
  |]
  ==> F(j) ∪ {L} ⊆ X &
  (forall x<a. x ≤ j | (exists xa ∈ (F(j) ∪ {L})). P(x, xa)) →
  (exists ! Y. Y ∈ (F(j) ∪ {L}) & P(x, Y))"
⟨proof⟩

```

```

lemma main_induct:
  "[| b < a; f ∈ bij(a, {Y ∈ Pow(A) . Y ≈ succ(k #+ m)});
   h ∈ bij(a, {Y ∈ Pow(A) . Y ≈ succ(k)});
   ~Finite(a); Card(a); A ≈ a; k ∈ nat; m ∈ nat |]
  ==> recfunAC16(f, h, b, a) ⊆ {X ∈ Pow(A) . X ≈ succ(k #+ m)} &
  (forall x<a. x < b | (exists Y ∈ recfunAC16(f, h, b, a). h ` x ⊆ Y) →
  (exists ! Y. Y ∈ recfunAC16(f, h, b, a) & h ` x ⊆ Y))"
⟨proof⟩

```

```

lemma lemma_simp_induct:
  "[| ∀ b. b < a → F(b) ⊆ S & (forall x<a. (x < b | (exists Y ∈ F(b). f ` x ⊆ Y))
   → (exists ! Y. Y ∈ F(b) & f ` x ⊆ Y));
   f ∈ a->f``(a); Limit(a);
   ∀ i j. i ≤ j → F(i) ⊆ F(j) |]
  ==> (∪ j < a. F(j)) ⊆ S &
  (forall x ∈ f `` a. exists ! Y. Y ∈ (∪ j < a. F(j)) & x ⊆ Y)"
⟨proof⟩

```

**theorem W02\_AC16:** "[| W02; 0 < m; k ∈ nat; m ∈ nat |] ==> AC16(k #+ m, k)"  
 ⟨proof⟩

**end**

```

theory AC16_W04
imports AC16_lemmas
begin

lemma lemma1:
  "[| Finite(A); 0 < m; m ∈ nat |]
  ==> ∃ a f. Ord(a) & domain(f) = a &
             (∪ b < a. f ` b) = A & (∀ b < a. f ` b ≤ m)"
  ⟨proof⟩

lemmas well_ord_paired = paired_bij [THEN bij_is_inj, THEN well_ord_rvimage]

lemma lepoll_trans1: "[| A ≤ B; ~ A ≤ C |] ==> ~ B ≤ C"
  ⟨proof⟩

lemmas lepoll_paired = paired_eqpoll [THEN eqpoll_sym, THEN eqpoll_imp_lepoll]

lemma lemma2: "∃ y R. well_ord(y, R) & x ∩ y = 0 & ~y ≤ z & ~Finite(y)"
  ⟨proof⟩

lemma infinite_Un: "~Finite(B) ==> ~Finite(A ∪ B)"
  ⟨proof⟩

```

```

lemma succ_not_lepoll_lemma:
  "[| ~(\exists x ∈ A. f ` x = y); f ∈ inj(A, B); y ∈ B |]
   ==> (λa ∈ succ(A). if(a=A, y, f ` a)) ∈ inj(succ(A), B)"
⟨proof⟩

lemma succ_not_lepoll_imp_eqpoll: "[| ~A ≈ B; A ≤ B |] ==> succ(A)
≤ B"
⟨proof⟩

lemmas ordertype_eqpoll =
  ordermap_bij [THEN exI [THEN eqpoll_def [THEN def_imp_iff, THEN
iffD2]]]

lemma cons_cons_subset:
  "[| a ⊆ y; b ∈ y-a; u ∈ x |] ==> cons(b, cons(u, a)) ∈ Pow(x ∪
y)"
⟨proof⟩

lemma cons_cons_eqpoll:
  "[| a ≈ k; a ⊆ y; b ∈ y-a; u ∈ x; x ∩ y = 0 |]
   ==> cons(b, cons(u, a)) ≈ succ(succ(k))"
⟨proof⟩

lemma set_eq_cons:
  "[| succ(k) ≈ A; k ≈ B; B ⊆ A; a ∈ A-B; k ∈ nat |] ==> A = cons(a,
B)"
⟨proof⟩

lemma cons_eqE: "[| cons(x, a) = cons(y, a); x ≠ a |] ==> x = y"
⟨proof⟩

lemma eq_imp_Int_eq: "A = B ==> A ∩ C = B ∩ C"
⟨proof⟩

lemma eqpoll_sum_imp_Diff_lepoll_lemma [rule_format]:
  "[| k ∈ nat; m ∈ nat |]
   ==> ∀ A B. A ≈ k #+ m & k ≤ B & B ⊆ A —> A-B ≤ m"
⟨proof⟩

```

```

lemma eqpoll_sum_imp_Diff_lepoll:
  "[| A ≈ succ(k #+ m); B ⊆ A; succ(k) ⪻ B; k ∈ nat; m ∈ nat |]
   ==> A-B ⪻ m"
⟨proof⟩

lemma eqpoll_sum_imp_Diff_eqpoll_lemma [rule_format]:
  "[| k ∈ nat; m ∈ nat |]
   ==> ∀ A B. A ≈ k #+ m & k ≈ B & B ⊆ A → A-B ≈ m"
⟨proof⟩

lemma eqpoll_sum_imp_Diff_eqpoll:
  "[| A ≈ succ(k #+ m); B ⊆ A; succ(k) ≈ B; k ∈ nat; m ∈ nat |]
   ==> A-B ≈ m"
⟨proof⟩

lemma subsets_lepoll_0_eq_unit: "{x ∈ Pow(X). x ⪻ 0} = {0}"
⟨proof⟩

lemma subsets_lepoll_succ:
  "n ∈ nat ==> {z ∈ Pow(y). z ⪻ succ(n)} =
   {z ∈ Pow(y). z ⪻ n} ∪ {z ∈ Pow(y). z ≈ succ(n)}"
⟨proof⟩

lemma Int_empty:
  "n ∈ nat ==> {z ∈ Pow(y). z ⪻ n} ∩ {z ∈ Pow(y). z ≈ succ(n)} =
  0"
⟨proof⟩

locale AC16 =
  fixes x and y and k and l and m and t_n and R and MM and LL and
  GG and s
  defines k_def: "k == succ(1)"
    and MM_def: "MM == {v ∈ t_n. succ(k) ⪻ v ∩ y}"
    and LL_def: "LL == {v ∩ y. v ∈ MM}"
    and GG_def: "GG == λv ∈ LL. (THE w. w ∈ MM & v ⊆ w) - v"
    and s_def: "s(u) == {v ∈ t_n. u ∈ v & k ⪻ v ∩ y}"
  assumes all_ex: "∀z ∈ {z ∈ Pow(x ∪ y) . z ≈ succ(k)}.
```

```

 $\exists ! w. w \in t\_n \ \& \ z \subseteq w$ 
and disjoint[iff]: "x ∩ y = 0"
and "includes": "t_n ⊆ {v ∈ Pow(x ∪ y). v ≈ succ(k #+ m)}"
and WO_R[iff]: "well_ord(y, R)"
and lnat[iff]: "l ∈ nat"
and mnat[iff]: "m ∈ nat"
and mpos[iff]: "0 < m"
and Infinite[iff]: "¬ Finite(y)"
and noLepoll: "¬ y ⊲ {v ∈ Pow(x). v ≈ m}"
begin

lemma knat [iff]: "k ∈ nat"
⟨proof⟩

lemma Diff_Finite_eqpoll: "[| l ≈ a; a ⊆ y |] ==> y - a ≈ y"
⟨proof⟩

lemma s_subset: "s(u) ⊆ t_n"
⟨proof⟩

lemma sI:
  "[| w ∈ t_n; cons(b, cons(u, a)) ⊆ w; a ⊆ y; b ∈ y - a; l ≈ a |]
   ==> w ∈ s(u)"
⟨proof⟩

lemma in_s_imp_u_in: "v ∈ s(u) ==> u ∈ v"
⟨proof⟩

lemma ex1_superset_a:
  "[| l ≈ a; a ⊆ y; b ∈ y - a; u ∈ x |]
   ==> ∃ ! c. c ∈ s(u) & a ⊆ c & b ∈ c"
⟨proof⟩

lemma the_eq_cons:
  "[| ∀ v ∈ s(u). succ(l) ≈ v ∩ y;
     l ≈ a; a ⊆ y; b ∈ y - a; u ∈ x |]
   ==> (THE c. c ∈ s(u) & a ⊆ c & b ∈ c) ∩ y = cons(b, a)"
⟨proof⟩

lemma y_lepoll_subset_s:

```

$\text{succ}(l) \approx v \cap y;$   
 $l \approx a; a \subseteq y; u \in x /]$   
 $\Rightarrow y \lesssim \{v \in s(u). a \subseteq v\}$ "  
*(proof)*

**lemma** *x\_imp\_not\_y* [*dest*]: "a ∈ x ==> a ⊈ y"  
*(proof)*

**lemma** *w\_Int\_eq\_w\_Diff*:  
 $w \subseteq x \cup y ==> w \cap (x - \{u\}) = w - cons(u, w \cap y)"$   
*(proof)*

**lemma** *w\_Int\_eqpoll\_m*:  
 $[| w \in \{v \in s(u). a \subseteq v\};$   
 $l \approx a; u \in x;$   
 $\forall v \in s(u). succ(l) \approx v \cap y |]$   
 $\Rightarrow w \cap (x - \{u\}) \approx m"$   
*(proof)*

**lemma** *eqpoll\_m\_not\_empty*: "a ≈ m ==> a ≠ 0"  
*(proof)*

**lemma** *cons\_cons\_in*:  
 $[| z \in xa \cap (x - \{u\}); l \approx a; a \subseteq y; u \in x |]$   
 $\Rightarrow \exists! w. w \in t\_n \& cons(z, cons(u, a)) \subseteq w"$   
*(proof)*

**lemma** *subset\_s\_lepoll\_w*:  
 $[| \forall v \in s(u). succ(l) \approx v \cap y; a \subseteq y; l \approx a; u \in x |]$   
 $\Rightarrow \{v \in s(u). a \subseteq v\} \lesssim \{v \in Pow(x). v \approx m\}"$   
*(proof)*

```

lemma well_ord_subsets_eqpoll_n:
  " $n \in \text{nat} \Rightarrow \exists S. \text{well\_ord}(\{z \in \text{Pow}(y) . z \approx \text{succ}(n)\}, S)$ "
(proof)

```

```

lemma well_ord_subsets_lepoll_n:
  " $n \in \text{nat} \Rightarrow \exists R. \text{well\_ord}(\{z \in \text{Pow}(y) . z \lesssim n\}, R)$ "
(proof)

```

```

lemma LL_subset: " $\text{LL} \subseteq \{z \in \text{Pow}(y) . z \lesssim \text{succ}(k \#+ m)\}$ "
(proof)

```

```

lemma well_ord_LL: " $\exists S. \text{well\_ord}(\text{LL}, S)$ "
(proof)

```

```

lemma unique_superset_in_MM:
  " $v \in \text{LL} \Rightarrow \exists ! w. w \in \text{MM} \& v \subseteq w$ "
(proof)

```

```

lemma Int_in_LL: " $w \in \text{MM} \Rightarrow w \cap y \in \text{LL}$ "
(proof)

```

```

lemma in_LL_eq_Int:
  " $v \in \text{LL} \Rightarrow v = (\text{THE } x. x \in \text{MM} \& v \subseteq x) \cap y$ "
(proof)

```

```

lemma unique_superset1: " $a \in \text{LL} \Rightarrow (\text{THE } x. x \in \text{MM} \wedge a \subseteq x) \in \text{MM}$ "
(proof)

```

```

lemma the_in_MM_subset:
  " $v \in \text{LL} \Rightarrow (\text{THE } x. x \in \text{MM} \& v \subseteq x) \subseteq x \cup y$ "
(proof)

```

```

lemma GG_subset: " $v \in \text{LL} \Rightarrow GG ` v \subseteq x$ "
(proof)

```

```

lemma nat_lepoll_ordertype: "nat  $\lesssim$  ordertype(y, R)"
<proof>

lemma ex_subset_eqpoll_n: "n  $\in$  nat ==>  $\exists z. z \subseteq y \ \& \ n \approx z$ "
<proof>

lemma exists_proper_in_s: "u  $\in$  x ==>  $\exists v \in s(u). succ(k) \lesssim v \cap y$ "
<proof>

lemma exists_in_MM: "u  $\in$  x ==>  $\exists w \in MM. u \in w$ "
<proof>

lemma exists_in_LL: "u  $\in$  x ==>  $\exists w \in LL. u \in GG^w$ "
<proof>

lemma OUN_eq_x: "well_ord(LL, S) ==>
  (\bigcup b < ordertype(LL, S). GG ` (converse(ordermap(LL, S)) ` b)) = x"
<proof>

lemma in_MM_eqpoll_n: "w  $\in$  MM ==> w  $\approx$  succ(k #+ m)"
<proof>

lemma in_LL_eqpoll_n: "w  $\in$  LL ==> succ(k)  $\lesssim$  w"
<proof>

lemma in_LL: "w  $\in$  LL ==> w  $\subseteq$  (THE x. x  $\in$  MM  $\wedge$  w  $\subseteq$  x)"
<proof>

lemma all_in_lepoll_m:
  "well_ord(LL, S) ==>
   \forall b < ordertype(LL, S). GG ` (converse(ordermap(LL, S)) ` b)  $\lesssim$  m"
<proof>

lemma "conclusion":
  "\exists a f. Ord(a)  $\wedge$  domain(f) = a  $\wedge$  (\bigcup b < a. f ` b) = x  $\wedge$  (\forall b < a. f ` b  $\lesssim$  m)"
<proof>

end

```

```

theorem AC16_W04:
  "[| AC_Equiv.AC16(k #+ m, k); 0 < k; 0 < m; k ∈ nat; m ∈ nat |]
==> W04(m)"
⟨proof⟩

end

theory AC17_AC1
imports HH
begin

lemma AC0_AC1_lemma: "[| f:(Π X ∈ A. X); D ⊆ A |] ==> ∃ g. g:(Π X ∈
D. X)"
⟨proof⟩

lemma AC0_AC1: "AC0 ==> AC1"
⟨proof⟩

lemma AC1_AC0: "AC1 ==> AC0"
⟨proof⟩

lemma AC1_AC17_lemma: "f ∈ (Π X ∈ Pow(A) - {0}. X) ==> f ∈ (Pow(A)
- {0} → A)"
⟨proof⟩

lemma AC1_AC17: "AC1 ==> AC17"
⟨proof⟩

lemma UN_eq_imp_well_ord:
  "[| x - (⋃ j ∈ μ i. HH(λX ∈ Pow(x)-{0}. {f`X}, x, i)) = {x}.
  HH(λX ∈ Pow(x)-{0}. {f`X}, x, j)) = 0;
  f ∈ Pow(x)-{0} → x |]
==> ∃ r. well_ord(x,r)"

```

$\langle proof \rangle$

**lemma** *not\_AC1\_imp\_ex*:  
 $"\neg AC1 \implies \exists A. \forall f \in Pow(A) - \{0\} \rightarrow A. \exists u \in Pow(A) - \{0\}. f'u \notin u"$   
 $\langle proof \rangle$

**lemma** *AC17\_AC1\_aux1*:  
 $"[\forall f \in Pow(x) - \{0\} \rightarrow x. \exists u \in Pow(x) - \{0\}. f'u \notin u;$   
 $\exists f \in Pow(x) - \{0\} \rightarrow x.$   
 $x = (\bigcup a \in (\mu i. HH(\lambda X \in Pow(x) - \{0\}. \{f'X\}, x, i) = \{x\}).$   
 $HH(\lambda X \in Pow(x) - \{0\}. \{f'X\}, x, a)) = 0]$   
 $\implies P"$   
 $\langle proof \rangle$

**lemma** *AC17\_AC1\_aux2*:  
 $"\neg (\exists f \in Pow(x) - \{0\} \rightarrow x. x - F(f) = 0)$   
 $\implies (\lambda f \in Pow(x) - \{0\} \rightarrow x . x - F(f))$   
 $\in (Pow(x) - \{0\} \rightarrow x) \rightarrow Pow(x) - \{0\}"$   
 $\langle proof \rangle$

**lemma** *AC17\_AC1\_aux3*:  
 $"[\forall f'Z \in Z; Z \in Pow(x) - \{0\}]$   
 $\implies (\lambda X \in Pow(x) - \{0\}. \{f'X\})'Z \in Pow(Z) - \{0\}"$   
 $\langle proof \rangle$

**lemma** *AC17\_AC1\_aux4*:  
 $"\exists f \in F. f'((\lambda f \in F. Q(f))'f) \in (\lambda f \in F. Q(f))'f$   
 $\implies \exists f \in F. f'Q(f) \in Q(f)"$   
 $\langle proof \rangle$

**lemma** *AC17\_AC1*: "AC17  $\implies$  AC1"  
 $\langle proof \rangle$

**lemma** *AC1\_AC2\_aux1*:  
 $"[\forall f : (\prod X \in A. X); B \in A; 0 \notin A] \implies \{f'B\} \subseteq B \cap \{f'C. C \in A\}"$   
 $\langle proof \rangle$

```

lemma AC1_AC2_aux2:
  "[| pairwise_disjoint(A); B ∈ A; C ∈ A; D ∈ B; D ∈ C |] ==>
  f ` B = f ` C"
  ⟨proof⟩

lemma AC1_AC2: "AC1 ==> AC2"
  ⟨proof⟩

lemma AC2_AC1_aux1: "0 ∉ A ==> 0 ∉ {B * {B}. B ∈ A}"
  ⟨proof⟩

lemma AC2_AC1_aux2: "[| X * {X} ∩ C = {y}; X ∈ A |]
  ==> (THE y. X * {X} ∩ C = {y}): X * A"
  ⟨proof⟩

lemma AC2_AC1_aux3:
  "∀ D ∈ {E * {E}. E ∈ A}. ∃ y. D ∩ C = {y}
   ==> (λx ∈ A. fst(THE z. (x * {x} ∩ C = {z}))) ∈ (Π X ∈ A. X)"
  ⟨proof⟩

lemma AC2_AC1: "AC2 ==> AC1"
  ⟨proof⟩

lemma empty_notin_images: "0 ∉ {R `` {x}. x ∈ domain(R)}"
  ⟨proof⟩

lemma AC1_AC4: "AC1 ==> AC4"
  ⟨proof⟩

lemma AC4_AC3_aux1: "f ∈ A -> B ==> (∪ z ∈ A. {z} * f ` z) ⊆ A * ∪ (B)"
  ⟨proof⟩

lemma AC4_AC3_aux2: "domain(∪ z ∈ A. {z} * f(z)) = {a ∈ A. f(a) ≠ 0}"
  ⟨proof⟩

```

**lemma** *AC4\_AC3\_aux3*: " $x \in A \Rightarrow (\bigcup z \in A. \{z\} * f(z)) `` \{x\} = f(x)$ "  
*(proof)*

**lemma** *AC4\_AC3*: "AC4  $\Rightarrow$  AC3"  
*(proof)*

**lemma** *AC3\_AC1\_lemma*:  
"  $b \notin A \Rightarrow (\prod x \in \{a \in A. id(A) ` a \neq b\}. id(A) ` x) = (\prod x \in A. x)$ "  
*(proof)*

**lemma** *AC3\_AC1*: "AC3  $\Rightarrow$  AC1"  
*(proof)*

**lemma** *AC4\_AC5*: "AC4  $\Rightarrow$  AC5"  
*(proof)*

**lemma** *AC5\_AC4\_aux1*: " $R \subseteq A * B \Rightarrow (\lambda x \in R. fst(x)) \in R \rightarrow A$ "  
*(proof)*

**lemma** *AC5\_AC4\_aux2*: " $R \subseteq A * B \Rightarrow range(\lambda x \in R. fst(x)) = domain(R)$ "  
*(proof)*

**lemma** *AC5\_AC4\_aux3*: "[| \exists f \in A \rightarrow C. P(f, domain(f)); A = B |] \Rightarrow \exists f \in B \rightarrow C. P(f, B)"  
*(proof)*

**lemma** *AC5\_AC4\_aux4*: "[| R \subseteq A \* B; g \in C \rightarrow R; \forall x \in C. (\lambda z \in R. fst(z)) ` (g ` x) = x |]  
 $\Rightarrow (\lambda x \in C. snd(g ` x)) : (\prod x \in C. R `` \{x\})$ "  
*(proof)*

**lemma** *AC5\_AC4*: "AC5  $\Rightarrow$  AC4"  
*(proof)*

```

lemma AC1_iff_AC6: "AC1  $\longleftrightarrow$  AC6"
⟨proof⟩

end

theory AC18_AC19
imports AC_Equiv
begin

definition
uu :: "i => i" where
"uu(a) == {c ∪ {0}. c ∈ a}"

lemma PROD_subsets:
" [| f ∈ (Π b ∈ {P(a). a ∈ A}. b); ∀ a ∈ A. P(a) ≤ Q(a) |]
  ==> (λa ∈ A. f ` P(a)) ∈ (Π a ∈ A. Q(a))"
⟨proof⟩

lemma lemma_AC18:
" [| ∀ A. 0 ∉ A → (∃ f. f ∈ (Π X ∈ A. X)); A ≠ 0 |]
  ==> (⋂ a ∈ A. ⋃ b ∈ B(a). X(a, b)) ⊆
      (⋃ f ∈ Π a ∈ A. B(a). ⋂ a ∈ A. X(a, f ` a))"
⟨proof⟩

lemma AC1_AC18: "AC1 ==> PROP AC18"
⟨proof⟩

theorem (in AC18) AC19
⟨proof⟩

lemma RepRep_conj:

```

```

    "[| A ≠ 0; 0 ∉ A |] ==> {uu(a). a ∈ A} ≠ 0 & 0 ∉ {uu(a). a
∈ A}""
⟨proof⟩

lemma lemma1_1: "[|c ∈ a; x = c ∪ {0}; x ∉ a |] ==> x - {0} ∈ a"
⟨proof⟩

lemma lemma1_2:
    "[| f‘(uu(a)) ∉ a; f ∈ (Π B ∈ {uu(a). a ∈ A}. B); a ∈ A |]
==> f‘(uu(a))-{0} ∈ a"
⟨proof⟩

lemma lemma1: "∃ f. f ∈ (Π B ∈ {uu(a). a ∈ A}. B) ==> ∃ f. f ∈ (Π B
∈ A. B)"
⟨proof⟩

lemma lemma2_1: "a ≠ 0 ==> 0 ∈ (⋃ b ∈ uu(a). b)"
⟨proof⟩

lemma lemma2: "[| A ≠ 0; 0 ∉ A |] ==> (⋂ x ∈ {uu(a). a ∈ A}. ⋃ b ∈ x.
b) ≠ 0"
⟨proof⟩

lemma AC19_AC1: "AC19 ==> AC1"
⟨proof⟩

end

```

```

theory DC
imports AC_Equiv Hartog Cardinal_aux
begin

lemma RepFun_lepoll: "Ord(a) ==> {P(b). b ∈ a} ≤ a"
⟨proof⟩

Trivial in the presence of AC, but here we need a wellordering of X

lemma image_Ord_lepoll: "[| f ∈ X → Y; Ord(X) |] ==> f‘‘X ≤ X"
⟨proof⟩

lemma range_subset_domain:
    "[| R ⊆ X * X; !!g. g ∈ X ==> ∃ u. ⟨g, u⟩ ∈ R |]
==> range(R) ⊆ domain(R)"
⟨proof⟩

lemma cons_fun_type: "g ∈ n → X ==> cons(<n, x>, g) ∈ succ(n) → cons(x,
X)"
⟨proof⟩

```

```

lemma cons_fun_type2:
  "[| g ∈ n->X; x ∈ X |] ==> cons(<n,x>, g) ∈ succ(n) -> X"
⟨proof⟩

lemma cons_image_n: "n ∈ nat ==> cons(<n,x>, g)`n = g`n"
⟨proof⟩

lemma cons_val_n: "g ∈ n->X ==> cons(<n,x>, g)`n = x"
⟨proof⟩

lemma cons_image_k: "k ∈ n ==> cons(<n,x>, g)`k = g`k"
⟨proof⟩

lemma cons_val_k: "[| k ∈ n; g ∈ n->X |] ==> cons(<n,x>, g)`k = g`k"
⟨proof⟩

lemma domain_cons_eq_succ: "domain(f)=x ==> domain(cons(<x,y>, f)) =
succ(x)"
⟨proof⟩

lemma restrict_cons_eq: "g ∈ n->X ==> restrict(cons(<n,x>, g), n) =
g"
⟨proof⟩

lemma succ_in_succ: "[| Ord(k); i ∈ k |] ==> succ(i) ∈ succ(k)"
⟨proof⟩

lemma restrict_eq_imp_val_eq:
  "[| restrict(f, domain(g)) = g; x ∈ domain(g) |]
  ==> f`x = g`x"
⟨proof⟩

lemma domain_eq_imp_fun_type: "[| domain(f)=A; f ∈ B->C |] ==> f ∈ A->C"
⟨proof⟩

lemma ex_in_domain: "[| R ⊆ A * B; R ≠ 0 |] ==> ∃x. x ∈ domain(R)"
⟨proof⟩

```

#### definition

```

DC :: "i => o" where
  "DC(a) == ∀X R. R ⊆ Pow(X)*X &
    (∀Y ∈ Pow(X). Y ⊂ a → (∃x ∈ X. <Y,x> ∈ R))
    → (∃f ∈ a->X. ∀b<a. <f`b, f`b> ∈ R)"

```

#### definition

```

DC0 :: o where
  "DC0 == ∀A B R. R ⊆ A*B & R≠0 & range(R) ⊆ domain(R)
    → (∃f ∈ nat->domain(R). ∀n ∈ nat. <f`n, f`succ(n)>:R)"

```

```

definition
  ff :: "[i, i, i, i] => i" where
    "ff(b, X, Q, R) ==
      transrec(b, %c r. THE x. first(x, {x ∈ X. <r `` c, x> ∈ R},
      Q))"

locale DCO_imp =
  fixes XX and RR and X and R
  assumes all_ex: "∀ Y ∈ Pow(X). Y ⊲ nat → (∃ x ∈ X. <Y, x> ∈ R)"
  defines XX_def: "XX == (⋃ n ∈ nat. {f ∈ n->X. ∀ k ∈ n. <f `` k, f ` k> ∈
  R})"
    and RR_def: "RR == {<z1, z2>:XX*XX. domain(z2)=succ(domain(z1))
    & restrict(z2, domain(z1)) = z1}"
begin

lemma lemma1_1: "RR ⊆ XX*XX"
  ⟨proof⟩

lemma lemma1_2: "RR ≠ 0"

```

```

⟨proof⟩

lemma lemma1_3: "range(RR) ⊆ domain(RR)"
⟨proof⟩

lemma lemma2:
  "[| ∀ n ∈ nat. <f‘n, f‘succ(n)> ∈ RR; f ∈ nat -> XX; n ∈ nat |]

  ==> ∃ k ∈ nat. f‘succ(n) ∈ k -> X & n ∈ k
    & <f‘succ(n)‘‘n, f‘succ(n)‘n> ∈ R"
⟨proof⟩

lemma lemma3_1:
  "[| ∀ n ∈ nat. <f‘n, f‘succ(n)> ∈ RR; f ∈ nat -> XX; m ∈ nat |]

  ==> {f‘succ(x)‘x. x ∈ m} = {f‘succ(m)‘x. x ∈ m}"
⟨proof⟩

lemma lemma3:
  "[| ∀ n ∈ nat. <f‘n, f‘succ(n)> ∈ RR; f ∈ nat -> XX; m ∈ nat |]

  ==> (λx ∈ nat. f‘succ(x)‘x) ‘‘ m = f‘succ(m)‘‘m"
⟨proof⟩

end

theorem DCO_imp_DC_nat: "DCO ==> DC(nat)"
⟨proof⟩

lemma singleton_in_funcs:
  "x ∈ X ==> {<0,x>} ∈
   (⋃n ∈ nat. {f ∈ succ(n)->X. ∀ k ∈ n. <f‘k, f‘succ(k)> ∈
   R})"
⟨proof⟩

locale imp_DCO =
  fixes XX and RR and x and R and f and allRR
  defines XX_def: "XX == (⋃n ∈ nat.
    {f ∈ succ(n)->domain(R). ∀ k ∈ n. <f‘k, f‘succ(k)> ∈
    R})"
  and RR_def:
    "RR == {<z1,z2>:Fin(XX)*XX.
      (domain(z2)=succ(⋃f ∈ z1. domain(f)))
      & (∀f ∈ z1. restrict(z2, domain(f)) = f))
      | (¬ (∃g ∈ XX. domain(g)=succ(⋃f ∈ z1. domain(f)))
```

```

& ( $\forall f \in z1. \text{restrict}(g, \text{domain}(f)) = f$ ) &  $z2 = \{<0, x>\}\})\}"$ 
```

and `allRR_def`:

```

"allRR ==  $\forall b < \text{nat}.$ 
< $f`b, f`b> \in$ 
 $\{<z1, z2> \in \text{Fin}(XX) * XX. (\text{domain}(z2) = \text{succ}(\bigcup f \in z1. \text{domain}(f))$ 
&  $(\bigcup f \in z1. \text{domain}(f)) = b$ 
&  $(\forall f \in z1. \text{restrict}(z2, \text{domain}(f))$ 
=  $f$ )\}"
```

`begin`

**lemma lemma4:**

```

"[] range(R)  $\subseteq \text{domain}(R); x \in \text{domain}(R) []$ 
==>  $RR \subseteq \text{Pow}(XX) * XX \&$ 
 $(\forall Y \in \text{Pow}(XX). Y \prec \text{nat} \longrightarrow (\exists x \in XX. <Y, x> : RR))$ "
```

*(proof)*

**lemma UN\_image\_succ\_eq:**

```

"[]  $f \in \text{nat} \rightarrow X; n \in \text{nat} []$ 
==>  $(\bigcup x \in f``\text{succ}(n). P(x)) = P(f`n) \cup (\bigcup x \in f``n. P(x))$ "
```

*(proof)*

**lemma UN\_image\_succ\_eq\_succ:**

```

"[]  $(\bigcup x \in f``n. P(x)) = y; P(f`n) = \text{succ}(y);$ 
 $f \in \text{nat} \rightarrow X; n \in \text{nat} [] \Rightarrow (\bigcup x \in f``\text{succ}(n). P(x)) = \text{succ}(y)$ "
```

*(proof)*

**lemma apply\_domain\_type:**

```

"[]  $h \in \text{succ}(n) \rightarrow D; n \in \text{nat}; \text{domain}(h) = \text{succ}(y) [] \Rightarrow h`y \in D$ "
```

*(proof)*

**lemma image\_fun\_succ:**

```

"[]  $h \in \text{nat} \rightarrow X; n \in \text{nat} [] \Rightarrow h``\text{succ}(n) = \text{cons}(h`n, h``n)$ "
```

*(proof)*

**lemma f\_n\_type:**

```

"[]  $\text{domain}(f`n) = \text{succ}(k); f \in \text{nat} \rightarrow XX; n \in \text{nat} []$ 
==>  $f`n \in \text{succ}(k) \rightarrow \text{domain}(R)$ "
```

*(proof)*

**lemma f\_n\_pairs\_in\_R [rule\_format]:**

```

"[]  $h \in \text{nat} \rightarrow XX; \text{domain}(h`n) = \text{succ}(k); n \in \text{nat} []$ 
==>  $\forall i \in k. <h`n`i, h`n`\text{succ}(i)> \in R$ "
```

*(proof)*

**lemma restrict\_cons\_eq\_restrict:**

```

"[]  $\text{restrict}(h, \text{domain}(u)) = u; h \in n \rightarrow X; \text{domain}(u) \subseteq n []$ 
==>  $\text{restrict}(\text{cons}(<n, y>, h), \text{domain}(u)) = u$ "
```

*(proof)*

```

lemma all_in_image_restrict_eq:
  "[| ∀ x ∈ f `` n. restrict(f ` n, domain(x)) = x;
    f ∈ nat → XX;
    n ∈ nat; domain(f ` n) = succ(n);
    (⋃ x ∈ f `` n. domain(x)) ⊆ n |]
  ==> ∀ x ∈ f `` succ(n). restrict(cons(<succ(n), y>, f ` n), domain(x))
  = x"
⟨proof⟩

lemma simplify_recursion:
  "[| ∀ b < nat. <f `` b, f ` b> ∈ RR;
    f ∈ nat → XX; range(R) ⊆ domain(R); x ∈ domain(R) |]
  ==> allRR"
⟨proof⟩

lemma lemma2:
  "[| allRR; f ∈ nat → XX; range(R) ⊆ domain(R); x ∈ domain(R); n ∈
  nat |]
  ==> f ` n ∈ succ(n) → domain(R) & (∀ i ∈ n. <f ` n ` i, f ` n ` succ(i)> : R)"
⟨proof⟩

lemma lemma3:
  "[| allRR; f ∈ nat → XX; n ∈ nat; range(R) ⊆ domain(R); x ∈ domain(R)
  |]
  ==> f ` n ` n = f ` succ(n) ` n"
⟨proof⟩

end

theorem DC_nat_imp_DC0: "DC(nat) ==> DC0"
⟨proof⟩

lemma fun_Ord_inj:
  "[| f ∈ a → X; Ord(a);
    !!b c. [| b < c; c ∈ a |] ==> f ` b ≠ f ` c |]
  ==> f ∈ inj(a, X)"
⟨proof⟩

lemma value_in_image: "[| f ∈ X → Y; A ⊆ X; a ∈ A |] ==> f ` a ∈ f `` A"
⟨proof⟩

lemma lesspoll_lemma: "[| ~ A ⊑ B; C ⊑ B |] ==> A - C ≠ 0"

```

$\langle proof \rangle$

**theorem** DC\_W03: " $(\forall K. Card(K) \rightarrow DC(K)) \Rightarrow W03$ "  
 $\langle proof \rangle$

**lemma** images\_eq:  
" $[| \forall x \in A. f'x = g'x; f \in Df \rightarrow Cf; g \in Dg \rightarrow Cg; A \subseteq Df; A \subseteq Dg |]$

$\Rightarrow f''A = g''A$ "

$\langle proof \rangle$

**lemma** lam\_images\_eq:  
" $[| Ord(a); b \in a |] \Rightarrow (\lambda x \in a. h(x))''b = (\lambda x \in b. h(x))''b$ "  
 $\langle proof \rangle$

**lemma** lam\_type\_RepFun: " $(\lambda b \in a. h(b)) \in a \rightarrow \{h(b). b \in a\}$ "  
 $\langle proof \rangle$

**lemma** lemmaX:  
" $[| \forall Y \in Pow(X). Y \prec K \rightarrow (\exists x \in X. \langle Y, x \rangle \in R); b \in K; Z \in Pow(X); Z \prec K |]$   
 $\Rightarrow \{x \in X. \langle Z, x \rangle \in R\} \neq 0$ "  
 $\langle proof \rangle$

**lemma** W01\_DC\_lemma:  
" $[| Card(K); well_ord(X, Q); \forall Y \in Pow(X). Y \prec K \rightarrow (\exists x \in X. \langle Y, x \rangle \in R); b \in K |]$   
 $\Rightarrow ff(b, X, Q, R) \in \{x \in X. \langle (\lambda c \in b. ff(c, X, Q, R))''b, x \rangle \in R\}$ "  
 $\langle proof \rangle$

**theorem** W01\_DC\_Card: " $W01 \Rightarrow \forall K. Card(K) \rightarrow DC(K)$ "  
 $\langle proof \rangle$

end

## References

- [1] Lawrence C. Paulson and Krzysztof Grąbczewski. Mechanizing set theory: Cardinal arithmetic and the axiom of choice. *Journal of Automated Reasoning*, 17(3):291–323, December 1996.
- [2] Herman Rubin and Jean E. Rubin. *Equivalents of the Axiom of Choice*,

*II.* North-Holland, 1985.