

FCFamilies

Filip Marić, Bojan Vučković, Miodrag Živković

April 8, 2018

Contents

1	Combinatorics	4
1.1	Generating all permutations	4
1.2	Generating all combinations	6
1.3	Generating all nonempty subsets	17
1.4	Generating all m-elements subsets of n-element set	17
2	Representing lists of natural numbers by natural numbers	18
2.1	sumpows	18
2.2	nat2list	24
2.3	list2nat	30
3	Union closed families	32
3.1	Union closed families	32
3.1.1	Closure – minimal union closed family containing F	33
3.2	Union closed families closed to unions with an additional family	36
3.2.1	Union closed extensions	38
4	Families of sets	38
5	Frankl’s condition	39
5.1	Frankl’s condition	39
5.1.1	Frankl’s function	39
5.2	FC Families	42
6	Executable implementations of set families	44
6.1	Abstract representation of sets and families	44
6.1.1	Implementation of sets by sorted and distinct lists	46
6.1.2	Implementation of sets of nats by nats	47
6.2	Abstract representation of union closed families	50
6.2.1	Implementation by sorted and distinct lists	57
6.2.2	Implementation by sets represented as natural numbers	61

7	Weights and shares	65
7.1	Weight functions	65
7.2	Shares	67
7.3	Hypercube construction	69
7.4	Frankl's families characterization using weights	72
7.5	Frankl's families characterization using shares	74
8	FC families characterization using shares (Poonen's theorem)	75
8.1	HyperCube projection	75
8.2	All shares non-negative	78
9	nonFC families characterization using shares (Poonen's theorem)	81
10	SomeShareNegative — combinatorial search for union-closed extension with a negative share	98
10.1	Abstract representation of sets and families with weights and shares	103
10.1.1	Implementation by sorted and distinct lists	104
10.1.2	Implementation by natural numbers	104
10.2	Weight functions implemented by mappings	105
10.2.1	Representation of sets by lists	105
10.2.2	Representation of sets by natural numbers	105
11	SomeShareNegative – executable implementation	106
11.1	Refinement of SomeShareNegativeFunction	106
11.1.1	Initial version	106
11.1.2	Passing current family share as a parameter	109
11.1.3	Caching set shares	110
11.1.4	Passing the sum of shares of elements in the current list as a parameter	113
11.1.5	Incremental insert and close operation and calculation of share of the extended family	115
11.1.6	Final implementation	118
11.2	Interpretations	122
11.2.1	Representation of sets by lists	122
11.2.2	Representation of sets by natural numbers	122
12	Isomorphisms of set families	123
12.1	Properties preserved by injective functions	124
12.2	Injective embedding	129
12.3	Isomorphism definition	131
12.4	Iso-representing collections of families	133

12.5	Non-isomorphic families of sets	133
12.5.1	Implementation by sets represented by (sorted and distinct) lists	138
13	Expressible sets and irreducible families	142
13.1	Irreducible families	143
13.2	Removing all expressible sets	143
13.3	Uniqueness of irreducible subfamily for the given closure . . .	147
13.3.1	Implementation by sorted and distinct lists	150
14	Covering	152
14.1	Definition of Covering	152
14.2	Covering and empty set	154
14.3	Covering and isomorphic families	154
14.4	Covering, closure and irreducible families	158
14.4.1	FC and nonFC covering implementation	159
15	L-partitioned families	163
15.1	Ordering of L-partitions	168
16	Generating all L-partitioned families with some given prop- erties	172
16.1	Implementation by sorted and distinct lists	176
16.2	Recursive enumeration procedure	181
16.3	Dynamic programming enumeration procedure	183
16.4	Generating all irreducible families	194
16.5	Generating all families that are not FCs covered by the given collection	196
16.5.1	Implementation	198
16.6	Generating all irreducible families that are not FCs covered by the given collection	200
16.6.1	Implementation	201
17	Minimal FC(6) families and maximal nonFC(6) families	204
18	Tactics	211
18.1	Tactics for verifying FC families	211
18.2	Tactics for verifying nonFC families	212
19	FC status proofs	214
20	Proof that all families are covered	359

1 Combinatorics

```

theory Combinatorics
imports Main
        HOL-Library.Permutation
        HOL-Library.List-lexord
        More.MoreList
begin

```

1.1 Generating all permutations

```

primrec interleave :: 'a  $\Rightarrow$  'a list  $\Rightarrow$  'a list list where
  interleave x [] = [[x]]
| interleave x (h # t) = (x # (h # t)) # (map ( $\lambda$  l. h # l) (interleave x t))

```

For example, *interleave* 1 [2, 3, 4] = [[1, 2, 3, 4], [2, 1, 3, 4], [2, 3, 1, 4], [2, 3, 4, 1]].

```

primrec permute :: 'a list  $\Rightarrow$  'a list list where
  permute [] = [[]]
| permute (h # t) = concat (map ( $\lambda$  l. interleave h l) (permute t))

```

For example, *permute* [1, 2, 3] = [[1, 2, 3], [2, 1, 3], [2, 3, 1], [1, 3, 2], [3, 1, 2], [3, 2, 1]].

```

lemma multiset-interleave:
  shows  $p \in \text{set } (\text{interleave } h \ a) \implies \text{mset } p = \text{mset } a + \{\#h\# \}$ 
proof (induct a arbitrary: p)
  case Nil
  thus ?case
  by simp
next
  case (Cons h' t')
  thus ?case
    using add.commute [of {#h#} {#h'#}]
    using add.assoc [of mset t' {#h#} {#h'#}]
    using add.assoc [of mset t' {#h'#} {#h#}]
    by auto
qed

```

```

lemma interleave-hd [simp]:  $h \# t \in \text{set } (\text{interleave } h \ t)$ 
by (cases t) auto

```

```

lemma interleave-append [simp]:  $t1 \ @ \ [h] \ @ \ t2 \in \text{set } (\text{interleave } h \ (t1 \ @ \ t2))$ 
by (induct t1) auto

```

```

lemma isPermutation-permute:
  shows  $p \in \text{set } (\text{permute } l) \implies p <\sim\sim> l$ 
proof (induct l arbitrary: p)
  case Nil
  thus ?case

```

```

      by (simp add: mset-eq-perm)
next
  case (Cons h t)
  thus ?case
    unfolding mset-eq-perm[THEN sym]
    by (auto simp add: multiset-interleave)
qed

```

```

lemma mset-append:
  assumes mset p = mset t + {#h#}
  shows  $\exists t1\ t2. p = t1 @ [h] @ t2 \wedge mset\ t = mset\ t1 + mset\ t2$ 
using assms
proof (induct p arbitrary: t)
  case Nil
  thus ?case
    by simp
next
  case (Cons a p')
  show ?case
  proof (cases a = h)
    case True
    thus ?thesis
      using Cons(2)
      by (rule-tac x=[] in exI) auto
  next
    case False
    hence  $h \in\# mset\ p'$ 
      using Cons(2) insert-noteq-member[of a mset p']
      by auto
    hence  $mset\ p' = mset\ p' - \{ \#h\# \} + \{ \#h\# \}$ 
      using insert-DiffM2[of h mset p']
      by simp
    thus ?thesis
      using Cons(1)[of remove1 h p'] Cons(2)
      by (auto, rule-tac x=a#t1 in exI, auto)
  qed
qed

```

```

lemma permute-isPermutation:
   $p <\sim\sim> l \implies p \in set\ (permute\ l)$ 
proof (induct l arbitrary: p)
  case Nil
  thus ?case
    by simp
next
  case (Cons h t)
  obtain t1 t2 where  $p = t1 @ [h] @ t2$  and  $mset\ t = mset\ t1 + mset\ t2$ 

```

```

using Cons(2)
unfolding mset-eq-perm[THEN sym]
using mset-append[of p t h]
by auto
thus ?case
using Cons(1)[of t1 @ t2] Cons(2)
using add.assoc [of mset t1 mset t2 {#h#}]
using interleave-append[of t1 h t2]
unfolding mset-eq-perm[THEN sym]
by auto
qed

```

```

lemma permute-bij:
assumes p ∈ set (permute l)
shows ∃ f. bij-betw f {..using assms
using isPermutation-permute[of p l]
using permutation-Ex-bij[of p l]
by auto

```

1.2 Generating all combinations

```

fun combine-aux :: 'a list ⇒ nat ⇒ nat ⇒ 'a list list where
  combine-aux l n k =
    (if k = 0 then [[]] else
     if k = n then [l] else
     (case l of
      [] ⇒ []
      | (h # t) ⇒
        (map (λ l'. h # l') (combine-aux t (n-(1::nat)) (k-(1::nat)))) @
        combine-aux t (n-(1::nat)) k))
declare combine-aux.simps[simp del]

```

```

lemma combine-aux-induct:
assumes
  ∧ l n. P [[]] l n 0
  ∧ l n. 0 < n ⇒ P [l] l n n
  ∧ k n. [0 < k; k ≠ n] ⇒ P [] [] n k
  ∧ h t n k. [
    P (combine-aux t (n-(1::nat)) (k-(1::nat))) t (n-(1::nat)) (k-(1::nat));
    P (combine-aux t (n-(1::nat)) k) t (n-(1::nat)) k] ⇒
    P (map (op # h) (combine-aux t (n-(1::nat)) (k-(1::nat)))) @
      combine-aux t (n-(1::nat)) k (h # t) n k
shows P (combine-aux l n k) l n k
using assms
proof (induct l n k rule: combine-aux.induct)
case (1 l n k)

```

```

show ?case
proof (cases k=0)
  case True
  thus ?thesis
    using 1(3)
    by (simp add: combine-aux.simps)
next
case False
show ?thesis
proof (cases k = n)
  case True
  thus ?thesis
    using ⟨k ≠ 0⟩
    using 1(4)
    by (simp add: combine-aux.simps)
next
case False
show ?thesis
proof (cases l = [])
  case True
  thus ?thesis
    using ⟨k ≠ 0⟩ ⟨k ≠ n⟩ 1(5)
    by (simp add: combine-aux.simps)
next
case False
then obtain h t where l = h # t by (auto simp add: neq-Nil-conv)
let ?l1 = combine-aux t (n-(1::nat)) (k-(1::nat))
let ?l2 = combine-aux t (n-(1::nat)) k
have P ?l1 t (n-(1::nat)) (k-(1::nat))
  using 1(1)[of h t] 1(3-6) ⟨k ≠ 0⟩ ⟨k ≠ n⟩ ⟨l = h # t⟩
  by auto
moreover
have P ?l2 t (n-(1::nat)) k
  using 1(2)[of h t] 1(3-6) ⟨k ≠ 0⟩ ⟨k ≠ n⟩ ⟨l = h # t⟩
  by auto
ultimately
show ?thesis
  using ⟨l = h # t⟩ ⟨k ≠ 0⟩ ⟨k ≠ n⟩
  using combine-aux.simps[of l n k] 1(6)
  by simp
qed
qed
qed
qed

```

definition *combine* :: 'a list \Rightarrow nat \Rightarrow 'a list list **where**
combine l k = combine-aux l (length l) k

For example, *combine* [1, 2, 3] 2 = [[1, 2], [1, 3], [2, 3]].

```

lemma combine-aux-subset:
  shows  $\forall A. A \in \text{set } (\text{combine-aux } l \ n \ k) \longrightarrow \text{set } A \subseteq \text{set } l$ 
by (rule combine-aux-induct) (auto, force)

lemma combine-subset:
  assumes  $A \in \text{set } (\text{combine } l \ k)$ 
  shows  $\text{set } A \subseteq \text{set } l$ 
using assms
using combine-aux-subset[rule-format, of  $A \ l \ \text{length } l \ k$ ]
unfolding combine-def
by simp

lemma combine-aux-sublist:
  shows  $\forall A. A \subseteq \text{set } L \wedge n = \text{length } L \wedge m = \text{card } A \longrightarrow (\exists x \in \text{set } (\text{combine-aux } L \ n \ m). A = \text{set } x)$ 
proof (rule combine-aux-induct[where  $P = \lambda c \ l \ n \ m. (\forall A. A \subseteq \text{set } l \wedge n = \text{length } l \wedge m = \text{card } A \longrightarrow (\exists x \in \text{set } c. A = \text{set } x))$ ])
  fix  $l :: 'a \text{ list}$  and  $n :: \text{nat}$ 
  show  $\forall A. A \subseteq \text{set } l \wedge n = \text{length } l \wedge 0 = \text{card } A \longrightarrow (\exists x \in \text{set } []. A = \text{set } x)$ 
  by (auto simp add: card-eq-0-iff finite-subset)
next
  fix  $l :: 'a \text{ list}$  and  $n :: \text{nat}$ 
  assume  $n > (0 :: \text{nat})$ 
  thus  $\forall A. A \subseteq \text{set } l \wedge n = \text{length } l \wedge n = \text{card } A \longrightarrow (\exists x \in \text{set } [l]. A = \text{set } x)$ 
  using subset-card-length
  by auto
next
  fix  $h :: 'a$  and  $t :: 'a \text{ list}$  and  $n :: \text{nat}$  and  $k :: \text{nat}$ 
  let  $?l1 = \text{combine-aux } t \ (n - 1) \ (k - 1)$  and
   $?l2 = (\text{combine-aux } t \ (n - 1) \ k)$ 
  assume  $*$ :  $\forall A. A \subseteq \text{set } t \wedge n - 1 = \text{length } t \wedge k - 1 = \text{card } A \longrightarrow (\exists x \in \text{set } ?l1. A = \text{set } x)$  and
   $**$ :  $\forall A. A \subseteq \text{set } t \wedge n - 1 = \text{length } t \wedge k = \text{card } A \longrightarrow (\exists x \in \text{set } ?l2. A = \text{set } x)$ 
  show  $\forall A. A \subseteq \text{set } (h \# t) \wedge n = \text{length } (h \# t) \wedge k = \text{card } A \longrightarrow$ 
   $(\exists x \in \text{set } (\text{map } (\text{op } \# \ h) \ ?l1) \ @ \ ?l2). A = \text{set } x)$ 
proof (rule allI, rule impI, (erule conjE)+)
  fix  $A'$ 
  assume  $A' \subseteq \text{set } (h \# t) \wedge n = \text{length } (h \# t) \wedge k = \text{card } A'$ 
  show  $\exists x \in \text{set } (\text{map } (\text{op } \# \ h) \ ?l1) \ @ \ ?l2. A' = \text{set } x$ 
  proof (cases  $h \in A'$ )
  case False
  hence  $A' \subseteq \text{set } t$ 
  using  $\langle A' \subseteq \text{set } (h \# t) \rangle$ 
  by auto
  thus ?thesis
  using **[rule-format, of  $A'$ ]  $\langle n = \text{length } (h \# t) \rangle \langle k = \text{card } A' \rangle$ 
  by auto
next

```



```

    case True
    hence  $\exists x \in \text{set } ?l1. (A' - \{h\}) = \text{set } x$ 
    using  $*[\text{rule-format, of } A' - \{h\}] \langle n = \text{length } (h \# t) \rangle \langle k = \text{card } A' \rangle \langle A' \subseteq \text{set } (h \# t) \rangle$ 
    using  $\text{card-Diff-singleton}[\text{of } A' h] \text{ finite-subset}[\text{of } A' \text{ set } (h \# t)]$ 
    by auto
    then obtain  $x$  where  $x \in \text{set } ?l1 \ (A' - \{h\}) = \text{set } x$ 
    by auto
    thus ?thesis
    using  $\langle h \in A' \rangle$ 
    by (rule-tac  $x=h \# x$  in  $\text{bexI}$ ) auto
  qed
qed
qed simp

```

```

lemma combine-sublist:
  assumes  $A \subseteq \text{set } L$ 
  shows  $\exists x \in \text{set } (\text{combine } L (\text{card } A)). A = \text{set } x$ 
using assms
using combine-aux-sublist[rule-format, of  $A \ L \ \text{length } L \ \text{card } A$ ]
unfolding combine-def
by simp

```

```

lemma combine-aux-combines:
  assumes sorted  $l$  and distinct  $l$  and  $n = \text{length } l$ 
  shows  $A \in \text{set } (\text{combine-aux } l \ n \ k) \longleftrightarrow \text{sorted } A \wedge \text{distinct } A \wedge \text{length } A = k \wedge \text{set } A \subseteq \text{set } l$ 
using assms
proof (induct  $l \ n \ k$  arbitrary:  $A$  rule: combine-aux.induct)
  case (1  $l \ n \ k$ )
  show ?case
  proof (cases  $k = 0$ )
    case True
    thus ?thesis
    by (auto simp add: combine-aux.simps)
  next
    case False
    show ?thesis
    proof (cases  $k = n$ )
      case True
      thus ?thesis
      using  $\langle k \neq 0 \rangle \langle \text{sorted } l \rangle \langle \text{distinct } l \rangle \langle n = \text{length } l \rangle$ 
      using distinct-card[of  $l$ ] distinct-card[of  $A$ ]
      using card-seteq[of  $\text{set } l \ \text{set } A$ ]
      using sorted-distinct-set-unique[of  $A \ l$ ]
      by (auto simp add: combine-aux.simps)
    next
      case False
      show ?thesis

```

```

proof (cases l = [])
  case True
  thus ?thesis
    using ⟨k ≠ 0⟩ ⟨k ≠ n⟩
    by (auto simp add: combine-aux.simps)
next
  case False
  then obtain h t where l = h # t by (auto simp add: neg-Nil-conv)
  let ?l1 = combine-aux t (n-(1::nat)) (k-(1::nat))
  let ?l2 = combine-aux t (n-(1::nat)) k
  have sorted t
    using 1(3) ⟨l = h # t⟩
    by (auto simp add: sorted-Cons)

  have A ∈ set (combine-aux l n k) = (A ∈ set (map (λ l'. h # l') ?l1) ∨ A
∈ set ?l2)
    using ⟨k ≠ 0⟩ ⟨k ≠ n⟩ ⟨l = h # t⟩
    by (simp add: combine-aux.simps)
  moreover
  have A ∈ set ?l2 = (sorted A ∧ distinct A ∧ length A = k ∧ set A ⊆ set t)
    using 1(2)[of h t A] ⟨k ≠ 0⟩ ⟨k ≠ n⟩ ⟨l = h # t⟩ ⟨sorted t⟩ 1(4) 1(5)
    by simp
  moreover
  have A ∈ set (map (λ l'. h # l') ?l1) = (sorted A ∧ distinct A ∧ length A
= k ∧ ¬ set A ⊆ set t ∧ set A ⊆ set l)
  proof–
    have (tl A ∈ set ?l1) =
      (sorted (tl A) ∧ distinct (tl A) ∧ length (tl A) = k - 1 ∧ set (tl A)
⊆ set t)
    using 1(1)[of h t tl A] ⟨l = h # t⟩ ⟨sorted t⟩ 1(4) 1(5) ⟨k ≠ 0⟩ ⟨k ≠ n⟩
    by simp
  moreover
  have A ∈ set (map (λ l'. h # l') ?l1) = (A = h # (tl A) ∧ (tl A ∈ set
?l1))
    by auto
  moreover
  have ∀ x ∈ set t. h ≤ x ∧ h ∉ set t
    using ⟨sorted l⟩ ⟨distinct l⟩ ⟨l = h # t⟩
    by (auto simp add: sorted-Cons)

  have (A = h # tl A ∧ sorted (tl A) ∧ distinct (tl A) ∧ length A - 1 = k
- 1 ∧ set (tl A) ⊆ set t) =
    (sorted A ∧ distinct A ∧ length A = k ∧ ¬ set A ⊆ set t ∧ set A ⊆
insert h (set t)) (is ?lhs = ?rhs)
  proof (rule iffI, (erule-tac[!] conjE)+)
    assume A = h # tl A sorted (tl A) distinct (tl A) length A - 1 = k -
1 set (tl A) ⊆ set t
    show ?rhs
    proof (safe)

```

```

show length A = k
proof-
  have length A ≠ 0
    using ⟨A = h # tl A⟩
    by auto
  show ?thesis
    using ⟨length A - 1 = k - 1⟩ ⟨k ≠ 0⟩ ⟨length A ≠ 0⟩
    by arith
qed
next
show sorted A
  apply (subst ⟨A = h # tl A⟩, subst sorted-Cons)
  using ⟨sorted (tl A)⟩ ⟨set (tl A) ⊆ set t⟩ ⟨∀ x ∈ set t. h ≤ x⟩
  by auto
next
show distinct A
  apply (subst ⟨A = h # tl A⟩)
  using ⟨distinct (tl A)⟩ ⟨h ∉ set t⟩ ⟨set (tl A) ⊆ set t⟩
  by auto
next
assume set A ⊆ set t
thus False
  apply (subst (asm) ⟨A = h # tl A⟩)
  using ⟨h ∉ set t⟩
  by auto
next
fix x
assume x ∈ set A x ∉ set t
thus x = h
  apply (subst (asm) ⟨A = h # tl A⟩)
  using ⟨set (tl A) ⊆ (set t)⟩
  by auto
qed
next
assume sorted A distinct A length A = k ¬ set A ⊆ set t set A ⊆ insert
h (set t)
show ?lhs
proof (safe)
  show sorted (tl A)
    using ⟨sorted A⟩
    by (rule linorder-class.sorted-tl)
next
show distinct (tl A)
  using ⟨distinct A⟩
  by (rule distinct-tl)
next
show A = h # tl A
proof-
  have h ∈ set A A ≠ []

```

```

    using  $\langle \neg \text{set } A \subseteq \text{set } t \rangle \langle \text{set } A \subseteq \text{insert } h (\text{set } t) \rangle$ 
    by auto
  have  $A = \text{hd } A \# \text{tl } A$ 
    using  $\langle A \neq [] \rangle$ 
    by simp
  show ?thesis
  proof (cases  $\text{hd } A = h$ )
    case True
    thus ?thesis
      using  $\langle A \neq [] \rangle$ 
      by auto
  next
    case False
    from  $\langle h \in \text{set } A \rangle$  have  $h = \text{hd } A \vee h \in \text{set } (\text{tl } A)$ 
      by (subst (asm)  $\langle A = \text{hd } A \# \text{tl } A \rangle$ ) auto
    hence  $h \in \text{set } (\text{tl } A)$ 
      using  $\langle \text{hd } A \neq h \rangle$ 
      by auto
    have  $\text{hd } A < h$ 
      using  $\langle \text{sorted } A \rangle$ 
      apply (subst (asm)  $\langle A = \text{hd } A \# \text{tl } A \rangle$ )
      using  $\langle h \in \text{set } (\text{tl } A) \rangle \langle \text{hd } A \neq h \rangle$ 
      by (auto simp add: sorted-Cons)
    moreover
    have  $\text{hd } A \in \text{set } t$ 
      using  $\langle \text{set } A \subseteq \text{insert } h (\text{set } t) \rangle$ 
      apply (subst (asm)  $\langle A = \text{hd } A \# \text{tl } A \rangle$ )
      using  $\langle \text{hd } A \neq h \rangle$ 
      by simp
    hence  $h \leq \text{hd } A$ 
      using  $\langle \forall x \in \text{set } t. h \leq x \rangle$ 
      by simp
    ultimately
    show ?thesis
      by simp
  qed
qed

fix x
assume  $x \in \text{set } (\text{tl } A)$ 
show  $x \in \text{set } t$ 
proof-
  have  $x \in \text{set } A$ 
    using  $\langle x \in \text{set } (\text{tl } A) \rangle$ 
    by (subst  $\langle A = h \# \text{tl } A \rangle$ ) simp
  hence  $x \in \text{insert } h (\text{set } t)$ 
    using  $\langle \text{set } A \subseteq \text{insert } h (\text{set } t) \rangle$ 
    by auto
  have  $x \neq h$ 

```

```

      using ⟨distinct A⟩
      apply (subst (asm) ⟨A = h # tl A⟩)
      using ⟨x ∈ set (tl A)⟩
      by auto
    show x ∈ set t
      using ⟨x ∈ insert h (set t)⟩ ⟨x ≠ h⟩
      by auto
  qed
next
  show length A - 1 = k - 1
    using ⟨length A = k⟩
    by simp
  qed
qed
ultimately
show ?thesis
  using ⟨l = h # t⟩
  by simp
qed
ultimately
show ?thesis
  using ⟨l = h # t⟩
  by auto
qed
qed
qed
qed

lemma combine-combines:
  assumes
    sorted l and distinct l
  shows
     $A \in \text{set } (\text{combine } l \ k) \iff (\text{sorted } A \wedge \text{distinct } A \wedge \text{length } A = k \wedge \text{set } A \subseteq \text{set } l)$ 
  using assms
  unfolding combine-def
  using combine-aux-combines[of l length l A k]
  by simp

lemma combine-aux-length:
  assumes A ∈ set (combine-aux l n k) and length l = n
  shows length A = k
  using assms
  proof (induct l n k arbitrary: A rule: combine-aux.induct)
    case (1 l n k)
    show ?case
    proof (cases k = 0)
      case True
      thus ?thesis

```

```

    using 1(3)
    by (simp add: combine-aux.simps)
next
case False
show ?thesis
proof (cases k = n)
case True
thus ?thesis
  using ⟨k ≠ 0⟩ 1(3) 1(4)
  by (simp add: combine-aux.simps)
next
case False
show ?thesis
proof (cases l)
case Nil
thus ?thesis
  using ⟨k ≠ 0⟩ ⟨k ≠ n⟩ 1(3)
  by (simp add: combine-aux.simps)
next
case (Cons h t)
thus ?thesis
  using ⟨k ≠ 0⟩ ⟨k ≠ n⟩ 1(3) 1(4)
  using combine-aux.simps[of l n k]
  using 1(1)[of h t tl A] 1(2)[of h t A]
  by auto
qed
qed
qed
qed

```

```

lemma combine-length:
  assumes A ∈ set (combine l k)
  shows length A = k
using assms
using combine-aux-length[of A l length l k]
unfolding combine-def
by simp

```

```

lemma distinct-combine-aux:
  assumes n = length l and distinct l
  shows distinct (combine-aux l n k)
using assms
proof (induct l n k rule: combine-aux.induct)
case (1 l n k)
show ?case
proof (cases k = 0)
case True
thus ?thesis
  by (simp add: combine-aux.simps)

```

```

next
  case False
  show ?thesis
  proof (cases  $k = n$ )
    case True
    thus ?thesis
      by (simp add: combine-aux.simps)
  next
  case False
  show ?thesis
  proof (cases  $l$ )
    case Nil
    thus ?thesis
      using  $\langle k \neq 0 \rangle \langle k \neq n \rangle$ 
      by (simp add: combine-aux.simps)
  next
  case (Cons  $h\ t$ )
  thus ?thesis
    using  $\langle k \neq 0 \rangle \langle k \neq n \rangle$ 
    using combine-aux.simps[of l n k]
    using 1(1)[of h t] 1(2)[of h t] 1(3) 1(4)
    apply (auto simp add: distinct-map inj-on-def)
    using combine-aux-subset[rule-format, of - t length t k]
    by force
  qed
qed
qed
qed
qed

lemma distinct-combine:
  assumes distinct l
  shows distinct (combine l k)
using assms
using distinct-combine-aux[of length l l k]
unfolding combine-def
by simp

lemma sorted-prepend:  $\text{sorted } l = \text{sorted } (\text{map } (op \# h) l)$ 
by (induct  $l$ ) (auto simp add: sorted-Cons)

lemma sorted-combine-aux-lemma:
  fixes  $h :: 'a::\text{linorder}$ 
  assumes  $\forall x \in \text{set } t. h \leq x$  and  $h \notin \text{set } t$  and  $\text{set } l' \subseteq \text{set } t$  and  $l' \neq []$ 
  shows  $h \# l \leq l'$ 
proof -
  have  $hd\ l' \in \text{set } t$ 
  using  $\langle l' \neq [] \rangle \langle \text{set } l' \subseteq \text{set } t \rangle$ 
  by auto
  hence  $h < hd\ l'$ 

```

```

    using ⟨ $\forall x \in \text{set } t. h \leq x$ ⟩ ⟨ $h \notin \text{set } t$ ⟩
    by (cases  $h = \text{hd } l'$ ) auto
  thus ?thesis
    using ⟨ $l' \neq []$ ⟩ Cons-le-Cons[of  $h \ l \ \text{hd } l' \ \text{tl } l'$ ]
    by simp
qed

lemma sorted-combine-aux:
  assumes  $n = \text{length } l$  and sorted  $l$  and distinct  $l$ 
  shows sorted (combine-aux  $l \ n \ k$ )
using assms
proof (induct  $l \ n \ k$  rule: combine-aux.induct)
  case (1  $l \ n \ k$ )
  show ?case
  proof (cases  $k = 0$ )
    case True
    thus ?thesis
      by (simp add: combine-aux.simps)
  next
    case False
    show ?thesis
    proof (cases  $k = n$ )
      case True
      thus ?thesis
        by (simp add: combine-aux.simps)
    next
      case False
      show ?thesis
      proof (cases  $l$ )
        case Nil
        thus ?thesis
          using ⟨ $k \neq 0$ ⟩ ⟨ $k \neq n$ ⟩
          by (simp add: combine-aux.simps)
      next
        case (Cons  $h \ t$ )
        thus ?thesis
          using ⟨ $k \neq 0$ ⟩ ⟨ $k \neq n$ ⟩
          using combine-aux.simps[of  $l \ n \ k$ ]
          using 1(1)[of  $h \ t$ ] 1(2)[of  $h \ t$ ] 1(3) 1(4) 1(5)
          apply (auto simp add: distinct-map inj-on-def sorted-Cons sorted-append)
          apply (subst sorted-prepend[THEN sym], simp)
          apply (rule sorted-combine-aux-lemma[of  $t$ ])
          apply (simp add: combine-aux-subset)+
          apply (subst length-0-conv[THEN sym])
          using combine-aux-length
          by force
      qed
    qed
  qed

```


qed

lemma *sorted-combine*:
 assumes *sorted l and distinct l*
 shows *sorted (combine l k)*
using *assms*
using *sorted-combine-aux*[of length l l k]
unfolding *combine-def*
by *simp*

1.3 Generating all nonempty subsets

definition *all-nonempty-subsets* **where**
 all-nonempty-subsets F = concat (map ($\lambda k.$ combine F k) [1.. $\text{length } F + 1$])

lemma *all-subset*: *set (all-nonempty-subsets l) \subseteq {A. set A \subseteq set l \wedge length A \geq 1}*
unfolding *all-nonempty-subsets-def*
using *combine-subset combine-length*
by *force*

1.4 Generating all m-elements subsets of n-element set

definition *all-mn-subsets* **where**
 all-mn-subsets n m = combine [0.. n] m

lemma *all-mn-subsets*:
 set (map set (all-mn-subsets n m)) = {A. card A = m \wedge A \subseteq {0.. n }} (**is** *?lhs*
 = *?rhs*)
unfolding *all-mn-subsets-def*
using *combine-combines*[of [0.. n] - m]
by (*auto simp add: distinct-card*) (*metis distinct-card finite-atLeastLessThan finite-sorted-distinct-unique image-iff rev-finite-subset*)

lemma *all-mn-subsets-completeness*:
 assumes *card A = m A \subseteq {0.. n }*
 shows $\exists Al \in \text{set } (all-mn-subsets n m).$ *set Al = A*

proof –
 from *assms* **have** *A \in set (map set (all-mn-subsets n m))*
 using *all-mn-subsets*[of n m]
 by *auto*
 thus *?thesis*
 by *auto*
qed

lemma *all-mn-subsets-sorted-distinct*:
 assumes *A \in set (all-mn-subsets n m)*
 shows *sorted A \wedge distinct A*
using *assms*
unfolding *all-mn-subsets-def*

by (metis combine-combines distinct-upt sorted-upt)

lemma [simp]: $A \in \text{set } (\text{all-mn-subsets } n \ m) \implies \text{set } A \subseteq \{0..<n\}$
 using all-mn-subsets[of n m]
 by auto

end

2 Representing lists of natural numbers by natural numbers

theory ListNat
 imports Main More.MoreNat More.MoreBigOperators More.MoreList
 begin

In this section, we define of representation of sets of natural numbers by single natural numbers. The set $\{a_0, a_1, \dots, a_n\}$ is represented by $2^{a_0} + 2^{a_1} + \dots + 2^{a_n}$.

2.1 sumpows

If the set is stored in a distinct list, the function *sumpows2* returns the natural number that represents it.

abbreviation *sumpows2* where
 $\text{sumpows2 } l \equiv \text{sum-list } (\text{map } (\text{op } ^ (2::\text{nat})) l)$

lemma *sumpows2-mod*:
 assumes *finite X*
 shows $(\text{sum } (\text{op } ^ 2) X) \bmod 2 \neq (0::\text{nat}) \longleftrightarrow 0 \in X$
 proof
 assume $0 \in X$
 hence $X = (X - \{0\}) \cup \{0\}$
 by auto
 hence $\text{sum } (\text{op } ^ (2::\text{nat})) X = \text{sum } (\text{op } ^ 2) (\text{insert } 0 (X - \{0\}))$
 by simp
 also have $\dots = \text{sum } (\text{op } ^ 2) (X - \{0\}) + 1$
 by (subst sum.insert, simp-all add: (finite X))
 finally have $\text{sum } (\text{op } ^ (2::\text{nat})) X = \dots$
 .
 moreover
 have $\text{sum } (\text{op } ^ 2) (X - \{0\}) \bmod 2 = (0::\text{nat})$
 by (subst sum-mod, auto simp add: (finite X) power-mod)
 ultimately
 show $\text{sum } (\text{op } ^ 2) X \bmod 2 \neq (0::\text{nat})$
 by auto
 next
 assume $\text{sum } (\text{op } ^ 2) X \bmod 2 \neq (0::\text{nat})$

```

show  $0 \in X$ 
proof (rule ccontr)
  assume  $0 \notin X$ 
  have  $\text{sum } (op \wedge 2) X \bmod 2 = (0::nat)$ 
  proof (subst sum-mod, auto simp add:  $\langle \text{finite } X \rangle$ )
    fix  $x$ 
    assume  $x \in X$ 
    with  $\langle 0 \notin X \rangle$  have  $x \neq 0$ 
    by simp (rule ccontr, simp)
    thus  $2 \wedge x \bmod 2 = (0::nat)$ 
    using power-mod
    by simp
  qed
  thus False
  using  $\langle \text{sum } (op \wedge 2) X \bmod 2 \neq (0::nat) \rangle$ 
  by simp
qed
qed

lemma unique-sumpows2:
  fixes  $n::nat$ 
  shows  $\exists! X. \text{finite } X \wedge \text{sum } (op \wedge 2) X = n$ 
proof (induct  $n$  rule: nat-less-induct)
  fix  $n::nat$ 
  assume hyp:  $\forall m < n. \exists! X. \text{finite } X \wedge \text{sum } (op \wedge 2) X = m$ 
  show  $\exists! X. \text{finite } X \wedge \text{sum } (op \wedge 2) X = n$ 
  proof (cases  $n = 0$ )
    case True
    show ?thesis
    proof
      show  $\text{finite } \{\} \wedge \text{sum } (op \wedge 2) \{\} = n$ 
      using  $\langle n = 0 \rangle$ 
      by simp
    next
      fix  $X$ 
      assume  $\text{finite } X \wedge \text{sum } (op \wedge 2) X = n$ 
      hence  $\text{finite } X \text{ sum } (op \wedge 2) X = (0::nat)$ 
      using  $\langle n = 0 \rangle$ 
      by simp-all
      hence  $\forall a \in X. 2 \wedge a = (0::nat)$ 
      apply (subst sum-eq-0-iff[THEN sym])
      by simp
      thus  $X = \{\}$ 
      by auto
    qed
  next
    case False
    hence  $n \text{ div } 2 < n$ 
    by auto

```

```

show ?thesis
proof-
  obtain X' where finite X' sum (op ^ 2) X' = n div 2
  ∀ X''. finite X'' ∧ sum (op ^ 2) X'' = n div 2 ⟶ X'' = X'
  using hyp ⟨n div 2 < n⟩
  apply (erule-tac x=n div 2 in allE)
  by auto
show ?thesis
proof (cases n mod 2 = 0)
  case True
  let ?X' = (op + 1) ' X'
  show ?thesis
  proof (rule, rule conjI)
    show finite ?X'
    using ⟨finite X'⟩
    by simp
  next
    show sum (op ^ 2) ?X' = n
    proof (subst sum.reindex, simp-all)
      show sum (λ x. 2 * 2 ^ x) X' = n
      using ⟨sum (op ^ 2) X' = n div 2⟩
      apply (subst sum-distrib-left[THEN sym])
      using ⟨n mod 2 = 0⟩
      by auto
    qed
  next
    fix X
    assume finite X ∧ sum (op ^ 2) X = n
    hence finite X sum (op ^ 2) X = n
    by simp-all
    hence 0 ∉ X
    using sumpows2-mod[of X] ⟨n mod 2 = 0⟩
    by simp

    let ?X = (λ x. x - 1) ' X
    have ?X = X'
    proof (rule ⟨∀ X''. finite X'' ∧ sum (op ^ 2) X'' = n div 2 ⟶ X'' =
X'⟩[rule-format], rule conjI)
      show finite ?X
      using ⟨finite X⟩
      by simp
    next
      show sum (op ^ 2) ?X = n div 2
      proof (subst sum.reindex, simp-all)
        show inj-on (λ x. x - Suc 0) X
        unfolding inj-on-def
        proof (safe)
          fix x y
          assume x ∈ X y ∈ X

```

```

    hence  $x \neq 0 \ y \neq 0$  using  $\langle 0 \notin X \rangle$ 
    by  $-(rule\ ccontr, auto)+$ 
    assume  $x - Suc\ 0 = y - Suc\ 0$ 
    thus  $x = y$ 
    using  $\langle x \neq 0 \rangle \langle y \neq 0 \rangle$ 
    by auto
  qed
next
  have  $(2::nat) * sum\ (\lambda\ x.\ 2^\wedge (x - Suc\ 0))\ X = sum\ (op^\wedge 2)\ X$ 
  proof (subst sum-distrib-left, rule sum.cong, simp-all)
    fix  $x::nat$ 
    assume  $x \in X$ 
    have  $x \neq 0$ 
    apply (rule ccontr)
    using  $\langle 0 \notin X \rangle \langle x \in X \rangle$ 
    by auto
    thus  $(2::nat) * 2^\wedge (x - Suc\ 0) = 2^\wedge x$ 
    by (simp add: power-eq-if)
  qed
  thus  $sum\ (\lambda\ x.\ 2^\wedge (x - Suc\ 0))\ X = n\ div\ 2$ 
  using  $\langle sum\ (op^\wedge 2)\ X = n \rangle$ 
  using  $\langle n\ mod\ 2 = 0 \rangle$ 
  by auto
  qed
qed
show  $X = op + 1 \text{ ' } X'$ 
proof (safe)
  fix  $x$ 
  assume  $x \in X$ 
  hence  $x \neq 0 \ x - 1 \in X'$ 
  using  $\langle 0 \notin X \rangle \langle image\ (\lambda x.\ x - 1)\ X = X' \rangle$ 
  by auto (rule ccontr, simp)
  thus  $x \in op + 1 \text{ ' } X'$ 
  by (rule-tac  $x=x-1$  in rev-image-eqI) auto
next
  fix  $x'::nat$ 
  assume  $x' \in X'$ 
  then obtain  $x$  where  $x \in X \ x' = x - 1 \ x \neq 0$ 
  using  $\langle image\ (\lambda x.\ x - 1)\ X = X' \rangle \langle 0 \notin X \rangle$ 
  by auto
  thus  $1 + x' \in X$ 
  by simp
  qed
qed
next
case False
let  $?X' = (op + 1) \text{ ' } X' \cup \{0\}$ 
show ?thesis

```

```

proof (rule, rule conjI)
  show finite ?X'
    using ⟨finite X'⟩
    by simp
next
  show sum (op ^ 2) ?X' = n
  proof (simp, subst sum.insert)
    show finite (op + (Suc 0) ' X')
      using ⟨finite X'⟩
      by simp
  next
    show 0 ∉ op + (Suc 0) ' X'
      by auto
  next
    show 2 ^ 0 + sum (op ^ 2) (op + (Suc 0) ' X') = n
    proof–
      have sum (op ^ 2) (op + (Suc 0) ' X') = n - 1
      proof (subst sum.reindex, simp-all)
        show (∑ x∈X'. 2 * 2 ^ x) = n - Suc 0
        apply (subst sum-distrib-left[THEN sym])
        using ⟨sum (op ^ 2) X' = n div 2⟩ ⟨n mod 2 ≠ 0⟩
        using div-mult-mod-eq[of n 2]
        by auto
      qed
    thus ?thesis
      using ⟨n ≠ 0⟩
      by simp
    qed
  qed
next
  fix X
  assume finite X ∧ sum (op ^ 2) X = n
  hence finite X sum (op ^ 2) X = n
    by simp-all
  hence 0 ∈ X
    using sumpows2-mod[of X] ⟨n mod 2 ≠ 0⟩
    by simp

  let ?X = (λ x. x - 1) ' (X - {0})
  have ?X = X'
  proof (rule ⟨∀ X''. finite X'' ∧ sum (op ^ 2) X'' = n div 2 ⟶ X'' =
X'⟩[rule-format], rule conjI)
    show finite ?X
      using ⟨finite X⟩
      by simp
  next
    show sum (op ^ 2) ?X = n div 2
    proof (subst sum.reindex, simp-all)
      show inj-on (λx. x - Suc 0) (X - {0})

```

```

      unfolding inj-on-def
    by auto
  next
    have  $(2::nat) * \text{sum } (\lambda x. 2 ^ (x - \text{Suc } 0)) (X - \{0\}) = \text{sum } (op ^ 2) (X - \{0\})$ 
    proof (subst sum-distrib-left, rule sum.cong, simp-all)
      fix  $x::nat$ 
      assume  $x \in X \wedge 0 < x$ 
      thus  $(2::nat) * 2 ^ (x - \text{Suc } 0) = 2 ^ x$ 
        by (simp add: power-eq-if)
    qed
    moreover
    have  $\text{sum } (op ^ 2) (X - \{0\}) = \text{sum } (op ^ 2) X - (1::nat)$ 
      apply (subst sum-diff1-nat)
      using  $\langle 0 \in X \rangle$ 
      by simp
    ultimately
    show  $\text{sum } (\lambda x. 2 ^ (x - \text{Suc } 0)) (X - \{0\}) = n \text{ div } 2$ 
      using  $\langle \text{sum } (op ^ 2) X = n \rangle$ 
      using  $\langle n \bmod 2 \neq 0 \rangle$ 
      by auto
  qed
qed

show  $X = ?X'$ 
proof (safe)
  fix  $x$ 
  assume  $x \in X \wedge x \notin op + 1 ' X'$ 
  show  $x = 0$ 
  proof (rule ccontr)
    assume  $x \neq 0$ 
    hence  $x - 1 \in X'$ 
      using  $\langle x \in X \rangle \langle ?X = X' \rangle$ 
      by auto
    hence  $x \in op + 1 ' X'$ 
      using  $\langle x \neq 0 \rangle$ 
      by (rule-tac  $x=x-1$  in rev-image-eqI) auto
    thus False
      using  $\langle x \notin \text{image } (op + 1) X' \rangle$ 
      by simp
  qed
next
  fix  $x'::nat$ 
  assume  $x' \in X'$ 
  then obtain  $x$  where  $x \in X \wedge x' = x - 1 \wedge x \neq 0$ 
    using  $\langle ?X = X' \rangle \langle 0 \in X \rangle$ 
    by auto
  thus  $1 + x' \in X$ 
    by simp

```

```

      next
      show  $0 \in X$ 
      using  $\langle 0 \in X \rangle$ 
      by simp
    qed
  qed
  qed
  qed
  qed
  qed

```

```

lemma sumpows2-inj:
  assumes sorted  $l$  and distinct  $l$  and sorted  $l'$  and distinct  $l'$ 
    sumpows2  $l = \text{sumpows2 } l'$ 
  shows  $l = l'$ 
  proof -
    have sumpows2  $l = \text{Sum } ((\text{op } ^ (2::\text{nat})) \text{ ` set } l)$ 
      apply (subst distinct-sum-list-conv-Sum)
      using  $\langle \text{distinct } l \rangle$ 
      by (auto simp add: distinct-map inj-on-def)
    moreover
    have sumpows2  $l' = \text{Sum } ((\text{op } ^ (2::\text{nat})) \text{ ` set } l')$ 
      apply (subst distinct-sum-list-conv-Sum)
      using  $\langle \text{distinct } l' \rangle$ 
      by (auto simp add: distinct-map inj-on-def)
    ultimately
    have  $\sum (\text{op } ^ (2::\text{nat}) \text{ ` set } l) = \sum (\text{op } ^ (2::\text{nat}) \text{ ` set } l')$ 
      using  $\langle \text{sumpows2 } l = \text{sumpows2 } l' \rangle$ 
      by simp
    hence  $\text{sum } (\text{op } ^ (2::\text{nat})) (\text{set } l) = \text{sum } (\text{op } ^ 2) (\text{set } l')$ 
      by (subst (asm) sum.reindex, simp-all add: inj-on-def)
    hence  $\text{set } l = \text{set } l'$ 
      using unique-sumpows2[of  $\text{sum } (\text{op } ^ (2::\text{nat})) (\text{set } l)$ ]
      by auto
    thus ?thesis
      using  $\langle \text{distinct } l \rangle \langle \text{sorted } l \rangle \langle \text{distinct } l' \rangle \langle \text{sorted } l' \rangle$ 
      using sorted-distinct-set-unique
      by auto
  qed

```

```

lemma sumpows2-shift:
  shows  $2 * \text{sumpows2 } l = \text{sumpows2 } (\text{map } (\text{op } + 1) l)$ 
  by (induct  $l$ ) auto

```

2.2 nat2list

The function *nat2list* deconstructs the natural number back to the set it represents (for executability, given by a sorted, distinct list).

```

fun nat2list-aux ::  $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat list}$  where

```



```

nat2list-aux n k =
  (if n = 0 then
    []
  else if n mod 2 = 0 then
    nat2list-aux (n div 2) (k + 1)
  else
    k # nat2list-aux (n div 2) (k + 1))

```

definition *nat2list* **where**
nat2list n \equiv *nat2list-aux* n 0

lemma *sumpows2-nat2list-aux*:
sumpows2 (nat2list-aux n k) = n * 2^k
proof (induct n k rule: nat2list-aux.induct)
 case (1 n k)
 show ?case
proof (cases n = 0)
 case True
thus ?thesis
by simp
 next
 case False
 show ?thesis
proof (cases n mod 2 = 0)
 case True
thus ?thesis
using ⟨n ≠ 0⟩
using nat2list-aux.simps[of n k]
using 1(1)
by (auto simp del: nat2list-aux.simps)
 next
 case False
thus ?thesis
using ⟨n mod 2 ≠ 0⟩⟨n ≠ 0⟩
using mult-eq-if[of n 2 ^ k]
using mod-mult-div-eq[of n 2, THEN sym]
using nat2list-aux.simps[of n k]
using 1(2)
by (auto simp del: nat2list-aux.simps)
qed
qed
qed

lemma *sumpows2-nat2list*:
sumpows2 (nat2list n) = n
unfolding nat2list-def
using *sumpows2-nat2list-aux*[of n 0]
by auto

```

lemma nat2list-aux-empty:
  (nat2list-aux x k = []) = (x = 0)
by (induct x k rule: nat2list-aux.induct) auto

lemma nat2list-aux-shift:
  nat2list-aux n (k + 1) = map (op + 1) (nat2list-aux n k)
by (induct n k rule: nat2list-aux.induct) auto

lemma sorted-distinct-nat2list-aux:
  sorted (nat2list-aux n k) ∧
  distinct (nat2list-aux n k) ∧
  (∀ a ∈ List.set (nat2list-aux n k). a ≥ k)
proof (induct rule: nat2list-aux.induct)
  case (1 n k)
  show ?case
  proof (cases n = 0)
    case True
    thus ?thesis
    by simp
  next
  case False
  show ?thesis
  proof (cases n mod 2 = 0)
    case True
    thus ?thesis
    using ⟨n ≠ 0⟩
    using nat2list-aux.simps[of n k]
    using 1(1)
    by (auto simp del: nat2list-aux.simps)
  next
  case False
  thus ?thesis
  using ⟨n ≠ 0⟩
  using nat2list-aux.simps[of n k]
  using 1(2)
  by (auto simp del: nat2list-aux.simps intro!: sorted.Cons)
  qed
qed
qed

lemma
  sorted-nat2list: sorted (nat2list n) and
  distinct-nat2list: distinct (nat2list n)
using sorted-distinct-nat2list-aux[of n 0]
unfolding nat2list-def
by auto

lemma sumpows2-nat2list-unique:
  assumes sumpows2 l = n and sorted l and distinct l

```

```

    shows  $l = \text{nat2list } n$ 
using assms
using sumpows2-nat2list[of n]
using sumpows2-inj[of nat2list n l]
using sorted-nat2list distinct-nat2list
by simp

lemma inj-nat2list-aux':
  assumes  $\text{nat2list-aux } x \ k = \text{nat2list-aux } y \ k$ 
  shows  $x = y$ 
using assms
proof (induct x k arbitrary: y rule: nat2list-aux.induct)
  case ( $1 \ x \ k$ )
  show ?case
  proof (cases x = 0)
    case True
    thus ?thesis
      using  $1(3)$ 
      using nat2list-aux-empty[of y k]
      by simp
  next
  case False
  show ?thesis
  proof (cases x mod 2 = 0)
    case True
    hence  $\text{nat2list-aux } (x \text{ div } 2) \ (\text{Suc } k) = \text{nat2list-aux } y \ k$ 
      using  $\langle x \neq 0 \rangle \ 1(3)$ 
      using nat2list-aux.simps[of x k]
      by simp
    show ?thesis
    proof (cases y = 0)
      case True
      hence  $\text{nat2list-aux } (x \text{ div } 2) \ (\text{Suc } k) = []$ 
        using  $\langle \text{nat2list-aux } (x \text{ div } 2) \ (\text{Suc } k) = \text{nat2list-aux } y \ k \rangle$ 
        using nat2list-aux.simps[of y k]
        by (simp del: nat2list-aux.simps)
      hence  $x \text{ div } 2 = 0$ 
        using nat2list-aux-empty[of x div 2 Suc k]
        by simp
      thus ?thesis
        using  $\langle x \text{ mod } 2 = 0 \rangle$ 
        using  $\langle y = 0 \rangle$ 
        by auto
    next
    case False
    show ?thesis
    proof (cases y mod 2 = 0)
      case True
      hence  $\text{nat2list-aux } (x \text{ div } 2) \ (\text{Suc } k) = \text{nat2list-aux } (y \text{ div } 2) \ (\text{Suc } k)$ 

```

```

    using ⟨y ≠ 0⟩
    using ⟨nat2list-aux (x div 2) (Suc k) = nat2list-aux y k⟩
    using nat2list-aux.simps[of y k]
    by (simp del: nat2list-aux.simps)
  hence x div 2 = y div 2
    using ⟨x ≠ 0⟩ ⟨x mod 2 = 0⟩ 1(1)[of y div 2]
    by simp
  thus ?thesis
    using ⟨x mod 2 = 0⟩ ⟨y mod 2 = 0⟩
    by auto
next
case False
  hence nat2list-aux (x div 2) (Suc k) = k # nat2list-aux (y div 2) (Suc k)
    using ⟨y ≠ 0⟩
    using ⟨nat2list-aux (x div 2) (Suc k) = nat2list-aux y k⟩
    using nat2list-aux.simps[of y k]
    by (simp del: nat2list-aux.simps)
  thus ?thesis
    using sorted-distinct-nat2list-aux[of x div 2 Suc k]
    by auto
qed
qed
next
case False
  hence k # nat2list-aux (x div 2) (Suc k) = nat2list-aux y k
    using ⟨x ≠ 0⟩ 1(3)
    using nat2list-aux.simps[of x k]
    by simp
  show ?thesis
  proof (cases y = 0)
    case True
    thus ?thesis
      using ⟨k # nat2list-aux (x div 2) (Suc k) = nat2list-aux y k⟩
      by simp
  next
  case False
  show ?thesis
  proof (cases y mod 2 = 0)
    case True
    hence k # nat2list-aux (x div 2) (Suc k) = nat2list-aux (y div 2) (Suc k)
      using ⟨y ≠ 0⟩
      using ⟨k # nat2list-aux (x div 2) (Suc k) = nat2list-aux y k⟩
      using nat2list-aux.simps[of y k]
      by (simp del: nat2list-aux.simps)
    have k ∈ set (nat2list-aux (y div 2) (Suc k))
      using ⟨k # nat2list-aux (x div 2) (Suc k) = nat2list-aux (y div 2) (Suc
k)⟩[THEN sym]
      by simp
    thus ?thesis

```

```

      using sorted-distinct-nat2list-aux[of y div 2 Suc k]
    by auto
  next
  case False
  hence nat2list-aux (x div 2) (Suc k) = nat2list-aux (y div 2) (Suc k)
    using ⟨y ≠ 0⟩
    using ⟨k # nat2list-aux (x div 2) (Suc k) = nat2list-aux y k⟩
    using nat2list-aux.simps[of y k]
    by (simp del: nat2list-aux.simps)
  hence x div 2 = y div 2
    using 1(2)[of y div 2] ⟨x ≠ 0⟩ ⟨x mod 2 ≠ 0⟩
    by simp
  thus ?thesis
    using ⟨x mod 2 ≠ 0⟩ ⟨y mod 2 ≠ 0⟩
    using mod-mult-div-eq[of x 2, THEN sym]
    using mod-mult-div-eq[of y 2, THEN sym]
    by simp
qed
qed
qed
qed
qed

lemma inj-nat2list:
  inj nat2list
unfolding inj-on-def
proof (auto)
  fix x y
  assume nat2list x = nat2list y
  thus x = y
    unfolding nat2list-def
    using inj-nat2list-aux'[of x 0 y]
    by simp
qed

lemma nat2list-even:
  nat2list (2*n) = map (op + 1) (nat2list n)
using sumpows2-nat2list-unique[of map (op + 1) (nat2list n) 2*n]
using sumpows2-nat2list[of n]
using sumpows2-shift[of nat2list n]
by (auto simp add: sorted-nat2list sorted-map distinct-nat2list distinct-map inj-on-def)

lemma nat2list-odd:
  nat2list (2*n + 1) = 0 # map (op + 1) (nat2list n)
proof -
  have Suc (2 * n) div 2 = n
    by auto
  thus ?thesis
    unfolding nat2list-def

```

```

    using nat2list-aux-shift
    by auto
qed

```

2.3 list2nat

Although the function *sumpows2* converts a set (given by a distinct list) to its representing natural number, we define the function *list2nat* that does the same, but is somewhat more efficient.

```

fun list2nat-aux-measure where
list2nat-aux-measure (l, i, s, r) =
  (case l of
    []  $\Rightarrow$  0
  | (h # t)  $\Rightarrow$  if i > h then 0 else Max (set l) - i + 1)

```

```

function list2nat-aux :: nat list  $\Rightarrow$  nat  $\Rightarrow$  nat  $\Rightarrow$  nat  $\Rightarrow$  nat where
  list2nat-aux [] i s r = r
| list2nat-aux (h # t) i s r =
  (if i = h then
    list2nat-aux t (i+1) (2*s) (s + r)
  else if i < h then
    list2nat-aux (h # t) (i+1) (2*s) r
  else r)

```

```

by pat-completeness auto

```

termination

```

proof (relation measure list2nat-aux-measure)
  show wf (measure list2nat-aux-measure)
  by simp

```

next

```

fix h i s r :: nat and t

```

```

assume i = h

```

```

have list2nat-aux-measure (t, i + 1, 2 * s, s + r) <
  list2nat-aux-measure (h # t, i, s, r)

```

```

proof (cases t)

```

```

  case Nil

```

```

  thus ?thesis

```

```

    using ⟨i = h⟩

```

```

    by simp

```

next

```

  case (Cons h' t')

```

```

  thus ?thesis

```

```

    using ⟨i = h⟩

```

```

    using Max-insert[of set t' h]

```

```

    by (cases t'  $\neq$  []) auto

```

qed

```

thus ((t, i + 1, 2 * s, s + r), (h # t, i, s, r))  $\in$  measure list2nat-aux-measure
  by simp

```

next

```

fix h i s r :: nat and t

```

```

    assume  $i \neq h$   $i < h$ 
    thus  $((h \# t, i + 1, 2 * s, r), (h \# t, i, s, r)) \in \text{measure } \text{list2nat-aux-measure}$ 
      using Max-insert[of set t h]
      by (cases  $t = []$ ) auto
qed

```

```

definition list2nat where
  list2nat  $l = \text{list2nat-aux } l \ 0 \ 1 \ 0$ 

```

```

lemma list2nat-aux-sumpows2:
  assumes  $s = 2^i$  and sorted l and distinct l and
     $l \neq [] \longrightarrow i \leq \text{hd } l$ 
  shows  $\text{list2nat-aux } l \ i \ s \ r = r + \text{sumpows2 } l$ 
using assms
proof (induct l i s r rule: list2nat-aux.induct)
  case ( $1 \ i \ s \ r$ )
  thus ?case
    by simp
next
  case ( $2 \ h \ t \ i \ s \ r$ )
  show ?case
proof (cases h = i)
  case True
  have  $t \neq [] \longrightarrow i + 1 \leq \text{hd } t$ 
    using  $2(4) \ 2(5) \ 2(6)$ 
    by (cases t) auto
  thus ?thesis
    using  $\langle h = i \rangle$ 
    using  $2(1) \ 2(2) \ 2(3) \ 2(4) \ 2(5)$ 
    by (simp add: sorted-Cons)
next
  case False
  show ?thesis
proof (cases i < h)
  case True
  thus ?thesis
    using  $2(2) \ 2(3) \ 2(4) \ 2(5) \ \langle h \neq i \rangle$ 
    by (simp add: sorted-Cons)
next
  case False
  thus ?thesis
    using  $\langle h \neq i \rangle \ 2(6)$ 
    by simp
qed
qed
qed

```

```

lemma list2nat-sumpows2:
  assumes sorted l and distinct l

```

```

  shows list2nat l = sumpows2 l
unfolding list2nat-def
using assms
using list2nat-aux-sumpows2[of 1 0 l 0]
by simp

lemma nat2list-list2nat:
  assumes distinct l and sorted l
  shows nat2list (list2nat l) = l
using assms
using list2nat-sumpows2[of l]
using sumpows2-nat2list-unique[of l list2nat l]
by simp

lemma inj-list2nat:
  assumes sorted l1 and distinct l1 and sorted l2 and distinct l2
  assumes list2nat l1 = list2nat l2
  shows l1 = l2
proof-
  have sumpows2 l1 = sumpows2 l2
  using assms
  by (auto simp add: list2nat-sumpows2[THEN sym])
  thus ?thesis
  using assms
  by (simp add: sumpows2-inj)
qed

end

```

3 Union closed families

```

theory UnionClosed
imports Main
begin

```

3.1 Union closed families

```

definition sum-family (infixl  $\uplus$  100) where
   $A \uplus B = (\lambda (a, b). a \cup b) \cdot \{(a, b) . a \in A \wedge b \in B\}$ 

```

```

definition union-closed :: 'a set set  $\Rightarrow$  bool where
  union-closed F  $\equiv \forall A B. A \in F \wedge B \in F \longrightarrow A \cup B \in F$ 

```

```

lemma union-closed F  $\longleftrightarrow F \uplus F = F$ 
unfolding union-closed-def sum-family-def
by auto

```

```

abbreviation finite-union-closed where
  finite-union-closed F  $\equiv$  union-closed F  $\wedge$  finite ( $\bigcup F$ )

```



```

lemma union-closed-n:
  assumes finite  $F'$  and  $F' \neq \{\}$  and  $F' \subseteq F$ 
  assumes union-closed  $F$ 
  shows  $\bigcup F' \in F$ 
using assms
proof (induct  $F'$ )
  case empty
  thus ?case
    by simp
next
  case (insert  $a$   $F''$ )
  thus ?case
    using insert
    by (cases  $F'' = \{\}$ ) (auto simp add: union-closed-def)
qed

```

```

lemma union-closed-insert:
  assumes union-closed  $F$  and  $\forall B \in F. A \cup B \in F \cup \{A\}$ 
  shows union-closed ( $F \cup \{A\}$ )
using assms
unfolding union-closed-def
by (auto simp add: Un-commute)

```

3.1.1 Closure – minimal union closed family containing F

```

definition closure where
  closure  $F = \text{Union } \text{'(Pow } F - \{\{\}\})$ 

```

```

lemma closure-closed:
  union-closed (closure  $F$ )
unfolding union-closed-def
proof ((rule allI)+, rule impI, erule conjE)
  fix  $A\ B :: 'a\ \text{set}$ 
  assume  $A \in \text{closure } F\ B \in \text{closure } F$ 
  then obtain  $xa\ xb$  where
     $A = \bigcup xa\ xa \in \text{Pow } F - \{\{\}\}$ 
     $B = \bigcup xb\ xb \in \text{Pow } F - \{\{\}\}$ 
    by (auto simp add: closure-def)
  have  $\bigcup xa \cup \bigcup xb = \bigcup (xa \cup xb)$ 
    by auto
  moreover
  have  $xa \cup xb \in \text{Pow } F - \{\{\}\}$ 
    using  $\langle xa \in \text{Pow } F - \{\{\}\} \rangle \langle xb \in \text{Pow } F - \{\{\}\} \rangle$ 
    by auto
  ultimately
  show  $A \cup B \in \text{closure } F$ 
    using  $\langle A = \bigcup xa \rangle \langle B = \bigcup xb \rangle$ 
    unfolding closure-def

```

by *blast*
qed

lemma *closure-min-closed*:
 assumes *finite F* and $F \subseteq F'$ and *union-closed F'*
 shows $\text{closure } F \subseteq F'$
proof
 fix x
 assume $x \in \text{closure } F$
 then obtain S where $x = \bigcup S$ $S \in \text{Pow } F - \{\{\}\}$
 unfolding *closure-def*
 by *auto*
 thus $x \in F'$
 using *union-closed-n*[*of S F*] *union-closed F'* *finite F* $\langle F \subseteq F' \rangle$
 using *finite-subset*
 by *auto*
 qed

lemma *closure-union-closed-id*:
 assumes *finite F* and *union-closed F*
 shows $\text{closure } F = F$
 using *assms closure-min-closed*[*of F F*]
 by (*auto simp add: closure-def*)

lemma [*simp*]:
 shows $\bigcup (\text{closure } F) = \bigcup F$
 unfolding *closure-def*
 by *auto*

lemma *finite-closure*:
 assumes *finite* $(\bigcup F)$
 shows *finite* $(\bigcup (\text{closure } F))$
 using *assms*
 by *simp*

lemma *closure-mono*:
 assumes $F \subseteq F'$
 shows $\text{closure } F \subseteq \text{closure } F'$
 using *assms*
 unfolding *closure-def*
 by *auto*

Iterative closure: insert set to closed family and close.

abbreviation *insert-and-close* :: '*a set* \Rightarrow '*a set set* \Rightarrow '*a set set* **where**
insert-and-close A F $\equiv F \cup \{A\} \cup (\text{op } \cup A)$ '*F*

lemma *union-closed-insert-and-close*:
 assumes *union-closed F*
 shows *union-closed* (*insert-and-close A F*)

```

using assms
unfolding union-closed-def
by auto

lemma closure-insert:
  assumes finite F
  shows  $\text{closure } (F \cup \{A\}) = \text{insert-and-close } A \ (\text{closure } F)$  (is ?lhs = ?rhs)
proof
  show  $?lhs \subseteq ?rhs$ 
    using assms union-closed-insert-and-close[of closure F A]
    using closure-closed[of F]
    using closure-min-closed[of F  $\cup$  {A} ?rhs]
    by (auto simp add: closure-def)
  next
    show  $?rhs \subseteq ?lhs$ 
    proof
      fix x
      assume  $x \in ?rhs$ 
      hence  $x \in \text{closure } F \cup \{A\} \vee x \in \text{op} \cup A$  ‘ (closure F)
      by simp
      thus  $x \in ?lhs$ 
    proof
      assume  $x \in \text{closure } F \cup \{A\}$ 
      thus  $x \in \text{closure } (F \cup \{A\})$ 
      by (auto simp add: closure-def)
    next
      assume  $x \in \text{op} \cup A$  ‘ (closure F)
      then obtain y where  $x = y \cup A$   $y \in \text{closure } F$ 
      by auto
      then obtain k where  $k \in \text{Pow } F$   $y = \bigcup k$   $k \neq \{\}$ 
      unfolding closure-def
      by auto

      show ?thesis
      unfolding closure-def
      apply (rule rev-image-eqI[of k  $\cup$  {A}])
      using  $\langle x = y \cup A \rangle \langle y = \bigcup k \rangle \langle k \neq \{\} \rangle \langle k \in \text{Pow } F \rangle$ 
      by auto
    qed
  qed
qed

lemma closure-subset:
  shows  $F \subseteq \text{closure } F$ 
unfolding closure-def
by auto

lemma closure-empty:
  shows  $\{\} \in \text{closure } F \longleftrightarrow \{\} \in F$ 

```

```

using closure-subset[of F]
unfolding closure-def
by force

lemma insert-and-close-empty:
  shows insert-and-close {} F = F  $\cup$  {}
by auto

lemma closure-remove-empty:
  assumes finite F
  shows closure (F - {}) = closure F - {}
proof (cases {}  $\in$  F)
  case True
  hence F = (F - {})  $\cup$  {}
  by auto
  have closure F = closure (F - {})  $\cup$  {}
  apply (subst (F = (F - {})  $\cup$  {}))
  apply (subst closure-insert[of F - {} {}])
  apply (simp add: assms)
  apply (subst insert-and-close-empty)
  by simp
  thus ?thesis
  using True closure-empty[of F - {}]
  by simp
next
  case False
  thus ?thesis
  using closure-empty[of F]
  by simp
qed

```

```

lemma insert-and-close-closure:
  assumes finite F and union-closed F
  shows closure (F  $\cup$  {A}) = insert-and-close A F (is ?lhs = ?rhs)
using assms
using closure-insert[of F A]
using closure-union-closed-id[of F]
by simp

```

3.2 Union closed families closed to unions with an additional family

```

definition union-closed-additional where
  [simp]: union-closed-additional F Fc  $\equiv$ 
    union-closed F  $\wedge$  ( $\forall$  A  $\in$  F. (op  $\cup$  A) ' Fc  $\subseteq$  F)

```

```

lemma
  shows union-closed-additional F Fc  $\longleftrightarrow$  F  $\uplus$  F = F  $\wedge$  F  $\uplus$  Fc  $\subseteq$  F
unfolding union-closed-additional-def union-closed-def sum-family-def

```

by *auto force*

lemma *union-closed-additional*:

shows $(\forall A \in F. \forall Ai \in Fc. A \cup Ai \in F) \longleftrightarrow$
 $(\forall A \in F. (op \cup A) ' Fc \subseteq F)$

by *auto*

abbreviation *insert-and-close-additional* **where**

insert-and-close-additional $A F Fc \equiv F \cup \{A\} \cup ((op \cup A) ' F) \cup ((op \cup A) ' Fc)$

lemma *closure-additional-set*:

assumes *finite* F and $A \in \text{closure } F$ and

$A' \in F'$ and *union-closed* F' and $\forall A' \in F'. op \cup A' ' F \subseteq F'$

shows $A \cup A' \in F'$

proof –

have $op \cup A' ' F \subseteq F'$

using $\langle \forall A' \in F'. \text{image } (op \cup A') F \subseteq F' \rangle \langle A' \in F' \rangle$

by *simp*

obtain X where $A = \bigcup X$ $X \subseteq F$ $X \neq \{\}$

using $\langle A \in \text{closure } F \rangle$

unfolding *closure-def*

by *auto*

hence *finite* X

using $\langle \text{finite } F \rangle$

by (*auto simp add: finite-subset*)

have $A \cup A' = \bigcup ((op \cup A') ' X)$

using $\langle A = \bigcup X \rangle \langle X \neq \{\} \rangle$

by *auto*

show *?thesis*

proof (*subst* $\langle A \cup A' = \bigcup \text{image } (op \cup A') X \rangle$, *rule union-closed-n*)

show *finite* $(op \cup A' ' X)$

using $\langle \text{finite } X \rangle$

by *simp*

next

show $op \cup A' ' X \neq \{\}$

using $\langle X \neq \{\} \rangle$

by *simp*

next

show $op \cup A' ' X \subseteq F'$

using $\langle \text{image } (op \cup A') F \subseteq F' \rangle$

using $\langle X \subseteq F \rangle$

by *auto*

next

show *union-closed* F'

by *fact*

qed

qed

lemma *union-closed-additional-closure*:
assumes *finite Fc* **and** *union-closed-additional F Fc*
shows *union-closed-additional F (closure Fc)*
unfolding *union-closed-additional-def*
proof (*safe*)
 fix *x y*
 assume *x ∈ F y ∈ closure Fc*
 thus *x ∪ y ∈ F*
 using *closure-additional-set[of Fc y x F]*
 using $\langle \text{union-closed-additional } F \text{ Fc} \rangle \langle \text{finite } Fc \rangle$
 by (*auto simp add: Un-commute*)
next
 show *union-closed F*
 using $\langle \text{union-closed-additional } F \text{ Fc} \rangle$
 by *simp*
qed

3.2.1 Union closed extensions

definition *union-closed-extensions* **where**
 $[simp]: \text{union-closed-extensions } Fc \equiv$
 $\{F. F \subseteq \text{Pow } (\bigcup Fc) \wedge \text{union-closed-additional } F \text{ Fc}\}$
syntax
 $\text{-union-closed-extensions} :: 'a \text{ set set} \Rightarrow 'a \text{ set set set} \quad (\{\!\{-\}\!\})$
translations
 $\{\!\{Fc\}\!\} == \text{CONST union-closed-extensions } Fc$
end

4 Families of sets

theory *Family*
imports *Main*
begin

definition *count* $:: 'a \Rightarrow 'a \text{ set set} \Rightarrow \text{nat}$ **where**
 $\text{count } a \text{ } F \equiv \text{card } \{S \in F. a \in S\}$

lemma *count-Un-disjoint*:
assumes $A \cap B = \{\}$ **and** *finite A* **and** *finite B*
shows $\text{count } a \text{ } (A \cup B) = \text{count } a \text{ } A + \text{count } a \text{ } B$
proof –
 have $\{S \in A. a \in S\} \cap \{S \in B. a \in S\} = \{\}$
 $\{S \in A \cup B. a \in S\} = \{S \in A. a \in S\} \cup \{S \in B. a \in S\}$
 using *assms*
 by *auto*

```

thus ?thesis
  unfolding count-def
  using assms card-Un-disjoint[of {S ∈ A. a ∈ S} {S ∈ B. a ∈ S}]
  by auto
qed

end

```

5 Frankl's condition

```

theory Frankl
imports Family UnionClosed More.MoreSet
begin

```

5.1 Frankl's condition

Note that, in order to avoid using rational numbers, the following condition is not formulated as $\text{count } x \ F \geq (\text{card } F) / 2$, what would be more expected.

```

abbreviation frankl-element :: 'a ⇒ 'a set set ⇒ bool where
  frankl-element x F ≡ x ∈ ⋃ F ∧ 2 * count x F ≥ card F

```

```

definition frankl :: 'a set set ⇒ bool where
  frankl F ≡ (∃ x. frankl-element x F)

```

```

lemma obtain-frankl-element:
  assumes frankl F
  obtains x where x ∈ ⋃ F and 2 * count x F ≥ card F
using assms
unfolding frankl-def
by auto

```

5.1.1 Frankl's function

```

abbreviation frankl-fun :: 'a ⇒ 'a set set ⇒ int where
  frankl-fun x F ≡ 2 * int (count x F) - int (card F)

```

```

lemma
  shows frankl-element x F ⟷ (x ∈ ⋃ F ∧ frankl-fun x F ≥ 0)
by auto

```

```

lemma frankl-fun-Un-disjoint:
  assumes A ∩ B = {} and finite A and finite B
  shows frankl-fun a (A ∪ B) = frankl-fun a A + frankl-fun a B
proof –
  let ?A = {S. S ∈ A ∧ a ∈ S} and ?B = {S. S ∈ B ∧ a ∈ S}
  have *: {S. (S ∈ A ∪ B) ∧ a ∈ S} = ?A ∪ ?B ?A ∩ ?B = {}
  using assms
  by auto

```

```

hence card {S. (S ∈ A ∪ B) ∧ a ∈ S} = card ?A + card ?B
  using assms
  using card-Un-disjoint[of ?A ?B]
  by auto
thus ?thesis
  using assms
  unfolding count-def
  by (auto simp add: card-Un-disjoint)
qed

lemma frankl-fun-Un-disjoint-3:
  assumes finite A and finite B and finite C
    A ∩ B = {} and A ∩ C = {} and B ∩ C = {}
  shows frankl-fun i (A ∪ B ∪ C) =
    frankl-fun i A + frankl-fun i B + frankl-fun i C
using assms frankl-fun-Un-disjoint[of A ∪ B C i]
using assms frankl-fun-Un-disjoint[of A B i]
by auto

lemma frankl-fun-UN-disjoint:
  assumes finite (⋃ set l) and
    ∀ i j. i < length l ∧ j < length l ∧ i ≠ j ⟶ l ! i ∩ l ! j = {}
  shows frankl-fun i (⋃ set l) = sum-list (map (frankl-fun i) l)
using assms
proof (induct l)
  case Nil
  thus ?case
    by (simp add: count-def)
next
  case (Cons a l')
  have *: ∀ i j. i < length l' ∧ j < length l' ∧ i ≠ j ⟶ l' ! i ∩ l' ! j = {}
    using Cons(3)
    by force
  show ?case
  proof-
    have frankl-fun i (⋃ set (a # l')) = frankl-fun i (a ∪ (⋃ set l'))
      by simp
    also have ... = frankl-fun i a + frankl-fun i (⋃ set l')
    proof (subst frankl-fun-Un-disjoint[of a ⋃ (set l') i])
      show a ∩ ⋃ set l' = {}
        using Cons(3)
        by (auto simp add: nth-Cons) (metis Suc-less-eq Zero-not-Suc disjoint-iff-not-equal
in-set-conv-nth nth.simps nth-Cons-0 nth-Cons-Suc zero-less-Suc)
      qed (insert Cons(2), auto)
    finally
      show ?thesis
        using Cons(1,2) *
        by simp
    qed
  qed

```


qed

lemma *frankl-fun-Pow*:

assumes *finite A* and $a \in A$
 shows *frankl-fun a (Pow A) = 0*

proof –

have $\exists A'. A = A' \cup \{a\} \wedge a \notin A' \wedge \text{finite } A'$

using *assms*

by (*rule-tac x=A - {a} in exI*) *auto*

then obtain A' where $A = A' \cup \{a\}$ $a \notin A'$ *finite A'*

by *auto*

let $?PA' = \text{Pow } A'$ and $?aPA' = ((op \cup \{a\}) ' \text{Pow } A')$

have $\text{Pow } A = ?PA' \cup ?aPA'$

using *Pow-insert[of a A']* $\langle A = A' \cup \{a\} \rangle$

by (*simp add: image-def*)

moreover

have $?PA' \cap ?aPA' = \{\}$

using $\langle a \notin A' \rangle$

by *auto*

ultimately

have $\text{count } a (\text{Pow } A) = \text{count } a ?PA' + \text{count } a ?aPA'$

using $\langle \text{finite } A' \rangle$

by (*subst count-Un-disjoint[symmetric]*) *simp-all*

moreover

have $\{S. S \subseteq A' \wedge a \in S\} = \{\}$

using $\langle a \notin A' \rangle$

by *auto*

hence $\text{count } a ?PA' = 0$

using $\langle \text{finite } A' \rangle$

unfolding *count-def*

by *auto*

moreover

have $\text{count } a ?aPA' = \text{card } ?PA'$

proof –

have $\{S \in op \cup \{a\} ' \text{Pow } A'. a \in S\} = op \cup \{a\} ' \text{Pow } A'$

by *auto*

hence $\text{count } a (op \cup \{a\} ' \text{Pow } A') = \text{card } (op \cup \{a\} ' \text{Pow } A')$

unfolding *count-def*

by *simp*

also have $\dots = \text{card } (\text{Pow } A')$

using $\langle a \notin A' \rangle$

by (*subst card-image*) (*auto simp add: inj-on-def*)

finally

show *?thesis*

.

qed

ultimately

show *?thesis*

using $\langle A = A' \cup \{a\} \rangle \langle a \notin A' \rangle$

using $\langle \text{finite } A \rangle$
 by (auto simp add: card-Pow)
 qed

5.2 FC Families

definition *FC-family* where

$FC\text{-family } Fc \equiv$
 $\forall F. F \supseteq Fc \wedge \text{finite-union-closed } F \longrightarrow$
 $(\exists a \in \bigcup Fc. 2 * \text{count } a F \geq \text{card } F)$

lemma *FC-family-frankl*:

assumes *FC-family* Fc and $Fc \subseteq F$ and *finite-union-closed* F
 shows *frankl* F

proof–

have $\bigcup Fc \subseteq \bigcup F$
 using $\langle Fc \subseteq F \rangle$
 by auto
 thus ?thesis
 using *assms*
 unfolding *FC-family-def* *frankl-def*
 by blast

qed

lemma *FC-family-mono*:

assumes $Fc \subseteq Fc'$ and *FC-family* Fc
 shows *FC-family* Fc'

unfolding *FC-family-def*

proof (*safe*)

fix F
 assume $Fc' \subseteq F$ *union-closed* F *finite* $(\bigcup F)$
 hence $Fc \subseteq F$
 using $\langle Fc \subseteq Fc' \rangle$
 by auto
 then obtain a where $a \in \bigcup Fc$ $\text{card } F \leq 2 * \text{count } a F$
 using $\langle \text{FC-family } Fc \rangle \langle \text{union-closed } F \rangle \langle \text{finite } (\bigcup F) \rangle$
 unfolding *FC-family-def*
 by blast
 thus $\exists a \in \bigcup Fc'. \text{card } F \leq 2 * \text{count } a F$
 using $\langle Fc \subseteq Fc' \rangle \langle Fc' \subseteq F \rangle$
 by blast

qed

lemma *FC-family-empty-set-remove*:

shows *FC-family* $Fc \longleftrightarrow \text{FC-family } (Fc - \{\{\}\})$

proof

assume *FC-family* Fc
 show *FC-family* $(Fc - \{\{\}\})$
 unfolding *FC-family-def*

```

proof (safe)
  fix  $F$ 
  assume  $Fc - \{\{\}\} \subseteq F$  union-closed  $F$  finite  $(\bigcup F)$ 
  hence  $Fc \subseteq F \cup \{\{\}\}$  finite-union-closed  $(F \cup \{\{\}\})$ 
    by (auto simp add: union-closed-def)
  then obtain  $a$  where  $a \in \bigcup Fc$   $\text{card } (F \cup \{\{\}\}) \leq 2 * \text{count } a (F \cup \{\{\}\})$ 
    using  $\langle FC\text{-family } Fc \rangle$ 
    unfolding  $FC\text{-family-def}$ 
    by blast
  moreover
  have  $\text{card } F \leq \text{card } (F \cup \{\{\}\})$ 
    using  $\langle \text{finite } (\bigcup F) \rangle$ 
    by (auto simp add: finiteUn-iff card-insert-le)
  moreover
  have  $\{A. (A = \{\} \vee A \in F) \wedge a \in A\} = \{A \in F. a \in A\}$ 
    by auto
  hence  $2 * \text{count } a (F \cup \{\{\}\}) = 2 * \text{count } a F$ 
    by (auto simp add: count-def)
  ultimately
  have  $\text{card } F \leq 2 * \text{count } a F$   $a \in \bigcup (Fc - \{\{\}\})$ 
    by auto
  thus  $\exists a \in \bigcup (Fc - \{\{\}\}). \text{card } F \leq 2 * \text{count } a F$ 
    by blast
qed
next
  assume  $FC\text{-family } (Fc - \{\{\}\})$ 
  thus  $FC\text{-family } Fc$ 
    unfolding  $FC\text{-family-def}$ 
    by auto
qed

lemma  $FC\text{-family-empty-set-insert}$ :
  shows  $FC\text{-family } (Fc \cup \{\{\}\}) \longleftrightarrow FC\text{-family } Fc$ 
by (metis FC-family-empty-set-remove Diff-insert-absorb Un-empty-right Un-insert-right insert-absorb)

lemma  $FC\text{-family-closure}$ :
  assumes finite  $Fc$ 
  shows  $FC\text{-family } Fc \longleftrightarrow FC\text{-family } (\text{closure } Fc)$ 
proof
  assume  $FC\text{-family } Fc$ 
  moreover
  have  $Fc \subseteq \text{closure } Fc$ 
    by (auto simp add: closure-def)
  ultimately
  show  $FC\text{-family } (\text{closure } Fc)$ 
    unfolding  $FC\text{-family-def}$ 
    by auto
next

```

```

assume FC-family (closure Fc)
thus FC-family Fc
  unfolding FC-family-def
  using closure-min-closed[of Fc] (finite Fc)
  by auto
qed

end

```

6 Executable implementations of set families

6.1 Abstract representation of sets and families

```

theory FamilyImpl
imports Family
         Frankl
         More.MoreSet ListNat
         HOL-Library.List-lexord
         HOL-Library.Code-Target-Nat
begin

```

abbreviation *map* **where** *map* \equiv *List.map*

Set operations are specified in the following locale. Families are always represented as (distinct) lists of sets.

```

locale SetImpl =
  fixes to-set :: 's  $\Rightarrow$  'a set
  fixes inv :: 's  $\Rightarrow$  bool
  assumes to-set-inj:  $\llbracket \text{inv } s1; \text{inv } s2; \text{to-set } s1 = \text{to-set } s2 \rrbracket \Longrightarrow s1 = s2$ 
  assumes to-set-ex:  $\text{finite } a \Longrightarrow \exists s. a = \text{to-set } s \wedge \text{inv } s$ 
  assumes to-set-finite:  $\text{finite } (\text{to-set } s)$ 
begin

```

definition *f-to-set* :: 's list \Rightarrow 'a set set **where**
 $[\text{simp}]: f\text{-to-set } F = \text{set } (\text{map } \text{to-set } F)$

abbreviation *fs-to-set* :: 's list list \Rightarrow 'a set set set (\circ) **where**
 $\circ FF \equiv \text{set } (\text{map } f\text{-to-set } FF)$

lemma *to-set-inj-on*:
shows $\forall a \in \text{set } A. \text{inv } a \Longrightarrow \text{inj-on } \text{to-set } (\text{set } A)$
unfolding *inj-on-def*
by (*simp add: to-set-inj*)

lemma *set-set*:
 $\llbracket \forall a \in \text{set } A. \text{inv } a; \text{inv } h \rrbracket \Longrightarrow$
 $h \in \text{set } A \longleftrightarrow \text{to-set } h \in f\text{-to-set } A$
using *to-set-inj*
by *auto*

```

lemma f-to-set-ex:
  assumes finite ( $\bigcup F$ )
  shows  $\exists Fl. F = f\text{-to-set } Fl \wedge (\forall A \in \text{set } Fl. \text{inv } A)$ 
proof –
  from assms
  have finite  $F \forall A \in F. \text{finite } A$ 
    by (auto simp add: finiteUn-iff)
  thus ?thesis
proof (induct F)
  case empty
  thus ?case
    by auto
next
  case (insert A F)
  then obtain Fl Al where  $F = f\text{-to-set } Fl \wedge (\forall A \in \text{set } Fl. \text{inv } A) \wedge A = \text{to-set}$ 
     $Al \wedge \text{inv } Al$ 
    using to-set-ex[of A]
    by auto
  thus ?case
    by (rule-tac x=Al # Fl in exI) auto
qed
qed

abbreviation contains :: 'a  $\Rightarrow$  's  $\Rightarrow$  bool where
  contains a S  $\equiv a \in \text{to-set } S$ 

definition count :: 'a  $\Rightarrow$  's list  $\Rightarrow$  nat where
  count a F = length (filter (contains a) F)

lemma count-set:
  assumes distinct  $F \forall x \in \text{set } F. \text{inv } x$ 
  shows count a F = Family.count a (f-to-set F)
proof –
  have to-set ' $\{x \in \text{set } F. a \in \text{to-set } x\} = \{S \in \text{to-set} \text{ 'set } F. a \in S\}$ 
    by auto
  moreover
  have inj-on to-set ' $\{x \in \text{set } F. a \in \text{to-set } x\}$ 
    using to-set-inj ' $\langle \forall x \in \text{set } F. \text{inv } x \rangle$ 
    unfolding inj-on-def
    by auto
  ultimately
  have card ' $\{x \in \text{set } F. a \in \text{to-set } x\} = \text{card } \{S \in \text{to-set} \text{ 'set } F. a \in S\}$ 
    using card-image[of to-set ' $\{x \in \text{set } F. a \in \text{to-set } x\}$ ]
    by auto
  thus ?thesis
    using assms distinct-card[of filter (contains a) F, symmetric]
    unfolding count-def Family.count-def contains-def
    by simp

```

qed

definition *frankl-fun* :: 'a \Rightarrow 's list \Rightarrow int **where**
frankl-fun x F \equiv 2 * int (count x F) - int (length F)

lemma *frankl-fun-set*:
assumes *distinct Fs* \forall A \in set Fs. *inv A*
shows *frankl-fun a Fs* = *Frankl.frankl-fun a (f-to-set Fs)*
proof -
have *inj-on to-set (set Fs)*
using *assms to-set-inj*
by (*auto simp add: inj-on-def*)
hence *card (to-set ' set Fs)* = *card (set Fs)*
by (*rule card-image*)
thus ?thesis
using *distinct-card[OF assms(1)] count-set[OF assms, of a]*
unfolding *frankl-fun-def*
by *auto*

qed

end

6.1.1 Implementation of sets by sorted and distinct lists

abbreviation *sd* **where**
sd A \equiv *sorted A* \wedge *distinct A*
abbreviation *sdf* **where**
sdf F \equiv \forall A \in set F. *sd A*
abbreviation *sdff* **where**
sdff F \equiv \forall F \in set F. *sdf F*

For example, the family $\{\{1, 2, 3\}, \{2, 3, 4\}\}$ is represented by $[[1, 2, 3], [2, 3, 4]]$.

global-interpretation *SetImpl-lists*: *SetImpl set* :: nat list \Rightarrow nat set *sd*
defines *f-to-set-l* = *SetImpl-lists.f-to-set* **and**
count-l = *SetImpl-lists.count* **and**
frankl-fun-l = *SetImpl-lists.frankl-fun*
proof (*unfold-locales*)
fix a :: nat set
assume *finite a*
thus \exists s. a = set s \wedge *sd s*
apply (*rule-tac x=sorted-list-of-set a in exI*)
by *simp*
qed (*auto simp add: sorted-distinct-set-unique*)

abbreviation *dm* **where**
dm F n \equiv \bigcup *f-to-set-l F* \subseteq $\{0..<n\}$

abbreviation *dmf* **where**
 $dmf\ FF\ n \equiv \forall\ F \in set\ FF. dm\ F\ n$

abbreviation *fs-to-set-l* **where**
 $fs-to-set-l\ F \equiv set\ (map\ f-to-set-l\ F)$

6.1.2 Implementation of sets of nats by nats

For example, the family $\{\{1, 2, 3\}, \{2, 3, 4\}\}$ is represented by [14, 28]. Encoding and decoding functions are defined in the theory ListNat.

global-interpretation *SetImpl-nats*: *SetImpl set \circ nat2list $\lambda\ n.$ True*

defines *f-to-set-n* = *SetImpl-nats.f-to-set* **and**
count-n = *SetImpl-nats.count* **and**
frankl-fun-n = *SetImpl-nats.frankl-fun*

proof (*unfold-locales*)

fix *s1 s2*

assume (*set \circ nat2list*) *s1* = (*set \circ nat2list*) *s2*

thus *s1* = *s2*

using *sorted-nat2list*[*of s1*] *sorted-nat2list*[*of s2*]

using *distinct-nat2list*[*of s1*] *distinct-nat2list*[*of s2*]

using *sorted-distinct-set-unique*[*of nat2list s1 nat2list s2*]

using *inj-nat2list*

by (*simp add: inj-on-def*)

next

fix *a :: nat set*

assume *finite a*

thus $\exists\ s. a = (set \circ nat2list)\ s \wedge True$

by (*rule-tac x=list2nat (sorted-list-of-set a) in exI*) (*simp add: nat2list-list2nat*)

qed *simp-all*

ML⟨⟨

(* Converts a term that represents a family (set of sets) to a corresponding list of lists. If the term represents a list of families, then the transformation is applied to each family in the list and the term that represents the corresponding list of lists of lists is returned. *)

fun sets-to-lists F =

let

val c1 = Const (List.list.Cons, @{typ nat \Rightarrow nat list \Rightarrow nat list})

val c2 = Const (Set.insert, @{typ nat \Rightarrow nat set \Rightarrow nat set})

val c1' = Const (List.list.Cons, @{typ nat list \Rightarrow nat list list \Rightarrow nat list list})

val c2' = Const (Set.insert, @{typ nat set \Rightarrow nat set set \Rightarrow nat set set})

val c3 = Const (List.list.Cons, @{typ nat set set \Rightarrow nat set set list \Rightarrow nat set set list})

val c3' = Const (List.list.Cons, @{typ nat list list \Rightarrow nat list list list \Rightarrow nat list list list})

```

    val sub1 = (c2, c1)
    val sub2 = (c2', c1')
    val sub3 = (c3, c3')
    val sub4 = (@{term {}::nat set}, @term []::nat list)
    val sub5 = (@{term {}::nat set set}, @term []::nat list list)
    val sub6 = (@{term []::nat set set list}, @term []::nat list list list)
  in
    subst-free [sub1, sub2, sub3, sub4, sub5, sub6] F
  end;

(* Given a term that represents a family returns a theorem stating that it is obtained
   by applying f-to-set-l to the corresponding list of list representation.
   E.g. f-to-set-thm @term {{1}, {1, 2}}::nat set set returns
       {{1}, {1, 2}} = f-to-set-l [[1], [1, 2]]: thm
*)

fun f-to-set-l-thm F =
  let
    fun mk-f-to-set-l-thm F =
      let
        val term-eq = @term HOL.eq :: nat set set => nat set set => bool
        val term-f-to-set = @term f-to-set-l :: nat list list => nat set set
      in
        HOLogic.mk-Trueprop (list-comb (term-eq, [F, term-f-to-set $ sets-to-lists
F]))
      end
    in
      Goal.prove @context [] [] (mk-f-to-set-l-thm F) (fn - => (simp-tac @context
1))
    end
  end

(* Similar as the previous one, except f-to-set-l is applied to every family i the given
   list.
   E.g. list-f-to-set-thm @term [{1}, {2}], {{1, 2}, {1, 3}}::nat set set list
   returns
       [{1}, {2}], {{1, 2}, {1, 3}} = map f-to-set-l [[[1], [2]], [[1, 2], [1, 3]]] : thm
*)

fun list-f-to-set-l-thm F =
  let
    fun mk-list-f-to-set-l-thm FF =
      let
        val term-eq = @term HOL.eq :: nat set set list => nat set set list => bool
        val term-map-f-to-set-l = @term map f-to-set-l :: nat list list list => nat set
set list
      in
        HOLogic.mk-Trueprop (list-comb (term-eq, [FF, term-map-f-to-set-l $ sets-to-lists
FF]))
      end
    in
      Goal.prove @context [] [] (mk-list-f-to-set-l-thm FF) (fn - => (simp-tac @context
1))
    end
  end

```



```

in
  Goal.prove @{\context} [] [] (mk-list-f-to-set-l-thm F) (fn - => (simp-tac @{\context}
1))
end

(* Converts a term that represents a family (set of sets) to a corresponding list of
nats. *)

fun sets-to-nats t =
  let
    val cons-list = Const (List.list.Cons, @{\typ nat => nat list => nat list})
    val insert-list = Const (Set.insert, @{\typ nat set => nat set set => nat set set})
  in
    case t of insert-list $ S $ L =>
      cons-list $ (@{\term list2nat} $ sets-to-lists S) $ sets-to-nats L
    | @{\term {}::nat set set} => @{\term []:: nat list}
    | - => raise Fail (pattern not matched)
  end

(* Given a term that represents a family returns a theorem stating that it is obtained
by applying f-to-set-n to the corresponding list of nats representation.
E.g. f-to-set-n-thm @{\term {{1}, {1, 2}}::nat set set} returns
{{1}, {1, 2}} = f-to-set-n [list2nat [1], list2nat [1, 2]]: thm
*)

fun f-to-set-n-thm F =
  let
    fun mk-f-to-set-n-thm F =
      let
        val term-eq = @{\term HOL.eq :: nat set set => nat set set => bool}
        val term-f-to-set = @{\term f-to-set-n :: nat list => nat set set}
      in
        Hologic.mk-Trueprop (list-comb (term-eq, [F, term-f-to-set $ sets-to-nats
F]))
      end
    val ss = put-simpset HOL-ss @{\context} addsimps
      ([@{\thm nat2list-list2nat}, @{\thm SetImpl-nats.f-to-set-def}] @
@{\thms List.list.map} @
@{\thm comp-apply} @
@{\thms distinct.simps} @
@{\thms List.list.set} @
@{\thm empty-iff}, @{\thm insert-iff}] @
@{\thms eq-numeral-simps} @ @{\thm zero-neq-one}] @
@{\thms Num.num.distinct} @ @{\thms num.inject} @
@{\thm sorted.Nil}, @{\thm sorted-single}, @{\thm sorted-many}] @
@{\thms le-numeral-simps} @ @{\thms le-num-simps} @ @{\thms
less-num-simps} @ @{\thm le0})
  in
    Goal.prove @{\context} [] [] (mk-f-to-set-n-thm F) (fn - => (asm-full-simp-tac ss

```

```

1))
end
>>

```

```

end
theory UnionClosedImpl
imports UnionClosed FamilyImpl
begin

```

6.2 Abstract representation of union closed families

Sets that support empty set, unions and powerset.

```

locale SetUnionImpl = SetImpl to-set for to-set :: 's  $\Rightarrow$  'a set +
  fixes empty :: 's
  assumes empty-set: to-set empty = {}
  assumes empty-inv: inv empty

  fixes union :: 's  $\Rightarrow$  's  $\Rightarrow$  's (infixl  $\sqcup$  100)
  assumes union-set: to-set (s1  $\sqcup$  s2) = to-set s1  $\cup$  to-set s2
  assumes union-inv:  $\llbracket$ inv s1; inv s2 $\rrbracket \Longrightarrow$  inv (s1  $\sqcup$  s2)

  fixes pow :: 's  $\Rightarrow$  's list
  assumes pow-set: f-to-set (pow s) = Pow (to-set s)
  assumes pow-inv: inv s  $\Longrightarrow$   $\forall$  A  $\in$  set (pow s). inv A
  assumes pow-distinct: inv s  $\Longrightarrow$  distinct (pow s)
begin

```

```

lemma subset-in-pow:
  assumes inv A and inv S and to-set A  $\subseteq$  to-set S
  shows A  $\in$  set (pow S)
proof -
  have to-set A  $\in$  f-to-set (pow S)
  using  $\langle$ to-set A  $\subseteq$  to-set S $\rangle$  pow-set[of S]
  by simp
  thus ?thesis
  using  $\langle$ inv A $\rangle$   $\langle$ inv S $\rangle$ 
  using pow-inv to-set-inj
  by auto
qed

```

```

definition Union :: 's list  $\Rightarrow$  's where
  Union F = foldl union empty F

```

```

lemma Union-set:
  shows to-set (Union F) =  $\bigcup$  (f-to-set F)

```

by (induct F rule: rev-induct) (auto simp add: Union-def empty-set union-set)

lemma *Union-inv*:

shows $\forall A \in \text{set } F. \text{inv } A \implies \text{inv } (\text{Union } F)$

unfolding *Union-def*

by (induct F rule: rev-induct) (auto simp add: empty-inv union-inv)

abbreviation *union-with-all* :: $'s \Rightarrow 's \text{ list} \Rightarrow 's \text{ list}$ **where**

union-with-all $A F \equiv (\text{map } (op \sqcup A) F)$

lemma *union-with-all-set*:

shows $\text{map } \text{to-set } (\text{union-with-all } A F) = \text{map } (op \cup (\text{to-set } A) \circ \text{to-set}) F$

by (simp add: comp-def union-set)

lemma *union-with-all-set'*:

shows $(\text{to-set } '(\text{op } \sqcup A \text{ ' set } F)) = (\text{op } \cup (\text{to-set } A) \text{ ' to-set ' set } F)$

proof–

have $\text{set } (\text{map } \text{to-set } (\text{union-with-all } A F)) = \text{set } (\text{map } (op \cup (\text{to-set } A) \circ \text{to-set}) F)$

using *union-with-all-set*[of $A F$]

by (auto simp add: comp-def)

thus ?thesis

by (simp add: image-comp[symmetric])

qed

definition *insert-set* :: $'s \Rightarrow 's \text{ list} \Rightarrow 's \text{ list}$ **where**

insert-set $A F \equiv (\text{if } A \in \text{set } F \text{ then } F \text{ else } A \# F)$

lemma *insert-set-set* [simp]:

shows $\text{set } (\text{insert-set } A F) = \{A\} \cup \text{set } F$

by (auto simp add: insert-set-def)

definition *insert-sets* :: $'s \text{ list} \Rightarrow 's \text{ list} \Rightarrow 's \text{ list}$ **where**

insert-sets $l F = \text{foldl } (\lambda F' a. \text{insert-set } a F') F l$

lemma *insert-sets-set* [simp]:

shows $\text{set } (\text{insert-sets } l F) = \text{set } l \cup \text{set } F$

unfolding *insert-sets-def*

by (induct l rule: rev-induct) auto

lemma *insert-sets-remdups*:

shows $\text{distinct } F \implies \text{insert-sets } l F = \text{remdups } (\text{rev } l @ F)$

unfolding *insert-sets-def*

by (induct l rule: rev-induct) (auto simp add: insert-set-def distinct-remdups-id)

lemma *insert-sets-distinct*:
 shows $\text{distinct } F \implies \text{distinct } (\text{insert-sets } l \ F)$
 using *insert-sets-remdups*[of $F \ l$]
 by *auto*

definition *insert-and-close* :: $'s \Rightarrow 's \text{ list} \Rightarrow 's \text{ list}$ **where**
 [simp]: $\text{insert-and-close } A \ F \equiv \text{insert-sets } ([A] \ @ \ \text{union-with-all } A \ F) \ F$

lemma *insert-and-close-set*:
 assumes *distinct* F
 shows $f\text{-to-set } (\text{insert-and-close } h \ F) = \text{UnionClosed.insert-and-close } (to\text{-set } h) \ (f\text{-to-set } F)$
 using *assms*
 using *insert-sets-remdups*
 using *union-with-all-set'*[of $h \ F$, *THEN sym*]
 by *auto*

lemma *insert-and-close-inv*:
 assumes $\forall l \in \text{set } F. \text{inv } l \text{ and } \text{inv } h \text{ and } \text{distinct } F$
 shows $\forall l \in \text{set } (\text{insert-and-close } h \ F). \text{inv } l$
 using *assms*
 by (*auto simp add: insert-sets-remdups union-inv*)

lemma *insert-and-close-subset*:
 assumes $\forall l \in \text{set } F. \text{to-set } l \subseteq S \text{ and } \text{to-set } h \subseteq S \text{ and } \text{distinct } F$
 shows $\forall l \in \text{set } (\text{insert-and-close } h \ F). \text{to-set } l \subseteq S$
proof (*safe*)
 fix $x \ l$
 assume $l \in \text{set } (\text{insert-and-close } h \ F) \ x \in \text{to-set } l$
 hence $x \in \bigcup (f\text{-to-set } (\text{insert-and-close } h \ F))$
 by *auto*
 hence $x \in \bigcup (\text{UnionClosed.insert-and-close } (to\text{-set } h) \ (f\text{-to-set } F))$
 using $\langle \text{distinct } F \rangle$
 apply (*subst insert-and-close-set*[of $F \ h$, *THEN sym*])
 by *simp-all*
 thus $x \in S$
 using *assms*
 by (*auto split: if-split-asm*)
qed

definition *insert-and-close-additional* :: $'s \Rightarrow 's \text{ list} \Rightarrow 's \text{ list} \Rightarrow 's \text{ list}$ **where**
 [simp]: $\text{insert-and-close-additional } A \ F \ Fc \equiv \text{insert-sets } ([A] \ @ \ \text{union-with-all } A \ F \ @ \ \text{union-with-all } A \ Fc) \ F$

lemma *insert-and-close-additional-set*:

```

assumes distinct F
shows f-to-set (insert-and-close-additional A F Fc) = UnionClosed.insert-and-close-additional
(to-set A) (f-to-set F) (f-to-set Fc)
using assms
using insert-sets-remdups
using union-with-all-set'[of A F, THEN sym]
using union-with-all-set'[of A Fc, THEN sym]
by auto

```

lemma *insert-and-close-additional-subset:*

```

assumes to-set A ⊆ S and
   $\forall l \in \text{set } F. \text{to-set } l \subseteq S$  and
   $\forall l \in \text{set } Fc. \text{to-set } l \subseteq S$ 
assumes distinct F
shows  $\forall l \in \text{set } (\text{insert-and-close-additional } A \ F \ Fc). \text{to-set } l \subseteq S$ 
proof (safe)
  fix x l
  assume  $l \in \text{set } (\text{insert-and-close-additional } A \ F \ Fc) \ x \in \text{to-set } l$ 
  hence  $x \in \bigcup (\text{f-to-set } (\text{insert-and-close-additional } A \ F \ Fc))$ 
  by auto
  hence  $x \in \bigcup (\text{UnionClosed.insert-and-close-additional } (\text{to-set } A) \ (\text{f-to-set } F) \ (\text{f-to-set } Fc))$ 
  using <distinct F>
  apply (subst insert-and-close-additional-set[of F A Fc, THEN sym])
  by simp-all
  thus  $x \in S$ 
  using assms
  by (auto split: if-split-asm)
qed

```

lemma *insert-and-close-additional-inv:*

```

assumes inv A and  $\forall l \in \text{set } F. \text{inv } l$  and  $\forall l \in \text{set } Fc. \text{inv } l$ 
assumes distinct F
shows  $\forall l \in \text{set } (\text{insert-and-close-additional } A \ F \ Fc). \text{inv } l$ 
using assms
by (auto simp add: insert-sets-remdups union-inv)

```

lemma *insert-and-close-additional-cong:*

```

assumes set F = set F' and distinct F and distinct F'
shows set (insert-and-close-additional A F Fc) =
  set (insert-and-close-additional A F' Fc)
using assms
by (simp add: insert-sets-remdups)

```

definition *close :: 's list \Rightarrow 's list where*

```

close F = foldl ( $\lambda F A. \text{insert-and-close } A \ F$ ) [] F

```

```

lemma close-snoc:
  shows close ( $F @ [A]$ ) = insert-and-close  $A$  (close  $F$ )
unfolding close-def
by simp

lemma close-distinct:
  shows distinct (close  $A$ )
unfolding close-def
by (induct  $A$  rule: rev-induct) (auto simp add: insert-sets-distinct)

lemma close-set:
  shows f-to-set (close  $A$ ) = UnionClosed.closure (f-to-set  $A$ )
proof (induct  $A$  rule: rev-induct)
  case (snoc  $l$   $a$ )
  let  $?L = \lambda x. \text{foldl } (\lambda F A. \text{local.insert-and-close } A F) [] x$ 
  show  $?case$ 
    using close-distinct[of  $a$ ]
    using insert-and-close-set[of  $?L$   $a$ ]
    using closure-insert[of f-to-set  $a$  to-set  $l$ ]
    using snoc(1)
    unfolding close-def
    by simp
next
  case Nil
  thus  $?case$ 
    by (simp add: close-def closure-def)
qed

lemma close-completeness:
  assumes inv  $A$  and  $\forall a \in \text{set } F. \text{inv } a$ 
  assumes to-set  $A \in \text{UnionClosed.closure } (\text{f-to-set } F)$ 
  shows  $A \in \text{set } (\text{close } F)$ 
using assms
proof (induct  $F$  arbitrary:  $A$  rule: rev-induct)
  case Nil
  thus  $?case$ 
    by (simp add: closure-def)
next
  case (snoc  $a$   $l$ )
  from  $\langle \text{to-set } A \in \text{closure } (\text{f-to-set } (l @ [a])) \rangle$ 
  have  $\text{to-set } A = \text{to-set } a \vee \text{to-set } A \in \text{closure } (\text{f-to-set } l) \vee \text{to-set } A \in \text{op} \cup$ 
     $(\text{to-set } a) \text{ ' closure } (\text{to-set ' set } l)$ 
  using closure-insert[of f-to-set  $l$  to-set  $a$ ]
  by simp
  thus  $?case$ 
proof
  assume to-set  $A = \text{to-set } a$ 
  thus  $?thesis$ 
    using snoc(2) snoc(3)

```

```

    by (simp add: to-set-inj close-snoc)
next
  assume to-set A ∈ closure (f-to-set l) ∨
    to-set A ∈ op ∪ (to-set a) ‘ closure (to-set ‘ set l)
  thus ?thesis
proof
  assume to-set A ∈ closure (f-to-set l)
  thus ?thesis
    using snoc(1) snoc(2) snoc(3)
    by (simp add: close-snoc)
next
  assume to-set A ∈ op ∪ (to-set a) ‘ closure (to-set ‘ set l)
  then obtain X where X ∈ closure (f-to-set l) to-set A = to-set a ∪ X
    by auto
  hence finite X
    using finiteUn-iff[of closure (f-to-set l)]
    using finite-closure[of f-to-set l]
    by (auto simp add: to-set-finite)
  then obtain Y where X = to-set Y inv Y
    using to-set-ex[of X]
    by auto
  hence Y ∈ set (close l)
    using snoc(1)[of Y] ⟨X ∈ closure (f-to-set l)⟩ snoc(3)
    by simp
  hence A = a ∪ Y
    using ⟨to-set A = to-set a ∪ X⟩ ⟨X = to-set Y⟩
    using snoc(2) snoc(3) ⟨inv Y⟩
    by (auto simp add: union-set union-inv to-set-inj)
  hence A ∈ op ∪ a ‘ set (close l)
    using ⟨Y ∈ set (close l)⟩
    by simp
  thus ?thesis
    by (simp add: close-snoc)
qed
qed
qed

```

```

lemma close-inv:
  assumes ∀ l ∈ set F. inv l
  shows ∀ l ∈ set (close F). inv l
using assms
proof (induct F rule: rev-induct)
  case (snoc a l)
  thus ?case
    using close-distinct[of l]
    by (auto simp add: close-def insert-sets-remdups union-inv)
qed (simp add: close-def)

```

```

lemma close-subset:

```

```

assumes  $\forall l \in \text{set } F. \text{ to-set } l \subseteq S$ 
shows  $\forall l \in \text{set } (\text{close } F). \text{ to-set } l \subseteq S$ 
using assms
proof (induct F rule: rev-induct)
  case (snoc a l)
  thus ?case
    using insert-and-close-subset[of close l S a]
    using close-distinct[of l]
    unfolding close-def
    by simp
qed (simp add: close-def)

definition close-and-insert-empty :: 's list  $\Rightarrow$  's list where
  close-and-insert-empty F  $\equiv$ 
    let X = close F
    in (if empty  $\in$  set X then X else empty  $\#$  X)

lemma close-and-insert-empty-set':
  shows  $\text{set } (\text{close-and-insert-empty } F) = \text{set } (\text{close } F) \cup \{\text{empty}\}$ 
by (auto simp add: Let-def close-and-insert-empty-def)

lemma close-and-insert-empty-set:
  shows f-to-set (close-and-insert-empty F) =
    (UnionClosed.closure (f-to-set F))  $\cup$   $\{\{\}\}$ 
proof–
  have to-set ' set (close F) = closure (to-set ' set F)
  by (metis f-to-set-def close-set image-set)
  thus ?thesis
    unfolding close-and-insert-empty-def
    by (auto simp add: empty-set Let-def) (metis empty-set image-eqI)
qed

primrec union-closed :: 's list  $\Rightarrow$  bool where
  union-closed [] = True
  | union-closed (h # t) = (list-all ( $\lambda x. h \sqcup x \in \text{set } (h \# t)$ ) t)  $\wedge$  union-closed t

lemma union-closed-set:
  shows union-closed F  $\implies$  UnionClosed.union-closed (f-to-set F)
proof (induct F)
  case Nil
  thus ?case
    by (simp add: UnionClosed.union-closed-def)
next
  case (Cons h t)
  thus ?case
    using union-closed-insert[of f-to-set t to-set h]
    by (auto simp add: list-all-iff) (metis imageI union-set)
qed

```


primrec *union-closed-additional'* :: 's list \Rightarrow 's list \Rightarrow bool **where**
union-closed-additional' F [] = True
| *union-closed-additional'* F (h # t) = (list-all (λ x. h \sqcup x \in set F) F \wedge
union-closed-additional' F t)

lemma *union-closed-additional'-set*:
assumes *union-closed-additional'* F Fc
shows $\forall A \in f\text{-to-set } Fc. \forall B \in (f\text{-to-set } F). A \cup B \in (f\text{-to-set } F)$
using *assms*
proof (*induct* Fc)
 case Nil
 thus ?case
 by *simp*
next
 case (Cons h t)
 thus ?case
 by (*simp add: list-all-iff*) (*metis union-set imageI*)
qed

definition *union-closed-additional* :: 's list \Rightarrow 's list \Rightarrow bool **where**
union-closed-additional F Fc \longleftrightarrow *union-closed* F \wedge *union-closed-additional'* F Fc

lemma *union-closed-additional-set*:
assumes *union-closed-additional* F Fc
shows *UnionClosed.union-closed-additional* (f-to-set F) (f-to-set Fc)
using *assms*
unfolding *union-closed-additional-def*
using *union-closed-set[of F]* *union-closed-additional'-set[of F Fc]*
by (*auto*) (*metis sup-commute*)

end

6.2.1 Implementation by sorted and distinct lists

Power set of a list

primrec *Pow-l* :: 'a list \Rightarrow 'a list list **where**
Pow-l [] = [[]]
| *Pow-l* (h # t) =
 (let X = *Pow-l* t
 in X @ map (op # h) X)

lemma *Pow-l-set*:
shows set (map set (*Pow-l* X)) = *Pow* (set X)
proof (*induct* X)
 case Nil
 thus ?case
 by *simp*
next

```

    case (Cons a l')
    thus ?case
      using Cons[THEN sym]
      by (auto simp add: Let-def comp-def Pow-insert)
qed

lemma Pow-l-distinct:
  shows distinct A  $\implies$  distinct (Pow-l A)
proof (induct A)
  case Nil
  thus ?case
    by simp
next
  case (Cons a A)
  thus ?case
  proof (auto simp add: Let-def distinct-map inj-on-def)
    fix x
    assume a  $\notin$  set A
    assume a  $\#$  x  $\in$  set (Pow-l A)
    hence set (a  $\#$  x)  $\in$  Pow (set A)
      using Pow-l-set[of A]
    by (auto simp del: set-simps)
    thus False
      using  $\langle a \notin \text{set } A \rangle$ 
      by simp
  qed
qed

lemma Pow-l-elems-distinct:
  assumes distinct A and x  $\in$  set (Pow-l A)
  shows distinct x
using assms
proof (induct A arbitrary: x)
  case Nil
  thus ?case
    by simp
next
  case (Cons h t)
  thus ?case
  proof (cases x  $\in$  set (Pow-l t))
    case True
    thus ?thesis
      using Cons
      by simp
  next
    case False
    then obtain y where y  $\in$  set (Pow-l t) x = h  $\#$  y
      using Cons(3)
      by (auto simp add: Let-def)
  qed
qed

```

```

    thus ?thesis
      using Cons(2) Cons(1)
      using Pow-l-set[of t]
      by auto
  qed
qed

lemma Pow-l-elems-sorted:
  assumes sorted A and  $x \in \text{set } (\text{Pow-l } A)$ 
  shows sorted x
using assms
proof (induct A arbitrary: x)
  case Nil
  thus ?case
    by simp
next
  case (Cons t h)
  show ?case
  proof (cases  $x \in \text{set } (\text{Pow-l } t)$ )
    case True
    thus ?thesis
      using Cons
      by simp
  next
    case False
    then obtain y where  $y \in \text{set } (\text{Pow-l } t)$   $x = h \# y$ 
      using Cons(4)
      by (auto simp add: Let-def)
    thus ?thesis
      using Cons(1) Cons(2) Cons(3)
      using Pow-l-set[of t]
      using sorted-Cons[of h y]
      by auto
  qed
qed
qed

```

```

lemma Pow-l-no-perms:
  assumes distinct A and sorted A
  assumes  $x \in \text{set } (\text{Pow-l } A)$  and  $y \in \text{set } (\text{Pow-l } A)$ 
  assumes  $\text{set } x = \text{set } y$ 
  shows  $x = y$ 
using assms
using Pow-l-elems-sorted[of A x]
using Pow-l-elems-distinct[of A x]
using Pow-l-elems-sorted[of A y]
using Pow-l-elems-distinct[of A y]
by (simp add: sorted-distinct-set-unique)

```

Merging two sorted lists

```

fun merge :: 'a::linorder list  $\Rightarrow$  'a::linorder list  $\Rightarrow$  'a::linorder list (infixl  $\sqcup$  100)
where
  merge [] l = l
| merge l [] = l
| merge (h1 # t1) (h2 # t2) =
  (if h1 < h2 then
    h1 # merge t1 (h2 # t2)
  else if h1 > h2 then
    h2 # merge (h1 # t1) t2
  else
    h1 # merge t1 t2)

lemma merge-set:
  shows set (l1  $\sqcup$  l2) = set l1  $\cup$  set l2
by (induct l1 l2 rule: merge.induct) auto

lemma merge-sorted:
  assumes sorted l1 and sorted l2
  shows sorted (l1  $\sqcup$  l2)
using assms
by (induct l1 l2 rule: merge.induct) (auto simp add: sorted-Cons merge-set)

lemma merge-distinct:
  assumes distinct l1 and distinct l2 and sorted l1 and sorted l2
  shows distinct (l1  $\sqcup$  l2)
using assms
by (induct l1 l2 rule: merge.induct) (auto simp add: sorted-Cons merge-set)

global-interpretation SetUnionImpl-lists: SetUnionImpl  $\lambda$  (l::nat list). sorted l
 $\wedge$  distinct l set [] merge Pow-l
defines
  close-l = SetUnionImpl-lists.close and
  insert-set-l = SetUnionImpl-lists.insert-set and
  insert-sets-l = SetUnionImpl-lists.insert-sets and
  insert-and-close-l = SetUnionImpl-lists.insert-and-close and
  Union-l = SetUnionImpl-lists.Union and
  close-insert-empty-l = SetUnionImpl-lists.close-and-insert-empty and
  union-closed-l = SetUnionImpl-lists.union-closed and
  union-closed-additional'-l = SetUnionImpl-lists.union-closed-additional' and
  union-closed-additional-l = SetUnionImpl-lists.union-closed-additional
proof (unfold-locales)
  fix s :: nat list
  show SetImpl.f-to-set set (Pow-l s) = Pow (set s)
    unfolding f-to-set-l-def[symmetric]
    unfolding SetImpl-lists.f-to-set-def[of Pow-l s]
    by (rule Pow-l-set)
qed (auto simp add: Pow-l-elems-sorted Pow-l-elems-distinct merge-set merge-sorted
  merge-distinct sorted-distinct-set-unique Pow-l-distinct)

```

6.2.2 Implementation by sets represented as natural numbers

For example, the family $\{\{0, 1, 2\}, \{1, 2, 3\}\}$ is represented as $[7, 14]$.

Powerset

definition *pow-n* :: *nat* \Rightarrow *nat list* **where**
pow-n *n* = *map list2nat (Pow-l (nat2list n))*

Union is obtained by bitwise disjunction – this is a naive implementation

function *bitor* :: *nat* \Rightarrow *nat* \Rightarrow *nat* **where**

bitor *x y* =
 (if *x* = 0 then *y*
 else if *y* = 0 then *x* else
 let (*xd*, *xm*) = *Divides.divmod-nat* *x* 2;
 (*yd*, *ym*) = *Divides.divmod-nat* *y* 2 in
 if (*xm* = 1 | *ym* = 1)
 then 1 + 2 * *bitor* *xd yd*
 else 2 * *bitor* *xd yd*)

by *pat-completeness auto*

termination

by (*relation measure* ($\lambda (x, -). x$), *auto simp add: divmod-nat-div-mod*)

declare *bitor.simps*[*simp del*]

lemma *bitor-set*:

shows *set* (*nat2list* (*bitor* *x y*)) = *set* (*nat2list* *x*) \cup *set* (*nat2list* *y*)

proof (*induct* *x y* *rule: bitor.induct*)

case (1 *x' y'*)

show ?*case*

proof (*cases* *x' = 0*)

case *True*

thus ?*thesis*

by (*simp add: bitor.simps nat2list-def*)

next

case *False*

show ?*thesis*

proof (*cases* *y' = 0*)

case *True*

thus ?*thesis*

using $\langle x' \neq 0 \rangle$

by (*simp add: bitor.simps nat2list-def*)

next

case *False*

show ?*thesis*

proof (*cases* *x' mod 2* \neq *Suc 0* \wedge *y' mod 2* \neq *Suc 0*)

case *True*

hence *bitor* *x' y'* = 2 * *bitor* (*x' div 2*) (*y' div 2*)

using $\langle x' \neq 0 \rangle$ $\langle y' \neq 0 \rangle$

using *bitor.simps*[*of* *x' y'*]

by (*simp add: split-def Let-def div-nat-def*[*THEN sym*] *mod-nat-def*[*THEN*

```

sym] split: if-split-asm)
  hence nat2list (bitor x' y') = map (op + 1) (nat2list (bitor (x' div 2) (y'
div 2)))
    by (simp add: nat2list-even)
  moreover
  have set (nat2list (bitor (x' div 2) (y' div 2))) = set (nat2list (x' div 2))
  ∪ set (nat2list (y' div 2))
    using 1(2)[of Divides.divmod-nat x' 2 x' div 2 x' mod 2
      Divides.divmod-nat y' 2 y' div 2 y' mod 2]
    using ⟨x' ≠ 0⟩ ⟨y' ≠ 0⟩ True
    unfolding div-nat-def mod-nat-def
    by (simp add: divmod-nat-div-mod)
  moreover
  from True
  have x' mod 2 = 0 y' mod 2 = 0
    by auto
  hence x' = 2 * (x' div 2) y' = 2 * (y' div 2)
    by auto
  hence nat2list x' = map (op + 1) (nat2list (x' div 2))
    nat2list y' = map (op + 1) (nat2list (y' div 2))
    using nat2list-even[of x' div 2] nat2list-even[of y' div 2]
    by auto
  ultimately
  show ?thesis
    by auto
next
case False
  hence bitor x' y' = 2 * bitor (x' div 2) (y' div 2) + 1
    using ⟨x' ≠ 0⟩ ⟨y' ≠ 0⟩
    using bitor.simps[of x' y']
    by (simp add: split-def Let-def div-nat-def[THEN sym] mod-nat-def[THEN
sym] split: if-split-asm)
  hence nat2list (bitor x' y') = 0 # map (op + 1) (nat2list (bitor (x' div 2)
(y' div 2)))
    using nat2list-odd
    by simp
  moreover
  have set (nat2list (bitor (x' div 2) (y' div 2))) = set (nat2list (x' div 2))
  ∪ set (nat2list (y' div 2))
    using 1(1)[of Divides.divmod-nat x' 2 x' div 2 x' mod 2
      Divides.divmod-nat y' 2 y' div 2 y' mod 2]
    using ⟨x' ≠ 0⟩ ⟨y' ≠ 0⟩ False
    by simp (simp add: divmod-nat-div-mod)
  moreover
  from False
  have *: x' mod 2 = 0 ∧ y' mod 2 = 1 ∨ x' mod 2 = 1 ∧ y' mod 2 = 0 ∨
x' mod 2 = 1 ∧ y' mod 2 = 1
    by auto
  have set (nat2list x') ∪ set (nat2list y') = {0} ∪ (op + 1 ` (set (nat2list

```

```

(x' div 2)) ∪ set (nat2list (y' div 2))))
proof-
{
  assume x' mod 2 = 0 ∧ y' mod 2 = 1
  hence x' = 2 * (x' div 2) ∧ y' = 2 * (y' div 2) + 1
    using mod-mult-div-eq[of y' 2, THEN sym] mod-mult-div-eq[of x' 2,
THEN sym]
  by simp
  hence ?thesis
    using nat2list-even[of x' div 2] nat2list-odd[of y' div 2]
    by auto
}
moreover
{
  assume x' mod 2 = 1 ∧ y' mod 2 = 0
  hence x' = 2 * (x' div 2) + 1 ∧ y' = 2 * (y' div 2)
    using mod-mult-div-eq[of y' 2, THEN sym] mod-mult-div-eq[of x' 2,
THEN sym]
  by simp
  hence ?thesis
    using nat2list-even[of y' div 2] nat2list-odd[of x' div 2]
    by auto
}
moreover
{
  assume x' mod 2 = 1 ∧ y' mod 2 = 1
  hence x' = 2 * (x' div 2) + 1 ∧ y' = 2 * (y' div 2) + 1
    using mod-mult-div-eq[of y' 2, THEN sym] mod-mult-div-eq[of x' 2,
THEN sym]
  by simp
  hence ?thesis
    using nat2list-odd[of x' div 2] nat2list-odd[of y' div 2]
    by auto
}
ultimately
show ?thesis
  using *
  by blast
qed
ultimately
show ?thesis
  by simp
qed
qed
qed
qed

```

global-interpretation *SetUnionImpl-nats*: *SetUnionImpl* λ *n*. *True set* ∘ *nat2list*

```

0 bitor pow-n
defines
  close-n = SetUnionImpl-nats.close and
  insert-set-n = SetUnionImpl-nats.insert-set and
  insert-sets-n = SetUnionImpl-nats.insert-sets and
  insert-and-close-n = SetUnionImpl-nats.insert-and-close and
  Union-n = SetUnionImpl-nats.Union and
  close-and-insert-empty-n = SetUnionImpl-nats.close-and-insert-empty and
  union-closed-n = SetUnionImpl-nats.union-closed and
  union-closed-additional'-n = SetUnionImpl-nats.union-closed-additional' and
  union-closed-additional-n = SetUnionImpl-nats.union-closed-additional
proof (unfold-locales)
next
  show (set ∘ nat2list) 0 = {}
  by (simp add: nat2list-def)
next
  fix s1 s2
  show (set ∘ nat2list) (bitor s1 s2) = (set ∘ nat2list) s1 ∪ (set ∘ nat2list) s2
  using bitor-set
  by simp
next
  fix s
  show SetImpl.f-to-set (set ∘ nat2list) (pow-n s) = Pow ((set ∘ nat2list) s)
  proof–
    have (set ∘ nat2list ∘ list2nat) ‘ set (Pow-l (nat2list s)) = set ‘ set (Pow-l
(nat2list s))
    proof (rule image-cong, simp)
      fix x
      assume x ∈ set (Pow-l (nat2list s))
      thus (set ∘ nat2list ∘ list2nat) x = set x
      using sorted-nat2list[of s] distinct-nat2list[of s]
      using nat2list-list2nat[of x] Pow-l-elems-sorted[of nat2list s x] Pow-l-elems-distinct[of
nat2list s x]
      by simp
    qed
    thus ?thesis
      unfolding f-to-set-n-def[symmetric]
      unfolding SetImpl-nats.f-to-set-def
      unfolding pow-n-def
      using Pow-l-set[of nat2list s]
      by simp
  qed
next
  fix s
  show distinct (pow-n s)
  unfolding pow-n-def
  by (auto simp add: distinct-map distinct-nat2list Pow-l-distinct
inj-on-def inj-list2nat
Pow-l-elems-sorted[of nat2list s] sorted-nat2list

```



```

Pow-l-elems-distinct[of nat2list s])
qed simp-all

end

```

7 Weights and shares

```

theory WeightsShares
imports Main Frankl
begin

```

7.1 Weight functions

A technique, introduced by Poonen, used for analyzing Frankl's conjecture is based on the concept of weights and shares.

definition *weight-fun* :: $('a \Rightarrow 'b::\{\text{zero}, \text{ord}\}) \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$ **where**
 $\text{weight-fun } w \ X \equiv (\forall a \in X. w \ a \geq 0) \wedge (\exists a \in X. w \ a > 0)$

definition *set-weight* :: $('a \Rightarrow 'b::\{\text{comm-monoid-add}\}) \Rightarrow 'a \text{ set} \Rightarrow 'b$ (**infixl** \triangleright_s 100) **where**
 $w \triangleright_s S = (\sum_{x \in S. w \ x})$

definition *family-weight* :: $('a \Rightarrow 'b::\{\text{comm-monoid-add}\}) \Rightarrow 'a \text{ set set} \Rightarrow 'b$ (**infixl** \triangleright_f 100) **where**
 $w \triangleright_f F = (\sum_{S \in F. w \triangleright_s S})$

lemma *set-weight-cong*:
assumes $\forall v \in S. w \ v = w' \ v$
shows $w \triangleright_s S = w' \triangleright_s S$
using *assms*
unfolding *set-weight-def*
by (*subst sum.cong*) *auto*

lemma *family-weight-lemma*:
assumes *finite* $(\bigcup F)$
shows $w \triangleright_f F = (\sum_{a \in \bigcup F. (w \ a) * \text{count } a \ F})$
proof–
have $w \triangleright_f F = (\sum_{S \in F. \sum_{a \in S. w \ a})$
by (*simp add: family-weight-def set-weight-def*)
also have $\dots = (\sum_{a \in \bigcup F. (w \ a) * (\sum_{S \in F. a \in S. 1)})$
proof–
let $?S = \lambda a. \{S \in F. a \in S\} \times \{a\}$
let $?S' = \lambda x. \{(S, a). S \in F \wedge a \in S\}$
let $?CS = ?S \text{ ' } (\bigcup F)$

have $(\sum_{S \in F. \sum_{a \in S. w \ a}) = (\sum_{(S, a) \in (\text{SIGMA } S:F. S). w \ a})$
using *assms*
by (*simp add: sum.Sigma finiteUn-iff*)

```

also have ... = (∑ (S, a) ∈ ?S' a. w a)
  using Collect-case-prod-Sigma[of λ x. x ∈ F λ x y. y ∈ x, THEN sym]
  by auto
also have ... = (∑ (S, a) ∈ ∪ ?CS. w a)
  by (rule sum.cong) auto
also have ... = (∑ A ∈ ?CS. (∑ x ∈ A. w (snd x)))
  using assms
  by (subst sum.Union-disjoint) (auto simp add: split-def finiteUn-iff)
also have ... = (∑ a ∈ ∪ F. (∑ x ∈ ?S a. w (snd x)))
proof (subst sum.reindex)
  show sum (sum (λ x. w (snd x)) ∘ ?S) (∪ F) = (∑ a ∈ ∪ F. sum (λ x. w
(snd x)) (?S a))
  proof (rule sum.cong)
    fix a
    assume a ∈ ∪ F
    have sum (λ x. w (snd x)) (?S a) = sum (λ x. (w a)) (?S a)
      by (rule sum.cong) auto
    thus (sum (λ x. w (snd x)) ∘ ?S) a = (∑ x ∈ {S ∈ F. a ∈ S} × {a}. w (snd
x))
      using assms ⟨a ∈ ∪ F⟩
      by auto
  qed simp
next
show inj-on ?S (∪ F)
  unfolding inj-on-def
  by auto
qed
also have ... = (∑ a ∈ ∪ F. w a * (∑ x ∈ {S ∈ F. a ∈ S}. 1))
proof (rule sum.cong)
  fix a
  assume a ∈ ∪ F
  have sum (λ x. w (snd x)) (?S a) = sum (λ x. (w a)) (?S a)
    by (rule sum.cong) auto
  thus (∑ x ∈ {S ∈ F. a ∈ S} × {a}. w (snd x)) = w a * (∑ x ∈ {S ∈ F. a ∈
S}. 1)
    unfolding count-def
    using assms ⟨a ∈ ∪ F⟩
    by (auto simp add: finiteUn-iff)
  qed simp
finally show ?thesis
  .
qed
also have ... = (∑ a ∈ ∪ F. (w a) * (count a F))
  unfolding count-def
  by simp
finally
show ?thesis
  .
qed

```

lemma *sum-card-reorder*:
assumes *finite* $(\bigcup F)$
shows $(\sum A \in F. \text{card } A) = (\sum a \in \bigcup F. \text{count } a \ F)$
using *assms family-weight-lemma*[*of* $F \ \lambda \ -. \ 1$]
by (*simp add: family-weight-def set-weight-def*)

7.2 Shares

definition *set-share* :: $'a \text{ set} \Rightarrow ('a \Rightarrow \text{nat}) \Rightarrow 'a \text{ set} \Rightarrow \text{int}$ **where**
 $\text{set-share } S \ w \ X \equiv 2 * \text{int } (w \triangleright_s S) - \text{int } (w \triangleright_s X)$

syntax

$\text{-set-share} :: 'a \text{ set} \Rightarrow ('a \Rightarrow \text{nat}) \Rightarrow 'a \text{ set} \Rightarrow \text{int} \quad (- \bowtie_s -)$

translations

$w \bowtie_s S \ X == \text{CONST set-share } S \ w \ X$

definition *family-share* :: $'a \text{ set set} \Rightarrow ('a \Rightarrow \text{nat}) \Rightarrow 'a \text{ set} \Rightarrow \text{int}$ **where**
 $\text{family-share } F \ w \ X \equiv (\sum S \in F. (w \bowtie_s S \ X))$

syntax

$\text{-family-share} :: 'a \text{ set} \Rightarrow ('a \Rightarrow \text{nat}) \Rightarrow 'a \text{ set} \Rightarrow \text{int} \quad (- \bowtie_f -)$

translations

$w \bowtie_f F \ X == \text{CONST family-share } F \ w \ X$

lemma *family-share-cong*:

assumes $\forall v \in X. w \ v = w' \ v \ \bigcup F \subseteq X$
shows $(w \bowtie_f F \ X) = (w' \bowtie_f F \ X)$

proof –

have $w \triangleright_s X = w' \triangleright_s X$
using *set-weight-cong*[*of* $X \ w \ w'$]
using *assms*
by *auto*

moreover

{
fix S
assume $S \in F$
hence $w \triangleright_s S = w' \triangleright_s S$
using *set-weight-cong*[*of* $S \ w \ w'$]
using *assms*
by *auto*

}

ultimately

show *?thesis*

unfolding *family-share-def set-share-def*

by *simp*

qed

lemma *family-share-Pow*:

assumes *finite* $(\bigcup F)$

shows $(w \bowtie_f (\text{Pow } (\bigcup F)) (\bigcup F)) = 0$

proof–
let $?X = \bigcup F$
let $?P = \text{Pow } ?X$
have $\forall S \in ?P. (w \bowtie_s S ?X) + (w \bowtie_s (?X - S) ?X) = 0$
proof
fix S
assume $S \in ?P$
hence *finite* S *finite* $(?X - S) S \cup \bigcup F = \bigcup F$
using *assms*
by (*auto simp add: finite-subset*)
thus $(w \bowtie_s S ?X) + (w \bowtie_s (?X - S) ?X) = 0$
unfolding *set-share-def set-weight-def*
using *sum.union-disjoint[of S ?X - S w]*
by (*auto simp add: field-simps*)
qed
moreover
have $(\sum S \in ?P. (w \bowtie_s S ?X)) + (\sum S \in ?P. (w \bowtie_s (?X - S) ?X)) =$
 $(\sum S \in ?P. (w \bowtie_s S ?X) + (w \bowtie_s (?X - S) ?X))$
using *finite* $(\bigcup F)$
by (*auto simp add: sum.distrib*)
ultimately
have $(\sum S \in ?P. (w \bowtie_s S ?X)) + (\sum S \in ?P. (w \bowtie_s (?X - S) ?X)) = 0$
by *auto*
moreover
have $(\sum S \in ?P. (w \bowtie_s (?X - S) ?X)) = (\sum S \in \text{Pow } ?X. (w \bowtie_s S ?X))$
proof–
have *inj-on* $(\text{op} - (\bigcup F)) ?P$
unfolding *inj-on-def*
by *auto*
moreover
have $\text{op} - (\bigcup F) \text{ ‘ } ?P = ?P$
by *auto*
ultimately
show *?thesis*
using *sum.reindex[of $\lambda S. ?X - S \text{ Pow } ?X \lambda S. (w \bowtie_s S ?X)$, THEN sym]*
by *auto*
qed
hence $(\sum S \in ?P. (w \bowtie_s S ?X)) + (\sum S \in ?P. (w \bowtie_s (\bigcup F - S) ?X)) = 2 *$
 $(\sum S \in ?P. (w \bowtie_s S ?X))$
by *auto*
ultimately
show *?thesis*
unfolding *family-share-def*
by *simp*
qed

lemma *family-share-lemma:*

shows $(w \bowtie_f F X) = \text{int } (2 * w \triangleright_f F) - \text{int } (w \triangleright_s X * (\text{card } F))$
unfolding *family-share-def set-share-def family-weight-def*

unfolding *diff-conv-add-uminus*
by (*subst sum.distrib*, *subst sum-distrib-left*[*THEN sym*]) (*auto simp add: int-sum*)

7.3 Hypercube construction

definition *hypercube* :: '*a set* \Rightarrow '*a set* \Rightarrow '*a set set* **where**
hypercube *K S* $\equiv \{L. (K \subseteq L) \wedge L \subseteq (K \cup S)\}$

syntax

-hypercube :: '*a set* \Rightarrow '*a set* \Rightarrow '*a set set* ($\langle -, - \rangle$)

translations

$\langle K, S \rangle == \text{CONST } \textit{hypercube } K\ S$

lemma *hypercube-Pow*: $\langle K, S \rangle = (op \cup K) \text{ ' } Pow\ S$ (**is** $?H = ?KP$)

proof

show $?H \subseteq ?KP$

proof

fix *M*

assume $M \in \langle K, S \rangle$

then obtain *L* **where** $L \subseteq S$ $M = K \cup L$

unfolding *hypercube-def*

by *auto*

thus $M \in ?KP$

by *auto*

qed

show $?KP \subseteq ?H$

unfolding *hypercube-def*

by *auto*

qed

lemma *hypercube-inter*:

assumes $K1 \neq K2$ **and** $K1 \cap S = \{\}$ **and** $K2 \cap S = \{\}$

shows $\langle K1, S \rangle \cap \langle K2, S \rangle = \{\}$

using *assms*

unfolding *hypercube-def*

by *auto*

lemma *hypercube-UN-Pow*:

shows $\bigcup ((\lambda K. \langle K, S \rangle) \text{ ' } (Pow\ K)) = Pow\ (K \cup S)$ (**is** $?lhs = ?rhs$)

proof

show $?lhs \subseteq ?rhs$

unfolding *hypercube-def*

by *auto*

next

show $?rhs \subseteq ?lhs$

proof

fix *X*

assume $X \in ?rhs$

show $X \in ?lhs$

proof—

```

    let ?Kx = X ∩ K
    let ?Cx = ⟨?Kx, S⟩
    have ?Kx ∈ Pow K
      by auto
    moreover
    have X ∈ ?Cx
      using ⟨X ∈ ?rhs⟩
      unfolding hypercube-def
      by auto
    ultimately
    show ?thesis
      by blast
  qed
qed
qed

```

definition *hyper-share* **where**

hyper-share $K\ S\ F\ w\ X \equiv (\sum L \in \langle K, S \rangle \cap F. (w \bowtie_s L\ X))$

syntax

-hyper-share :: ('a ⇒ nat) ⇒ 'a set ⇒ 'a set ⇒ 'a set set ⇒ 'a set ⇒ int (-
 \odot_f - - - -)

translations

$w \odot_f K\ S\ F\ X == \text{CONST } \textit{hyper-share } K\ S\ F\ w\ X$

lemma *family-share-hyper-share*:

assumes *finite* ($\bigcup F$)

assumes $K' \cup S = (\bigcup F)$ **and** $K' \cap S = \{\}$

shows $(w \bowtie_f F (\bigcup F)) = (\sum K \in \text{Pow } K'. (w \odot_f K\ S\ F (\bigcup F)))$

proof–

let $?w = \lambda S. (w \bowtie_s S (\bigcup F))$

let $?H = \lambda K. \langle K, S \rangle \cap F$

let $?K = \text{Pow } K'$

let $?C = ?H \text{ ‘ } ?K$

have $\bigcup ?C = F$

using *hypercube-UN-Pow[of S K'] assms*

by *auto*

moreover

have $(\sum K \in ?K. (w \odot_f K\ S\ F (\bigcup F))) = (\sum L \in \bigcup ?C. ?w L)$

proof–

have $(\sum L \in \bigcup ?C. ?w L) = \text{sum } (\text{sum } ?w) ?C$

proof (*subst sum.Union-disjoint*)

show $\forall L \in ?C. \text{finite } L$

using $\langle \text{finite } (\bigcup F) \rangle$

by (*simp add: finiteUn-iff*)

next

show $\forall A \in ?C. \forall B \in ?C. A \neq B \longrightarrow A \cap B = \{\}$

proof

fix A **assume** $A \in ?C$

```

then obtain  $Ka$  where *:  $Ka \in ?K \ ?H \ Ka = A$ 
  by auto

show  $\forall B \in ?C. A \neq B \longrightarrow A \cap B = \{\}$ 
proof
  fix  $B$  assume  $B \in ?C$ 
  then obtain  $Kb$  where **:  $Kb \in ?K \ ?H \ Kb = B$ 
    by auto
  show  $A \neq B \longrightarrow A \cap B = \{\}$ 
  proof
    assume  $A \neq B$ 
    hence  $Ka \neq Kb$ 
      using * **
      by auto
    moreover
    have  $Ka \cap S = \{\} \ Kb \cap S = \{\}$ 
      using  $\langle Ka \in ?K \rangle \langle Kb \in ?K \rangle$ 
      using assms
      by auto
    ultimately
    have  $\langle Ka, S \rangle \cap \langle Kb, S \rangle = \{\}$ 
      using hypercube-inter[of  $Ka \ Kb \ S$ ]
      by simp
    thus  $A \cap B = \{\}$ 
      using * **
      by auto
  qed
qed
qed
qed simp
also have  $\dots = (\sum K \in ?K. \text{sum } ?w \ ( ?H \ K))$ 
proof (subst sum.reindex-nontrivial)
  show finite  $?K$ 
    using assms
    using finite-subset[of  $K' \cup F$ ]
    by auto
next
fix  $Kx \ Ky$ 
assume  $Kx \in ?K \ Ky \in ?K \ Kx \neq Ky \ ?H \ Kx = ?H \ Ky$ 
hence  $\langle Kx, S \rangle \cap \langle Ky, S \rangle = \{\}$ 
  using hypercube-inter[of  $Kx \ Ky \ S$ ] assms
  by force
with  $\langle ?H \ Kx = ?H \ Ky \rangle$ 
have  $?H \ Kx = \{\}$ 
  by auto
thus  $\text{sum } ?w \ ( ?H \ Kx) = 0$ 
  by auto
qed simp
finally show ?thesis

```

```

      unfolding hyper-share-def
    by simp
  qed
  ultimately
  show ?thesis
    unfolding family-share-def hyper-share-def
    by simp
  qed
end

```

7.4 Frankl's families characterization using weights

```

theory WeightsShares-Frankl
imports Main WeightsShares
        More.MoreBigOperators
begin

lemma frankl-weight':
  fixes w :: 'a ⇒ nat
  assumes F ≠ {} and finite (⋃ F) and weight-fun w (⋃ F) and
    2 * (w ▷f F) ≥ (w ▷s (⋃ F)) * card F
  shows ∃ x. frankl-element x F ∧ w x ≠ 0
proof (rule ccontr)
  assume ¬ ?thesis
  hence *: ∀ x. x ∈ ⋃ F ∧ w x ≠ 0 ⟶ 2 * count x F < card F
    by force
  obtain a0 where a0 ∈ ⋃ F w a0 > 0 ∀ a ∈ ⋃ F. 0 ≤ w a
    using ⟨weight-fun w (⋃ F)⟩
    unfolding weight-fun-def
    by auto

  have 2 * (w ▷f F) = 2 * (∑ a ∈ ⋃ F. (w a) * (count a F))
    using assms family-weight-lemma
    by simp

  have (∑ a ∈ ⋃ F. 2 * (w a) * (count a F)) < (∑ a ∈ ⋃ F. (w a) * card F)
  proof (rule sum-mono-single-lt-nat)
    show finite (⋃ F) using assms by simp
  next
    fix a
    assume a ∈ ⋃ F
    thus 2 * w a * (count a F) ≤ w a * (card F)
      using *
      by auto
  next
    show a0 ∈ ⋃ F using ⟨a0 ∈ ⋃ F⟩ .
  next
    show 2 * w a0 * (count a0 F) < w a0 * (card F)

```



```

    using * ⟨a0 ∈ ⋃ F⟩ ⟨w a0 > 0⟩
    by auto
qed
also have ... = (w ▷s (⋃ F)) * card F
    unfolding set-weight-def
    by (auto simp add: sum-distrib-right)
finally have 2 * (∑ a ∈ ⋃ F. (w a) * (count a F)) < (w ▷s (⋃ F)) * (card F)
    by (auto simp add: sum-distrib-left mult.assoc)

show False
    using ⟨2 * (∑ a ∈ ⋃ F. (w a) * (count a F)) < (w ▷s (⋃ F)) * (card F)⟩
    using ⟨2 * w ▷f F = 2 * (∑ a ∈ ⋃ F. (w a) * (count a F))⟩
    using ⟨2 * w ▷f F ≥ (w ▷s (⋃ F)) * (card F)⟩
    by simp
qed

lemma frankl-weight'':
  assumes frankl F and F ≠ {} and finite (⋃ F)
  shows ∃ w::('a ⇒ nat). weight-fun w (⋃ F) ∧
    2 * (w ▷f F) ≥ (w ▷s (⋃ F)) * card F
proof-
  from ⟨frankl F⟩ obtain a where a ∈ ⋃ F
    2 * count a F ≥ card F
    unfolding frankl-def
    by auto

  let ?w = λ x. if x = a then (1::nat) else 0
  have card (⋃ F ∩ {a}) = 1
    using ⟨a ∈ ⋃ F⟩
    by auto
  hence ?w ▷s (⋃ F) = 1
    using ⟨a ∈ ⋃ F⟩ ⟨finite (⋃ F)⟩
    unfolding set-weight-def
    by (auto simp add: sum.If-cases)

  have ∀ S ∈ F. ?w ▷s S = (if a ∈ S then 1 else 0)
    using ⟨finite (⋃ F)⟩
    unfolding set-weight-def
    by (auto simp add: sum.If-cases finiteUn-iff)
  hence ?w ▷f F = count a F
    unfolding count-def family-weight-def
    using ⟨finite (⋃ F)⟩
    by (auto simp add: sum.If-cases Collect-conj-eq finiteUn-iff)
  hence 2 * (?w ▷f F) ≥ card(F)
    using ⟨2 * count a F ≥ card F⟩
    by auto
  hence 2 * (?w ▷f F) ≥ (?w ▷s (⋃ F)) * card F
    by (subst ⟨?w ▷s (⋃ F) = 1⟩) simp

```

```

moreover
have weight-fun ?w ( $\bigcup F$ )
  using  $\langle a \in \bigcup F \rangle$ 
  unfolding weight-fun-def
  by auto
ultimately
show ?thesis
  by auto
qed

```

```

theorem frankl-weight:
  assumes  $F \neq \{\}$  and finite ( $\bigcup F$ )
  shows  $\text{frankl } F \longleftrightarrow$ 
     $(\exists w::'a \Rightarrow \text{nat. weight-fun } w (\bigcup F) \wedge$ 
       $2 * (w \triangleright_f F) \geq (w \triangleright_s (\bigcup F)) * \text{card } F) \text{ (is ?lhs } \longleftrightarrow \text{ ?rhs)}$ 

```

```

proof
  assume ?lhs
  thus ?rhs
    using assms frankl-weight''[of F]
    by auto
next
  assume ?rhs
  then obtain w where weight-fun w ( $\bigcup F$ )  $w \triangleright_s (\bigcup F) * \text{card } F \leq 2 * w \triangleright_f$ 
    F
    by auto
  thus ?lhs
    using assms frankl-weight'[of F w]
    unfolding frankl-def
    by auto
qed

```

7.5 Frankl's families characterization using shares

```

lemma frankl-share':
  assumes  $F \neq \{\}$  and finite ( $\bigcup F$ ) weight-fun w ( $\bigcup F$ )  $\wedge (w \bowtie_f F (\bigcup F)) \geq$ 
    0
  shows  $\exists a. \text{frankl-element } a F \wedge w a \neq (0::\text{nat})$ 
  using assms
  apply (subst frankl-weight', simp-all)
  using family-share-lemma[of F w  $\bigcup F$ ]
  by linarith

```

```

lemma frankl-share'':
  assumes  $F \neq \{\}$  and finite ( $\bigcup F$ ) and frankl F
  shows  $\exists w. \text{weight-fun } w (\bigcup F) \wedge (w \bowtie_f F (\bigcup F)) \geq 0$ 
  using assms
  using frankl-weight''[of F]
  using family-share-lemma[of F -  $\bigcup F$ ]
  by (smt of-nat-le-iff)

```

Sufficient condition for Frankl using hypershares.

theorem *frankl-share*:

assumes $F \neq \{\}$ and *finite* $(\bigcup F)$

shows $\text{frankl } F \longleftrightarrow (\exists w. \text{weight-fun } w (\bigcup F) \wedge (w \bowtie_f F (\bigcup F)) \geq 0)$

using *assms frankl-share'[of F] frankl-share''[of F]*

unfolding *frankl-def*

by *force*

theorem *frankl-hyper-share*:

assumes $F \neq \{\}$ and *finite* $(\bigcup F)$

assumes *weight-fun* $w (\bigcup F)$

assumes $K' \cup S = \bigcup F$ and $K' \cap S = \{\}$

assumes $\forall K \in \text{Pow } K'. (w \odot_f K S F (\bigcup F)) \geq 0$

shows $\exists a. \text{frankl-element } a F \wedge w a \neq 0$

using *assms*

apply (*subst frankl-share', simp-all*)

using *family-share-hyper-share[of F K' S] sum-nonneg*

by *fastforce*

end

8 FC families characterization using shares (Poonen's theorem)

theory *WeightsShares-FCFamily*

imports *WeightsShares-Frankl*

begin

8.1 HyperCube projection

abbreviation *hypercube-prj* **where**

hypercube-prj $K S F \equiv (\lambda S. S - K) ' (F \cap \text{op} \cup K ' \text{Pow } S)$

syntax

-hypercube-prj $:: 'a \text{ set} \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set set} \Rightarrow 'a \text{ set set} \quad ((-, -, -))$

translations

$\langle K, S, F \rangle == \text{CONST } \text{hypercube-prj } K S F$

lemma *hypercube-prj-union-closed*:

assumes *union-closed* F

shows *union-closed* $\langle K, S, F \rangle$

unfolding *union-closed-def*

proof ((*rule allI*)+, *rule impI*, *erule conjE*)

fix $A B$

let $?F = \text{hypercube-prj } K S F$

let $?HC = \text{op} \cup K ' \text{Pow } S$

assume $A \in ?F \ B \in ?F$

```

then obtain  $a\ b$  where
  *:  $a \in F\ a \in ?HC\ A = a - K\ A \in Pow\ S$ 
      $b \in F\ b \in ?HC\ B = b - K\ B \in Pow\ S$ 
  by auto

show  $A \cup B \in ?F$ 
proof (rule rev-image-eqI[of  $a \cup b$ ])
  show  $a \cup b \in F \cap ?HC$ 
  proof
    show  $a \cup b \in F$ 
    using  $\langle a \in F \rangle \langle b \in F \rangle$  (union-closed F)
    unfolding union-closed-def
    by auto
  next
    show  $a \cup b \in ?HC$ 
    apply (rule rev-image-eqI[of  $A \cup B$ ])
    using *
    by auto
  qed
next
  show  $A \cup B = a \cup b - K$ 
  using *
  by auto
qed
qed

lemma hypercube-prj-union-closed-additional:
  assumes union-closed F and  $Fc \subseteq F$  and  $S = \bigcup Fc$  and  $K \cap S = \{\}$ 
  shows union-closed-additional  $\langle K, S, F \rangle Fc$ 
unfolding union-closed-additional-def
proof (rule conjI)
  let  $?F = \langle K, S, F \rangle$ 
  show union-closed  $?F$ 
    using assms by (simp add: hypercube-prj-union-closed)

show  $\forall\ A' \in ?F. (op \cup A') ' Fc \subseteq ?F$ 
proof (rule ballI, rule subsetI)
  fix  $A'\ x$ 
  assume  $A' \in ?F\ x \in op \cup A' ' Fc$ 
  then obtain  $y$  where  $x = y \cup A'\ y \in Fc$ 
  by auto
  hence  $x \subseteq S$ 
  using  $\langle S = \bigcup Fc \rangle \langle A' \in ?F \rangle$ 
  by auto
  show  $x \in ?F$ 
proof (rule rev-image-eqI[of  $K \cup x$ ])
  show  $x = K \cup x - K$ 
  using  $\langle K \cap S = \{\} \rangle \langle x \subseteq S \rangle$ 
  by auto

```

```

next
  show  $K \cup x \in F \cap op \cup K \text{ ' Pow } S$ 
  proof
    show  $K \cup x \in op \cup K \text{ ' Pow } S$ 
    using  $\langle x \subseteq S \rangle$ 
    by auto
  next
    have  $y \in F$ 
    using  $\langle y \in Fc \rangle \langle Fc \subseteq F \rangle$ 
    by auto
    moreover
    have  $K \cup A' \in F$ 
    using  $\langle A' \in ?F \rangle$ 
    by auto
    ultimately
    have  $K \cup A' \cup y \in F$ 
    using assms
    unfolding union-closed-def
    by auto
    thus  $K \cup x \in F$ 
    by (subst  $\langle x = y \cup A' \rangle$ ) (subst Un-commute[of y A'], subst Un-assoc[THEN
sym])
  qed
qed
qed
qed

```

```

lemma set-weight-w0:
  assumes finite K and finite S
  assumes  $K \cap S = \{\}$  and  $\forall x \in K. w\ x = 0$  and  $S' \subseteq S$ 
  shows  $w \triangleright_s (K \cup S') = w \triangleright_s S'$ 
  unfolding set-weight-def
  using assms
  using sum.union-disjoint[of K S' w]
  by (auto simp add: finite-subset)

```

```

lemma hypercube-prj-hyper-share-w0:
  assumes finite S and finite K and  $\forall x \in K. w\ x = 0$ 
  shows  $(w \odot_f K\ S\ F\ X) = (w \bowtie_f \langle K, S, F \rangle\ X)$ 
  unfolding hyper-share-def family-share-def
  proof (subst sum.reindex)
    let  $?HCF = \langle K, S \rangle \cap F$ 
    let  $?HCF' = F \cap op \cup K \text{ ' Pow } S$ 
    let  $?mK = \lambda S. S - K$ 
    let  $?s = \lambda S. (w \bowtie_s S\ X)$ 
    show inj-on  $?mK\ ?HCF'$ 
    by (auto simp add: inj-on-def)

```

```

show  $(\sum L \in ?HCF. ?s\ L) = (\sum L \in ?HCF'. (?s \circ ?mK)\ L)$ 

```

```

proof (rule sum.cong)
  show  $?HCF = ?HCF'$ 
    by (auto simp add: hypercube-Pow)
next
  fix  $L$ 
  assume  $L \in ?HCF'$ 
  thus  $?s\ L = (?s \circ ?mK)\ L$ 
  proof (auto simp add: set-share-def)
    fix  $x$ 
    assume  $x \subseteq S$ 
    hence finite  $x$ 
    using  $\langle \text{finite } S \rangle$ 
    by (auto simp add: finite-subset)
    thus  $w \triangleright_s (K \cup x) = w \triangleright_s (K \cup x - K)$ 
    using set-weight-w0[of  $K\ K \cup x - K\ w\ K \cup x - K$ ]
    using  $\langle \text{finite } K \rangle \ \langle \forall x \in K. w\ x = 0 \rangle$ 
    by simp
  qed
qed
qed

```

8.2 All shares non-negative

abbreviation *uce-shares-nonneg* **where**

$$\text{uce-shares-nonneg } Fc\ w \equiv \forall F' \in \mathbb{F}Fc. (w \bowtie_f F' (\bigcup Fc)) \geq 0$$

theorem *frankl-uce-shares-nonneg*:

assumes $F \neq \{\}$ **and** *finite-union-closed* F **and** *weight-fun* $w\ (\bigcup F)$

assumes $Fc \subseteq F$ **and** $\forall x \in (\bigcup F) - (\bigcup Fc). w\ x = 0$

assumes *uce-shares-nonneg* $Fc\ w$

shows $\exists a \in \bigcup Fc. \text{frankl-element } a\ F$

proof—

have $\exists a. \text{frankl-element } a\ F \wedge w\ a \neq 0$

proof (rule *frankl-hyper-share*)

show $\forall K \in \text{Pow } (\bigcup F - \bigcup Fc). (w \odot_f K (\bigcup Fc)\ F (\bigcup F)) \geq 0$

proof

fix K

assume $K \in \text{Pow } (\bigcup F - \bigcup Fc)$

show $0 \leq (w \odot_f K (\bigcup Fc)\ F (\bigcup F))$

proof (*subst hypercube-prj-hyper-share-w0*)

show *finite* $(\bigcup Fc)$

using $\langle \text{finite-union-closed } F \rangle \ \langle Fc \subseteq F \rangle$

using *finite-subset*[of $\bigcup Fc \cup F$]

by *auto*

show *finite* K

using $\langle K \in \text{Pow } (\bigcup F - \bigcup Fc) \rangle \ \langle \text{finite } (\bigcup Fc) \rangle \ \langle \text{finite-union-closed } F \rangle$

by (auto simp add: *finite-subset*)

show $\forall x \in K. w\ x = 0$

using $\langle K \in \text{Pow } (\bigcup F - \bigcup Fc) \rangle$

```

    using  $\langle \forall x \in \bigcup F - \bigcup Fc. w\ x = 0 \rangle$ 
    by blast
  show  $0 \leq (w \bowtie_f \langle K, \bigcup Fc, F \rangle (\bigcup F))$ 
  proof-
    let  $?P = \langle K, \bigcup Fc, F \rangle$ 
    have  $0 \leq (w \bowtie_f ?P (\bigcup Fc))$ 
    proof (subst  $\langle \forall F' \in \{Fc\}. (w \bowtie_f F' (\bigcup Fc)) \geq 0 \rangle$ )
      show  $?P \in \{Fc\}$ 
      proof-
        have  $?P \subseteq Pow (\bigcup Fc)$ 
        by auto
      moreover
        have  $K \cap \bigcup Fc = \{\}$ 
        using  $\langle K \in Pow (\bigcup F - \bigcup Fc) \rangle$ 
        by auto
      hence union-closed-additional  $?P\ Fc$ 
      using hypercube-prj-union-closed-additional [of  $F\ Fc\ \bigcup Fc\ K$ ]
      using  $\langle Fc \subseteq F \rangle \langle K \in Pow (\bigcup F - \bigcup Fc) \rangle \langle \text{finite-union-closed } F \rangle$ 
      by simp
      ultimately
      show ?thesis
      by simp
    qed
  qed simp

  moreover
  have  $w \triangleright_s (\bigcup Fc) = w \triangleright_s (\bigcup F)$ 
  proof-
  have  $\bigcup F - \bigcup Fc \cup \bigcup Fc = \bigcup F$ 
  using  $\langle Fc \subseteq F \rangle$ 
  by auto
  thus ?thesis
  using set-weight-w0 [of  $\bigcup F - \bigcup Fc\ \bigcup Fc\ w\ \bigcup Fc$ ]
  using  $\langle \forall x \in (\bigcup F) - (\bigcup Fc). w\ x = 0 \rangle \langle \text{finite-union-closed } F \rangle \langle \text{finite} \rangle$ 
  by force
  qed
  ultimately
  show ?thesis
  unfolding family-share-def set-share-def
  by simp
  qed
  qed
  qed
  next
  show  $\bigcup F - \bigcup Fc \cup \bigcup Fc = \bigcup F$ 
  using  $\langle Fc \subseteq F \rangle$ 
  by auto
  qed (auto simp add: assms)

```

```

then obtain  $a$  where  $a \in \bigcup F$   $\text{card } F \leq 2 * \text{count } a$   $F$   $w$   $a \neq 0$ 
  by auto
moreover
have  $a \in \bigcup Fc$ 
proof–
  have  $a \in \bigcup Fc \vee a \in (\bigcup F - \bigcup Fc)$ 
    using  $\langle a \in \bigcup F \rangle \langle Fc \subseteq F \rangle$ 
    by auto
  moreover
have  $a \notin \bigcup F - \bigcup Fc$ 
    apply (rule ccontr)
    using  $\langle \forall a \in \bigcup F - \bigcup Fc. w\ a = 0 \rangle \langle w\ a \neq 0 \rangle$ 
    by auto
  ultimately
show ?thesis
    by auto
qed
ultimately
show ?thesis
  by auto
qed

theorem FC-family-uce-shares-nonneg:
  assumes weight-fun  $w$   $(\bigcup Fc)$  and uce-shares-nonneg  $Fc$   $w$ 
  shows FC-family  $Fc$ 
using assms
unfolding FC-family-def
proof (safe)
  fix  $F$ 
  assume uce-shares-nonneg  $Fc$   $w$   $Fc \subseteq F$  union-closed  $F$  finite  $(\bigcup F)$ 
  let  $?w' = \lambda a. \text{if } a \in \bigcup Fc \text{ then } w\ a \text{ else } 0$ 
  have  $\exists a \in \bigcup Fc. \text{frankl-element } a\ F$ 
  proof (rule frankl-uce-shares-nonneg)
    have  $\bigcup Fc \subseteq \bigcup F$ 
      using  $\langle Fc \subseteq F \rangle$ 
      by auto
    show weight-fun  $?w'$   $(\bigcup F)$ 
      using  $\langle \text{weight-fun } w\ (\bigcup Fc) \rangle \langle Fc \subseteq F \rangle$ 
      unfolding weight-fun-def
      by auto
  next
  show  $\forall x \in \bigcup F - \bigcup Fc. ?w'\ x = 0$ 
    by auto
  next
  show uce-shares-nonneg  $Fc$   $?w'$ 
  proof
    fix  $F'$ 
    assume  $F' \in \{\!\{Fc\}\!\}$ 
    hence  $\bigcup F' \subseteq \bigcup Fc$ 

```



```

    by auto
  hence  $(?w' \bowtie_f F' (\bigcup Fc)) = (w \bowtie_f F' (\bigcup Fc))$ 
    using family-share-cong[of  $\bigcup Fc$   $w$   $?w' F'$ ]
    by auto
  thus  $0 \leq (?w' \bowtie_f F' (\bigcup Fc))$ 
    using ⟨uce-shares-nonneg  $Fc$   $w$ ⟩  $\langle F' \in \mathbb{F}Fc \rangle$ 
    by simp
qed
next
  show  $Fc \subseteq F$ 
    by fact
next
  have  $Fc \neq \{\}$ 
    using ⟨weight-fun  $w$   $(\bigcup Fc)$ ⟩
    unfolding weight-fun-def
    by auto
  thus  $F \neq \{\}$ 
    using  $\langle Fc \subseteq F \rangle$ 
    by auto
next
  show finite-union-closed  $F$ 
    using ⟨union-closed  $F$ ⟩  $\langle \text{finite } (\bigcup F) \rangle$ 
    by simp
qed
thus  $\exists a \in \bigcup Fc. 2 * \text{count } a F \geq \text{card } F$ 
  by auto
qed
end

```

9 nonFC families characterization using shares (Poonen's theorem)

```

theory WeightsShares-NotFCFamily
imports Main WeightsShares-FCFamily
begin

```

lemma frankl-fun-hypercube:

```

  assumes  $S = \bigcup F$  and  $K \cap S = \{\}$  and  $a \notin K$ 
  shows  $\text{frankl-fun } a (op \cup K \text{ ` } F) = \text{frankl-fun } a F$ 
proof-
  have inj: inj-on  $(op \cup K) F$ 
    using assms
    by (auto simp add: inj-on-def)
  have count  $a (op \cup K \text{ ` } F) = \text{count } a F$ 
proof-
  have *:  $\{A \in op \cup K \text{ ` } F. a \in A\} = op \cup K \text{ ` } \{A \in F. a \in A\}$ 
    using  $\langle a \notin K \rangle$ 

```

```

    by auto
  show ?thesis
    unfolding count-def
    using  $\langle S = \bigcup F \rangle \langle K \cap S = \{\} \rangle inj$ 
    by (subst *) (rule card-image, simp add: inj inj-on-def)
qed
moreover
have card (op  $\cup$  K ' F) = card F
  by (rule card-image) (simp add: inj)
ultimately
show frankl-fun a (op  $\cup$  K ' F) = frankl-fun a F
  by simp
qed

lemma sum-over-spread:
  fixes d c :: nat and f
  assumes d  $\neq$  0
  shows  $(\sum s \leftarrow [0..<d * c]. f (s \text{ div } d)) = int\ d * (\sum s \leftarrow [0..<c]. f\ s)$ 
proof (induct c)
  case 0
  thus ?case
    by simp
next
  case (Suc c)
  show ?case
  proof-
    have  $[0..<d * Suc\ c] = [0..<d * c] @ [d * c..<d * Suc\ c]$ 
    using upt-add-eq-append[of 0 d * c d * (Suc c) - d * c]
    by (auto simp add: add.commute)
    hence  $(\sum s \leftarrow [0..<d * Suc\ c]. f (s \text{ div } d)) =$ 
       $(\sum s \leftarrow [0..<d * c]. f (s \text{ div } d)) + (\sum s \leftarrow [d * c..<d * Suc\ c]. f (s \text{ div } d))$ 
    by auto
  moreover
  have  $(\sum s \leftarrow [d * c..<d * Suc\ c]. f (s \text{ div } d)) = int\ d * f\ c$ 
  proof-
    have  $\forall s \in set\ [d * c..<d * Suc\ c]. s \text{ div } d = c$ 
    proof
      fix s
      assume  $s \in set\ [d * c..<d * Suc\ c]$ 
      hence  $s \geq d * c$  and  $s < d * Suc\ c$ 
      by auto
      thus  $s \text{ div } d = c$ 
      using split-div-lemma[of d c s]  $\langle d \neq 0 \rangle$ 
      by simp
    qed
    hence  $(\sum s \leftarrow [d * c..<d * Suc\ c]. f (s \text{ div } d)) = (\sum s \leftarrow [d * c..<d * Suc\ c].$ 
       $f\ c)$ 
    by (subst interv-sum-list-conv-sum-set-nat)+ auto

```

```

    thus ?thesis
      using sum-list-triv[of f c [d * c..<d * Suc c]]
      by auto
  qed
ultimately
show ?case
  using Suc
  by (auto simp add: int-distrib(2))
qed
qed

```

lemma archimedean-negative:

```

  fixes x y :: int
  assumes y < 0
  shows  $\exists d. x + \text{int } d * y < 0$ 
proof (cases x  $\leq 0$ )
  case True
  thus ?thesis
    using ⟨y < 0⟩
    by (rule-tac x=1 in exI) auto
next
  case False
  obtain k where k = -y k > 0
  using ⟨y < 0⟩ by auto
  have x * k  $\geq x$ 
  using False ⟨k > 0⟩
  by auto
  hence (x + 1) * k > x
  using ⟨k > 0⟩
  by (auto simp add: int-distrib(1))
  thus ?thesis
    using ⟨k = -y⟩
    by (rule-tac x=nat (x + 1) in exI) auto
qed

```

lemma nonFC:

```

  fixes Fc :: nat set set
  assumes length c = length Fs
  assumes  $\exists cj \in \text{set } c. cj > 0$ 
  assumes finite ( $\bigcup Fc$ )
  assumes union-closed Fc
  assumes  $\forall F \in \text{set } Fs. F \in \{\!\{Fc\}\!\}$ 
  assumes let Fs' = Fs
    in ( $\forall a \in \bigcup Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } c (\text{map }
(\text{frankl-fun } a) Fs')))) < 0$ )
  shows  $\neg \text{FC-family } Fc$ 
proof-
  let ?X =  $\bigcup Fc$ 

```

```

have  $\forall F \in \text{set } Fs. \bigcup F \subseteq ?X$ 
  using  $\langle \forall F \in \text{set } Fs. F \in \{\!\{Fc\}\!\} \rangle$ 
  by auto (metis Pow-iff UnionE set-mp)

have  $\forall F \in \text{set } Fs. \text{finite } F$ 
proof
  fix F
  assume  $F \in \text{set } Fs$ 
  hence  $\text{finite } (\bigcup F)$ 
    using  $\langle \forall F \in \text{set } Fs. \bigcup F \subseteq ?X \rangle \langle \text{finite } ?X \rangle$ 
    by (auto simp add: finite-subset)
  thus  $\text{finite } F$ 
    by (auto simp add: finiteUn-iff)
qed
hence  $\text{finite } (\bigcup (\text{set } Fs))$ 
  by (auto simp add: finiteUn-iff)

have  $\text{finite } Fc$ 
  using  $\langle \text{finite } (\bigcup Fc) \rangle$ 
  by (auto simp add: finiteUn-iff)

let ?sum-c = sum-list c

have ?sum-c > 0
proof-
  obtain wj where  $wj \in \text{set } c \wedge wj > 0$ 
    using  $\langle \exists wj \in \text{set } c. wj > 0 \rangle$ 
    by auto
  then obtain n where  $n < \text{length } c \wedge n = wj$ 
    by (auto simp add: in-set-conv-nth)
  thus ?thesis
    using sum-mono-single-lt-nat[of  $\{0..<\text{length } c\} \lambda -. 0 \text{ op } ! c \ n$ ]  $\langle wj > 0 \rangle$ 
    by (subst sum-list-sum-nth) auto
qed

have div:  $\forall d \ s. d \neq 0 \wedge s < d * ?sum-c \longrightarrow s \text{ div } d < ?sum-c$ 
proof (safe)
  fix d s
  assume  $d > 0 \wedge s < d * ?sum-c$ 
  have  $s \text{ div } d \leq ?sum-c$ 
    using  $\langle s < d * ?sum-c \rangle \langle d > 0 \rangle \text{div-le-mono}[of \ s \ d * ?sum-c \ d]$ 
    by auto
  thus  $s \text{ div } d < ?sum-c$ 
    using  $\langle s < d * ?sum-c \rangle \langle ?sum-c > 0 \rangle$ 
    by (metis Divides.div-mult2-eq div-eq-0-iff neq0-conv)

```

qed

let $?Gs = \text{spread } c \ Fs$
 have $\text{length } ?Gs = ?sum-c$
 using $\langle \text{length } c = \text{length } Fs \rangle$
 by (rule length-spread)

have $\forall i. i < ?sum-c \longrightarrow ?Gs ! i \in \text{set } Fs$

proof (safe)

fix i
 assume $i < ?sum-c$
 hence $?Gs ! i \in \text{set } ?Gs$
 using $\langle \text{length } ?Gs = ?sum-c \rangle \text{ div}$
 by auto
 thus $?Gs ! i \in \text{set } Fs$
 using $\text{set-spread}[of\ c\ Fs]$
 by auto

qed

hence $\forall d\ s. d \neq 0 \wedge s < d * ?sum-c \longrightarrow ?Gs ! (s \text{ div } d) \in \text{set } Fs$

using div

by auto

have $\forall i\ A\ B. i < ?sum-c \wedge A \in Fc \wedge B \in ?Gs ! i \longrightarrow A \cup B \in ?Gs ! i$

proof (safe)

fix $A\ B\ i$
 assume $A \in Fc\ i < ?sum-c\ B \in ?Gs ! i$
 then obtain j where $j < \text{length } Fs\ ?Gs ! i = Fs ! j$
 by (metis $\langle \forall i < \text{sum-list } c. \text{spread } c\ Fs ! i \in \text{set } Fs \rangle \text{ in-set-conv-nth}$)
 hence $?Gs ! i \in \{\!\{Fc\}\!\}$
 using $\langle \forall F \in \text{set } Fs. F \in \{\!\{Fc\}\!\} \rangle$
 by auto
 thus $A \cup B \in ?Gs ! i$
 using $\langle A \in Fc \rangle \langle B \in ?Gs ! i \rangle$
 unfolding $\text{union-closed-extensions-def}$
 by force

qed

hence $\forall d\ s\ A\ B. d \neq 0 \wedge s < d * ?sum-c \wedge A \in Fc \wedge B \in ?Gs ! (s \text{ div } d) \longrightarrow A \cup B \in ?Gs ! (s \text{ div } d)$

using div

by auto

have $\exists Bd. \forall d. \text{finite } (Bd\ d) \wedge \text{card } (Bd\ d) = d * ?sum-c + 1 \wedge (Bd\ d) \cap ?X = \{\}$

proof –

let $?Bd = \lambda d. (\text{SOME } B. (\text{finite } B \wedge \text{card } B = d * ?sum-c + 1 \wedge B \cap ?X =$

```

{}))
  show ?thesis
  proof (rule-tac x=?Bd in exI, rule allI, rule someI-ex)
    fix d
    show  $\exists B. \text{finite } B \wedge \text{card } B = d * ?\text{sum-c} + 1 \wedge B \cap ?X = \{\}$ 
      using finite-disjoint-set[of ?X d * ?sum-c + 1] (finite ?X)
      by auto
    qed
  qed
  then obtain Bd where Bd:  $\forall d. \text{finite } (Bd \ d) \wedge \text{card } (Bd \ d) = d * ?\text{sum-c} + 1$ 
     $\wedge (Bd \ d) \cap ?X = \{\}$ 
    by auto

```

```

let ?Bds =  $\lambda d \ s. \text{set-drop-nth } s \ (Bd \ d)$ 

```

```

have  $\forall d \ s. d > 0 \wedge s < d * ?\text{sum-c} \longrightarrow ?Bds \ d \ s \neq \{\}$ 
  proof ((rule allI)+, rule impI, erule conjE)
    fix d s
    assume  $d > 0 \wedge s < d * ?\text{sum-c}$ 
    hence  $0 < \text{card } (\text{set-drop-nth } s \ (Bd \ d))$ 
      using Bd (sum-c > 0)
    by (subst card-set-drop-nth) simp-all
    thus  $?Bds \ d \ s \neq \{\}$ 
      by auto
  qed

```

```

let ?Hds =  $\lambda d \ s. (\text{op} \cup (?Bds \ d \ s)) \ ' \ (?Gs \ ! \ (s \ \text{div} \ d))$ 
let ?Hdl =  $\lambda d. (\text{map } (?Hds \ d) \ [0..<d * ?\text{sum-c}])$ 
let ?Hd =  $\lambda d. (\text{op} \cup (Bd \ d)) \ ' \ \text{Pow } ?X$ 
let ?Fd =  $\lambda d. Fc \cup \bigcup \text{set } (?Hdl \ d) \cup (?Hd \ d)$ 

```

```

have  $\forall s \ d. s < d * ?\text{sum-c} \longrightarrow (?Hdl \ d) \ ! \ s = ?Hds \ d \ s$ 
  by simp
have  $\forall d \ s \ a. d \neq 0 \wedge a \in ?X \wedge s < d * ?\text{sum-c} \longrightarrow$ 
   $\text{frankl-fun } a \ (?Hds \ d \ s) = \text{frankl-fun } a \ (?Gs \ ! \ (s \ \text{div} \ d))$ 
  proof (rule+, (erule conjE)+, rule frankl-fun-hypercube)
    fix a s d
    assume  $a \in ?X$ 
    thus  $a \notin \text{set-drop-nth } s \ (Bd \ d)$ 
      using set-drop-nth-subset[of Bd d s] Bd
      by auto
  next
    fix d s a
    assume  $a \in ?X \wedge s < d * ?\text{sum-c} \wedge d \neq 0$ 
    hence  $?Gs \ ! \ (s \ \text{div} \ d) \in \text{set } Fs$ 

```

using $\langle \forall d s. d \neq 0 \wedge s < d * ?sum-c \longrightarrow ?Gs ! (s \text{ div } d) \in \text{set } Fs \rangle$
by *auto*
thus $\langle ?Bds d s \rangle \cap \left(\bigcup \langle ?Gs ! (s \text{ div } d) \rangle \right) = \{\}$
using $\langle \forall F \in \text{set } Fs. \bigcup F \subseteq ?X \rangle$
using *set-drop-nth-subset[of Bd d s] Bd*
by *blast*
qed *simp*

have $\forall d. d > 0 \longrightarrow Fc \cap \bigcup \text{set } \langle ?Hdl d \rangle = \{\}$
proof (*rule, rule, rule ccontr*)
fix *d*
assume $d > 0 \neg Fc \cap \bigcup \text{set } \langle ?Hdl d \rangle = \{\}$
then obtain $S S'$ **where** $S \in Fc S' \in \text{set } \langle ?Hdl d \rangle S \in S'$
by *auto*
then obtain $s A$ **where** $s < d * ?sum-c A \in ?Gs ! (s \text{ div } d) S = A \cup \langle ?Bds d$
 $s \rangle$
by *force*
obtain x **where** $x \in \langle ?Bds d s \rangle$
using $\langle s < d * ?sum-c \rangle \langle \forall d s. d > 0 \wedge s < d * ?sum-c \longrightarrow ?Bds d s \neq \{\} \rangle \langle d$
 $> 0 \rangle$
by *auto*
hence $x \in \bigcup Fc$
using $\langle S = A \cup \langle ?Bds d s \rangle \rangle \langle S \in Fc \rangle$
by *blast*
thus *False*
using $\langle x \in \langle ?Bds d s \rangle \rangle \text{set-drop-nth-subset[of Bd d s] Bd}$
by *auto*
qed

have $\forall d. d > 0 \longrightarrow Fc \cap \langle ?Hd d \rangle = \{\}$
proof (*rule, rule, rule ccontr*)
fix *d*
assume $d > 0 \neg Fc \cap \langle ?Hd d \rangle = \{\}$
then obtain S **where** $S \in Fc S \in \text{op} \cup (Bd d) \text{ 'Pow } (\bigcup Fc)$
by *auto*
then obtain A **where** $A \subseteq \bigcup Fc A \cup (Bd d) = S$
by *auto*
obtain x **where** $x \in (Bd d)$
using *Bd card-gt-0-iff[of Bd d]*
by *auto*
hence $x \in \bigcup Fc$
using $\langle A \cup (Bd d) = S \rangle \langle S \in Fc \rangle$
by *blast*
thus *False*
using $\langle x \in Bd d \rangle Bd$
by *auto*
qed

have $\forall d. d > 0 \longrightarrow \bigcup \text{set } (?Hdl\ d) \cap (?Hd\ d) = \{\}$
proof (*rule*, *rule*, *rule ccontr*)
 fix d
 assume $d > 0 \neg \bigcup \text{set } (?Hdl\ d) \cap (?Hd\ d) = \{\}$
 then obtain $S\ S'$ **where** $S \in (?Hd\ d)\ S' \in \text{set } (?Hdl\ d)\ S \in S'$
 by *auto*

 obtain A **where** $A \subseteq \bigcup Fc\ A \cup (Bd\ d) = S$
 using $\langle S \in (?Hd\ d) \rangle$
 by *auto*

 obtain $s\ B$ **where** $s < d * ?sum-c\ B \in ?Gs ! (s\ div\ d)\ S = B \cup (?Bds\ d\ s)$
 using $\langle S' \in \text{set } (?Hdl\ d) \rangle\ \langle S \in S' \rangle$
 by *force*

 obtain a **where** $a \in Bd\ d\ a \notin (?Bds\ d\ s)$
 using $\langle s < d * ?sum-c \rangle\ \text{set-drop-nth-drops}[of\ s\ Bd\ d]\ Bd$
 by *auto*
 moreover
 have $B \in \bigcup (\text{set } Fs)$
 using $\langle B \in ?Gs ! (s\ div\ d) \rangle\ \langle \forall d\ s. d \neq 0 \wedge s < d * ?sum-c \longrightarrow ?Gs ! (s\ div\ d) \in \text{set } Fs \rangle\ \langle d > 0 \rangle\ \langle s < d * ?sum-c \rangle$
 by *auto*
 hence $a \notin B$
 using $\langle \forall F \in \text{set } Fs. \bigcup F \subseteq ?X \rangle\ \langle a \in Bd\ d \rangle\ Bd$
 by *blast*
 ultimately
 have $a \in A \cup (Bd\ d)\ a \notin B \cup (?Bds\ d\ s)$
 by *auto*
 thus *False*
 using $\langle A \cup (Bd\ d) = S \rangle\ \langle S = B \cup (?Bds\ d\ s) \rangle$
 by *auto*
qed

have $\forall d. d > 0 \longrightarrow \text{finite } (\bigcup \text{set } (?Hdl\ d))$
 using $\langle \forall d\ s. d \neq 0 \wedge s < d * ?sum-c \longrightarrow ?Gs ! (s\ div\ d) \in \text{set } Fs \rangle$
 $\langle \forall F \in \text{set } Fs. \text{finite } F \rangle$
 by *auto*

have $\forall d. d > 0 \longrightarrow \text{finite } (?Hd\ d)$
 using $\langle \text{finite } ?X \rangle$
 by *auto*

let $?Si = \lambda a. \text{sum-list } (\text{map } (\lambda s. \text{frankl-fun } a\ (?Gs !\ s))\ [0..<?sum-c])$


```

have  $\forall d a. d > 0 \wedge a \in ?X \longrightarrow \text{frankl-fun } a \text{ } (?Fd d) = \text{frankl-fun } a \text{ } Fc + \text{int}$ 
 $d * ?Si a$ 
proof –

have  $\forall d a. d > 0 \longrightarrow$ 
 $\text{frankl-fun } a \text{ } (?Fd d) =$ 
 $\text{frankl-fun } a \text{ } Fc + \text{frankl-fun } a \text{ } (\bigcup \text{ set } (?Hdl d)) + \text{frankl-fun } a \text{ } (?Hd d)$ 
proof ((rule allI)+, rule impI)
  fix  $d i$ 
  assume  $d > (0::nat)$ 
  show  $\text{frankl-fun } i \text{ } (?Fd d) =$ 
 $\text{frankl-fun } i \text{ } Fc + \text{frankl-fun } i \text{ } (\bigcup \text{ set } (?Hdl d)) + \text{frankl-fun } i \text{ } (?Hd d)$ 
  proof (rule frankl-fun-Un-disjoint-3)
    show  $\text{finite } Fc$  by fact
  next
    show  $\text{finite } (\bigcup \text{ set } (?Hdl d))$ 
    using  $\langle d > 0 \rangle \langle \forall d. d > 0 \longrightarrow \text{finite } (\bigcup \text{ set } (?Hdl d)) \rangle$ 
    by simp
  next
    show  $\text{finite } (?Hd d)$ 
    using  $\langle d > 0 \rangle \langle \forall d. d > 0 \longrightarrow \text{finite } (?Hd d) \rangle$ 
    by simp
  next
    show  $Fc \cap (\bigcup \text{ set } (?Hdl d)) = \{\}$ 
    using  $\langle d > 0 \rangle \langle \forall d. d > 0 \longrightarrow Fc \cap \bigcup \text{ set } (?Hdl d) = \{\} \rangle$ 
    by simp
  next
    show  $Fc \cap (?Hd d) = \{\}$ 
    using  $\langle d > 0 \rangle \langle \forall d. d > 0 \longrightarrow Fc \cap (?Hd d) = \{\} \rangle$ 
    by simp
  next
    show  $(\bigcup \text{ set } (?Hdl d)) \cap (?Hd d) = \{\}$ 
    using  $\langle d > 0 \rangle \langle \forall d. d > 0 \longrightarrow (\bigcup \text{ set } (?Hdl d)) \cap (?Hd d) = \{\} \rangle$ 
    by simp
  qed
qed
moreover

  have  $\forall d. \forall a \in ?X. \text{frankl-fun } a \text{ } (?Hd d) = \text{frankl-fun } a \text{ } (\text{Pow } ?X)$ 
  apply (rule+, rule frankl-fun-hypercube)
  using  $Bd$ 
  by auto
  hence  $\forall d. \forall a \in ?X. \text{frankl-fun } a \text{ } (?Hd d) = 0$ 
  using  $\text{frankl-fun-Pow}[of ?X] \langle \text{finite } (\bigcup Fc) \rangle$ 
  by simp

moreover

```

```

    have  $\forall d a. d > 0 \longrightarrow \text{frankl-fun } a \ (\bigcup \text{ set } (?Hdl \ d)) = \text{sum-list } (\text{map}$ 
    ( $\text{frankl-fun } a$ )  $(?Hdl \ d))$ 
  proof (rule+, rule frankl-fun-UN-disjoint)
    fix d a
    assume  $d > (0::nat)$ 
    thus finite  $(\bigcup \text{ set } (?Hdl \ d))$ 
  proof (auto intro!: finite-imageI)
    fix s
    assume  $s < d * ?sum-c$ 
    hence  $?Gs \ ! \ (s \text{ div } d) \in \text{set } Fs$ 
    using  $\langle \forall d s. d \neq 0 \wedge s < d * ?sum-c \longrightarrow ?Gs \ ! \ (s \text{ div } d) \in \text{set } Fs \rangle \langle d$ 
     $> 0 \rangle$ 
    by auto
    hence  $\bigcup (?Gs \ ! \ (s \text{ div } d)) \subseteq ?X$ 
    using  $\langle \forall F \in \text{set } Fs. \bigcup F \subseteq ?X \rangle$ 
    by auto
    hence finite  $(\bigcup (?Gs \ ! \ (s \text{ div } d)))$ 
    using  $\langle \text{finite } ?X \rangle$ 
    by (auto simp add: finite-subset)
    thus finite  $(?Gs \ ! \ (s \text{ div } d))$ 
    by (auto simp add: finiteUn-iff)
  qed
next
fix d a
assume  $d > (0::nat)$ 
thus  $\forall i j. i < \text{length } (?Hdl \ d) \wedge j < \text{length } (?Hdl \ d) \wedge i \neq j \longrightarrow$ 
 $?Hdl \ d \ ! \ i \cap ?Hdl \ d \ ! \ j = \{\}$ 
proof auto
  fix i j a b
  assume  $i < d * ?sum-c \wedge j < d * ?sum-c \wedge i \neq j$ 
   $a \in ?Gs \ ! \ (i \text{ div } d) \wedge b \in ?Gs \ ! \ (j \text{ div } d)$ 
   $?Bds \ d \ i \cup a = ?Bds \ d \ j \cup b$ 
  have  $Bd \ d \cap ?X = \{\}$  finite  $(Bd \ d)$   $\text{card } (Bd \ d) = d * ?sum-c + 1$ 
  using  $Bd \ \langle d > 0 \rangle$ 
  by auto
  moreover
  have  $a \in \bigcup \text{ set } Fs \wedge b \in \bigcup \text{ set } Fs$ 
  using  $\langle a \in ?Gs \ ! \ (i \text{ div } d) \rangle \langle b \in ?Gs \ ! \ (j \text{ div } d) \rangle$ 
  using  $\langle i < d * ?sum-c \rangle \langle j < d * ?sum-c \rangle \langle d > 0 \rangle$ 
   $\langle \forall d s. d \neq 0 \wedge s < d * ?sum-c \longrightarrow ?Gs \ ! \ (s \text{ div } d) \in \text{set } Fs \rangle$ 
  by auto
  hence  $a \subseteq ?X \wedge b \subseteq ?X$ 
  using  $\langle \forall F \in \text{set } Fs. \bigcup F \subseteq ?X \rangle$ 
  by auto
  ultimately
  have  $a \cap ?Bds \ d \ i = \{\} \wedge a \cap ?Bds \ d \ j = \{\}$ 
   $b \cap ?Bds \ d \ i = \{\} \wedge b \cap ?Bds \ d \ j = \{\}$ 
  using  $\text{set-drop-nth-subset}[OF \ \langle \text{finite } (Bd \ d) \rangle]$ 
  by blast+

```

```

hence ?Bds d i = ?Bds d j
  using ⟨?Bds d i ∪ a = ?Bds d j ∪ b⟩
  using Un-iff ⟨Bd d ∩ ⋃ Fc = { }⟩ ⟨b ⊆ ⋃ Fc⟩ ⟨finite (Bd d)⟩
  using inf-sup-absorb set-drop-nth-subset
  by fastforce
thus False
  using ⟨i ≠ j⟩ set-drop-nth-inj[OF ⟨finite (Bd d)⟩]
  using ⟨card (Bd d) = d * ?sum-c + 1⟩ ⟨i < d * ?sum-c⟩ ⟨j < d * ?sum-c⟩
  by auto
qed
qed
moreover
have ∀ d a. d ≠ 0 ∧ a ∈ ?X ⟶
  sum-list (map (frankl-fun a) (?Hdl d)) =
  sum-list (map (λ s. frankl-fun a (?Gs ! (s div d))) [0..

```

hence $?Si\ a =$
 $\quad \text{sum-list } (\text{map } (\lambda(x, y). \text{int } x * y) (\text{zip } c (\text{map } (\text{frankl-fun } a) Fs)))$
using $\text{listsum-spread } \langle \text{length } c = \text{length } Fs \rangle$
by *auto*
thus $?Si\ a < 0$
using $\text{assms}(6)$
using $\langle \text{let } Fs' = Fs \text{ in } (\forall a \in \bigcup Fc. (\sum (x, y) \leftarrow \text{zip } c (\text{map } (\text{frankl-fun } a) Fs')). \text{int } x * y) < 0) \rangle \langle a \in ?X \rangle$
by *auto*
qed

hence $\forall a \in ?X. \exists da. \text{frankl-fun } a\ Fc + \text{int } da * ?Si\ a < 0$
using *archimedean-negative*
by *auto*

let $?da = \lambda a. \text{SOME } da. \text{frankl-fun } a\ Fc + \text{int } da * ?Si\ a < 0$
let $?d = \text{Max } (?da \text{ ' } ?X) + 1$

have $\forall a \in ?X. ?d \geq ?da\ a$
proof–
have $\bigwedge x\ y. x \leq y \implies x \leq y + (1::\text{nat})$
by *auto*
moreover
have $\forall a \in ?X. \text{Max } (?da \text{ ' } ?X) \geq ?da\ a$
using $\langle \text{finite } ?X \rangle$
by *auto*
ultimately
show *?thesis*
by *blast*
qed

have $\forall a \in ?X. \text{frankl-fun } a\ Fc + \text{int } (?da\ a) * ?Si\ a < 0$
apply $(\text{rule}, \text{rule someI-ex})$
using $\langle \forall a \in ?X. \exists da. \text{frankl-fun } a\ Fc + \text{int } da * ?Si\ a < 0 \rangle$
by *auto*

have $\forall a \in ?X. \text{frankl-fun } a\ Fc + \text{int } ?d * ?Si\ a < 0$
proof
fix a
assume $a \in ?X$
hence $** : ?d \geq ?da\ a \text{ frankl-fun } a\ Fc + \text{int } (?da\ a) * ?Si\ a < 0 \text{ ?Si } a < 0$
using $\langle \forall a \in ?X. ?d \geq ?da\ a \rangle \langle \forall a \in ?X. \text{frankl-fun } a\ Fc + \text{int } (?da\ a) * ?Si\ a < 0 \rangle \langle \forall a \in ?X. ?Si\ a < 0 \rangle$
by *auto*

show $\text{frankl-fun } a\ Fc + \text{int } ?d * ?Si\ a < 0$
proof–
have $** : \bigwedge x\ y\ y'\ z. \llbracket x + \text{int } y * z < 0; y \leq y'; z < 0 \rrbracket \implies x + \text{int } y' * z < 0$
 $(0::\text{int})$

```

proof–
  fix  $x\ y\ y'\ z$ 
  assume  $x + \text{int } y * z < 0 \ \ y \leq y' \ z < 0$ 
  have  $x + \text{int } y * z \geq x + \text{int } y' * z$ 
    using  $\langle y \leq y' \rangle \langle z < 0 \rangle$ 
    by auto
  thus  $x + \text{int } y' * z < 0$ 
    using  $\langle x + \text{int } y * z < 0 \rangle$ 
    by simp
qed
show ?thesis
  by (rule *[where  $y=?da\ a]$ ) fact+
qed
qed

moreover

have  $?d > 0$ 
  by auto

ultimately

show ?thesis
  by (rule-tac  $x=?d$  in exI) auto
qed
then obtain  $d$  where  $d > 0 \ \forall \ a \in ?X. \text{frankl-fun } a \ Fc + \text{int } d * ?Si\ a < 0$ 
  by auto

ultimately

have  $\forall \ a \in ?X. \text{frankl-fun } a \ (\ ?Fd\ d) < 0$ 
  by auto

have  $\exists \ F. Fc \subseteq F \wedge$ 
  union-closed  $F \wedge$ 
  finite  $(\bigcup F) \wedge (\forall a \in ?X. \text{card } F > 2 * \text{count } a\ F)$ 
proof (rule-tac  $x=?Fd\ d$  in exI, intro conjI)
  show  $\forall \ a \in ?X. 2 * \text{count } a \ (\ ?Fd\ d) < \text{card } (\ ?Fd\ d)$ 
    using  $\langle \forall \ a \in ?X. \text{frankl-fun } a \ (\ ?Fd\ d) < 0 \rangle$ 
    by force
  moreover
  show  $Fc \subseteq (\ ?Fd\ d)$ 
    by auto
  moreover
  show union-closed  $(\ ?Fd\ d)$ 
  proof–
    obtain  $Hdl\ Hd\ Fd$  where  $*$ :  $Hdl = \bigcup \text{set } (\ ?Hdl\ d)\ Hd = ?Hd\ d\ Fd = Fc$ 
     $\cup\ Hdl \cup Hd$ 
    by auto

```

```

have  $\forall A \in Fc. \forall B \in Fc. A \cup B \in Fd$ 
  using  $\langle \text{union-closed } Fc \rangle *$ 
  unfolding  $\text{union-closed-def}$ 
  by auto
moreover
have  $\forall A \in Fc. \forall B \in Hdl. A \cup B \in Fd$ 
proof (safe)
  fix  $A B$ 
  assume  $A \in Fc \ B \in Hdl$ 
  then obtain  $i \ a$  where  $i < d * \text{sum-list } c$ 
     $a \in ?Gs ! (i \text{ div } d) \ B = ?Bds \ d \ i \cup a$ 
    using *
    by auto

  hence  $A \cup a \in ?Gs ! (i \text{ div } d)$ 
    using  $\langle A \in Fc \rangle \langle \forall d \ s \ A \ B. d \neq 0 \wedge s < d * \text{sum-list } c \wedge A \in Fc \wedge B \in$ 
 $\text{spread } c \ Fs ! (s \text{ div } d) \longrightarrow A \cup B \in \text{spread } c \ Fs ! (s \text{ div } d) \rangle \langle d > 0 \rangle$ 
    by simp

  hence  $(A \cup a) \cup ?Bds \ d \ i \in \bigcup \text{set } (?Hdl \ d)$ 
    using  $\langle i < d * \text{sum-list } c \rangle$ 
    by auto (rule-tac  $x=i$  in  $\text{bexI}$ , auto)
moreover
have  $A \cup B = (A \cup a) \cup ?Bds \ d \ i$ 
  using  $\langle B = ?Bds \ d \ i \cup a \rangle$ 
  by auto
ultimately
show  $A \cup B \in Fd$ 
  using *
  by simp
qed
moreover
have  $\forall A \in Fc. \forall B \in Hd. A \cup B \in Fd$ 
proof safe
  fix  $A B$ 
  assume  $A \in Fc \ B \in Hd$ 
  then obtain  $b$  where  $b \subseteq \bigcup Fc \ B = b \cup (Bd \ d)$ 
    using *
    by auto
  hence  $A \cup b \subseteq \bigcup Fc$ 
    using  $\langle A \in Fc \rangle$ 
    by auto
  hence  $A \cup B \in ?Hd \ d$ 
    using  $\langle B = b \cup (Bd \ d) \rangle$ 
    by (smt  $\text{PowI Un-assoc Un-commute rev-image-eqI}$ )
  thus  $A \cup B \in Fd$ 
    using *
    by simp
qed

```

```

moreover
have  $\forall A \in \text{Hdl}. \forall B \in \text{Hdl}. A \cup B \in \text{Fd}$ 
proof safe
  fix  $A B$ 
  assume  $A \in \text{Hdl} B \in \text{Hdl}$ 
  then obtain  $i j a b$  where
     $i < d * \text{sum-list } c \ j < d * \text{sum-list } c$ 
     $a \in ?Gs ! (i \text{ div } d) \ A = ?Bds \ d \ i \cup a$ 
     $b \in ?Gs ! (j \text{ div } d) \ B = ?Bds \ d \ j \cup b$ 
    using *
    by auto
  hence  $a \in \bigcup \text{set } Fs \ b \in \bigcup \text{set } Fs$ 
    using  $\langle 0 < d \rangle \ \langle \forall d \ s. \ d \neq 0 \wedge s < d * \text{sum-list } c \longrightarrow \text{spread } c \ Fs ! (s \text{ div } d) \in \text{set } Fs \rangle$ 
    by auto
  hence  $a \cup b \subseteq ?X$ 
    using  $\langle \forall F \in \text{set } Fs. \bigcup F \subseteq \bigcup Fc \rangle$ 
    by auto
  show  $A \cup B \in \text{Fd}$ 
  proof (cases  $i = j$ )
    case True
      hence  $a \cup b \in ?Gs ! (i \text{ div } d)$ 
        using  $\langle a \in ?Gs ! (i \text{ div } d) \rangle \ \langle b \in ?Gs ! (j \text{ div } d) \rangle$ 
        using  $\langle \forall F \in \text{set } Fs. F \in \{Fc\} \rangle \text{ div } \langle \text{length } ?Gs = ?\text{sum-}c \rangle$ 
        using  $\langle \forall F \in \text{set } Fs. \bigcup F \subseteq \bigcup Fc \rangle$ 
      unfolding union-closed-extensions-def union-closed-additional-def union-closed-def
        using  $\langle 0 < d \rangle \ \langle \forall i < \text{sum-list } c. \text{spread } c \ Fs ! i \in \text{set } Fs \rangle \ \langle j < d * \text{sum-list } c \rangle$ 
        by simp
      hence  $(a \cup b) \cup ?Bds \ d \ i \in \bigcup \text{set } (?Hdl \ d)$ 
        using  $\langle i < d * ?\text{sum-}c \rangle$ 
        by auto (rule-tac  $x=i$  in beXI, auto)
      hence  $A \cup B \in \bigcup \text{set } (?Hdl \ d)$ 
        using  $\langle A = ?Bds \ d \ i \cup a \rangle \ \langle B = ?Bds \ d \ j \cup b \rangle \ \langle i = j \rangle$ 
        by (smt Un-assoc Un-commute sup.right-idem)
      thus ?thesis
        using *
        by simp
    next
      case False
      hence  $?Bds \ d \ i \cup ?Bds \ d \ j = Bd \ d$ 
        unfolding set-drop-nth-def
        using  $\langle i < d * ?\text{sum-}c \rangle \ \langle j < d * ?\text{sum-}c \rangle$ 
        using distinct-sorted-list-of-set[of  $Bd \ d$ ]
        using set-drop-nth-distinct[of  $i$  sorted-list-of-set ( $Bd \ d$ )]
        using set-drop-nth-distinct[of  $j$  sorted-list-of-set ( $Bd \ d$ )]
        using length-sorted-list-of-set[of  $Bd \ d$ ]  $Bd$ 
        using nth-eq-iff-index-eq[of sorted-list-of-set ( $Bd \ d$ )  $i \ j$ ]

```

```

    by auto
  hence  $A \cup B \in ?Hd\ d$ 
    using  $\langle A = ?Bds\ d\ i \cup a \rangle \langle B = ?Bds\ d\ j \cup b \rangle$ 
    using  $\langle a \cup b \subseteq \bigcup Fc \rangle$ 
    by (smt PowI Un-assoc image-iff sup-left-commute)
  thus  $A \cup B \in Fd$ 
    using *
    by simp
qed
qed
moreover
have  $\forall A \in Hdl. \forall B \in Hd. A \cup B \in Fd$ 
proof safe
  fix  $A\ B$ 
  assume  $A \in Hdl\ B \in Hd$ 
  then obtain  $i\ a\ b$  where
     $i < d * \text{sum-list } c$ 
     $a \in ?Gs\ !\ (i \text{ div } d)\ A = ?Bds\ d\ i \cup a$ 
     $b \subseteq \bigcup Fc\ B = b \cup (Bd\ d)$ 
    using *
    by auto blast
  hence  $a \cup b \subseteq ?X$ 
    using  $\langle 0 < d \rangle \langle \forall d\ s. d \neq 0 \wedge s < d * \text{sum-list } c \longrightarrow \text{spread } c\ Fs\ !\ (s \text{ div } d) \in \text{set } Fs \rangle$ 
    using  $\langle \forall F \in \text{set } Fs. \bigcup F \subseteq \bigcup Fc \rangle \langle b \subseteq \bigcup Fc \rangle$ 
    by (metis (mono-tags, lifting) Union-upper le-supI neq0-conv subset-trans)
  hence  $(a \cup b) \cup Bd\ d \in ?Hd\ d$ 
    by auto
  moreover
  have  $?Bds\ d\ i \cup Bd\ d = Bd\ d$ 
    using set-drop-nth-subset Bd
    by auto
  ultimately
  have  $A \cup B \in ?Hd\ d$ 
    using  $\langle A = ?Bds\ d\ i \cup a \rangle \langle B = b \cup (Bd\ d) \rangle$ 
    by (simp add: Un-left-commute sup-assoc)
  thus  $A \cup B \in Fd$ 
    using *
    by simp
qed
moreover
have  $\forall A \in Hd. \forall B \in Hd. A \cup B \in Fd$ 
proof safe
  fix  $A\ B$ 
  assume  $A \in Hd\ B \in Hd$ 
  then obtain  $a\ b$  where  $a \subseteq \bigcup Fc\ A = a \cup (Bd\ d)\ b \subseteq \bigcup Fc\ B = b \cup$ 
  ( $Bd\ d$ )
    using *
    by auto

```



```

    hence  $a \cup b \subseteq \bigcup Fc$ 
      by auto
    hence  $A \cup B \in ?Hd\ d$ 
      using  $\langle A = a \cup (Bd\ d) \rangle \langle B = b \cup (Bd\ d) \rangle$ 
      by (smt PowI Un-commute image-iff sup.right-idem sup-left-commute)
    thus  $A \cup B \in Fd$ 
      using *
      by simp

  qed
  ultimately
  have union-closed Fd
    using  $\langle Fd = Fc \cup Hdl \cup Hd \rangle$ 
    unfolding union-closed-def
    by (metis UnE Un-commute)+
  thus ?thesis
    using *
    by simp
  qed
next
  show finite  $(\bigcup (?Fd\ d))$ 
  proof (auto)
    show finite  $(\bigcup Fc)$  by fact
  next
    show finite  $(Bd\ d)$  using Bd by simp
  next
  show finite
     $(\bigcup (\bigcup_{x \in \{0..<d * \text{sum-list } c\}} \text{op } \cup (\text{set-drop-nth } x\ (Bd\ d)) \text{ 'spread } c\ Fs\ !\ (x\ \text{div } d)))$ 
    using  $\langle d > 0 \rangle \langle \forall\ d.\ d > 0 \longrightarrow \text{finite } (\bigcup \text{set } (?Hdl\ d)) \rangle$ 
  proof (auto simp add: finiteUn-iff)
    fix a
    show finite  $(\text{set-drop-nth } a\ (Bd\ d))$ 
      using Bd set-drop-nth-subset[of Bd d a]
      by (auto simp add: finite-subset)
  next
    fix a x
    assume  $x \in \text{spread } c\ Fs\ !\ (a\ \text{div } d)\ a < d * ?\text{sum-c}$ 
    hence  $x \in \bigcup (\text{set } Fs)$ 
      using  $\langle \forall\ d\ s.\ d \neq 0 \wedge s < d * ?\text{sum-c} \longrightarrow ?Gs\ !\ (s\ \text{div } d) \in \text{set } Fs \rangle \langle d$ 
       $> 0 \rangle$ 
      by auto
    then obtain F' where  $F' \in \text{set } Fs\ x \in F'$ 
      by auto
    hence finite  $(\bigcup F')$ 
      using  $\langle \forall\ F \in \text{set } Fs.\ \bigcup F \subseteq ?X \rangle$ 
      using  $\langle \text{finite } ?X \rangle$ 
      by (auto simp add: finite-subset)
    thus finite x

```

```

      using ⟨x ∈ F'⟩
      by (auto simp add: finiteUn-iff)
    qed
  next
    show finite (⋃ Fc)
      by fact
    qed
  qed
  thus ?thesis
    unfolding FC-family-def
    by (metis linorder-not-less)
qed
end

```

10 SomeShareNegative — combinatorial search for union-closed extension with a negative share

```

theory SomeShareNegative
imports Main WeightsShares-FCFamily
begin

```

According to the theorem

$$\llbracket \text{weight-fun } w \ (\bigcup Fc); \forall F' \in \llbracket Fc \rrbracket. 0 \leq w \bowtie_f F' \bigcup Fc \rrbracket \implies \text{FC-family } Fc$$

to show that a family is an FC family, it suffices to show that there is no member of union closed extension of a given family with a negative share. The function *some-share-negative* performs a combinatorial enumeration of union closed extensions of a given family and checks whether it contains a family with a negative share with respect to the given weight function. We give a rather abstract (nonexecutable) definition of this function and prove its main properties. This function will be later refined and a more efficient, executable implementation will be given.

```

primrec some-share-negative-aux :: 'a set list  $\Rightarrow$  'a set set  $\Rightarrow$  'a set set  $\Rightarrow$  ('a  $\Rightarrow$ 
nat)  $\Rightarrow$  'a set  $\Rightarrow$  bool where
  some-share-negative-aux [] Ft Fc w X = ((w  $\bowtie_f$  Ft X) < 0)
| some-share-negative-aux (h # t) Ft Fc w X =
  (if (w  $\bowtie_f$  Ft X) + sum-list (map (λ A. (w  $\bowtie_s$  A X)) (h # t))  $\geq$  0 then
    False
  else if some-share-negative-aux t Ft Fc w X then
    True
  else if h ∈ Ft then
    False
  else
    some-share-negative-aux t (insert-and-close-additional h Ft Fc) Fc w X
)

```

lemma *some-share-negative-aux-soundness*:

assumes *some-share-negative-aux* $L\ Ft\ Fc\ w\ X = False$ **and**
 $\forall A \in \text{set } L. (w \bowtie_s A\ X) < 0$ **and**
distinct L **and**
finite F **and** $F \supseteq Ft$ **and**
union-closed-additional $F\ Fc$ **and**
 $\forall A \in F - Ft. (w \bowtie_s A\ X) < 0 \longrightarrow A \in \text{List.set } L$

shows $(w \bowtie_f F\ X) \geq 0$

using *assms*

proof (*induct* L *arbitrary*: Ft)

case *Nil*

show *?case*

proof–

let $?ss = \lambda A. (w \bowtie_s A\ X)$

have $\text{sum } ?ss\ F = \text{sum } ?ss\ (Ft \cup (F - Ft))$

apply (*subst sum.cong*[*of* $F\ Ft \cup (F - Ft)\ ?ss\ ?ss$])

using $\langle F \supseteq Ft \rangle$

by *auto*

also have $\dots = \text{sum } ?ss\ Ft + \text{sum } ?ss\ (F - Ft)$

apply (*rule sum.union-disjoint*)

using $\langle \text{finite } F \rangle\ \langle F \supseteq Ft \rangle$

by (*auto simp add: finite-subset*)

finally have $\text{sum } ?ss\ F = \text{sum } ?ss\ Ft + \text{sum } ?ss\ (F - Ft)$

.

moreover

have $\text{sum } ?ss\ (F - Ft) \geq 0$

apply (*rule sum-nonneg*)

using $\langle \forall A \in F - Ft. \text{set-share } A\ w\ X < 0 \longrightarrow A \in \text{set } [] \rangle$

by *force*

moreover

have $\text{sum } ?ss\ Ft \geq 0$

using $\langle \text{some-share-negative-aux } []\ Ft\ Fc\ w\ X = False \rangle$

by (*simp add: family-share-def*)

ultimately

show *?thesis*

by (*simp add: family-share-def*)

qed

next

case (*Cons* $h\ t$)

let $?ss = \lambda A. (w \bowtie_s A\ X)$

show *?case*

proof (*cases family-share* $Ft\ w\ X + \text{sum-list } (\text{map } ?ss\ (h \# t)) \geq 0$)

case *True*

let $?Fp = \{A. A \in F - Ft \wedge ?ss\ A \geq 0\}$

let $?Fn = \{A. A \in F - Ft \wedge ?ss\ A < 0\}$

have $(\sum A \in F. ?ss\ A) = (\sum A \in (F - Ft) \cup Ft. ?ss\ A)$

```

    apply (rule sum.cong)
    using ⟨Ft ⊆ F⟩
    by auto
  also have ... = (∑ A∈(F - Ft). ?ss A) + (∑ A∈Ft. ?ss A)
    apply (rule sum.union-disjoint)
    using ⟨finite F⟩ ⟨F ⊇ Ft⟩
    by (auto simp add: finite-subset)
  finally have *: (∑ A∈F. ?ss A) = (∑ A∈(F - Ft). ?ss A) + (∑ A∈Ft. ?ss
A)
.

  have (∑ A∈(F - Ft). ?ss A) = (∑ A∈?Fp ∪ ?Fn. ?ss A)
    by (rule sum.cong) auto
  also have ... = (∑ A∈?Fp. ?ss A) + (∑ A∈?Fn. ?ss A)
    apply (rule sum.union-disjoint)
    using ⟨finite F⟩
    by auto
  finally have **: (∑ A∈(F - Ft). ?ss A) = (∑ A∈?Fp. ?ss A) + (∑ A∈?Fn.
?ss A)
    by auto

  have (∑ A∈?Fp. ?ss A) ≥ 0
    by (rule sum-nonneg) auto
  hence (∑ A∈F. ?ss A) ≥ (∑ A∈?Fn. ?ss A) + (∑ A∈Ft. ?ss A)
    using * **
    by simp

  let ?L = set (h # t)
  have sum-list (map ?ss (h # t)) = (∑ A∈?L. ?ss A)
    using ⟨distinct (h # t)⟩
    by (simp add: sum-list-distinct-conv-sum-set)
  moreover
  have (∑ A∈?L. ?ss A) = (∑ A∈(?Fn ∪ (?L - ?Fn)). ?ss A)
    apply (rule sum.cong)
    using ⟨∀ A∈F - Ft. ?ss A < 0 ⟶ A ∈ List.set (h # t)⟩
    by force auto
  also have ... = (∑ A∈?Fn. ?ss A) + (∑ A∈(?L - ?Fn). ?ss A)
    apply (rule sum.union-disjoint)
    using ⟨finite F⟩
    by auto
  also have ... ≤ (∑ A∈?Fn. ?ss A)
  proof-
    have (∑ A∈(?L - ?Fn). ?ss A) ≤ 0
      apply (rule sum-nonpos)
      using ⟨∀ A ∈ set (h # t). ?ss A < 0⟩
      by auto
    thus ?thesis
      by simp
  qed

```

```

finally
have  $(w \bowtie_f F X) \geq 0$ 
  using  $\langle (\sum A \in F. ?ss A) \geq (\sum A \in ?Fn. ?ss A) + (\sum A \in Ft. ?ss A) \rangle$ 
  using  $\langle (w \bowtie_f Ft X) + \text{sum-list } (\text{map } ?ss (h \# t)) \geq 0 \rangle$ 
  unfolding family-share-def
  by simp
thus ?thesis
  by simp
next
case False
note * = this
have  $\neg \text{some-share-negative-aux } t \ Ft \ Fc \ w \ X$ 
  using  $\langle \text{some-share-negative-aux } (h \# t) \ Ft \ Fc \ w \ X = \text{False} \rangle *$ 
  by auto
note * = * this
note * = *  $\langle \text{finite } F \rangle \langle Ft \subseteq F \rangle \langle \text{union-closed-additional } F \ Fc \rangle$ 
have  $\forall A \in \text{set } t. ?ss A < 0 \text{ distinct } t$ 
  using  $\langle \forall A \in \text{set } (h \# t). ?ss A < 0 \rangle \langle \text{distinct } (h \# t) \rangle$ 
  by auto
note * = * this

show ?thesis
proof (cases  $h \in Ft$ )
case True
thus ?thesis
  using * Cons
  by auto
next
case False
let ?Ft' = insert-and-close-additional  $h \ Ft \ Fc$ 
let ?s' = family-share ?Ft'  $w \ X$ 
show ?thesis
proof (cases  $h \in F$ )
case True
show ?thesis
proof (rule Cons(1))
show  $\text{some-share-negative-aux } t \ ?Ft' \ Fc \ w \ X = \text{False}$ 
  using  $\langle \text{some-share-negative-aux } (h \# t) \ Ft \ Fc \ w \ X = \text{False} \rangle * \langle h \notin Ft \rangle$ 
  by auto
next
show  $\forall A \in F - ?Ft'. ?ss A < 0 \longrightarrow A \in \text{set } t$ 
  using  $\langle \forall A \in F - Ft. ?ss A < 0 \longrightarrow A \in \text{set } (h \# t) \rangle$ 
  using  $\langle h \notin Ft \rangle$ 
  by auto
next
show ?Ft'  $\subseteq F$ 
  using  $\langle Ft \subseteq F \rangle \langle h \in F \rangle \langle \text{union-closed-additional } F \ Fc \rangle$ 
  by (auto simp add: union-closed-def)
next

```

```

      show union-closed-additional F Fc
      using *
      by auto
    qed (auto simp add: *)
  next
  case False
  thus ?thesis
    using Cons(1)  $\langle \forall A \in F - Ft. ?ss A < 0 \longrightarrow A \in set (h \# t) \rangle *$ 
    by auto
  qed
qed
qed
qed

```

definition *some-share-negative* :: 'a set set \Rightarrow ('a \Rightarrow nat) \Rightarrow bool **where**

```

some-share-negative Fc w  $\equiv$ 
  let X = ( $\bigcup$  Fc);
  P = Pow X;
  Fc = closure Fc;
  N = {x. x  $\in$  P  $\wedge$  (w  $\bowtie_s$  x X) < 0};
  L = (SOME L. distinct L  $\wedge$  set L = N) in
  some-share-negative-aux L {} Fc w X

```

lemma *some-share-negative-soundness'*:

```

  assumes some-share-negative Fc w = False and
    finite ( $\bigcup$  Fc) and F  $\in$   $\{\!\!|$  Fc  $\!\!\}$ 
  shows (w  $\bowtie_f$  F ( $\bigcup$  Fc))  $\geq$  0

```

proof (rule some-share-negative-aux-soundness)

```

  let ?X =  $\bigcup$  Fc
  let ?P = Pow ?X
  let ?Fc = closure Fc
  let ?N = {x. x  $\in$  ?P  $\wedge$  set-share x w ?X < 0}
  let ?L = SOME L. distinct L  $\wedge$  set L = ?N

```

have *: distinct ?L \wedge set ?L = ?N

proof (rule someI-ex)

```

  have finite ?N
    using  $\langle$ finite ( $\bigcup$  Fc) $\rangle$ 
    by simp
  thus  $\exists$  L. distinct L  $\wedge$  set L = ?N
    by (rule ex-list-of-set)

```

qed

show some-share-negative-aux ?L {} ?Fc w (\bigcup Fc) = False

```

  using assms
  unfolding some-share-negative-def Let-def
  by simp

```

show $\forall a \in set ?L. (w \bowtie_s a (\bigcup Fc)) < 0$ distinct ?L

```

using *
by simp-all

show  $\forall a \in F - \{\}. (w \bowtie_s a (\bigcup Fc)) < 0 \longrightarrow a \in \text{set } ?L$ 
using  $\langle F \in \{\!\!| Fc |\!\!\rangle *$ 
by auto

show finite F
using  $\langle F \in \{\!\!| Fc |\!\!\rangle \langle \text{finite } (\bigcup Fc) \rangle$ 
by (auto simp add: finite-subset)
next
show union-closed-additional F (closure Fc)
using  $\langle F \in \{\!\!| Fc |\!\!\rangle \langle \text{finite } (\bigcup Fc) \rangle \text{union-closed-additional-closure}[of Fc F]$ 
by (auto simp add: finiteUn-iff)
qed (auto simp add: assms)

theorem some-share-negative-soundness:
assumes finite ( $\bigcup Fc$ )
shows some-share-negative Fc w = False  $\implies$  uce-shares-nonneg Fc w
using assms
using some-share-negative-soundness'
by auto

end

```

10.1 Abstract representation of sets and families with weights and shares

```

theory WeightsSharesImpl
imports WeightsShares FamilyImpl
          HOL-Library.Mapping
          HOL-Library.RBT-Mapping HOL-Library.RBT-Impl
begin

locale SetWeightsSharesImpl = SetImpl to-set inv for
  inv :: 's  $\Rightarrow$  bool and to-set :: 's  $\Rightarrow$  'a set +
  fixes set-weight :: ('a  $\Rightarrow$  nat)  $\Rightarrow$  's  $\Rightarrow$  nat
  assumes set-weight-set: inv A  $\implies$  set-weight w A = w  $\triangleright_s$  to-set A
begin

definition set-share :: 's  $\Rightarrow$  ('a  $\Rightarrow$  nat)  $\Rightarrow$  's  $\Rightarrow$  int where
  set-share A w X = 2 * int (set-weight w A) - int (set-weight w X)

definition family-share :: 's list  $\Rightarrow$  ('a  $\Rightarrow$  nat)  $\Rightarrow$  's  $\Rightarrow$  int where
  family-share F w X = sum-list (map ( $\lambda A.$  set-share A w X) F)

lemma set-share-set:
  assumes inv x and inv X
  shows set-share x w X = (w  $\bowtie_s$  (to-set x) (to-set X))

```

```

unfolding WeightsShares.set-share-def set-share-def
using assms
by (auto simp add: set-weight-set)

lemma family-share-set:
  assumes distinct F and  $\forall l \in \text{set } F. \text{inv } l$  and inv X
  shows family-share F w X = (w  $\bowtie_f$  (f-to-set F) (to-set X))
unfolding family-share-def WeightsShares.family-share-def
apply (subst sum-list-distinct-conv-sum-set)
using assms
apply simp-all
apply (subst sum.reindex)
using assms
apply (auto simp add: to-set-inj-on)
apply (rule sum.cong)
using assms
by (auto simp add: set-share-set)

end

```

10.1.1 Implementation by sorted and distinct lists

```

definition set-weight-l :: ('a  $\Rightarrow$  nat)  $\Rightarrow$  'a list  $\Rightarrow$  nat where
  set-weight-l w S = sum-list (map w S)

```

```

lemma set-weight-l:
  assumes distinct A
  shows set-weight-l w A = w  $\triangleright_s$  set A
unfolding set-weight-l-def set-weight-def
apply (subst sum-list-distinct-conv-sum-set)
using assms
by simp-all

```

```

global-interpretation SetWeightsSharesImpl-lists: SetWeightsSharesImpl  $\lambda$  (l::nat list). sorted l  $\wedge$  distinct l set set-weight-l
by (unfold-locales) (simp add: set-weight-l)

```

10.1.2 Implementation by natural numbers

```

definition set-weight-n :: (nat  $\Rightarrow$  nat)  $\Rightarrow$  nat  $\Rightarrow$  nat where
  set-weight-n w n = sum-list (map w (nat2list n))

```

```

lemma set-weight-n:
  shows set-weight-n w A = w  $\triangleright_s$  set (nat2list A)
unfolding set-weight-n-def set-weight-def
by (subst sum-list-distinct-conv-sum-set) (simp-all add: distinct-nat2list)

```

```

global-interpretation SetWeightsSharesImpl-nats: SetWeightsSharesImpl  $\lambda$  -. True
set  $\circ$  nat2list set-weight-n
by (unfold-locales) (simp add: set-weight-n)

```


10.2 Weight functions implemented by mappings

locale *SetWeightsSharesMapImpl* = *SetWeightsSharesImpl* *inv* *to-set* *set-weight*
for
inv :: 's \Rightarrow bool **and** *to-set* :: 's \Rightarrow 'a set **and** *set-weight* :: ('a \Rightarrow nat) \Rightarrow 's \Rightarrow nat +

fixes *set-weight-map* :: ('a, nat) mapping \Rightarrow 's \Rightarrow nat

assumes *set-weight-map*:

$\llbracket \text{inv } A; \forall x \in \text{to-set } A. \text{Mapping.lookup } \text{wm } x = \text{Some } (w \ x) \rrbracket \Longrightarrow$

set-weight-map *wm* *A* = *set-weight* *w* *A*

begin

abbreviation *set-share-map* :: 's \Rightarrow ('a, nat) mapping \Rightarrow 's \Rightarrow int **where**

set-share-map *A* *w* *X* \equiv 2 * int(*set-weight-map* *w* *A*) - int (*set-weight-map* *w* *X*)

end

10.2.1 Representation of sets by lists

definition *set-weight-map-l* :: ('a, nat) mapping \Rightarrow 'a list \Rightarrow nat **where**

set-weight-map-l *w* *S* = *sum-list* (map (*the* \circ (*Mapping.lookup* *w*)) *S*)

lemma *set-weight-map-l*:

$\llbracket \text{distinct } A; \forall x \in \text{set } A. \text{Mapping.lookup } \text{wm } x = \text{Some } (w \ x) \rrbracket \Longrightarrow$

set-weight-map-l *wm* *A* = *set-weight-l* *w* *A*

unfolding *set-weight-l-def* *set-weight-map-l-def*

using *sum-list-distinct-conv-sum-set*[of *A* *w*]

using *sum-list-distinct-conv-sum-set*[of *A* *the* \circ (*Mapping.lookup* *wm*)]

by *auto*

global-interpretation *SetWeightsSharesMapImpl-lists*:

SetWeightsSharesMapImpl λ (*l*::nat list). sorted *l* \wedge distinct *l* *List.set* *set-weight-l* *set-weight-map-l*

proof (*unfold-locales*)

fix *A* **and** *wm* :: ('a::linorder, nat) mapping **and** *w*

assume sorted *A* \wedge distinct *A* $\forall x \in \text{set } A. \text{Mapping.lookup } \text{wm } x = \text{Some } (w \ x)$

thus *set-weight-map-l* *wm* *A* = *set-weight-l* *w* *A*

by (*simp* *add*: *set-weight-map-l*)

qed

10.2.2 Representation of sets by natural numbers

definition *set-weight-map-n* :: (nat, nat) Mapping.mapping \Rightarrow nat \Rightarrow nat **where**

set-weight-map-n *w* *n* = *sum-list* (map (*the* \circ (*Mapping.lookup* *w*)) (nat2list *n*))

global-interpretation *SetWeightsSharesCachedImpl-nats*:

SetWeightsSharesMapImpl λ -. True *List.set* \circ nat2list *set-weight-n* *set-weight-map-n*

proof

fix *A* *wm* **and** *w* :: nat \Rightarrow nat

```

assume  $\forall x \in (\text{set} \circ \text{nat2list}) A. \text{Mapping.lookup } \text{wm } x = \text{Some } (w \ x)$ 
thus  $\text{set-weight-map-n } \text{wm } A = \text{set-weight-n } w \ A$ 
unfolding  $\text{set-weight-n-def } \text{set-weight-map-n-def}$ 
using  $\text{distinct-nat2list[of } A]$ 
by  $(\text{auto simp add: sum-list-distinct-conv-sum-set})$ 
qed

end

```

11 SomeShareNegative – executable implementation

```

theory SomeShareNegativeImpl
imports Main
         HOL-Library.List-lexord
         More.MoreMap
         SomeShareNegative UnionClosedImpl WeightsSharesImpl
begin

```

In this section we define a refinement of the *some-share-negative* function in several steps in order to obtain a more efficient, executable implementation. Optimizations are introduced gradually, through separate refinement steps.

11.1 Refinement of SomeShareNegativeFunction

```

locale SomeShareNegativeImpl =
  SetWeightsSharesMapImpl inv to-set set-weight set-weight-map +
  SetUnionImpl inv to-set empty union pow
  for  $\text{inv} :: 's \Rightarrow \text{bool}$  and  $\text{to-set} :: 's \Rightarrow 'a \text{ set}$  and  $\text{set-weight} :: ('a \Rightarrow \text{nat}) \Rightarrow 's \Rightarrow \text{nat}$ 
  and  $\text{set-weight-map} :: ('a, \text{nat}) \text{ mapping} \Rightarrow 's \Rightarrow \text{nat}$  and
   $\text{empty} :: 's$  and  $\text{union} :: 's \Rightarrow 's \Rightarrow 's$  and  $\text{pow} :: 's \Rightarrow 's \text{ list}$ 
begin

```

11.1.1 Initial version

```

fun some-share-negative-aux-1 ::  $'s \text{ list} \Rightarrow 's \text{ list} \Rightarrow 's \text{ list} \Rightarrow ('a \Rightarrow \text{nat}) \Rightarrow 's \Rightarrow \text{bool}$  where
  some-share-negative-aux-1 []  $F \text{ Init } w \ X = (\text{family-share } F \ w \ X < 0)$ 
| some-share-negative-aux-1 ( $h \ \# \ t$ )  $F \text{ Init } w \ X =$ 
   $(\text{if } \text{family-share } F \ w \ X + \text{sum-list } (\text{map } (\lambda A. \text{set-share } A \ w \ X)) \ (h \ \# \ t)) \geq 0$ 
then
  False
else if some-share-negative-aux-1  $t \ F \text{ Init } w \ X$  then
  True
else if  $h \in \text{set } F$  then
  False
else
  some-share-negative-aux-1  $t \ (\text{insert-and-close-additional } h \ F \text{ Init}) \text{ Init } w \ X$ 

```

)

lemma *some-share-negative-aux-1-some-share-negative-aux:*

assumes *distinct F*

$\forall l \in \text{set } F. \text{inv } l$

$\forall l \in \text{set } L. \text{inv } l$

$\forall l \in \text{set } \text{Init}. \text{inv } l$

inv X

shows *some-share-negative-aux-1 L F Init w X = some-share-negative-aux (map to-set L) (f-to-set F) (f-to-set Init) w (to-set X)*

using *assms*

proof (*induct L arbitrary: F*)

case *Nil*

thus *?case*

using *family-share-set[of F X w]*

by *simp*

next

case (*Cons h t*)

have **: family-share F w X + sum-list (map ($\lambda A. \text{set-share } A w X$) (h # t))*

=

(w \bowtie_f (f-to-set F) (to-set X)) + sum-list (map ($\lambda A. (w \bowtie_s A (to-set X))$) (map to-set (h # t)))

proof–

have *sum-list (map ($\lambda A. \text{set-share } A w X$) (h # t)) = sum-list (map ($\lambda A. (w \bowtie_s A (to-set X))$) (map to-set (h # t)))*

proof–

have *map ($\lambda A. \text{set-share } A w X$) t = map ($\lambda x. (w \bowtie_s (to-set x) (to-set X))$) t*

using $\langle \forall a \in \text{set } (h \# t). \text{inv } a \rangle \langle \text{inv } X \rangle$

using *set-share-set*

by *auto*

hence $(\sum A \leftarrow t. \text{set-share } A w X) = (\sum x \leftarrow t. (w \bowtie_s (to-set x) (to-set X)))$

by *metis*

thus *?thesis*

using $\langle \forall a \in \text{set } (h \# t). \text{inv } a \rangle \langle \text{inv } X \rangle$

by (*auto simp add: set-share-set comp-def*)

qed

thus *?thesis*

using *family-share-set[of F X w]*

using $\langle \text{distinct } F \rangle \langle \forall a \in \text{set } F. \text{inv } a \rangle \langle \text{inv } X \rangle$

by *simp*

qed

show *?case*

proof (*cases family-share F w X + sum-list (map ($\lambda A. \text{set-share } A w X$) (h # t)) ≥ 0*)

case *True*

thus *?thesis*

```

    using *
    by simp
next
case False
  have **: some-share-negative-aux-1 t F Init w X = some-share-negative-aux
    (map to-set t) (f-to-set F) (f-to-set Init) w (to-set X)
    using Cons
    by auto

  show ?thesis
  proof (cases some-share-negative-aux-1 t F Init w X)
    case True
      show ?thesis
      using  $\langle \neg \text{family-share } F \text{ w } X + \text{sum-list } (\lambda A. \text{set-share } A \text{ w } X) (h \# t) \rangle \geq 0 \rangle *$ 
      using  $\langle \text{some-share-negative-aux-1 } t \text{ F Init w X} \rangle **$ 
      by auto
    next
      case False
        have ***:  $(h \in \text{set } F) = (to\text{-set } h \in f\text{-to-set } F)$ 
        apply (rule set-set[of F h])
        using  $\langle \forall a \in \text{set } F. \text{inv } a \rangle \langle \forall a \in \text{set } (h \# t). \text{inv } a \rangle$ 
        by auto
        show ?thesis
        proof (cases  $h \in \text{set } F$ )
          case True
            show ?thesis
            using  $\langle \neg \text{family-share } F \text{ w } X + \text{sum-list } (\lambda A. \text{set-share } A \text{ w } X) (h \# t) \rangle \geq 0 \rangle *$ 
            using  $\langle \neg \text{some-share-negative-aux-1 } t \text{ F Init w X} \rangle **$ 
            using  $\langle h \in \text{set } F \rangle ***$ 
            by simp
          next
            case False
              let  $?F' = \text{insert-and-close-additional } h \text{ F Init}$ 

              have some-share-negative-aux-1 t (insert-and-close-additional h F Init) Init
                w X =
                  some-share-negative-aux (map to-set t) (UnionClosed.insert-and-close-additional
                    (to-set h) (f-to-set F) (f-to-set Init)) (f-to-set Init) w (to-set X)
              proof (subst Cons(1))
                show distinct ?F'
                using  $\langle \text{distinct } F \rangle$ 
                by (simp add: insert-sets-distinct)
              next
                show  $\text{inv } X \ \forall l \in \text{set } t. \text{inv } l \ \forall l \in \text{set } \text{Init}. \text{inv } l$ 
                using  $\langle \text{inv } X \rangle \langle \forall l \in \text{set } (h \# t). \text{inv } l \rangle \langle \forall l \in \text{set } \text{Init}. \text{inv } l \rangle$ 
                by simp-all
              next

```

```

show  $\forall l \in (\text{set } ?F'). \text{ inv } l$ 
  using  $\langle \forall a \in \text{set } F. \text{ inv } a \rangle \langle \forall a \in \text{set } (h \# t). \text{ inv } a \rangle \langle \forall a \in \text{set } \text{Init}. \text{ inv } a \rangle$ 
  using  $\langle \text{distinct } F \rangle$ 
  by (auto simp add: insert-sets-remdups union-inv)
next
show some-share-negative-aux (map to-set t) (f-to-set (insert-and-close-additional
h F Init)) (f-to-set Init) w (to-set X) =
  some-share-negative-aux (map to-set t) (UnionClosed.insert-and-close-additional
(to-set h) (f-to-set F) (f-to-set Init)) (f-to-set Init) w
  (to-set X)
  using  $\langle \text{distinct } F \rangle$ 
  using insert-and-close-additional-set
  by simp
qed
thus ?thesis
  using  $\langle \neg \text{family-share } F \text{ w } X + \text{sum-list } (\text{map } (\lambda A. \text{set-share } A \text{ w } X) (h \# t)) \geq 0 \rangle *$ 
  using  $\langle \neg \text{some-share-negative-aux-1 } t \text{ F Init w } X \rangle **$ 
  using  $\langle \neg h \in \text{set } F \rangle ***$ 
  by simp
qed
qed
qed
qed

```

11.1.2 Passing current family share as a parameter

```

fun some-share-negative-aux-2 :: ('s × int) list ⇒ ('s list × int) ⇒ 's list ⇒ ('a
⇒ nat) ⇒ 's ⇒ bool where
  some-share-negative-aux-2 [] (F, s) Init w X = (s < 0)
| some-share-negative-aux-2 (h # t) (F, s) Init w X =
  (if s + sum-list (map snd (h # t)) ≥ 0 then
    False
  else if some-share-negative-aux-2 t (F, s) Init w X then
    True
  else if fst h ∈ set F then
    False
  else
    let F' = insert-and-close-additional (fst h) F Init;
    s' = family-share F' w X in
    some-share-negative-aux-2 t (F', s') Init w X
  )

```

lemma some-share-negative-aux-2-some-share-negative-aux-1:

```

assumes L = zip l (map (λ F. set-share F w X) l) c = family-share F w X
shows some-share-negative-aux-2 L (F, c) Init w X = some-share-negative-aux-1
l F Init w X
using assms

```

by (induct l arbitrary: F L c) (auto simp add: Let-def)

11.1.3 Caching set shares

Instead of calculating set share every time from scratch when calculating, family shares, we shall maintain a mapping from sets to their shares.

abbreviation *family-share-cached* :: 's list \Rightarrow ('s, int) mapping \Rightarrow int **where**
family-share-cached F shares \equiv sum-list (map (the \circ (λ A. Mapping.lookup shares A)) F)

lemma *family-share-cached-family-share*:

assumes \forall A. inv A \wedge to-set A \subseteq S \longrightarrow Mapping.lookup shares A = Some (set-share A w X)

\forall A \in set F. inv A \wedge to-set A \subseteq S

shows *family-share-cached* F shares = *family-share* F w X

proof –

have map (λ x. the (Mapping.lookup shares x)) F = map (λ A. local.set-share A w X) F

using *assms*

by *auto*

thus ?thesis

unfolding *family-share-def comp-def*

by *metis*

qed

lemma *family-share-cached-cong*:

assumes set F = set F' distinct F distinct F'

shows *family-share-cached* F shares = *family-share-cached* F' shares

using *assms*

using *sum-list-distinct-conv-sum-set*[of F the \circ (λ A. Mapping.lookup shares A)]

using *sum-list-distinct-conv-sum-set*[of F' the \circ (λ A. Mapping.lookup shares A)]

by *simp*

fun *some-share-negative-aux-3* :: ('s \times int) list \Rightarrow ('s list \times int) \Rightarrow 's list \Rightarrow ('s, int) mapping \Rightarrow bool **where**

some-share-negative-aux-3 [] (F, s) Init shares = (s < 0)

| *some-share-negative-aux-3* (h # t) (F, s) Init shares =

(if s + sum-list (map snd (h # t)) \geq 0 then

False

else if *some-share-negative-aux-3* t (F, s) Init shares then

True

else if fst h \in set F then

False

else

let F' = insert-and-close-additional (fst h) F Init;

s' = *family-share-cached* F' shares in

some-share-negative-aux-3 t (F', s') Init shares

)

```

lemma some-share-negative-aux-3-some-share-negative-aux-2:
  assumes  $\forall A. \text{inv } A \wedge \text{to-set } A \subseteq S \longrightarrow \text{Mapping.lookup shares } A = \text{Some}$ 
    (set-share  $A$   $w$   $X$ )
     $\forall A \in \text{set } F. \text{inv } A \wedge \text{to-set } A \subseteq S$ 
     $\forall A \in \text{set } \text{Init}. \text{inv } A \wedge \text{to-set } A \subseteq S$ 
     $\forall A \in \text{set } (\text{map fst } l). \text{inv } A \wedge \text{to-set } A \subseteq S$ 
    distinct  $F$ 
  shows some-share-negative-aux-3  $l$  ( $F, s$ ) Init shares = some-share-negative-aux-2
     $l$  ( $F, s$ ) Init  $w$   $X$ 
  using assms
  proof (induct  $l$  arbitrary:  $F$   $s$ )
    case Nil
    thus ?case
    by simp
  next
    case (Cons  $h$   $t$ )
    show ?case
    proof (cases  $s + \text{sum-list } (\text{map snd } (h \# t)) \geq 0$ )
      case True
      thus ?thesis
      by simp
    next
      case False
      show ?thesis
      proof (cases  $\text{fst } h \in \text{set } F$ )
        case True
        thus ?thesis
        using  $\langle \neg s + \text{sum-list } (\text{map snd } (h \# t)) \geq 0 \rangle$  Cons
        by simp
      next
        case False
        let  $?F' = \text{insert-and-close-additional } (\text{fst } h) F$  Init
        let  $?s' = \text{family-share-cached } ?F'$  shares
        let  $?s'' = \text{family-share } ?F'$   $w$   $X$ 

        have  $\forall A \in \text{set } ?F'. \text{to-set } A \subseteq S$ 
          apply (rule insert-and-close-additional-subset)
          using Cons(3) Cons(4) Cons(5)
          using  $\langle \text{distinct } F \rangle$ 
          by simp-all

        have  $\forall A \in \text{set } ?F'. \text{inv } A$ 
          apply (rule insert-and-close-additional-inv)
          using Cons(3) Cons(4) Cons(5)
          using  $\langle \text{distinct } F \rangle$ 
          by simp-all

        have distinct  $?F'$ 
          using  $\langle \text{distinct } F \rangle$ 

```

```

    by (simp add: insert-sets-distinct)

  have ?s' = ?s''
  apply (rule family-share-cached-family-share[of S shares w X ?F])
  using ⟨∀ A ∈ set ?F'. to-set A ⊆ S⟩ ⟨∀ A ∈ set ?F'. inv A⟩ Cons(2)
  by simp-all

  hence some-share-negative-aux-3 t (?F', ?s') Init shares = some-share-negative-aux-2
  t (?F', ?s'') Init w X
  using Cons(1)[of ?F' ?s'] Cons(2) Cons(3) Cons(4) Cons(5)
  using ⟨∀ A ∈ set ?F'. to-set A ⊆ S⟩ ⟨∀ A ∈ set ?F'. inv A⟩
  using ⟨distinct ?F'⟩
  by auto
  thus ?thesis
  using ⟨¬ s + sum-list (map snd (h # t)) ≥ 0⟩ ⟨fst h ∉ set F⟩ Cons
  by auto
qed
qed
qed

lemma some-share-negative-aux-3-correct:
  assumes some-share-negative-aux-3 L (F, c) Init shares = False
  ∀ (x, y) ∈ set L. y = set-share x w X
  c = family-share F w X
  ∀ A. inv A ∧ to-set A ⊆ S ⟶ Mapping.lookup shares A = Some (set-share A w
X)
  ∀ A ∈ set F. inv A ∧ to-set A ⊆ S
  ∀ A ∈ set Init. inv A ∧ to-set A ⊆ S
  ∀ A ∈ set (map fst L). inv A ∧ to-set A ⊆ S
  distinct F
  inv X
  ∀ A ∈ set (map fst L). (w ⋈s (to-set A) (to-set X)) < 0
  finite F'
  UnionClosed.union-closed-additional F' (f-to-set Init)
  f-to-set F ⊆ F'
  ∀ A ∈ F' - f-to-set F. ((w ⋈s A (to-set X)) < 0 ⟶ A ∈ f-to-set (map fst L))
  distinct L
  shows (w ⋈f F' (to-set X)) ≥ 0
using assms
using exists-zip[of L λ x. set-share x w X]
apply (subst (asm) some-share-negative-aux-3-some-share-negative-aux-2[of S shares
w X])
apply simp-all
apply (subst (asm) some-share-negative-aux-2-some-share-negative-aux-1[of L map
fst L])
apply (simp-all add: comp-def)
apply (subst (asm) some-share-negative-aux-1-some-share-negative-aux)
apply simp-all
apply (rule some-share-negative-aux-soundness[of map to-set (map fst L) f-to-set

```



```

F f-to-set Init w to-set X F']
apply (simp-all add: comp-def)
proof -
  assume *:  $\forall A \in \text{set } L. \text{inv } (\text{fst } A) \wedge \text{to-set } (\text{fst } A) \subseteq S \text{ distinct } L \forall x \in \text{set } L. \text{case}$ 
  x of (x, y)  $\Rightarrow y = \text{set-share } x \text{ w } X$ 
  have  $\forall a \in \text{set } (\text{map } \text{fst } L). \text{inv } a$ 
  using  $\langle \forall A \in \text{set } L. \text{inv } (\text{fst } A) \wedge \text{to-set } (\text{fst } A) \subseteq S \rangle$ 
  by simp
  hence inj-on to-set (set (map fst L))
  by (rule to-set-inj-on)
  thus distinct (map ( $\lambda x. \text{to-set } (\text{fst } x)$ ) L)
  using comp-inj-on-iff[of fst set L to-set] * inj-on-fst[of L]
  by (auto simp add: distinct-map comp-def)
qed

```

11.1.4 Passing the sum of shares of elements in the current list as a parameter

```

fun some-share-negative-aux-4 :: (('s  $\times$  int) list  $\times$  int)  $\Rightarrow$  ('s list  $\times$  int)  $\Rightarrow$  's list
 $\Rightarrow$  ('s, int) mapping  $\Rightarrow$  bool where
  some-share-negative-aux-4 ([], -) (F, s) Init shares = (s < 0)
| some-share-negative-aux-4 ((h # t), ls) (F, s) Init shares =
  (if s + ls  $\geq$  0 then
    False
  else if some-share-negative-aux-4 (t, ls - snd h) (F, s) Init shares then
    True
  else if fst h  $\in$  set F then
    False
  else
    let F' = insert-and-close-additional (fst h) F Init;
    s' = family-share-cached F' shares in
    some-share-negative-aux-4 (t, ls - snd h) (F', s') Init shares
  )

```

lemma *some-share-negative-aux-4-some-share-negative-aux-3*:
shows *some-share-negative-aux-4* (l, sum-list (map snd l)) (F, s) Init shares =
some-share-negative-aux-3 l (F, s) Init shares
by (induct l arbitrary: F s) auto

lemma *some-share-negative-aux-4-equal-set*:
assumes *set F = set F' distinct F distinct F'*
shows *some-share-negative-aux-4* (l, ls) (F, s) Init shares =
some-share-negative-aux-4 (l, ls) (F', s) Init shares
using *assms*
proof (induct l arbitrary: F F' s ls)
case Nil
thus ?case
by simp

```

next
  case (Cons h t)
  show ?case
  proof (cases  $s + ls \geq 0$ )
    case True
    thus ?thesis
      by simp
  next
  case False
  show ?thesis
  proof (cases some-share-negative-aux-4 (t, ls - snd h) (F, s) Init shares)
    case True
    thus ?thesis
      using  $\langle \neg s + ls \geq 0 \rangle$ 
      using Cons(1)[of F F' ls - snd h s] Cons(2) Cons(3) Cons(4)
      by auto
  next
  case False
  show ?thesis
  proof (cases fst h  $\in$  set F)
    case True
    thus ?thesis
      using  $\langle \neg s + ls \geq 0 \rangle$ 
      using  $\langle \neg$  some-share-negative-aux-4 (t, ls - snd h) (F, s) Init shares $\rangle$ 
      using Cons(1)[of F F' ls - snd h s] Cons(2) Cons(3) Cons(4)
      by auto
  next
  case False
  let ?F = insert-and-close-additional (fst h) F Init
  let ?F' = insert-and-close-additional (fst h) F' Init
  have set ?F = set ?F'
    using  $\langle$ set F = set F' $\rangle$   $\langle$ distinct F $\rangle$   $\langle$ distinct F' $\rangle$ 
    by (rule insert-and-close-additional-cong)
  hence family-share-cached ?F shares = family-share-cached ?F' shares
    using family-share-cached-cong[of ?F ?F' shares]
    using  $\langle$ distinct F $\rangle$   $\langle$ distinct F' $\rangle$ 
    by (simp add: insert-sets-distinct)
  thus ?thesis
    using  $\langle$ fst h  $\notin$  set F $\rangle$ 
    using  $\langle \neg s + ls \geq 0 \rangle$ 
    using  $\langle \neg$  some-share-negative-aux-4 (t, ls - snd h) (F, s) Init shares $\rangle$ 
    using Cons(1)[of F F' ls - snd h s] Cons(2) Cons(3) Cons(4)
    using  $\langle$ set ?F = set ?F' $\rangle$ 
    using Cons(1)[of ?F ?F']
    by (simp add: Let-def insert-sets-distinct)
qed
qed
qed
qed

```

11.1.5 Incremental insert and close operation and calculation of share of the extended family

fun *insert-and-close-additional-cached* **where**

insert-and-close-additional-cached $A (Ft, s) Fc \text{ shares} =$
 $(\text{let } add = [A] @ \text{union-with-all } A \text{ } Ft @ \text{union-with-all } A \text{ } Fc;$
 $add = \text{filter } (\lambda x. x \notin \text{set } Ft) (\text{remdups } add)$
 $\text{in } (add @ Ft, s + \text{family-share-cached } add \text{ shares}))$

lemma *insert-and-close-additional-cached-family-share-cached*:

shows $\text{let } (Ft', s') = \text{insert-and-close-additional-cached } A (Ft, \text{family-share-cached } Ft \text{ shares}) Fc \text{ shares}$

$\text{in } s' = \text{family-share-cached } Ft' \text{ shares}$

by $(\text{auto simp add: Let-def})$

lemma *insert-and-close-additional-cached*:

assumes

distinct Ft

insert-and-close-additional-cached $A (Ft, \text{family-share-cached } Ft \text{ shares}) Fc \text{ shares}$
 $=$

(Ft', s')

shows

$\text{set } Ft' = \text{set } (\text{insert-and-close-additional } A \text{ } Ft \text{ } Fc) \wedge$
 $s' = \text{family-share-cached } Ft' \text{ shares}$

proof –

let $?add = [A] @ \text{union-with-all } A \text{ } Ft @ \text{union-with-all } A \text{ } Fc$

let $?add = \text{filter } (\lambda x. x \notin \text{set } Ft) (\text{remdups } ?add)$

have $Ft' = ?add @ Ft$ *distinct* Ft'

$s' = \text{family-share-cached } Ft \text{ shares} + \text{family-share-cached } ?add \text{ shares}$

using $(\text{insert-and-close-additional-cached } A (Ft, \text{family-share-cached } Ft \text{ shares})$

$Fc \text{ shares} = (Ft', s'))$

using $(\text{distinct } Ft)$

by $(\text{auto simp add: Let-def})$

thus $?thesis$

by *auto*

qed

function *some-share-negative-aux-5* $:: ((s \times \text{int}) \text{ list} \times \text{int}) \Rightarrow (s \text{ list} \times \text{int}) \Rightarrow$
 $s \text{ list} \Rightarrow (s, \text{int}) \text{ mapping} \Rightarrow \text{bool}$ **where**

some-share-negative-aux-5 $([], -) (F, s) \text{ Init shares} =$

$(\text{if } s < 0 \text{ then True else False})$

| *some-share-negative-aux-5* $((h \# t), ls) (F, s) \text{ Init shares} =$

$(\text{if } s + ls \geq 0 \text{ then}$

False

$\text{else if } \text{some-share-negative-aux-5 } (t, ls - \text{snd } h) (F, s) \text{ Init shares then}$

True

$\text{else if } \text{fst } h \in \text{set } F \text{ then}$

False

else

```

    let (F', s') = insert-and-close-additional-cached (fst h) (F, s) Init shares in
    some-share-negative-aux-5 (t, ls - snd h) (F', s') Init shares
  )
by pat-completeness auto
termination
by (relation measure (λ ((L, -), -, -, -). length L)) auto

lemma some-share-negative-aux-5-some-share-negative-aux-4:
assumes s = family-share-cached F shares distinct F
shows some-share-negative-aux-5 (L, ls) (F, s) Init shares =
some-share-negative-aux-4 (L, ls) (F, s) Init shares
using assms
proof (induct L arbitrary: F s ls)
  case Nil
  thus ?case
  by simp
next
  case (Cons h t)
  show ?case
  proof (cases 0 ≤ s + ls)
    case True
    thus ?thesis
    unfolding some-share-negative-aux-4.simps some-share-negative-aux-5.simps
    by simp
  next
    case False
    hence *: some-share-negative-aux-5 (t, ls - snd h) (F, s) Init shares =
some-share-negative-aux-4 (t, ls - snd h) (F, s) Init shares
    using Cons(1)[of s F ls - snd h] Cons(2) Cons(3)
    by auto
    show ?thesis
  proof (cases some-share-negative-aux-5 (t, ls - snd h) (F, s) Init shares)
    case True
    thus ?thesis
    using ⟨¬ 0 ≤ s + ls⟩ *
    by simp
  next
    case False
    show ?thesis
  proof (cases fst h ∈ set F)
    case True
    thus ?thesis
    using ⟨¬ 0 ≤ s + ls⟩
    using ⟨¬ some-share-negative-aux-5 (t, ls - snd h) (F, s) Init shares⟩
    using *
    by simp
  next
    case False
    let ?F1s1 = insert-and-close-additional-cached (fst h) (F, s) Init shares

```

```

let ?F1 = fst ?F1s1
let ?s1 = snd ?F1s1
let ?F2 = insert-and-close-additional (fst h) F Init
let ?s2 = family-share-cached ?F2 shares

have distinct ?F1 distinct ?F2
  using ⟨distinct F⟩
  unfolding Let-def
  by (auto simp add: insert-sets-distinct Let-def)

hence set ?F1 = set ?F2 ∧ ?s1 = ?s2
  using insert-and-close-additional-cached[OF ⟨distinct F⟩, of fst h shares
Init ?F1 ?s1] Cons(2) by (metis family-share-cached-cong prod.collapse)
hence set ?F1 = set ?F2 ?s1 = ?s2
  by (rule conjunct1, rule conjunct2)

have some-share-negative-aux-4 (t, ls - snd h) (?F1, ?s1) Init shares =
  some-share-negative-aux-4 (t, ls - snd h) (?F2, ?s2) Init shares
proof (subst some-share-negative-aux-4-equal-set[of ?F1 ?F2 t ls - snd h
?s1 Init shares])
  show set ?F1 = set ?F2
    by fact
next
  show distinct ?F1 distinct ?F2
    by fact+
next
  show some-share-negative-aux-4 (t, ls - snd h) (?F2, ?s1) Init shares =
    some-share-negative-aux-4 (t, ls - snd h) (?F2, ?s2) Init shares
    by (subst ⟨?s1 = ?s2⟩, rule refl)
qed
moreover
  have some-share-negative-aux-5 (t, ls - snd h) (?F1, ?s1) Init shares =
some-share-negative-aux-4 (t, ls - snd h) (?F1, ?s1) Init shares
  apply (rule Cons(1)[of ?s1 ?F1 ls - snd h])
  using Cons(2) ⟨distinct ?F1⟩
  using insert-and-close-additional-cached-family-share-cached[of fst h F
shares Init]
  by (auto simp add: Let-def comp-def)
ultimately
show ?thesis
  using ⟨fst h ∉ set F⟩
  using ⟨¬ 0 ≤ s + ls⟩
  using ⟨¬ some-share-negative-aux-5 (t, ls - snd h) (F, s) Init shares⟩ *
  by (simp add: Let-def)
qed
qed
qed
qed

```

11.1.6 Final implementation

definition *some-share-negative* :: 's list \Rightarrow ('a, nat) mapping \Rightarrow bool where
some-share-negative A w \equiv

```

let S = Union A;
P = pow S;
A = close A;
U' = map ( $\lambda$  A. set-share-map A w S) P;
U = sort-key snd (zip P U');
shares = tabulate U;
L = takeWhile ( $\lambda$  (a, b). b < 0) U;
F = [] in
some-share-negative-aux-5 (L, sum-list (map snd L)) (F, 0) A shares

```

lemma *some-share-negativeFalse-FamilyShare*:

assumes

some-share-negative A w = False

$A' = f\text{-to-set } A$

$\forall l \in \text{set } A. \text{inv } l$

finite F

UnionClosed.union-closed-additional F (closure A')

$\forall x \in F. x \subseteq \bigcup A'$

$\forall x \in \bigcup A'. \text{Mapping.lookup } w \ x = \text{Some } (w' \ x)$

shows $(w' \bowtie_f F (\bigcup A')) \geq 0$

proof–

let ?S = Union A

let ?P = pow ?S

let ?A = close A

let ?U' = map (λ A. set-share-map A w ?S) ?P

let ?U = sort-key snd (zip ?P ?U')

let ?shares = tabulate ?U

let ?L = takeWhile (λ (a, b). b < 0) ?U

let ?F = []

have inv (Union A)

using $\langle \forall l \in \text{set } A. \text{inv } l \rangle$

using *Union-inv*[of A]

by simp

have *: $\forall x \in \text{set } ?P. \text{set-share-map } x \ w \ (\text{Union } A) = \text{set-share } x \ w' \ (\text{Union } A)$

proof

fix x

assume $x \in \text{set } ?P$

hence *to-set* x $\subseteq \bigcup A' \text{ inv } x$

using *pow-set*[of Union A]

using *pow-inv*[of Union A]

using *Union-set*[of A] $\langle \text{inv } (\text{Union } A) \rangle$ $\langle A' = f\text{-to-set } A \rangle$

by auto

```

hence  $\forall a \in \text{to-set } x. \text{Mapping.lookup } w \ a = \text{Some } (w' \ a)$ 
  using  $\langle \forall a \in \bigcup A'. \text{Mapping.lookup } w \ a = \text{Some } (w' \ a) \rangle$ 
  by auto
thus  $\text{set-share-map } x \ w \ (\text{Union } A) = \text{set-share } x \ w' \ (\text{Union } A)$ 
  using  $\text{set-weight-map}[of \ x \ w \ w']$ 
  using  $\text{set-weight-map}[of \ \text{Union } A \ w \ w']$ 
  using  $\langle \forall a \in \bigcup A'. \text{Mapping.lookup } w \ a = \text{Some } (w' \ a) \rangle$ 
  using  $\text{Union-set}[of \ A]$ 
  using  $\langle \text{inv } (\text{Union } A) \rangle \langle \text{inv } x \rangle \langle A' = f\text{-to-set } A \rangle$ 
  by (simp add: set-share-def)
qed

show ?thesis
proof (subst  $\langle A' = f\text{-to-set } A \rangle$ , subst  $\text{Union-set}[THEN \ \text{sym}]$ , rule some-share-negative-aux-3-correct[of
?L ?F 0 ?A ?shares])
  show  $\text{some-share-negative-aux-3 } ?L \ ( ?F, 0) \ ?A \ ?shares = \text{False}$ 
    using  $\langle \text{some-share-negative } A \ w = \text{False} \rangle$ 
    unfolding some-share-negative-def Let-def
    using some-share-negative-aux-4-some-share-negative-aux-3
    using some-share-negative-aux-5-some-share-negative-aux-4
    by simp
  next
  show  $\forall (x, y) \in \text{set } ?L. y = \text{set-share } x \ w' \ (\text{Union } A)$ 
    using *
    by (auto dest!: set-takeWhileD simp add: set-zip)
  next
  show  $0 = \text{family-share } [] \ w' \ (\text{Union } A)$ 
    by (simp add: family-share-def)
  next
  show  $\forall A'. \text{inv } A' \wedge \text{to-set } A' \subseteq \text{to-set } ?S \longrightarrow \text{Mapping.lookup } ?shares \ A' =$ 
    Some  $(\text{set-share } A' \ w' \ (\text{Union } A))$ 
    proof-
      have distinct ?P
        using  $\langle \text{inv } (\text{Union } A) \rangle$ 
        by (simp add: pow-distinct)
      hence distinct  $(\text{map } \text{fst } ?U)$ 
        using inj-on-fst[of ?U]
        by (force simp add: distinct-map distinct-zipI1 set-zip)
      thus ?thesis
        using  $\langle \text{distinct } ?P \rangle$ 
        using  $\langle \text{inv } (\text{Union } A) \rangle$ 
        using subset-in-pow
        using *
        using tabulate-map-of[of ?U]
        by (auto simp add: tabulate-map-of map-of-sort-key map-of-zip-map Mapping.lookup.abs-eq)
    qed
  next
  show  $\forall A' \in \text{set } (\text{close } A). \text{inv } A' \wedge \text{to-set } A' \subseteq \text{to-set } (\text{Union } A)$ 

```

```

    using close-inv[of A]  $\langle \forall l \in \text{set } A. \text{ inv } l \rangle$ 
    using close-subset[of A to-set (Union A)]
    using Union-set[of A]
    by auto
next
show  $\forall A' \in \text{set } (\text{map fst } ?L). \text{ inv } A' \wedge \text{to-set } A' \subseteq \text{to-set } (\text{Union } A)$ 
    using pow-inv[of Union A] using pow-set[of Union A]  $\langle \text{inv } (\text{Union } A) \rangle$ 
    by (auto dest!: set-takeWhileD set-zip-leftD) force
next
show inv (Union A)
    by fact
next
show  $\forall A' \in \text{set } (\text{map fst } ?L). (w' \bowtie_s (\text{to-set } A') (\text{to-set } (\text{Union } A))) < 0$ 
proof-
{
  fix a b
  assume  $(a, b) \in \text{set } (\text{zip } ?P ?U') \ b < 0$ 
  hence  $a \in \text{set } ?P \text{ set-share-map } a \ w \ (\text{Union } A) < 0$ 
    by (auto simp add: set-zip)
  hence  $(w' \bowtie_s (\text{to-set } a) (\text{to-set } (\text{Union } A))) < 0$ 
    using * set-share-set[of a Union A w]
    using  $\langle \text{inv } (\text{Union } A) \rangle$ 
    using pow-inv[of Union A]
    by auto
}
thus ?thesis
  by (auto dest!: set-takeWhileD)
qed
next
show finite F
  by fact
next
show UnionClosed.union-closed-additional F (f-to-set (close A))
  using  $\langle \text{UnionClosed.union-closed-additional } F \ (\text{closure } A') \rangle \langle A' = \text{f-to-set } A \rangle$ 
  using close-set[of A]
  by simp
next
show  $\forall x \in F - \text{f-to-set } []. ((w' \bowtie_s x (\text{to-set } (\text{Union } A))) < 0 \longrightarrow$ 
 $x \in \text{f-to-set } (\text{map fst } ?L))$ 
proof(safe)
  fix x
  assume  $x \in F \ (w' \bowtie_s x (\text{to-set } (\text{Union } A))) < 0$ 
  have  $x \subseteq \text{to-set } (\text{Union } A)$ 
    using  $\langle \forall x \in F. x \subseteq \bigcup A' \rangle \langle x \in F \rangle$ 
    using Union-set[of A]  $\langle A' = \text{f-to-set } A \rangle$ 
    by simp
  hence  $x \in \text{f-to-set } ?P$ 
    using pow-set[of Union A]
    by simp

```



```

then obtain  $x'$  where  $x = \text{to-set } x' \ x' \in \text{set } ?P \ \text{inv } x'$ 
  using  $\text{pow-inv}[\text{of } \text{Union } A] \ \langle \text{inv } (\text{Union } A) \rangle$ 
  by auto
have  $(x', \text{set-share-map } x' \ w \ (\text{Union } A)) \in \text{set } ?L$ 
  apply  $(\text{rule sorted-takeWhile-snd-neg}[\text{of } ?U \ ?P])$ 
  using  $\langle x' \in \text{set } ?P \rangle \ \langle x = \text{to-set } x' \rangle$ 
  using  $\langle (w' \bowtie_s x \ (\text{to-set } (\text{Union } A))) < 0 \rangle \ \langle \text{inv } (\text{Union } A) \rangle \ \langle \text{inv } x' \rangle$ 
  using  $\text{set-share-set } *$ 
  by auto
thus  $x \in \text{f-to-set } (\text{map fst } ?L)$ 
  using  $\langle x = \text{to-set } x' \rangle$ 
  by  $(\text{force simp add: comp-def})$ 
qed
next
  show  $\text{distinct } ?L$ 
  using  $\langle \text{inv } (\text{Union } A) \rangle$ 
  by  $(\text{auto intro!: distinct-takeWhile distinct-zipI1 pow-distinct})$ 
qed simp-all
qed

```

lemma *some-share-negativeSound:*

assumes

$\forall l \in \text{set } A. \ \text{inv } l$

$A' = \text{f-to-set } A$

$\forall x \in A'. \ \text{finite } x$

$\forall x \in \bigcup A'. \ \text{Mapping.lookup } w \ x = \text{Some } (w' \ x)$

shows $\text{some-share-negative } A \ w = \text{False} \implies \text{uce-shares-nonneg } A' \ w'$

proof

fix F

assume $\text{some-share-negative } A \ w = \text{False} \ F \in \mathbb{A}$

hence $\text{UnionClosed.union-closed } F \ F \subseteq \text{Pow } (\bigcup A') \ \forall A'' \in F. \ \text{op } \cup \ A'' \ ' \ A' \subseteq F$

by *auto*

have $\text{finite } F \ \forall \ x \in F. \ \text{finite } x$

using $\langle F \subseteq \text{Pow } (\bigcup A') \rangle \ \langle \forall \ X \in A'. \ \text{finite } X \rangle \ \langle A' = \text{f-to-set } A \rangle$

by $(\text{auto simp add: finite-subset})$

show $0 \leq (w' \bowtie_f F \ (\bigcup A'))$

proof $(\text{subst } \langle A' = \text{f-to-set } A \rangle, \text{rule some-share-negativeFalse-FamilyShare})$

show $\text{some-share-negative } A \ w = \text{False}$

by *fact*

next

show $\text{UnionClosed.union-closed-additional } F \ (\text{closure } (\text{f-to-set } A))$

apply $(\text{subst } \langle A' = \text{f-to-set } A \rangle [\text{THEN sym}])$

using $\langle \text{UnionClosed.union-closed } F \rangle$

proof (auto)

fix $A'' \ x$

assume $A'' \in F \ x \in \text{closure } A'$

```

    hence  $x \cup A'' \in F$ 
    using  $\langle \text{UnionClosed.union-closed } F \rangle \langle \forall A'' \in F. \text{image } (op \cup A'') A' \subseteq F \rangle$ 
    using  $\text{closure-additional-set}[of A' x A'' F] \langle A' = f\text{-to-set } A \rangle$ 
    by simp
    thus  $A'' \cup x \in F$ 
    by (simp add: Un-commute)
  qed
next
  show finite F
  by fact
next
  show  $\forall l \in \text{set } A. \text{inv } l$ 
  by fact
next
  show  $\forall x \in F. x \subseteq \bigcup f\text{-to-set } A$ 
  using  $\langle A' = f\text{-to-set } A \rangle$ 
  using  $\langle F \subseteq \text{Pow } (\bigcup A') \rangle$ 
  by auto
next
  show  $\forall a \in \bigcup f\text{-to-set } A. \text{Mapping.lookup } w a = \text{Some } (w' a)$ 
  using  $\langle \forall x \in \bigcup A'. \text{Mapping.lookup } w x = \text{Some } (w' x) \rangle$ 
  using  $\langle A' = f\text{-to-set } A \rangle$ 
  by auto
  qed simp
qed
end

```

11.2 Interpretations

11.2.1 Representation of sets by lists

global-interpretation *SomeShareNegativeImpl-lists:*

SomeShareNegativeImpl $\lambda (l :: \text{nat list}). \text{sorted } l \wedge \text{distinct } l \text{ List.set set-weight-}l$
 $\text{set-weight-map-}l \sqsubseteq \text{merge Pow-}l$

proof *qed*

11.2.2 Representation of sets by natural numbers

global-interpretation *SomeShareNegativeImpl-nats:*

SomeShareNegativeImpl $\lambda n. \text{True set} \circ \text{nat2list set-weight-}n \text{ set-weight-map-}n 0$
 $\text{bitor pow-}n$

defines

insert-and-close-additional-cached-}n = \text{SomeShareNegativeImpl-nats.insert-and-close-additional-cached}

and

some-share-negative-aux-5-}n = \text{SomeShareNegativeImpl-nats.some-share-negative-aux-5}

and

some-share-negative-}n = \text{SomeShareNegativeImpl-nats.some-share-negative}

by (*unfold-locales*)

definition *some-share-negative* **where**

some-share-negative $A\ w \equiv \text{some-share-negative-n } (\text{map list2nat } A)\ w$

lemma *some-share-negative-soundness-nats*:

assumes *finite* $A\ \forall\ X \in A. \text{finite } X$

set $(\text{map set } A') = A$

$\forall\ X \in \text{set } A'. \text{distinct } X \wedge \text{sorted } X$

$w' = \text{Mapping.tabulate } (\text{sorted-list-of-set } (\bigcup A))\ w$

shows *some-share-negative* $A'\ w' = \text{False} \implies \text{uce-shares-nonneg } A\ w$

proof (rule *SomeShareNegativeImpl-nats.some-share-negativeSound*)

show $A = \text{f-to-set-n } (\text{map list2nat } A')$

proof–

have $(\text{set} \circ \text{nat2list} \circ \text{list2nat})\ \text{'set } A' = \text{set}\ \text{'set } A'$

using $\langle \forall\ X \in \text{set } A'. \text{distinct } X \wedge \text{sorted } X \rangle$

using *nat2list-list2nat*

by force

thus *?thesis*

using $\langle \text{set } (\text{map set } A') = A \rangle$

by simp

qed

show $\forall x \in \bigcup A. \text{Mapping.lookup } w'\ x = \text{Some } (w\ x)$

using $\langle w' = \text{Mapping.tabulate } (\text{sorted-list-of-set } (\bigcup A))\ w \rangle$

using $\langle A = \text{SetImpl-nats.f-to-set } (\text{map list2nat } A') \rangle$

by auto

qed (*simp-all add: assms some-share-negative-def*)

definition *weights2map* **where**

weights2map $l = \text{foldl } (\lambda\ w\ (k, v). w\ (k := v))\ (\lambda\ -. 0)\ l$

lemma *weights2map-snoc[simp]*: *weights2map* $(xs\ @\ [x]) = (\text{weights2map } xs)\ (\text{fst } x := \text{snd } x)$

by (*simp add: weights2map-def split-def*)

definition *ssn* **where**

ssn $f\ w = \text{some-share-negative } f\ (\text{tabulate } w)$

end

12 Isomorphisms of set families

theory *IsomorphicFamilies*

imports *Main*

More/MoreSet More/MoreFun

UnionClosed Frankl

begin

Each injective function f on the domain $\bigcup F$ maps the set family F to its isomorphic image $op \text{ ' } f \text{ ' } F$. Elements of $\bigcup F$ are mapped using f , while elements of F are mapped using $op \text{ ' } f$.

12.1 Properites preserved by injective functions

lemma *empty-iso*:

shows $F = \{\}$ \longleftrightarrow $(op \text{ ' } f \text{ ' } F) = \{\}$

by *auto*

lemma *inj-on-iso*:

assumes *inj-on* f $(\bigcup F)$

shows *inj-on* $(op \text{ ' } f)$ F

using *assms*

unfolding *inj-on-def*

by *blast*

lemma *inj-on-iso-strong*:

assumes *inj-on* f $(\bigcup F)$

shows *inj-on* $(op \text{ ' } f)$ $(Pow (\bigcup F))$

using *assms*

unfolding *inj-on-def*

by *blast*

lemma *finite-iso*:

assumes *inj-on* f $(\bigcup F)$

shows *finite* $F \longleftrightarrow$ *finite* $(op \text{ ' } f \text{ ' } F)$

using *assms*

using *inj-on-iso*[*of* f F]

by (*auto dest: finite-imageD*)

lemma *card-iso*:

assumes *inj-on* f $(\bigcup F)$

shows *card* $(op \text{ ' } f \text{ ' } F) =$ *card* F

proof (*subst card-image*)

let $?f = op \text{ ' } f$

show *inj-on* $?f$ F

using *assms*

by (*simp add: inj-on-iso*)

qed *simp*

lemma *finite-elems-iso*:

assumes *inj-on* f $(\bigcup F)$

shows $(\forall S \in F. \text{finite } S) \longleftrightarrow (\forall S \in (op \text{ ' } f \text{ ' } F). \text{finite } S)$

proof (*safe*)

let $?F = op \text{ ' } f \text{ ' } F$

fix S

```

assume  $\forall S \in ?F. \text{finite } S \implies S \in F$ 

show  $\text{finite } S$ 
proof (rule finite-imageD)
  show  $\text{finite } (f \text{ `` } S)$ 
    using  $\langle \forall S \in ?F. \text{finite } S \rangle \langle S \in F \rangle$ 
    by simp
next
  show  $\text{inj-on } f \text{ } S$ 
  proof (rule subset-inj-on)
    show  $S \subseteq \bigcup F$ 
    using  $\langle S \in F \rangle$ 
    by auto
  qed (simp add: assms)
qed
qed auto

lemma card-elem-iso:
  assumes  $\text{inj-on } f \text{ } (\bigcup F) \implies A \in F \implies \text{card } A > 0$ 
  shows  $\text{card } (f \text{ `` } A) = \text{card } A$ 
using assms
using inj-on-iff-eq-card[of  $A \text{ } f$ ]
using subset-inj-on[of  $f \text{ } \bigcup F \text{ } A$ ]
using card-ge-0-finite[of  $A$ ]
by blast

lemma finite-Union-iso:
  assumes  $\text{inj-on } f \text{ } (\bigcup F)$ 
  shows  $\text{finite } (\bigcup F) = \text{finite } (\bigcup (\text{op `` } f \text{ `` } F))$ 
using assms
by ((subst finiteUn-iff)+, simp add: finite-iso finite-elems-iso)

lemma union-closed-iso:
  assumes  $\text{inj-on } f \text{ } (\bigcup F)$ 
  shows  $\text{union-closed } F \iff \text{union-closed } (\text{op `` } f \text{ `` } F)$ 
proof
  assume  $\text{union-closed } F$ 
  show  $\text{union-closed } (\text{op `` } f \text{ `` } F)$ 
    unfolding union-closed-def
  proof (safe)
    fix  $A \ B \ x \ y$ 
    assume  $x \in F \ y \in F$ 
    show  $f \text{ `` } x \cup f \text{ `` } y \in \text{op `` } f \text{ `` } F$ 
    proof (rule rev-image-eqI[of  $x \cup y$ ])
      show  $x \cup y \in F$ 
      using  $\langle \text{union-closed } F \rangle \langle x \in F \rangle \langle y \in F \rangle$ 
      unfolding union-closed-def
      by simp
    qed auto
  qed auto

```

```

qed
next
  assume *: union-closed (op ' f ' F)
  show union-closed F
    unfolding union-closed-def
  proof (safe)
    fix A B
    assume A ∈ F B ∈ F
    hence f ' A ∪ f ' B ∈ op ' f ' F
      using *
      unfolding union-closed-def
    by auto
    then obtain X where f ' X = f ' A ∪ f ' B X ∈ F
      by auto
    hence f ' X = f ' (A ∪ B)
      by auto
    hence X = A ∪ B
      using assms ⟨X ∈ F⟩ ⟨A ∈ F⟩ ⟨B ∈ F⟩
      using inj-on-iso-strong[of f F]
      unfolding inj-on-def
      by blast
    thus A ∪ B ∈ F
      using ⟨X ∈ F⟩
      by simp
  qed
qed

lemma finite-union-closed-iso:
  assumes inj-on f (⋃ F)
  shows finite-union-closed F ⟷ finite-union-closed (op ' f ' F)
using assms
using finite-Union-iso[of f F] union-closed-iso[of f F]
by (simp add: finiteUn-iff)

lemma count-iso:
  assumes inj-on f (⋃ F) and a ∈ ⋃ F
  shows count a F = count (f a) (op ' f ' F)
using assms
unfolding count-def
proof (subst card-image[THEN sym])
  let ?f = op ' f
  let ?aF = {S ∈ F. a ∈ S}
  let ?faF = {S ∈ op ' f ' F. f a ∈ S}
  show inj-on ?f ?aF
    using assms
    using inj-on-iso[of f F]
    using subset-inj-on[of op ' f F ?aF]
    by auto

```

```

show card (?f ‘ ?aF) = card ?fafF
proof-
  have ?f ‘ ?aF = ?fafF
  proof (auto)
    fix A a'
    assume A ∈ F a' ∈ A f a = f a'
    hence a = a' using assms
      unfolding inj-on-def
      by auto
    thus f ‘ A ∈ ?f ‘ ?aF
      using rev-image-eqI[of A ?aF f ‘ A op ‘ f] ⟨A ∈ F⟩ ⟨a' ∈ A⟩
      by simp
  qed
  thus ?thesis
    by simp
qed
qed

lemma Frankl-iso:
  assumes inj-on f (⋃ F)
  shows frankl F ⟷ frankl (op ‘ f ‘ F)
proof-
  let ?F = op ‘ f ‘ F
  show ?thesis
    unfolding frankl-def
  proof (safe)
    fix X x
    assume *: X ∈ F x ∈ X card F ≤ 2 * count x F
    show ∃ x. x ∈ ⋃ (op ‘ f ‘ F) ∧ card (op ‘ f ‘ F) ≤ 2 * count x (op ‘ f ‘ F)
      apply (rule-tac x=f x in exI)
      using * ⟨inj-on f (⋃ F)⟩
      by (subst count-iso[THEN sym]) (auto simp add: card-iso)
  next
    fix X x
    assume *: card ?F ≤ 2 * count (f x) ?F X ∈ F x ∈ X
    show ∃ x. x ∈ ⋃ F ∧ card F ≤ 2 * count x F
      apply (rule-tac x=x in exI)
      using * ⟨inj-on f (⋃ F)⟩
      by (subst count-iso) (auto simp add: card-iso)
  qed
qed

lemma closure-iso:
  shows closure (op ‘ g ‘ F) = op ‘ g ‘ (closure F)
proof (safe)
  fix x
  assume x ∈ closure (op ‘ g ‘ F)
  then obtain F' where ⋃ F' = x F' ≠ {} and *: F' ⊆ op ‘ g ‘ F
    unfolding closure-def

```

```

    by auto
  from * obtain  $F''$  where  $F'' \subseteq F$   $F' = \text{op } 'g' 'F''$ 
    by (metis subset-image-iff)
  moreover
  hence  $F'' \neq \{\}$   $g' (\bigcup F'') = x$ 
    using  $\langle F' \neq \{\} \rangle \langle \bigcup F' = x \rangle$ 
    by auto
  ultimately
  have  $\bigcup F'' \in \text{closure } F$   $g' (\bigcup F'') = x$ 
    by (auto simp add: closure-def)
  thus  $x \in \text{op } 'g' ' \text{closure } F$ 
    by auto
next
fix  $x$ 
assume  $x \in \text{closure } F$ 
then obtain  $F'$  where  $F' \subseteq F$   $F' \neq \{\}$   $x = \bigcup F'$ 
  by (auto simp add: closure-def)
hence  $\text{op } 'g' 'F' \subseteq (\text{op } 'g' 'F) \text{ op } 'g' 'F' \neq \{\}$   $g' x = \bigcup (\text{op } 'g' 'F')$ 
  by auto
thus  $g' x \in \text{closure } (\text{op } 'g' 'F)$ 
  unfolding closure-def
  by blast
qed

```

lemma *FC-family-iso*:

```

  fixes  $f :: 'a \Rightarrow \text{nat}$ 
  assumes inj-on  $f$   $(\bigcup Fc)$  FC-family  $(\text{op } 'f' 'Fc)$ 
  shows FC-family  $Fc$ 
using assms
unfolding FC-family-def
proof (safe)
  fix  $F$ 
  assume *:  $\forall F. \text{op } 'f' 'Fc \subseteq F \wedge \text{finite-union-closed } F \longrightarrow (\exists a \in (\bigcup (\text{op } 'f' 'Fc)). \text{card } F \leq 2 * \text{count } a F)$ 
  assume inj-on  $f$   $(\bigcup Fc)$   $Fc \subseteq F$  union-closed  $F$  finite  $(\bigcup F)$ 
  have  $\bigcup Fc \subseteq \bigcup F$  finite  $(\bigcup Fc)$   $\bigcup Fc \cup \bigcup F = \bigcup F$ 
    using  $\langle Fc \subseteq F \rangle \langle \text{finite } (\bigcup F) \rangle$  finite-subset[of  $\bigcup Fc \cup F$ ]
    by auto
  then obtain  $f'$  where  $\forall x \in \bigcup Fc. f' x = f x$  inj-on  $f'$   $(\bigcup F)$ 
    using bij-betw-inj-extend[of  $f \bigcup Fc$   $f' (\bigcup Fc) \bigcup F - \bigcup Fc$ ]
    using  $\langle \text{inj-on } f (\bigcup Fc) \rangle \langle \text{finite } (\bigcup F) \rangle \langle Fc \subseteq F \rangle$ 
    unfolding bij-betw-def
    by auto
  let  $?F' = \text{op } 'f' 'F$ 
  have  $\text{op } 'f' 'Fc \subseteq ?F'$ 
  proof (safe)
    fix  $x$ 
    assume  $x \in Fc$ 

```



```

show  $f \text{ ' } x \in op \text{ ' } f' \text{ ' } F$ 
proof (rule rev-image-eqI)
  show  $x \in F$  using  $\langle x \in Fc \rangle \langle Fc \subseteq F \rangle$ 
  by auto
next
  show  $f \text{ ' } x = f' \text{ ' } x$ 
  using  $\langle x \in Fc \rangle \langle \forall x \in \bigcup Fc. f' x = f x \rangle$ 
  by force
qed
qed
moreover
have finite-union-closed ?F'
  using finite-union-closed-iso
  using  $\langle inj\text{-on } f' (\bigcup F) \rangle \langle finite (\bigcup F) \rangle \langle union\text{-closed } F \rangle$ 
  by auto
ultimately
have  $\exists a \in (\bigcup (op \text{ ' } f' Fc)). card \text{ ?F}' \leq 2 * count a \text{ ?F}'$ 
  using *
  by blast
then obtain  $a$  where  $a \in \bigcup Fc$   $a \in \bigcup F$   $card \text{ ?F}' \leq 2 * count (f a) \text{ ?F}'$ 
  using  $\langle \bigcup Fc \subseteq \bigcup F \rangle$ 
  by blast
moreover
have  $f a = f' a$ 
  using  $\langle a \in \bigcup Fc \rangle \langle \forall x \in \bigcup Fc. f' x = f x \rangle$ 
  by auto
ultimately
show  $\exists a \in \bigcup Fc. card F \leq 2 * count a F$ 
  using count-iso[of f' F a] card-iso[of f' F]  $\langle inj\text{-on } f' (\bigcup F) \rangle$ 
  by (rule-tac  $x=a$  in bexI) auto
qed

```

12.2 Injective embedding

definition *inj-embed* **where**

[simp]: $inj\text{-embed } F F' \longleftrightarrow (\exists f. inj\text{-on } f (\bigcup F) \wedge F' = (op \text{ ' } f) \text{ ' } F)$

lemma *inj-embed-refl* [simp]:

shows *inj-embed* $F F$

unfolding *inj-embed-def*

by (rule-tac $x=id$ **in** *exI*) auto

lemma *inj-embed-sym*:

assumes *inj-embed* $F F'$

shows *inj-embed* $F' F$

proof—

obtain f **where** *: $inj\text{-on } f (\bigcup F) F' = (op \text{ ' } f) \text{ ' } F$

using *assms*

by auto

```

let ?f' = inv-into ( $\bigcup F$ ) f
show ?thesis
  unfolding inj-embed-def
proof (rule-tac x=?f' in exI, rule conjI)
  show inj-on ?f' ( $\bigcup F'$ )
    apply (rule inj-on-inv-into)
    using *(2)
    by auto
next
show  $F = \text{op } \langle ?f' \rangle F'$ 
proof (safe)
  fix x
  assume  $x \in F$ 
  hence  $x \subseteq \bigcup F$ 
    by auto
  show  $x \in \text{op } \langle ?f' \rangle F'$ 
  proof (rule-tac rev-image-eqI[of f' x])
    show  $f' x \in F'$ 
      using *(2)  $\langle x \in F \rangle$ 
      by simp
  next
  show  $x = \text{inv-into } (\bigcup F) f' f' x$ 
    using inv-into-image-cancel[of f'  $\bigcup F$  x]
    using *(1)  $\langle x \subseteq \bigcup F \rangle$ 
    by auto
qed
next
fix x
assume  $x \in F'$ 
then obtain y where  $x = f' y$   $y \in F$   $y \subseteq \bigcup F$ 
  using *(2)
  by auto
thus ?f'  $\langle x \in F$ 
  using inv-into-image-cancel[of f'  $\bigcup F$  y] *(1)
  by auto
qed
qed
qed

lemma inj-embed-trans:
  assumes inj-embed  $F F'$  and inj-embed  $F' F''$ 
  shows inj-embed  $F F''$ 
proof-
  from assms
  obtain f g where inj-on f ( $\bigcup F$ )  $F' = \text{op } \langle f \rangle F$  inj-on g ( $\bigcup F'$ )  $F'' = \text{op } \langle g \rangle F'$ 
  unfolding inj-embed-def
  by auto
thus ?thesis

```

```

    unfolding inj-embed-def
    by (rule-tac x=g ∘ f in exI) (auto simp add: comp-def inj-on-def)
qed

```

12.3 Isomorphism definition

There is a bijective mapping between equivalent families.

definition *iso* **where**

```

[simp]: iso F F' ⟷ (∃ h. bij-betw h (⋃ F) (⋃ F') ∧ F' = (op ' h) ' F)

```

lemma *inj-embed-iso*:

```

shows inj-embed F F' ⟷ iso F F'

```

proof

```

assume inj-embed F F'

```

```

then obtain f where *: inj-on f (⋃ F) F' = (op ' f) ' F

```

```

  by auto

```

```

show iso F F'

```

```

  unfolding iso-def bij-betw-def

```

```

  apply (rule-tac x=f in exI, rule conjI)

```

```

  using *

```

```

  by auto

```

next

```

assume iso F F'

```

```

thus inj-embed F F'

```

```

  unfolding iso-def inj-embed-def bij-betw-def

```

```

  by auto

```

qed

lemma *iso-refl*:

```

shows iso F F

```

```

using inj-embed-iso[of F F] inj-embed-refl

```

```

by blast

```

lemma *iso-sym*:

```

assumes iso F F'

```

```

shows iso F' F

```

```

using assms

```

```

using inj-embed-iso[of F F'] inj-embed-iso[of F' F] inj-embed-sym

```

```

by blast

```

lemma *iso-trans*:

```

assumes iso F F' and iso F' F''

```

```

shows iso F F''

```

```

using assms

```

```

using inj-embed-iso[of F F'] inj-embed-iso[of F' F''] inj-embed-iso[of F F''] inj-embed-trans[of
F F' F'']

```

```

by blast

```

lemma *iso-finite*:

```

    assumes iso  $F F'$ 
    shows  $\text{finite } F \longleftrightarrow \text{finite } F'$ 
using assms
unfolding iso-def
by (metis bij-betw-imp-inj-on finite-iso)

lemma iso-insert:
  assumes inj-on  $f (\bigcup F \cup A)$ 
  shows iso  $(F \cup \{A\}) ((\text{op } f F) \cup \{f A\})$ 
     $\text{card } (f A) = \text{card } A$ 
     $f A \subseteq f (\bigcup F \cup A)$ 
     $A \notin F \implies f A \notin \text{op } f F$ 
proof -
  show iso  $(F \cup \{A\}) ((\text{op } f F) \cup \{f A\})$ 
    unfolding iso-def
  proof (rule-tac  $x=f$  in exI, rule conjI)
    show bij-betw  $f (\bigcup (F \cup \{A\})) (\bigcup ((\text{op } f F) \cup \{f A\}))$ 
      using assms(1)
      unfolding bij-betw-def
      by (auto simp add: Un-commute)
    next
      show  $\text{op } f F \cup \{f A\} = \text{op } f (F \cup \{A\})$ 
        by simp
    qed
  next
    show  $\text{card } (f A) = \text{card } A$ 
      using assms(1)
      using card-image[of  $f A$ ]
      by (auto simp add: inj-on-def)
  next
    show  $f A \subseteq f (\bigcup F \cup A)$ 
      by auto
  next
    assume  $A \notin F$ 
    show  $f A \notin \text{op } f F$ 
    proof (rule ccontr)
      assume  $\neg ?thesis$ 
      then obtain  $A'$  where  $f A = f A' A' \in F$ 
        by auto
      moreover
      have inj-on  $f (A \cup A')$ 
        using assms(1)  $\langle A' \in F \rangle$ 
        unfolding inj-on-def
        by auto
      ultimately
      have  $A = A'$ 
        using inj-on-Un-image-eq-iff[of  $f A A'$ ]
        by simp
      thus False

```

```

    using ⟨A ∉ F⟩ ⟨A' ∈ F⟩
    by simp
  qed
qed

```

12.4 Iso-representing collections of families

FF' iso-represents FF if an isomorphic image of each element of FF is in FF'

definition *iso-represents* **where**

iso-represents $FF' FF \longleftrightarrow (\forall F \in FF. \exists F' \in FF'. \text{iso } F F')$

abbreviation *iso-representing-subset* **where**

iso-representing-subset $FF' FF \equiv \text{iso-represents } FF' FF \wedge FF' \subseteq FF$

end

12.5 Non-isomorphic families of sets

theory *NonIsomorphicFamilies*

imports *Main*

IsomorphicFamilies

FamilyImpl

begin

In this section we define a combinatorial algorithm (implemented by a function *non-isomorphic-families*) that computes non isomorphic families of sets. Two uniform families (over the same domain) are called isomorphic if one can be obtained from the other by permuting elements of the domain. This function is used to remove symmetric cases in verification of FC families.

locale *Mapping* =

fixes *to-fun* :: $'m \Rightarrow ('a \Rightarrow 'a)$

locale *SetPermutations* = *SetImpl to-set inv* + *Mapping to-fun* **for**

inv :: $'s::\text{linorder} \Rightarrow \text{bool}$ **and** *to-set* :: $'s::\text{linorder} \Rightarrow 'a \text{ set}$ **and** *to-fun* :: $'m \Rightarrow ('a \Rightarrow 'a)$ +

fixes *permute-set* :: $'m \Rightarrow 's \Rightarrow 's$

assumes *permute-set-set*: $\text{to-set } (\text{permute-set } p \ s) = (\text{to-fun } p) \circ (\text{to-set } s)$

assumes *permute-set-inv*: $\llbracket \text{inj-on } (\text{to-fun } p) (\text{to-set } s); \text{inv } s \rrbracket \implies \text{inv } (\text{permute-set } p \ s)$

begin

definition *permute-family* :: $'m \Rightarrow 's \text{ list} \Rightarrow 's \text{ list}$ **where**

permute-family $p \ F = \text{map } (\text{permute-set } p) \ F$

lemma *permute-family-set*:

shows $f\text{-to-set } (\text{permute-family } p \ F) = \text{op } \circ (\text{to-fun } p) \circ f\text{-to-set } F$

unfolding *permute-family-def*

using *permute-set-set*
by *force*

lemma *permute-family-inj-embed*:
assumes *inj-on* (*to-fun p*) (\bigcup *f-to-set F*)
shows *inj-embed* (*f-to-set F*) (*f-to-set* (*permute-family p F*))
using *assms permute-family-set*
unfolding *inj-embed-def*
by (*rule-tac x=to-fun p in exI*) *auto*

function *non-isomorphic-families-aux* **where**
non-isomorphic-families-aux perms fams res =
(*case fams of*
 $\square \Rightarrow res$
 $| h \# t \Rightarrow$
 $let\ hp = remdups\ (h \# map\ (\lambda p.\ permute-family\ p\ h)\ perms)\ in$
 $non-isomorphic-families-aux\ perms\ (filter\ (\lambda l.\ sort\ l\ \notin\ set\ (map\ sort\ hp))\ fams)$
(*h # res*))
by *pat-completeness auto*
termination
by (*relation measure* ($\lambda (p, f, r).\ length\ f$)) *auto*
declare *non-isomorphic-families-aux.simps* [*simp del*]

definition *non-isomorphic-families* **where**
[*simp*]: *non-isomorphic-families perms fams = non-isomorphic-families-aux perms fams* \square

lemma *non-isomorphic-families-aux-res-mono*:
shows $set\ res \subseteq set\ (non-isomorphic-families-aux\ perms\ fams\ res)$
proof (*induct perms fams res rule: non-isomorphic-families-aux.induct*)
case (*1 perms fams res*)
show *?case*
proof (*cases fams*)
case *Nil*
thus *?thesis*
by (*simp add: non-isomorphic-families-aux.simps*)
next
case (*Cons a fams'*)
thus *?thesis*
using *non-isomorphic-families-aux.simps* [*of perms fams res*]
using *1* [*of a fams' remdups (a # FamilyImpl.map (\p. permute-family p a) perms)*]
by (*auto simp add: Let-def*)
qed
qed

lemma *non-isomorphic-families-aux-subset*:
assumes $F \supseteq set\ res$ $F \supseteq set\ fams$
shows $F \supseteq set\ (non-isomorphic-families-aux\ perms\ fams\ res)$

```

using assms
proof (induct perms fams res rule: non-isomorphic-families-aux.induct)
  case (1 perms fams res)
  show ?case
  proof (cases fams)
    case Nil
    thus ?thesis
    using 1(2)
    by (simp add: non-isomorphic-families-aux.simps)
  next
  case (Cons a fams')
  let ?hp = remdups (a # FamilyImpl.map (λp. permute-family p a) perms)
  have set (let hp = ?hp
    in non-isomorphic-families-aux perms [l ← fams . sort l ∉ set (map sort
hp])
     $(a \# res)) \subseteq F$ 
  unfolding Let-def
  apply (rule 1(1)[of a fams'])
  using 1(2) 1(3) Cons
  by auto
  thus ?thesis
  apply (subst non-isomorphic-families-aux.simps[of perms fams res])
  using Cons
  by simp
qed
qed

lemma non-isomorphic-families-subset:
  shows set (non-isomorphic-families perms F) ⊆ set F
unfolding non-isomorphic-families-def
by (simp add: non-isomorphic-families-aux-subset)

lemma non-isomorphic-families-aux:
  assumes  $\forall p \in \text{set perms}. \forall F \in \text{set fams}. \text{inj-on } (\text{to-fun } p) (\bigcup f\text{-to-set } F)$ 
  shows  $\forall F \in \text{set fams}. \exists F' \in \text{set } (\text{non-isomorphic-families-aux perms fams res}). \text{inj-embed } (f\text{-to-set } F) (f\text{-to-set } F')$ 
using assms
proof (induct perms fams res rule: non-isomorphic-families-aux.induct)
  case (1 perms fams res)
  show ?case
  proof (cases fams)
    case Nil
    thus ?thesis
    by simp
  next
  case (Cons F fams')

  let ?nefams = non-isomorphic-families-aux perms fams res
  let ?hp = remdups (F # (map (λ p. permute-family p F) perms))

```

```

let ?filt = filter (λ l. sort l ∉ set (map sort ?hp)) fams
let ?nef = non-isomorphic-families-aux perms ?filt (F # res)

have ?nefams = ?nef
  using Cons
  using non-isomorphic-families-aux.simps[of perms fams res]
  by simp

show ?thesis
proof (rule ballI)
  fix Fa
  assume Fa ∈ set fams
  show ∃ a ∈ set ?nefams. inj-embed (f-to-set Fa) (f-to-set a)
  proof (cases sort Fa = sort F)
    case True
    show ?thesis
    proof (rule-tac x=F in bexI)
      show F ∈ set (non-isomorphic-families-aux perms fams res)
      using ⟨?nefams = ?nef⟩
      using non-isomorphic-families-aux-res-mono[of F # res perms ?filt]
      by auto
    next
      show inj-embed (f-to-set Fa) (f-to-set F)
      using ⟨sort Fa = sort F⟩
      by (metis f-to-set-def image-set inj-embed-refl set-sort)
    qed
  next
    case False
    hence Fa ∈ set fams'
    using Cons ⟨Fa ∈ set fams⟩
    by auto
    show ?thesis
    proof (cases sort Fa ∈ set (map sort ?hp))
      case False
      thus ?thesis
      using Cons ⟨Fa ∈ set fams'⟩
      using 1(1)[of F fams' ?hp] 1(2) Cons
      using ⟨?nefams = ?nef⟩
      by auto
    next
      case True
      then obtain p where p ∈ set (perms) sort Fa = sort (permute-family p
F) inj-on (to-fun p) (⋃ f-to-set F)
      using ⟨sort Fa ≠ sort F⟩ 1(2) Cons
      by (auto split: if-split-asm)
      hence inj-embed (f-to-set F) (f-to-set Fa)
      using permute-family-inj-embed[of p F]
      by (metis f-to-set-def image-set set-sort)
    moreover

```



```

    have  $F \in \text{set } ?\text{nefams}$ 
      using  $\langle ?\text{nefams} = ?\text{nef} \rangle$ 
      using  $\text{non-isomorphic-families-aux-res-mono}[of\ F\ \# \text{ res perms } ?\text{filt}]$ 
      by simp
    ultimately
    show  $?thesis$ 
      using  $\text{inj-embed-sym}[of\ f\text{-to-set } F\ f\text{-to-set } Fa]$ 
      by auto
  qed
qed
qed
qed
qed

lemma non-isomorphic-families':
  assumes  $\forall p \in \text{set perms}. \forall F \in \text{set fams}. \text{inj-on } (to\text{-fun } p) (\bigcup f\text{-to-set } F)$ 
  shows  $\forall F \in \text{set fams}. \exists F' \in \text{set } (non\text{-isomorphic-families perms fams}).$ 
   $\text{inj-embed } (f\text{-to-set } F) (f\text{-to-set } F')$ 
  using assms
  unfolding non-isomorphic-families-def
  using non-isomorphic-families-aux
  by simp

lemma generating-subset-non-isomorphic-families:
  assumes  $\forall p \in \text{set perms}. \forall F \in \text{set } FF. \text{inj-on } (to\text{-fun } p) (\bigcup f\text{-to-set } F)$ 
   $\text{iso-representing-subset } (\bigcirc FF) FF'$ 
  shows  $\text{iso-representing-subset } (\bigcirc (non\text{-isomorphic-families perms } FF)) FF'$ 
proof
  show  $\bigcirc (non\text{-isomorphic-families perms } FF) \subseteq FF'$ 
    using  $\langle \text{iso-representing-subset } (\bigcirc FF) FF' \rangle \text{non-isomorphic-families-subset}[of\ perms]$ 
    by auto
next
  show  $\text{iso-represents } (\bigcirc (non\text{-isomorphic-families perms } FF)) FF'$ 
    unfolding iso-represents-def
  proof
    fix  $F'$ 
    assume  $F' \in FF'$ 
    then obtain  $F$  where  $F \in (\bigcirc FF) \text{ iso } F'$ 
      using  $\langle \text{iso-representing-subset } (\bigcirc FF) FF' \rangle$ 
      unfolding iso-represents-def
      by blast
    then obtain  $F_l$  where  $F_l \in \text{set } FF\ F = f\text{-to-set } F_l$ 
      by auto
    then obtain  $F''$  where  $F'' \in \bigcirc (non\text{-isomorphic-families perms } FF) \text{ inj-embed } F\ F''$ 
      using non-isomorphic-families'[of perms FF, OF assms(1)]
      by auto (metis f-to-set-def imageI image-set)
    thus  $\text{Bex } (\bigcirc (non\text{-isomorphic-families perms } FF)) (\text{iso } F')$ 

```

```

    using ⟨iso F' F⟩ iso-trans inj-embed-iso
    by blast
qed
qed
end

end

```

12.5.1 Implementation by sets represented by (sorted and distinct) lists

```

theory NonIsomorphicFamiliesImpl
imports NonIsomorphicFamilies
        Combinatorics
        HOL-Library.List-lexord
begin

```

```

definition list-to-fun :: nat list ⇒ (nat ⇒ nat) where
  list-to-fun l = (λ n. l ! n)

```

```

definition permute-set-l :: nat list ⇒ nat list ⇒ nat list where
  permute-set-l p A = sort (map (list-to-fun p) A)

```

```

global-interpretation SetPermutations-lists: SetPermutations sd set list-to-fun
permute-set-l

```

```

  defines
    non-isomorphic-families-aux-l = SetPermutations-lists.non-isomorphic-families-aux
and
    permute-family-l = SetPermutations-lists.permute-family and
    non-isomorphic-families-l = SetPermutations-lists.non-isomorphic-families
proof (unfold-locales)
  fix p s
  show set (permute-set-l p s) = list-to-fun p ‘ set s
    unfolding permute-set-l-def
    by simp

```

```

next
  fix s::nat list and p
  assume sorted s ∧ distinct s inj-on (list-to-fun p) (set s)
  thus sorted (permute-set-l p s) ∧ distinct (permute-set-l p s)
    by (auto simp add: permute-set-l-def list-to-fun-def distinct-map)
qed

```

```

lemma nat-list-to-fun-inj-on':
  assumes distinct p p <~> [0.. $n$ ] X ⊆ {0.. $\text{length } p$ }
  shows inj-on (list-to-fun p) X
  using assms

```

```

using distinct-conv-nth[of p]
apply (auto simp add: list-to-fun-def inj-on-def)
apply (rule ccontr)
apply (erule-tac x=x in allE, drule mp, force)
apply (erule-tac x=y in allE, drule mp, force)
by simp

lemma nat-list-to-fun-inj-on:
  assumes
     $\forall p \in \text{set perms. } p <\sim\sim> [0..<n]$  and
     $\forall F \in \text{set fams. } \bigcup f\text{-to-set-l } F \subseteq \{0..<n\}$ 
  shows  $\forall p \in \text{set perms. } \forall F \in \text{set fams. } \text{inj-on } (\text{list-to-fun } p) (\bigcup f\text{-to-set-l } F)$ 
unfolding inj-on-def list-to-fun-def SetImpl-lists.f-to-set-def
proof (auto)
  fix p F x y Ax Ay
  assume p  $\in \text{set perms}$  and
     $*$ :  $F \in \text{set fams } Ax \in \text{set } F \ x \in \text{set } Ax \ Ay \in \text{set } F \ y \in \text{set } Ay$  and
     $p ! x = p ! y$ 
  have distinct p length p = n
    using  $\langle p \in \text{set perms} \rangle \text{ assms}(1)$ 
    using perm-distinct-iff[of p [0..<n]] perm-length[of p [0..<n]]
    by auto
  moreover
  have  $x < n \ y < n$ 
    using  $*$  assms(2)
    unfolding SetImpl-lists.f-to-set-def
    by force+
  ultimately
  show  $x = y$ 
    using  $\langle p ! x = p ! y \rangle$ 
    using nth-eq-iff-index-eq [of p x y] assms(2)  $*$ 
    by simp
qed

lemma iso-permute-family-l:
  assumes iso (f-to-set-l F) (f-to-set-l F') sdf F' sdf F
  assumes dm F n dm F' n perms = permute [0..<n]
  shows  $\exists p \in \text{set perms. } \text{set } F' = \text{set } (\text{permute-family-l } p F)$ 
using assms
proof –
  from assms obtain f where bij: bij-betw f ( $\bigcup f\text{-to-set-l } F$ ) ( $\bigcup f\text{-to-set-l } F'$ )
and
     $*$ :  $f\text{-to-set-l } F' = \text{op } 'f' \text{ } f\text{-to-set-l } F$ 
    unfolding iso-def
    by auto
  have  $\text{card } (\{0..<n\} - (\bigcup f\text{-to-set-l } F)) = \text{card } (\{0..<n\} - (\bigcup f\text{-to-set-l } F'))$ 
    using bij  $\langle \text{dm } F \ n \rangle \langle \text{dm } F' \ n \rangle$ 
    by (metis bij-betw-same-card card-Diff-subset finite-atLeastLessThan finite-subset)
  then obtain f' where bij: bij-betw f'  $\{0..<n\}$   $\{0..<n\} \ \forall x \in \bigcup f\text{-to-set-l } F$ .

```

```

f' x = f x
  using bij-betw-extend[OF bij, of {0.. $n$ } - (⋃ f-to-set-l F) {0.. $n$ } - (⋃
f-to-set-l F')]
  using ⟨dm F' n⟩ ⟨dm F n⟩
  by auto (metis subset-Un-eq)
have *: f-to-set-l F' = op 'f' 'f-to-set-l F
  using ⟨∀ x ∈ ⋃ f-to-set-l F. f' x = f x⟩ *
  by (metis map-fam-cong)
let ?perm = map f' [0.. $n$ ]
have ?perm <~~> [0.. $n$ ]
  unfolding mset-eq-perm[symmetric]
proof (subst set-eq-iff-mset-eq-distinct[symmetric])
  show distinct (map f' [0.. $n$ ])
    using distinct-map[of f' [0.. $n$ ]]
    using bij
    by (auto simp add: bij-betw-def)
next
  show distinct [0.. $n$ ]
    by simp
next
  show set (map f' [0.. $n$ ]) = set [0.. $n$ ]
    using bij
    unfolding bij-betw-def
    by auto
qed
show ?thesis
proof (rule-tac x=?perm in bexI)
  show ?perm ∈ set perms
    using ⟨perms = permute [0.. $n$ ]⟩ ⟨?perm <~~> [0.. $n$ ]⟩
    using permute-isPermutation[of ?perm [0.. $n$ ]]
    by simp
next
  show set F' = set (permute-family-l ?perm F)
proof (safe)
  fix x
  assume x ∈ set F'
  then obtain y where y ∈ set F and **: set x = f' ' set y
    using *
    by (auto, smt image-eqI image-iff)
  have set y ⊆ {0.. $n$ }
    using ⟨y ∈ set F⟩ ⟨dm F n⟩
    by auto
  have x = permute-set-l ?perm y
proof-
  have map (op ! (FamilyImpl.map f' [0.. $n$ ])) y = map f' y
    using ⟨y ∈ set F⟩ ⟨dm F n⟩
    by auto
  moreover
  have sd x distinct y

```

```

    using ⟨y ∈ set F⟩ ⟨sdf F⟩ ⟨x ∈ set F'⟩ ⟨sdf F'⟩
    by auto
  hence x = sort (map f' y)
    using ** ⟨set y ⊆ {0..<n}⟩
    using sorted-distinct-set-unique[of x sort (map f' y)]
    using bij distinct-map[of f' y]
    using subset-inj-on[of f' {0..<n} set y]
    unfolding bij-betw-def
    by simp
  ultimately
  show ?thesis
    unfolding permute-set-l-def list-to-fun-def
    by metis
qed
thus x ∈ set (permute-family-l ?perm F)
  using ⟨y ∈ set F⟩
  unfolding SetPermutations-lists.permute-family-def
  by auto
next
fix x
assume x ∈ set (permute-family-l ?perm F)
then obtain y where x = permute-set-l ?perm y y ∈ set F
  unfolding SetPermutations-lists.permute-family-def
  by auto
hence x = sort (map f' y)
proof-
  have map (op ! (FamilyImpl.map f' [0..<n])) y = map f' y
    using ⟨y ∈ set F⟩ ⟨dm F n⟩
    by auto
  thus ?thesis
    using ⟨x = permute-set-l ?perm y⟩
    unfolding permute-set-l-def list-to-fun-def
    by metis
qed
hence set x ∈ set (map set F')
  using ⟨y ∈ set F⟩ *
  by auto
moreover
have distinct y set y ⊆ {0..<n}
  using ⟨y ∈ set F⟩ ⟨sdf F⟩ ⟨dm F n⟩
  by auto
hence sd x
  using ⟨x = sort (map f' y)⟩ ⟨y ∈ set F⟩ distinct-map[of f' y]
  using subset-inj-on[of f' {0..<n} set y]
  using bij
  unfolding bij-betw-def
  by auto
ultimately
show x ∈ set F'

```

```

    using ⟨sdf F'⟩
    by (metis SetImpl-lists.f-to-set-def SetImpl-lists.set-set)
  qed
qed
qed

```

```

lemmas non-isomorphic-families-soundness = SetPermutations-lists.non-isomorphic-families'[OF
nat-list-to-fun-inj-on]
lemmas permute-set-inv-l = SetPermutations-lists.permute-set-inv[OF nat-list-to-fun-inj-on]
lemmas iso-representing-subset-non-isomorphic-families-l = SetPermutations-lists.generating-subset-non-ison
nat-list-to-fun-inj-on]

end

```

13 Expressible sets and irreducible families

```

theory IrreducibleFamilies
imports UnionClosed
begin

```

```

definition expressible where
  expressible A F  $\longleftrightarrow (\exists F'. F' \subseteq F \wedge F' \neq \{\} \wedge A = \bigcup F')$ 

```

```

lemma expressible A F  $\longleftrightarrow (\exists F' \subseteq F. F' \neq \{\} \wedge (\forall A' \in F'. A' \subseteq A) \wedge \bigcup F' = A)$ 
unfolding expressible-def
by auto

```

```

lemma expressible-closure1:
  assumes expressible A F
  shows A ∈ closure F
using assms
unfolding expressible-def closure-def
by auto

```

```

lemma expressible-closure2:
  assumes expressible A F
  shows (op ∪ A) ' (closure F) ⊆ closure F
proof(safe)
  fix x
  from ⟨expressible A F⟩ obtain F' where *: F' ∈ Pow F A = ∪ F'
  by (auto simp add: expressible-def)
  assume x ∈ closure F
  then obtain F'' where F'' ∈ Pow F - {\{\}} x = ∪ F''
  unfolding closure-def
  by auto
  have F'' ∪ F' ∈ Pow F - {\{\}}
  using ⟨F' ∈ Pow F⟩ ⟨F'' ∈ Pow F - {\{\}}⟩

```

```

    by auto
  moreover
  have  $A \cup x = \bigcup (F'' \cup F')$ 
    using  $\langle A = \bigcup F' \rangle \langle x = \bigcup F'' \rangle$ 
    by auto
  ultimately
  show  $A \cup x \in \text{closure } F$ 
    unfolding closure-def
    by (metis image-iff)
qed

```

```

lemma expressible-closure:
  assumes expressible A F finite F
  shows closure (F  $\cup$  {A}) = closure F
using assms
using closure-insert[of F A]
using expressible-closure1[of A F]
using expressible-closure2[of A F]
by auto

```

13.1 Irreducible families

```

definition reducible :: 'a set set  $\Rightarrow$  bool where
  reducible F  $\longleftrightarrow$ 
    ( $\exists A F'. A \in F \wedge F' \subseteq F \wedge F' \neq \{\}$   $\wedge A \notin F' \wedge A = \bigcup F'$ )

```

```

abbreviation irreducible :: 'a set set  $\Rightarrow$  bool where
  irreducible F  $\equiv \neg$  reducible F

```

```

lemma reducible F  $\longleftrightarrow$  ( $\exists A \in F. \text{expressible } A (F - \{A\})$ )
unfolding reducible-def expressible-def
by auto

```

13.2 Removing all expressible sets

```

function (domintros) remove-expressible where
  remove-expressible F =
    (if ( $\exists A \in F. \text{expressible } A (F - \{A\})$ ) then
      let A = SOME A. A  $\in$  F  $\wedge$  expressible A (F - {A})
      in remove-expressible (F - {A})
    else
      F)
by pat-completeness auto

```

```

lemma remove-expressible-dom:
  assumes finite F
  shows remove-expressible-dom F
using assms
proof (induct F rule: wf-induct[of measure card])
  show wf (measure card)

```

```

    by auto
next
  fix x::'a set set
  assume finite x and *:  $\forall y. (y, x) \in \text{measure card} \longrightarrow \text{finite } y \longrightarrow \text{remove-expressible-dom } y$ 
  show remove-expressible-dom x
  proof (rule remove-expressible.domintros)
    fix A
    assume ++:  $A \in x \text{ expressible } A (x - \{A\})$ 
    let ?A = SOME A.  $A \in x \wedge \text{expressible } A (x - \{A\})$ 
    let ?y =  $x - \{?A\}$ 
    show remove-expressible-dom ?y
    proof (rule *[rule-format])
      show  $(x - \{?A\}, x) \in \text{measure card}$ 
      proof (simp, rule card-Diff1-less)
        show finite x by fact
      next
        show ?A  $\in x$ 
        using ++
        by (metis (lifting) someI-ex)
      qed
    next
      show finite  $(x - \{?A\})$ 
      using ⟨finite x⟩
      by auto
    qed
  qed
qed

lemma remove-expressible-subset:
  assumes finite F
  shows remove-expressible  $F \subseteq F$ 
proof-
  have remove-expressible-dom F
    using ⟨finite F⟩
    by (rule remove-expressible-dom)
  thus ?thesis
    using assms
  proof (induct F rule: remove-expressible.pinduct)
    case (1 F)
    show ?case
    proof (cases  $\exists A \in F. \text{expressible } A (F - \{A\})$ )
      case False
      thus ?thesis
        using ⟨remove-expressible-dom F⟩ remove-expressible.psimps[of F]
        by simp
    next
      case True
      thus ?thesis

```



```

      using ⟨remove-expressible-dom F⟩ remove-expressible.psimps[of F]
      using 1
      unfolding Let-def
      by auto
    qed
  qed
qed

lemma remove-expressible-closure:
  assumes finite F
  shows closure F = closure (remove-expressible F)
proof-
  have remove-expressible-dom F
  using ⟨finite F⟩
  by (rule remove-expressible-dom)
  thus ?thesis
  using assms
proof (induct F rule: remove-expressible.pinduct)
  case (1 F)
  show ?case
  proof (cases  $\exists A \in F. \text{expressible } A (F - \{A\})$ )
    case False
    thus ?thesis
      using ⟨remove-expressible-dom F⟩ remove-expressible.psimps[of F]
      by simp
  next
    case True
    let ?A = SOME A. A ∈ F ∧ expressible A (F - {A})
    have *: closure (F - {?A}) = closure (remove-expressible (F - {?A}))
      using 1(2)[of ?A] 1(3) True
      by simp
    moreover
    have closure F = closure (F - {SOME A. A ∈ F ∧ expressible A (F - {A})})
    proof-
      have ?A ∈ F ∧ expressible ?A (F - {?A})
      using True
      by (metis (lifting) someI-ex)
      thus ?thesis
      using expressible-closure[of ?A F - {?A}] 1(3)
      by simp (metis (lifting) insert-absorb)
    qed
  thus ?thesis
    using ⟨remove-expressible-dom F⟩ remove-expressible.psimps[of F] True *
    by simp
  qed
qed
qed
qed

```

```

lemma remove-expressible-irreducible:
  assumes finite F
  shows irreducible (remove-expressible F)
proof -
  have let F' = remove-expressible F in
     $\neg (\exists A \in F'. \text{expressible } A (F' - \{A\}))$ 
  proof -
    have remove-expressible-dom F
    using ⟨finite F⟩
    by (rule remove-expressible-dom)
    thus ?thesis
    using assms
  proof (induct F rule: remove-expressible.pinduct)
    case (1 F)
    show ?case
    proof (cases  $\exists A \in F. \text{expressible } A (F - \{A\})$ )
      case False
      thus ?thesis
      using ⟨remove-expressible-dom F⟩ remove-expressible.psimps[of F]
      by simp
    next
      case True
      thus ?thesis
      using ⟨remove-expressible-dom F⟩ remove-expressible.psimps[of F]
      using 1
      unfolding Let-def
      by simp
    qed
  qed
  thus ?thesis
  using assms
  unfolding Let-def
  unfolding expressible-def reducible-def
  by auto
qed

```

```

lemma ex-irreducible-closure:
  fixes F
  assumes finite F
  shows  $\exists F' \subseteq F. \text{irreducible } F' \wedge \text{closure } F' = \text{closure } F$ 
  using assms
  using remove-expressible-closure[OF assms]
  using remove-expressible-subset[OF assms]
  using remove-expressible-irreducible[OF assms]
  by auto

```

13.3 Uniqueness of irreducible subfamily for the given closure

```

lemma irreducible-closure':
  assumes irreducible F' and closure F = closure F'
  shows F'  $\subseteq$  F
proof (rule ccontr)
  assume  $\neg$  ?thesis
  then obtain A where A  $\in$  F' A  $\notin$  F
    by auto
  have A  $\in$  closure F
    using  $\langle$ closure F = closure F' $\rangle$   $\langle$ A  $\in$  F' $\rangle$ 
    unfolding closure-def
    by auto
  then obtain Fa where Fa  $\subseteq$  F Fa  $\neq$  {}  $\cup$  Fa = A
    by (auto simp add: closure-def)
  let ?F1 = Fa - F' and ?F2 = Fa  $\cap$  F'
  have ?F1  $\subseteq$  closure F
    using  $\langle$ Fa  $\subseteq$  F $\rangle$ 
    by (auto simp add: closure-def)
  hence  $\forall$  A'  $\in$  ?F1. A'  $\in$  closure F'
    using  $\langle$ closure F = closure F' $\rangle$   $\langle$ A  $\notin$  F $\rangle$   $\langle$ Fa  $\subseteq$  F $\rangle$ 
    by (auto simp add: closure-def)

  let ?f =  $\lambda$  A' F''. F''  $\subseteq$  F'  $\wedge$  F''  $\neq$  {}  $\wedge$  A' =  $\bigcup$  F''

  have *:  $\forall$  A'  $\in$  ?F1.  $\exists$  F''. ?f A' F''
  proof
    fix A'
    assume A'  $\in$  ?F1
    then obtain F'' where F''  $\in$  Pow F' - {[]} A' = Union F''
      using  $\langle$  $\forall$  A'  $\in$  ?F1. A'  $\in$  closure F' $\rangle$ 
      unfolding closure-def
      by (metis image-iff)
    thus  $\exists$  F''. ?f A' F''
      by auto
  qed

  let ?fs =  $\lambda$  A'. SOME F''. ?f A' F''
  let ?F'' = ?F2  $\cup$   $\bigcup$  (?fs ` ?F1)

  have ?F''  $\subseteq$  F'
  proof (safe)
    fix x A'
    assume A'  $\in$  Fa A'  $\notin$  F'
    hence  $\exists$  F''. ?f A' F''
      using *
      by auto
    assume x  $\in$  ?fs A'
    thus x  $\in$  F'

```

```

    using someI-ex[of ?f A', OF  $\exists F''. ?f A' F''$ ]
    by auto
qed
moreover
have  $\bigcup \bigcup (?fs \text{ ' } ?F1) = \bigcup ?F1$ 
proof
  show  $\bigcup \bigcup (?fs \text{ ' } ?F1) \subseteq \bigcup ?F1$ 
  proof
    fix x
    assume  $x \in \bigcup \bigcup (?fs \text{ ' } ?F1)$ 
    then obtain A' where  $x \in \bigcup ?fs A' A' \in ?F1$ 
    by auto
    thus  $x \in \bigcup ?F1$ 
    using * someI-ex[of ?f A']
    by auto
  qed
next
show  $\bigcup ?F1 \subseteq \bigcup \bigcup (?fs \text{ ' } ?F1)$ 
proof
  fix x
  assume  $x \in \bigcup ?F1$ 
  then obtain A' where  $x \in A' A' \in ?F1$ 
  by auto
  hence  $\exists F''. ?f A' F''$ 
  using *
  by auto
  show  $x \in \bigcup \bigcup (?fs \text{ ' } ?F1)$ 
  using  $\langle x \in A' \rangle \langle A' \in ?F1 \rangle$ 
  using someI-ex[of ?f A', OF  $\exists F''. ?f A' F''$ ]
  by auto
qed
qed
hence  $\bigcup ?F'' = A$ 
  using  $\langle \bigcup Fa = A \rangle$ 
  by auto
moreover
have  $A \notin ?F''$ 
proof (rule ccontr)
  assume  $\neg ?thesis$ 
  hence  $A \in ?F''$ 
  by simp
  hence  $A \in \bigcup (?fs \text{ ' } ?F1)$ 
  using  $\langle A \notin F \rangle \langle Fa \subseteq F \rangle$ 
  by auto
  then obtain A' where  $A' \in ?F1 \ A \in ?fs A'$ 
  by auto
  have  $A' \subseteq A$ 
  using  $\langle \bigcup Fa = A \rangle \langle A' \in ?F1 \rangle$ 
  by auto

```

```

moreover
have  $\exists F''. \text{?f } A' F''$ 
  using *  $\langle A' \in ?F1 \rangle$ 
  by auto
have  $A \subseteq A'$ 
  using  $\langle A \in \text{?fs } A' \rangle \langle A' \in ?F1 \rangle$ 
  using someI-ex[of  $\text{?f } A', OF \langle \exists F''. \text{?f } A' F'' \rangle$ ]
  by auto
ultimately
have  $A \in ?F1$ 
  using  $\langle A' \in ?F1 \rangle$ 
  by auto
thus False
  using  $\langle A \notin F \rangle \langle Fa \subseteq F \rangle$ 
  by auto
qed
moreover
have  $\text{?F}'' \neq \{\}$ 
proof–
  from  $\langle Fa \neq \{\} \rangle$ 
  obtain  $A'$  where  $A' \in Fa$ 
    by auto
  show ?thesis
  proof (cases  $A' \in F'$ )
    case True
    thus ?thesis
      using  $\langle A' \in Fa \rangle$ 
      by auto
  next
  case False
  hence  $A' \in Fa - F'$ 
    using  $\langle A' \in Fa \rangle$ 
    by auto
  hence  $\exists F''. \text{?f } A' F''$ 
    using *
    by auto
  have  $\bigcup (\text{?fs } ' ?F1) \neq \{\}$ 
    using someI-ex[of  $\text{?f } A', OF \langle \exists F''. \text{?f } A' F'' \rangle$ ]  $\langle A' \in Fa - F' \rangle$ 
    by auto
  thus ?thesis
    by simp
qed
qed
ultimately
have expressible  $A (F' - \{A\})$ 
  unfolding expressible-def
  by (rule-tac  $x = ?F''$  in exI) auto
thus False
  using  $\langle \text{irreducible } F' \rangle \langle A \in F' \rangle$ 

```

```

    unfolding reducible-def expressible-def
    by auto
qed

```

```

lemma irreducible-closure:
  assumes irreducible F and irreducible F' and closure F = closure F'
  shows F = F'
using assms
using irreducible-closure'
by auto

end

```

13.3.1 Implementation by sorted and distinct lists

```

theory IrreducibleFamiliesImpl
imports IrreducibleFamilies UnionClosedImpl Combinatorics
begin

```

```

definition expressible-l where
  expressible-l A F  $\equiv$   $A \in \text{set } (\text{map } \text{Union-l } (\text{all-nonempty-subsets } F))$ 

```

```

lemma expressible-l-soundness:
  assumes expressible-l A F
  shows expressible (set A) (f-to-set-l F)
proof-
  from assms obtain X' where  $X' \in \text{set } (\text{all-nonempty-subsets } F)$   $A = \text{Union-l } X'$ 
  unfolding expressible-l-def
  by auto
  then obtain X where  $\text{set } X \subseteq \text{set } F$   $1 \leq \text{length } X$   $\text{set } A = \bigcup (\text{f-to-set-l } X)$ 
  using SetUnionImpl-lists.Union-set[of X] all-subset[of F]
  by auto
  thus ?thesis
  unfolding expressible-def
  by (rule-tac x=f-to-set-l X in exI) auto
qed

```

```

lemma f-to-set-l-subset-ex:
  fixes F'::nat set set
  assumes  $F' \subseteq \text{f-to-set-l } Fl$ 
  shows  $\exists F'l. \text{set } F'l \subseteq \text{set } Fl \wedge \text{f-to-set-l } F'l = F' \wedge \text{distinct } F'l \wedge \text{length } F'l \leq \text{length } Fl$ 
using assms
using assms
proof (induct Fl arbitrary: F')
  case Nil
  thus ?case
  by auto

```

```

next
  case (Cons a Fl)
  show ?case
  proof (cases  $F' \subseteq f\text{-to-set-}l\ Fl$ )
    case True
    thus ?thesis
      using Cons(1)[of F']
      by auto
  next
  case False
  hence  $F' - \{set\ a\} \subseteq f\text{-to-set-}l\ Fl$   $a \notin set\ Fl$   $set\ a \in F'$ 
    using Cons(2)
    by auto
  then obtain F'l where
     $set\ F'l \subseteq set\ Fl$   $distinct\ F'l\ f\text{-to-set-}l\ F'l = F' - \{set\ a\}$   $length\ F'l \leq length$ 
    Fl
    using Cons(1)[of  $F' - \{set\ a\}$ ]
    by auto
  thus ?thesis
    using  $\langle a \notin set\ Fl \rangle \langle set\ a \in F' \rangle$ 
    by (rule-tac  $x=F'l$  @ [a] in exI) auto
qed
qed

lemma expressible-l-completeness-lemma:
  assumes  $F' \subseteq f\text{-to-set-}l\ Fl$   $F' \neq \{\}$ 
  shows  $\bigcup F' \in set\ (map\ set\ (map\ Union-l\ (all\ nonempty\ subsets\ Fl)))$ 
proof-
  obtain F'l where  $set\ F'l \subseteq set\ Fl$   $distinct\ F'l\ f\text{-to-set-}l\ F'l = F'$   $length\ F'l \leq$ 
  length Fl
    using f-to-set-l-subset-ex[OF assms(1)]
    by auto
  have  $length\ F'l \geq 1$ 
    using  $\langle f\text{-to-set-}l\ F'l = F' \rangle \langle F' \neq \{\} \rangle$ 
    by (auto simp add: not-less-eq-eq)
  have  $length\ F'l \in set\ [1..length\ Fl + 1]$ 
    using  $\langle length\ F'l \geq 1 \rangle \langle length\ F'l \leq length\ Fl \rangle$ 
    by auto
  moreover
  have  $set\ (Union-l\ F'l) \in (set \circ Union-l)\ 'set\ (Combinatorics.combine\ Fl\ (length\ F'l))$ 
    using combine-sublist[of set F'l Fl]  $\langle set\ F'l \subseteq set\ Fl \rangle \langle distinct\ F'l \rangle \langle length\ F'l$ 
     $\leq length\ Fl \rangle distinct-card$ [of F'l]
    by (auto simp add: SetUnionImpl-lists.Union-set)
  ultimately
  show ?thesis
    using  $\langle f\text{-to-set-}l\ F'l = F' \rangle$ [symmetric]
    unfolding all-nonempty-subsets-def
    by (simp add: SetUnionImpl-lists.Union-set, auto, force)

```

qed

lemma *expressible-l-completeness*:

assumes *sd A sdf F*

assumes

expressible (set A) (f-to-set-l F)

shows *expressible-l A F*

proof –

have *sdff (all-nonempty-subsets F)*

unfolding *all-nonempty-subsets-def*

using *⟨sdf F⟩*

using *combine-subset[of - F -]*

by *auto*

hence *sdf (map Union-l (all-nonempty-subsets F))*

using *SetUnionImpl-lists.Union-inv*

by *force*

from *⟨expressible (set A) (f-to-set-l F)⟩*

obtain *F' where F' ⊆ f-to-set-l F F' ≠ {} set A = ⋃ F'*

unfolding *expressible-def*

by *auto*

hence *set A ∈ set ‘ set (map Union-l (all-nonempty-subsets F))*

unfolding *expressible-l-def*

using *expressible-l-completeness-lemma[of F' F]*

by *auto*

thus *?thesis*

unfolding *expressible-l-def*

using *⟨sd A⟩ ⟨sdf (map Union-l (all-nonempty-subsets F))⟩*

using *SetImpl-lists.set-set[of map Union-l (all-nonempty-subsets F) A]*

by *auto*

qed

end

14 Covering

theory *Covering*

imports *UnionClosed Frankl IsomorphicFamilies IrreducibleFamilies*

begin

14.1 Definition of Covering

definition *FC-covered* **where**

FC-covered Fc F ≡ ∃ Fc'. iso Fc' Fc ∧ Fc' ⊆ closure F

definition *nonFC-covered* **where**

nonFC-covered Nc F ≡ ∃ Nc'. iso Nc Nc' ∧ closure F ⊆ closure Nc' ∪ {{}}

abbreviation *FCs-covered* **where**

FCs-covered $\mathcal{F} F \equiv \exists Fc \in \mathcal{F}. FC\text{-covered } Fc F$

abbreviation *nonFCs-covered* **where**

nonFCs-covered $\mathcal{N} F \equiv \exists Nc \in \mathcal{N}. nonFC\text{-covered } Nc F$

definition *covered* **where**

covered $\mathcal{F} \mathcal{N} F \equiv FCs\text{-covered } \mathcal{F} F \vee nonFCs\text{-covered } \mathcal{N} F$

lemma *FC-covered-sound*:

fixes $Fc :: nat \text{ set set}$

assumes *finite* F **and** *FC-covered* $Fc F$ **and** *FC-family* Fc

shows *FC-family* F

proof –

from $\langle FC\text{-covered } Fc F \rangle$ **obtain** Fc' **where** $iso Fc' Fc Fc' \subseteq closure F$

unfolding *FC-covered-def*

by *auto*

thus *?thesis*

using $\langle finite F \rangle$ *FC-family* Fc

using *FC-family-closure* $[of F]$

using *FC-family-mono* $[of Fc' closure F]$

using *FC-family-iso* $[of - Fc']$

unfolding *iso-def bij-betw-def*

by *auto*

qed

lemma *nonFC-covered-sound*:

fixes $F :: nat \text{ set set}$

assumes *finite* F' **and** *nonFC-covered* $F' F$ **and** $\neg FC\text{-family } F'$

shows $\neg FC\text{-family } F$

proof (*rule ccontr*)

assume $\neg ?thesis$

from $\langle nonFC\text{-covered } F' F \rangle$

obtain F'' **where** $iso F' F'' closure F \subseteq closure F'' \cup \{\{\}\}$ *finite* F''

using $\langle finite F' \rangle$

by (*auto simp add: nonFC-covered-def finite-iso*)

hence *FC-family* F''

using *closure-subset* $[of F]$ *FC-family-empty-set-insert*

using *FC-family-mono* $[of F closure F'' \cup \{\{\}\}]$ $\langle \neg \neg FC\text{-family } F \rangle$

using *assms FC-family-closure* $[of F'']$

by *auto*

thus *False*

using $\langle \neg FC\text{-family } F' \rangle$

using $\langle iso F' F'' \rangle$

using *FC-family-iso* $[of - F']$

unfolding *iso-def bij-betw-def*

by *auto*

qed

lemma *FC-covered-mono*:
assumes $F \subseteq F'$ *FC-covered* \mathcal{F} F
shows *FC-covered* \mathcal{F} F'
using *assms closure-mono*[of F F']
unfolding *FC-covered-def*
by *auto*

lemma *nonFC-covered-mono*:
assumes $F \supseteq F'$ *nonFC-covered* \mathcal{F} F
shows *nonFC-covered* \mathcal{F} F'
using *assms closure-mono*[of F' F]
unfolding *nonFC-covered-def*
by *auto*

14.2 Covering and empty set

lemma *FC-covered-remove-empty*:
assumes *finite* F
shows *FC-covered* F_c $(F - \{\{\}\}) \longrightarrow$ *FC-covered* F_c F
unfolding *FC-covered-def*
using *closure-remove-empty*[of F] *assms*
by *auto*

lemma *nonFC-covered-remove-empty*:
assumes *finite* F
shows *nonFC-covered* N_c $(F - \{\{\}\}) \longrightarrow$ *nonFC-covered* N_c F
unfolding *nonFC-covered-def*
using *closure-remove-empty*[of F] *assms*
by *auto*

lemma *covered-remove-empty*:
assumes *finite* F
shows *covered* \mathcal{F} \mathcal{N} $(F - \{\{\}\}) \longrightarrow$ *covered* \mathcal{F} \mathcal{N} F
using *assms*
using *FC-covered-remove-empty*[of F] *nonFC-covered-remove-empty*[of F]
unfolding *covered-def*
by *auto*

14.3 Covering and isomorphic families

lemma *FC-covered-iso*:
assumes *iso* F F' **and** *FC-covered* F_c F
shows *FC-covered* F_c F'
using *assms*
unfolding *FC-covered-def*
proof (*safe*)
fix F_c'
assume *iso* F F' *iso* F_c' F_c $F_c' \subseteq$ *closure* F
then obtain f g **where** $*$: $F_c' = op \text{ ' } f \text{ ' } F_c$ *bij-betw* f $(\bigcup F_c)$ $(\bigcup F_c')$ $F' = op$
 $\text{ ' } g \text{ ' } F$ *bij-betw* g $(\bigcup F)$ $(\bigcup F')$

```

    using iso-sym[of  $Fc'$   $Fc$ ]
    unfolding iso-def
    by metis
  hence inj-on  $g$  ( $\bigcup F$ )
    by (simp add: bij-betw-def)

  have  $\bigcup Fc' \subseteq \bigcup F$ 
    using Union-mono[OF  $\langle Fc' \subseteq \text{closure } F \rangle$ ]
    by simp
  have inj-on  $g$  ( $\bigcup Fc'$ )
    by (rule subset-inj-on[of  $\bigcup F$ ], fact+)

  let ? $Fc'' = op \text{ ` } g \text{ ` } Fc'$ 
  show  $\exists Fc''. \text{iso } Fc'' Fc \wedge Fc'' \subseteq \text{closure } F'$ 
  proof (rule-tac  $x=?Fc''$  in exI, safe)
    show iso ? $Fc'' Fc$ 
    proof-
      have iso  $Fc' ?Fc''$ 
        unfolding iso-def
        apply (rule-tac  $x=g$  in exI)
        using  $\langle \text{inj-on } g (\bigcup Fc') \rangle$ 
        by (auto simp add: bij-betw-def)
      hence iso ? $Fc'' Fc'$ 
        by (rule iso-sym)
      thus ?thesis
        using  $\langle \text{iso } Fc' Fc \rangle$ 
        by (rule iso-trans)
    qed
  next
  fix  $x$ 
  assume  $x \in Fc'$ 
  hence  $x \in \text{closure } F$ 
    using  $\langle Fc' \subseteq \text{closure } F \rangle$ 
    by auto
  thus  $g \text{ ` } x \in \text{closure } F'$ 
    using * closure-iso
    by auto
  qed
qed

lemma nonFC-covered-iso:
  fixes  $F F' :: \text{nat set set}$ 
  assumes iso  $F F'$  and nonFC-covered  $Nc F$  and
    finite ( $\bigcup F'$ ) and finite ( $\bigcup Nc$ )
  shows nonFC-covered  $Nc F'$ 
using assms
unfolding nonFC-covered-def
proof (safe)
  fix  $Nc'$ 

```

```

assume  $iso\ F\ F'\ iso\ Nc\ Nc'\ closure\ F \subseteq closure\ Nc' \cup \{\{\}\}$ 
then obtain  $f\ g$  where  $*$ :  $Nc' = op\ 'f'\ Nc\ bij\text{-}betw\ f\ (\bigcup Nc)\ (\bigcup Nc')\ F' =$ 
 $op\ 'g'\ F\ bij\text{-}betw\ g\ (\bigcup F)\ (\bigcup F')$ 
using  $iso\text{-}sym[of\ Nc'\ Nc]$ 
unfolding  $iso\text{-}def$ 
by  $force$ 

have  $finite\ (\bigcup Nc')$ 
using  $\langle finite\ (\bigcup Nc) \rangle \langle bij\text{-}betw\ f\ (\bigcup Nc)\ (\bigcup Nc') \rangle$ 
by  $(metis\ bij\text{-}betw\text{-}finite)$ 
moreover
have  $\bigcup F \subseteq \bigcup Nc'$ 
using  $Union\text{-}mono[OF\ \langle closure\ F \subseteq closure\ Nc' \cup \{\{\}\} \rangle]$ 
by  $simp$ 
ultimately
obtain  $g'$  where  $inj\text{-}on\ g'\ (\bigcup Nc')\ \forall\ x \in (\bigcup F). g'\ x = g\ x$ 
using  $\langle bij\text{-}betw\ g\ (\bigcup F)\ (\bigcup F') \rangle \langle finite\ (\bigcup F') \rangle$ 
using  $bij\text{-}betw\text{-}inj\text{-}extend[of\ g\ \bigcup F\ \bigcup F'\ \bigcup Nc' - \bigcup F]$ 
using  $finite\text{-}subset[of\ \bigcup Nc' - \bigcup F\ \bigcup Nc']$ 
by  $(auto, metis\ sup\text{-}absorb2)$ 

let  $?Nc'' = op\ 'g'\ Nc'$ 
show  $\exists Nc'. iso\ Nc\ Nc' \wedge closure\ F' \subseteq closure\ Nc' \cup \{\{\}\}$ 
proof  $(rule\text{-}tac\ x=?Nc''\ in\ exI, rule\ conjI)$ 
show  $iso\ Nc\ ?Nc''$ 
proof–
have  $iso\ Nc'\ ?Nc''$ 
using  $\langle inj\text{-}on\ g'\ (\bigcup Nc') \rangle$ 
unfolding  $iso\text{-}def$ 
by  $(rule\text{-}tac\ x=g'\ in\ exI)\ (auto\ simp\ add: bij\text{-}betw\text{-}def)$ 
thus  $?thesis$ 
using  $\langle iso\ Nc\ Nc' \rangle iso\text{-}trans$ 
by  $blast$ 
qed
next
show  $closure\ F' \subseteq closure\ (op\ 'g'\ Nc') \cup \{\{\}\}$ 
proof
fix  $x$ 
assume  $x \in closure\ F'$ 
hence  $x \in op\ 'g'\ closure\ F$ 
using  $*$ 
by  $(simp\ add: closure\text{-}iso)$ 
moreover
have  $op\ 'g'\ closure\ F = op\ 'g'\ closure\ F$ 
using  $\langle \forall x \in (\bigcup F). g'\ x = g\ x \rangle$ 
using  $map\text{-}fam\text{-}cong[of\ closure\ F\ g'\ g]$ 
by  $simp$ 
moreover
have  $op\ 'g'\ closure\ F \subseteq op\ 'g'\ (closure\ Nc' \cup \{\{\}\})$ 

```

```

    using ⟨closure F ⊆ closure Nc' ∪ {{}}⟩
    by auto
  moreover
  have op ' g' ' (closure Nc' ∪ {{}}) = closure (op ' g' ' Nc') ∪ {{}}
    using closure-iso[of g' Nc']
    by auto
  ultimately
  show x ∈ closure (op ' g' ' Nc') ∪ {{}}
    by auto
qed
qed
qed

```

```

lemma covered-iso:
  fixes F F' :: nat set set
  assumes finite (⋃ F') and
    ∀ Fc ∈ ℱ. finite (⋃ Fc) and ∀ Nc ∈ ℒ. finite (⋃ Nc)
  assumes iso F F'
  shows covered ℱ ℒ F ⇒ covered ℱ ℒ F'
using assms
unfolding covered-def
using FC-covered-iso[of F F'] nonFC-covered-iso[of F F']
by blast

```

```

lemma iso-represents-FCs-covered:
  fixes FF FFb :: nat set set set
  assumes iso-represents FFb FF and ∀ F ∈ FFb. FCs-covered ℱ F
  shows ∀ F ∈ FF. FCs-covered ℱ F
proof
  fix F
  assume F ∈ FF
  then obtain F' where F' ∈ FFb iso F' F
    using ⟨iso-represents FFb FF⟩ iso-sym
  unfolding iso-represents-def
  by blast
  then obtain Fc where Fc ∈ ℱ FC-covered Fc F'
    using ⟨∀ F ∈ FFb. FCs-covered ℱ F⟩
  by auto
  hence FC-covered Fc F
    using ⟨F ∈ FF⟩ FC-covered-iso[of F' F Fc] ⟨iso F' F⟩
  by auto
  thus FCs-covered ℱ F
    using ⟨Fc ∈ ℱ⟩
  by auto
qed

```

```

lemma iso-represents-nonFCs-covered:
  fixes FF FFb :: nat set set set
  assumes ∀ F ∈ FF. finite (⋃ F) ∀ F ∈ ℒ. finite (⋃ F)

```

```

assumes iso-represents FFb FF  $\forall F \in FFb. \text{nonFCs-covered } \mathcal{N} F$ 
shows  $\forall F \in FF. \text{nonFCs-covered } \mathcal{N} F$ 
proof
  fix  $F :: \text{nat set set}$ 
  assume  $F \in FF$ 
  then obtain  $F'$  where  $F' \in FFb \text{ iso } F' F$ 
    using  $\langle \text{iso-represents FFb FF} \rangle \text{ iso-sym}$ 
    unfolding iso-represents-def
    by blast
  then obtain  $N$  where  $N \in \mathcal{N} \text{ nonFC-covered } N F'$ 
    using  $\langle \forall F \in FFb. \text{nonFCs-covered } \mathcal{N} F \rangle$ 
    by auto
  hence nonFC-covered N F
    using  $\langle F \in FF \rangle \text{nonFC-covered-iso}[of F' F N] \langle \forall F \in FF. \text{finite } (\bigcup F) \rangle \langle \forall F \in \mathcal{N}. \text{finite } (\bigcup F) \rangle \langle \text{iso } F' F \rangle$ 
    by auto
  thus nonFCs-covered N F
    using  $\langle N \in \mathcal{N} \rangle$ 
    by auto
qed

```

14.4 Covering, closure and irreducible families

```

lemma closure-covered:
  assumes closure F = closure F'
  shows  $\text{covered } \mathcal{F} \mathcal{N} F \longleftrightarrow \text{covered } \mathcal{F} \mathcal{N} F'$ 
using assms
unfolding covered-def FC-covered-def nonFC-covered-def
by auto

```

```

lemma ex-irreducible-covered:
  fixes  $F$ 
  assumes  $\text{finite } (\bigcup F)$ 
  shows  $\exists F' \subseteq F. \text{irreducible } F' \wedge (\text{covered } \mathcal{F} \mathcal{N} F \longleftrightarrow \text{covered } \mathcal{F} \mathcal{N} F')$ 
proof–
  from assms
  have  $\text{finite } F$ 
    by  $(\text{auto simp add: finiteUn-iff})$ 
  hence  $\exists F' \subseteq F. \text{irreducible } F' \wedge (\text{closure } F = \text{closure } F')$ 
    using ex-irreducible-closure[of F]
    by auto
  thus ?thesis
    using closure-covered
    by metis
qed

```

```

lemma all-irreducible-covered-all-covered:
  assumes  $\forall F. \text{irreducible } F \wedge \bigcup F \subseteq \{0..<n::\text{nat}\} \longrightarrow \text{covered } \mathcal{F} \mathcal{N} F$ 
  shows  $\forall F. \bigcup F \subseteq \{0..<n\} \longrightarrow \text{covered } \mathcal{F} \mathcal{N} F$ 

```

```

proof (safe)
  fix F
  assume  $\bigcup F \subseteq \{0..<n\}$ 
  then obtain  $F'$  where  $F' \subseteq F$  irreducible  $F'$  covered  $\mathcal{F} \mathcal{N} F \longleftrightarrow \text{covered } \mathcal{F} \mathcal{N} F'$ 
    using finite-subset[of  $\bigcup F \{0..<n\}$ ]
    using ex-irreducible-covered[of  $F \mathcal{F} \mathcal{N}$ ]
    by auto
  have irreducible  $F' \wedge \bigcup F' \subseteq \{0..<n\}$ 
    using  $\langle F' \subseteq F \rangle \langle \text{irreducible } F' \rangle \langle \bigcup F \subseteq \{0..<n\} \rangle$ 
    by blast
  hence covered  $\mathcal{F} \mathcal{N} F'$ 
    by (rule assms[rule-format])
  thus covered  $\mathcal{F} \mathcal{N} F$ 
    using  $\langle \text{covered } \mathcal{F} \mathcal{N} F \longleftrightarrow \text{covered } \mathcal{F} \mathcal{N} F' \rangle$ 
    by simp
qed

end

```

14.4.1 FC and nonFC covering implementation

```

theory CoveringImpl
  imports
    Covering
    FamilyImpl UnionClosedImpl NonIsomorphicFamiliesImpl
    More.MoreBinomial
begin

```

definition *FC-covered-l* :: $\text{nat list list} \Rightarrow \text{nat list list} \Rightarrow \text{nat list list} \Rightarrow \text{bool}$ **where**
 $\text{FC-covered-l } Fc \ F \ \text{perms} \longleftrightarrow \text{list-ex } (\lambda Fc'. \text{set } Fc' \subseteq \text{set } (\text{close-l } F)) \ (\text{map } (\lambda p. \text{permute-family-l } p \ Fc) \ \text{perms})$

lemma *FC-covered-l-soundness*:

assumes $\forall p \in \text{set perms}. p <\sim\sim> [0..<n] \ \text{dm } Fc \ n$

assumes *FC-covered-l* $Fc \ F \ \text{perms}$

shows *FC-covered* (*f-to-set-l* Fc) (*f-to-set-l* F)

proof—

obtain p **where** $p \in \text{set perms set } (\text{permute-family-l } p \ Fc) \subseteq \text{set } (\text{close-l } F)$

using *assms*

unfolding *FC-covered-l-def*

by (*auto simp add: list-ex-iff*)

show *?thesis*

unfolding *FC-covered-def*

proof (*rule-tac* $x = \text{f-to-set-l } (\text{permute-family-l } p \ Fc)$ **in** *exI*, *rule*)

show *iso* (*f-to-set-l* (*permute-family-l* $p \ Fc$)) (*f-to-set-l* Fc)

proof—

have *inj-on* (*list-to-fun* p) ($\bigcup \text{f-to-set-l } Fc$)

```

    using perm-distinct-iff [of p [0.. $n$ ]] perm-length [of p [0.. $n$ ]]
      ⟨p ∈ set perms⟩ assms(1) assms(2)
    using nat-list-to-fun-inj-on' [of p n ∪ f-to-set-l Fc]
    by simp
  thus ?thesis
    using SetPermutations-lists.permute-family-inj-embed [of p Fc] inj-embed-iso [of
f-to-set-l Fc f-to-set-l (permute-family-l p Fc)] iso-sym
    by blast
  qed
next
  show f-to-set-l (permute-family-l p Fc) ⊆ closure (f-to-set-l F)
    using ⟨set (permute-family-l p Fc) ⊆ set (close-l F)⟩
    using SetUnionImpl-lists.close-set [of F]
    by auto
  qed
qed

```

lemma *FC-covered-l-completeness*:

```

  assumes sdf F sdf Fc dm Fc n dm F n perms = permute [0.. $n$ ]
  assumes FC-covered (f-to-set-l Fc) (f-to-set-l F)
  shows FC-covered-l Fc F perms
proof-
  from assms
  obtain Fc' where iso Fc' (f-to-set-l Fc) Fc' ⊆ closure (f-to-set-l F)
    unfolding FC-covered-def
    by auto
  have ∪ Fc' ⊆ ∪ closure (f-to-set-l F)
    using ⟨Fc' ⊆ closure (f-to-set-l F)⟩
    by (metis Union-mono)
  hence ∪ Fc' ⊆ {0.. $n$ }
    using ⟨dm F n⟩
    by auto
  have finite (∪ Fc')
    using ⟨iso Fc' (f-to-set-l Fc)⟩ bij-betw-finite
    unfolding iso-def
    by auto
  then obtain Fc'l where Fc' = f-to-set-l Fc'l sdf Fc'l
    using SetImpl-lists.f-to-set-ex [of Fc']
    by auto
  then obtain p where p ∈ set perms set Fc'l = set (permute-family-l p Fc)
    using ⟨iso Fc' (f-to-set-l Fc)⟩ iso-permute-family-l [of Fc Fc'l n perms]
    using ⟨dm Fc n⟩ ⟨∪ Fc' ⊆ {0.. $n$ }⟩ ⟨perms = permute [0.. $n$ ]] ⟨sdf Fc⟩
    using iso-sym [of f-to-set-l Fc'l f-to-set-l Fc]
    by auto
  show ?thesis
    unfolding FC-covered-l-def
  proof (subst list-ex-iff, rule-tac x=permute-family-l p Fc in becI)
    show permute-family-l p Fc ∈ set (FamilyImpl.map (λp. permute-family-l p

```



```

Fc) perms)
  using ⟨p ∈ set perms⟩
  by auto
next
show set (permute-family-l p Fc) ⊆ set (close-l F)
proof-
  have set Fc'l ⊆ set (close-l F)
  proof
    fix A
    assume A ∈ set Fc'l
    show A ∈ set (close-l F)
    proof (rule SetUnionImpl-lists.close-completeness)
      show sorted A ∧ distinct A
      using ⟨∀ A ∈ set Fc'l. sorted A ∧ distinct A⟩ ⟨A ∈ set Fc'l⟩
      by auto
    next
      show ∀ a ∈ set F. sorted a ∧ distinct a
      by fact
    next
      show set A ∈ closure (f-to-set-l F)
      using ⟨A ∈ set Fc'l⟩ ⟨Fc' = f-to-set-l Fc'l⟩ ⟨Fc' ⊆ closure (f-to-set-l F)⟩
      by auto
    qed
  qed
  thus ?thesis
  using ⟨set Fc'l = set (permute-family-l p Fc)⟩
  by auto
qed
qed
qed

```

definition *FCs-covered-l* **where**

FCs-covered-l \mathcal{F} F $perms \longleftrightarrow list-ex (\lambda Fc. FC-covered-l Fc F perms) \mathcal{F}$

definition *nonFC-covered-l* **where**

nonFC-covered-l Nc F $perms \longleftrightarrow$
 $list-ex (\lambda Nc'. set (close-l F) \subseteq set Nc')$
 $(map (\lambda p. (close-insert-empty-l (permute-family-l p Nc))) perms)$

lemma *nonFC-covered-l-soundness:*

assumes $\forall p \in set perms. p <\sim\sim> [0..<n] \ dm \ Nc \ n$

assumes *nonFC-covered-l* Nc F $perms$

shows *nonFC-covered* (f-to-set-l Nc) (f-to-set-l F)

proof–

obtain p **where** $p \in set perms$ $set (close-insert-empty-l (permute-family-l p Nc))$

$\supseteq set (close-l F)$

using *assms*

```

    unfolding nonFC-covered-l-def
    by (auto simp add: list-ex-iff)
show ?thesis
    unfolding nonFC-covered-def
proof (rule-tac x=f-to-set-l (permute-family-l p Nc) in exI, rule)
    show iso (f-to-set-l Nc) (f-to-set-l (permute-family-l p Nc))
    proof-
        have inj-on (list-to-fun p) (⋃ f-to-set-l Nc)
        using perm-distinct-iff[of p [0..

```

definition *nonFCs-covered-l* **where**
nonFCs-covered-l \mathcal{N} F perms \longleftrightarrow
list-ex (λ Nc. *nonFC-covered-l* Nc F perms) \mathcal{N}

lemma *nonFCs-covered-l-soundness*:
assumes $\forall p \in \text{set perms}. p < \sim \sim > [0.. *dmf* \mathcal{N} n
assumes *nonFCs-covered-l* \mathcal{N} F perms
shows *nonFCs-covered* (*fs-to-set-l* \mathcal{N}) (*f-to-set-l* F)
using *assms*
unfolding *nonFCs-covered-l-def*
using *nonFC-covered-l-soundness*[of perms n]
by (auto simp add: list-ex-iff) (rule-tac x=Nc in bexI, auto)$

definition *nonFC-covered-l-opt* :: *nat list list list* \Rightarrow *nat list list* \Rightarrow *bool* **where**
nonFC-covered-l-opt Nc-perms $F \longleftrightarrow$
list-ex (λ Nc'. *set* (*close-l* F) \subseteq *set* Nc') Nc-perms

definition *nonFCs-covered-l-opt* **where**
nonFCs-covered-l-opt \mathcal{N} -perms $F \longleftrightarrow$
list-ex (λ Nc. *nonFC-covered-l-opt* Nc F) \mathcal{N} -perms

lemma *nonFCs-covered-l-opt*:

```

    assumes  $\mathcal{N}$ -perms = (map ( $\lambda$  Nc. (map ( $\lambda$  p. close-insert-empty-l (permute-family-l
p Nc)) perms))  $\mathcal{N}$ )
    shows nonFCs-covered-l-opt  $\mathcal{N}$ -perms  $F \longleftrightarrow$  nonFCs-covered-l  $\mathcal{N}$   $F$  perms
using assms
unfolding nonFCs-covered-l-def nonFCs-covered-l-opt-def
unfolding nonFC-covered-l-def nonFC-covered-l-opt-def
by (auto simp add: list-ex-iff)

end

```

15 L-partitioned families

```

theory LPartitioning
imports Main More.MoreSet IsomorphicFamilies
begin

```

definition *is-L-part* where

```

is-L-part  $n$   $L$   $F \longleftrightarrow$ 
   $\bigcup F \subseteq \{0..<n::nat\} \wedge$ 
   $(\forall A \in F. \text{card } A < \text{length } L) \wedge$ 
   $(\forall n < \text{length } L. \text{card } \{A \in F. \text{card } A = n\} = L ! n)$ 

```

abbreviation *L-part* where

```

L-part  $n$   $L \equiv \{F. \text{is-L-part } n \ L \ F\}$ 

```

lemma *is-L-part-finite*:

```

    assumes is-L-part  $n$   $L$   $F$ 
    shows finite  $F \ \forall \ A \in F. \text{finite } A$ 
using assms
unfolding is-L-part-def
using finite-subset[of  $\bigcup F \ \{0..<n\}$ ]
by (auto simp add: finiteUn-iff)

```

lemma *is-L-part-zero*:

```

    shows is-L-part  $n$   $L$   $F \longleftrightarrow$  is-L-part  $n$  ( $L @ [0]$ )  $F$ 
proof-
  have *: finite ( $\bigcup F$ )  $\implies (\forall A \in F. \text{card } A < \text{length } L) \longleftrightarrow ((\forall A \in F. \text{card } A < \text{length } L + 1) \wedge \text{card } \{A \in F. \text{card } A = \text{length } L\} = 0)$ 
  by (auto simp add: finiteUn-iff) fastforce
  show ?thesis
  using *
  unfolding is-L-part-def
  by (auto simp add: nth-append finite-subset[of  $\bigcup F \ \{0..<n\}$ ]) (subgoal-tac na
= length  $L$ , simp+)
qed

```

lemma *is-L-part-zeros*:

```

    assumes  $\forall \ k'. \ k' > k \wedge k' < \text{length } L \longrightarrow L ! k' = 0$ 
    shows is-L-part  $n$   $L$   $F \longleftrightarrow$  is-L-part  $n$  (take  $(k+1)$   $L$ )  $F$ 

```

```

proof (cases  $k < \text{length } L$ )
  case False
  thus ?thesis
    by simp
next
  case True
  thus ?thesis
    using assms
  proof (induct  $L$  rule: rev-induct)
    case Nil
    thus ?case
      by simp
  next
    case (snoc  $a$   $L$ )
    show ?case
    proof (cases  $k = \text{length } L$ )
      case True
      thus ?thesis
        by auto
    next
      case False
      hence  $k < \text{length } L$ 
        using  $\langle k < \text{length } (L @ [a]) \rangle$ 
        by simp
      moreover
      hence  $a = 0$ 
      using  $\langle \forall k'. k < k' \wedge k' < \text{length } (L @ [a]) \longrightarrow (L @ [a]) ! k' = 0 \rangle$  [rule-format,
of length  $L$ ]
        by simp
      moreover
      have  $\forall k'. k < k' \wedge k' < \text{length } L \longrightarrow L ! k' = 0$ 
        using  $\langle \forall k'. k < k' \wedge k' < \text{length } (L @ [a]) \longrightarrow (L @ [a]) ! k' = 0 \rangle$ 
        by (auto simp add: nth-append)
      ultimately
      show ?thesis
        using snoc(1)
        using is-L-part-zero[of n L F]
        by simp
    qed
  qed
qed

lemma is-L-part-zeros-replicate:
   $\text{is-L-part } n \text{ (replicate } k \ 0) \ F \longleftrightarrow \text{is-L-part } n \ [] \ F$ 
proof (induct  $k$ )
  case  $0$ 
  thus ?case
    by simp
next

```

```

case (Suc k)
thus ?case
  using replicate-add[of k 1 0::nat] is-L-part-zero[of n replicate k 0 F]
  by simp
qed

```

```

lemma is-L-part-mem:
  assumes is-L-part n L F A ∈ F
  shows card A < length L ∧ L ! card A > 0
proof–
  let ?X = {A' ∈ F. card A' = card A}
  have A ∈ ?X
    using ⟨A ∈ F⟩
    by simp
  moreover
  have ?X ⊆ F
    by auto
  moreover
  have finite F
    using ⟨is-L-part n L F⟩
    by (simp add: is-L-part-finite)
  ultimately
  have card ?X > 0
    using finite-subset[of ?X F]
    using card-eq-0-iff[of ?X]
    by blast
  thus card A < length L ∧ L ! card A > 0
    using assms ⟨A ∈ F⟩
    unfolding is-L-part-def
    by auto
qed

```

```

lemma is-L-part-empty-mem:
  assumes
    is-L-part n L F hd L > 0 length L > 0
  shows {} ∈ F
proof–
  let ?A0 = {A' ∈ F. card A' = 0}
  have card ?A0 > 0
    using assms
    unfolding is-L-part-def
    by (auto simp add: hd-conv-nth)
  moreover
  have ∀ A' ∈ F. card A' = 0 ⟶ A' = {}
    using ⟨is-L-part n L F⟩
    using card-eq-0-iff
    by (auto simp add: is-L-part-finite)
  ultimately
  show ?thesis

```

```

    using card-gt-0-iff[of { $A' \in F$ .  $\text{card } A' = 0$ }]
    by auto
qed

lemma is-L-part-remove:
  assumes  $A \in F$  is-L-part  $n$   $L$   $F$ 
  shows is-L-part  $n$  ( $L[\text{card } A := (L ! \text{card } A) - 1]$ ) ( $F - \{A\}$ )
proof-
  let ?F =  $F - \{A\}$ 
  let ?X = { $A' \in F$ .  $\text{card } A' = \text{card } A$ }
  let ?XA = ?X - { $A$ }
  let ?L' =  $L[\text{card } A := (L ! \text{card } A) - 1]$ 
  have *:  $\forall n. n \neq \text{card } A \longrightarrow \{A' \in ?F. \text{card } A' = n\} = \{A' \in F. \text{card } A' = n\}$ 
  { $A' \in ?F. \text{card } A' = \text{card } A$ } = ?XA
  by auto

  have  $\text{card } A < \text{length } L$   $L ! \text{card } A > 0$ 
  using assms is-L-part-mem[of  $n$   $L$   $F$   $A$ ]
  by auto

  have  $\text{card } ?X = L ! \text{card } A$ 
  using assms  $\langle \text{card } A < \text{length } L \rangle$ 
  unfolding is-L-part-def
  by auto

  have  $L ! \text{card } A = \text{Suc } ((L ! \text{card } A) - 1)$ 
  using is-L-part-mem[of  $n$   $L$   $F$   $A$ ]  $\langle L ! \text{card } A > 0 \rangle$ 
  by auto
  hence  $\text{card } ?XA = (L ! \text{card } A) - 1$ 
  using  $\langle \text{card } ?X = L ! \text{card } A \rangle$ 
  using card-Suc'[of ?X ( $L ! \text{card } A) - 1$   $A$ ]  $\langle A \in F \rangle$ 
  by simp

  have  $\bigcup ?F \subseteq \{0..<n\}$ 
  using  $\langle \text{is-L-part } n \text{ } L \text{ } F \rangle$ 
  unfolding is-L-part-def Let-def
  by auto
  moreover
  {
    fix  $A'$ 
    assume  $A' \in ?F$ 
    hence  $\text{card } A' < \text{length } L$ 
    using  $\langle \text{is-L-part } n \text{ } L \text{ } F \rangle$ 
    unfolding is-L-part-def Let-def
    by auto
  }
  moreover
  have  $(\forall n < \text{length } ?L'. \text{card } \{A' \in ?F. \text{card } A' = n\} = ?L' ! n)$ 
  proof(safe)

```

```

fix  $n'$ 
assume  $n' < \text{length } ?L'$ 
show  $\text{card } \{A' \in ?F. \text{card } A' = n'\} = ?L' ! n'$ 
proof (cases  $n' \neq \text{card } A$ )
  case True
    thus ?thesis
      using  $*(1) \langle \text{is-L-part } n \ L \ F \rangle \langle n' < \text{length } ?L' \rangle$ 
      unfolding is-L-part-def Let-def
      by (auto simp add: nth-list-update)
  next
    case False
    thus ?thesis
      using  $*(2) \langle \text{card } ?XA = (L ! \text{card } A) - 1 \rangle \langle \text{card } A < \text{length } L \rangle$ 
      by (simp add: nth-list-update)
  qed
qed
ultimately
show ?thesis
  unfolding is-L-part-def Let-def
  by simp
qed

```

```

lemma is-L-part-remove-last:
  assumes is-L-part  $n \ (L @ [k + 1]) \ F \ \text{card } A = \text{length } L \ A \in F$ 
  shows is-L-part  $n \ (L @ [k]) \ (F - \{A\})$ 
using assms
using is-L-part-remove[of  $A \ F \ n \ L @ [k + 1]$ ]
by auto

```

```

lemma is-L-part-insert-last:
  assumes is-L-part  $n \ (L @ [k]) \ F \ A \notin F \ A \subseteq \{0..<n\} \ \text{card } A = \text{length } L$ 
  shows is-L-part  $n \ (L @ [k + 1]) \ (F \cup \{A\})$ 
proof–
  have finite  $F$ 
    using  $\langle \text{is-L-part } n \ (L @ [k]) \ F \rangle$ 
    by (simp add: is-L-part-finite)

```

```

  have  $*$ :  $\forall n < \text{length } L. \{X \in F \cup \{A\}. \text{card } X = n\} = \{X \in F. \text{card } X = n\}$ 
   $\{X \in F \cup \{A\}. \text{card } X = \text{length } L\} = \{X \in F. \text{card } X = \text{length } L\} \cup \{A\}$ 
    using  $\langle \text{card } A = \text{length } L \rangle \langle A \notin F \rangle$ 
    by auto

```

```

  have  $\bigcup (F \cup \{A\}) \subseteq \{0..<n\}$ 
    using  $\langle A \subseteq \{0..<n\} \rangle$ 
    using  $\langle \text{is-L-part } n \ (L @ [k]) \ F \rangle$ 
    by (auto simp add: is-L-part-def)

```

```

moreover
have  $\forall X \in F \cup \{A\}. \text{card } X < \text{length } (L @ [k + 1])$ 

```

```

    using ⟨is-L-part n (L @ [k]) F⟩ ⟨card A = length L⟩
    by (auto simp add: is-L-part-def)
moreover
{
  fix n'
  assume n' < length (L @ [k + 1])
  have card {X ∈ F ∪ {A}. card X = n'} = (L @ [k + 1]) ! n'
  proof (cases n' < length L)
    case True
    thus ?thesis
      using *(1) ⟨is-L-part n (L @ [k]) F⟩
      by (auto simp add: is-L-part-def nth-append)
  next
    case False
    hence n' = length L
      using ⟨n' < length (L @ [k + 1])⟩
      by simp
    thus ?thesis
      using *(2) ⟨card A = length L⟩ ⟨is-L-part n (L @ [k]) F⟩ ⟨A ∉ F⟩
      using card-insert-disjoint[of {A ∈ F. card A = length L}] ⟨finite F⟩
      by (auto simp add: is-L-part-def nth-append finite-subset)
  qed
}
ultimately
show ?thesis
  by (simp add: is-L-part-def)
qed

```

15.1 Ordering of L-partitions

definition *pwge* (infixl \succeq 100) **where**

$$L' \succeq L \longleftrightarrow (\text{length } L' = \text{length } L) \wedge (\forall i. i < \text{length } L \longrightarrow L' ! i \geq L ! i)$$

lemma *pwge-Cons*:

$$(a \# L) \succeq (b \# L') \longleftrightarrow (a \geq b \wedge L \succeq L')$$

unfolding *pwge-def*

by (auto simp add: nth-Cons) (metis Suc-eq-plus1-left diff-is-0-eq le-add-diff-inverse le-cases less-antisym less-imp-diff-less nth-Cons nth-Cons')

lemma *pwge-Nil*:

$$[] \succeq []$$

unfolding *pwge-def*

by *simp*

lemma *pwge-refl*:

fixes *L::'a::linorder list*

shows [*simp*]: $L \succeq L$

unfolding *pwge-def*

by *auto*


```

lemma pwge-trans [trans]:
  fixes L::'a::linorder list
  assumes  $L \succeq L'$   $L' \succeq L''$ 
  shows  $L \succeq L''$ 
using assms
proof (induct L arbitrary: L' L'')
  case Nil
  thus ?case
    unfolding pwge-def
    by simp
next
  case (Cons a l)
  obtain b l' c l'' where  $L' = b \# l'$   $L'' = c \# l''$ 
  using  $\langle a \# l \rangle \succeq L'$   $\langle L' \rangle \succeq L''$ 
  unfolding pwge-def
  by auto (metis Suc-length-conv)
  thus ?case
    using Cons
    by (auto simp add: pwge-Cons)
qed

```

```

lemma pwge-replicate-0:
  assumes  $\text{length } X = n$ 
  shows  $X \succeq \text{replicate } n \ (0::\text{nat})$ 
using assms
proof (induct n arbitrary: X)
  case 0
  thus ?case
    by simp
next
  case (Suc n)
  thus ?case
    by (cases X) (auto simp add: pwge-Cons)
qed

```

```

lemma pwge-list-update:
  fixes L::nat list
  assumes  $n < \text{length } L$   $nk \leq L ! n$ 
  shows  $L \succeq L [n := nk]$ 
using assms
unfolding pwge-def
by (auto simp add: nth-list-update)

```

```

lemma is-L-part-subset:
  assumes  $L' \succeq L$  is-L-part n L' F'
  shows  $\exists F. F \subseteq F' \wedge \text{is-L-part } n L F$ 
proof –
  let ?F'k =  $\lambda k. \{A. A \in F' \wedge \text{card } A = k\}$ 

```

```

let ?Fk =  $\lambda k. \text{SOME } S. S \subseteq ?F'k \ k \wedge \text{card } S = L ! k$ 
have  $\forall k. k < \text{length } L \longrightarrow (\exists S. S \subseteq ?F'k \ k \wedge \text{card } S = L ! k)$ 
proof (safe)
  fix k
  assume  $k < \text{length } L$ 
  show  $\exists S. S \subseteq ?F'k \ k \wedge \text{card } S = L ! k$ 
  proof (rule card-le)
    show  $L ! k \leq L' ! k$ 
    using assms  $\langle k < \text{length } L \rangle$ 
    unfolding pwge-def
    by simp
  next
  show  $\text{card } (?F'k \ k) = L' ! k$ 
  using assms  $\langle k < \text{length } L \rangle$ 
  unfolding is-L-part-def pwge-def
  by simp
  qed
qed
have *:  $\forall k < \text{length } L. ?Fk \ k \subseteq ?F'k \ k \wedge \text{card } (?Fk \ k) = L ! k$ 
proof (safe)
  fix k x
  assume  $k < \text{length } L$ 
  thus  $x \in ?Fk \ k \implies x \in F' \ x \in ?Fk \ k \implies \text{card } x = k$ 
   $\text{card } (?Fk \ k) = L ! k$ 
  using  $\langle \forall k. k < \text{length } L \longrightarrow (\exists S. S \subseteq ?F'k \ k \wedge \text{card } S = L ! k) \rangle$ 
  using tft-some[rule-format, where  $P = \lambda S. S \subseteq \{A \in F'. \text{card } A = k\} \wedge$ 
 $\text{card } S = L ! k]$ 
  by auto
  qed
qed

let ?F =  $\bigcup \{ ?Fk \ k \mid k. k < \text{length } L \}$ 
show ?thesis
proof (rule-tac  $x = ?F$  in exI, rule conjI)
  show  $?F \subseteq F'$ 
  using *
  by auto
next
  have  $\bigcup ?F \subseteq \{0..<n\}$ 
  using * is-L-part n L' F'
  unfolding is-L-part-def
  by auto
moreover
  have  $\forall A \in ?F. \text{card } A < \text{length } L$ 
  using *
  by auto
moreover
  have  $\forall k < \text{length } L. \{A \in ?F. \text{card } A = k\} = ?Fk \ k$ 
  using *
  by auto

```

```

hence  $\forall k < \text{length } L. \text{ card } \{A \in ?F. \text{ card } A = k\} = L ! k$ 
  using *
  by auto
ultimately
show is-L-part  $n$   $L$   $?F$ 
  unfolding is-L-part-def
  by simp
qed
qed

lemma is-L-part-subset':
  assumes  $F' \subseteq F$  is-L-part  $n$   $L$   $F$ 
  shows  $\exists L'. L \succeq L' \wedge \text{is-L-part } n \ L' \ F'$ 
proof -
  have finite  $(F - F')$ 
    using  $\langle \text{is-L-part } n \ L \ F \rangle$ 
    using is-L-part-def finite-subset[of  $\bigcup F \ \{0..<n\}$ ] finite-subset[of  $F - F'$   $F$ ]
    finiteUn-iff[of  $F$ ]
    by auto
  thus ?thesis
    using assms
  proof (induct  $F - F'$  arbitrary:  $F \ F' \ L$  rule: finite-induct)
    case empty
    hence  $F = F'$ 
      by auto
    thus ?case
      using  $\langle \text{is-L-part } n \ L \ F \rangle$ 
      by (rule-tac  $x=L$  in exI) (auto simp add: pwge-def)
  next
    case (insert  $X \ Y$ )
    have  $X \in F$ 
      using  $\langle \text{insert } X \ Y = F - F' \rangle$ 
      by auto
    then obtain  $L''$  where  $L \succeq L''$  is-L-part  $n$   $L''$   $(F - \{X\})$ 
      using is-L-part-remove[of  $X \ F \ n \ L$ ]  $\langle \text{is-L-part } n \ L \ F \rangle$ 
      using pwge-list-update[of  $\text{card } X \ L \ (L ! \text{card } X) - 1$ ]
      using is-L-part-mem
      by auto
    have  $\exists L'. L'' \succeq L' \wedge \text{is-L-part } n \ L' \ F'$ 
    proof (rule insert(3))
      show  $Y = (F - \{X\}) - F'$ 
        using  $\langle \text{insert } X \ Y = F - F' \rangle \ \langle X \notin Y \rangle$ 
        by auto
    next
      show  $F' \subseteq F - \{X\}$ 
        using  $\langle F' \subseteq F \rangle \ \langle \text{insert } X \ Y = F - F' \rangle$ 
        by auto
    next
      show is-L-part  $n$   $L''$   $(F - \{X\})$ 

```

by *fact*
 qed
 thus ?*case*
 using $\langle L \succeq L'' \rangle$
 using *pwge-trans*[*of L L''*]
 by *auto*
 qed
 qed

16 Generating all L-partitioned families with some given properties

abbreviation *L-part-P* **where**

L-part-P P n L $\equiv \{F \in L\text{-part } n \ L. \ P \ F\}$

abbreviation *mult-P* **where**

mult-P P F A $\equiv \{f \cup \{A\} \mid f. f \in F \wedge A \notin f \wedge P \ A \ f\}$

abbreviation

mult-all-P P F n m $\equiv \bigcup \{mult\text{-}P \ P \ F \ A \mid A. \ card \ A = m \wedge A \subseteq \{0..<n\}\}$

abbreviation *incrementally-checks* **where**

incrementally-checks Pinc P \equiv

$(\forall \ A \ F. (finite (\bigcup F \cup A) \wedge (\forall \ A' \in F. \ card \ A \geq \ card \ A') \wedge A \notin F) \longrightarrow$
 $(P \ (F \cup \{A\}) \longleftrightarrow P \ F \wedge Pinc \ A \ F))$

lemma *L-part-mult-P*:

assumes *incrementally-checks Pinc P*

shows *L-part-P P n (L @ [k + 1]) = mult-all-P Pinc (L-part-P P n (L @ [k]))*
n (length L)

(**is** ?*lhs* = ?*rhs*)

proof

show ?*lhs* \subseteq ?*rhs*

proof(*safe*)

fix *F'*

assume *is-L-part n (L @ [k + 1]) F' P F'*

hence $\bigcup F' \subseteq \{0..<n\} \ card \ \{A \in F'. \ card \ A = \ length \ L\} = k + 1$

unfolding *is-L-part-def*

by *auto*

then obtain *A* **where** *card A = length L A ∈ F'*

using *card-eq-0-iff*[*of {A ∈ F'. card A = length L}*]

by *auto*

let ?*F* = *F' - {A}*

have *F' ∈ {f ∪ {A} | f.*

f ∈ L-part n (L @ [k]) ∧ P f ∧ A ∉ f ∧ Pinc A f}

proof (*rule, rule-tac x=?F in exI, safe*)

show *is-L-part n (L @ [k]) ?F*

using *is-L-part n (L @ [k + 1]) F'*

using $\langle \ card \ A = \ length \ L \rangle \ \langle A \in F' \rangle$ *is-L-part-remove-last*

```

    by auto
next
  show  $A \in F'$ 
  by fact
next
  have  $\text{finite } (\bigcup (F' - \{A\}))$   $\text{finite } A$ 
  using  $\langle \bigcup F' \subseteq \{0..<n\} \rangle$   $\text{finite-subset}[of \bigcup F' \{0..<n\}] \langle A \in F' \rangle$ 
     $\text{finite-subset}[of \bigcup (F' - \{A\}) \bigcup F']$   $\text{finiteUn-iff}[of F']$ 
  by auto
  moreover
  have  $(\forall A' \in F' - \{A\}. \text{card } A' \leq \text{card } A)$ 
  using  $\langle \text{card } A = \text{length } L \rangle$   $\langle \text{is-L-part } n \ (L @ [k + 1]) \ F' \rangle$ 
  unfolding  $\text{is-L-part-def}$ 
  by auto
  ultimately
  show  $\text{Pinc } A \ ?F \ P \ (F' - \{A\})$ 
  using  $\langle P \ F' \rangle \langle A \in F' \rangle$ 
  using  $\text{assms}(1)[\text{rule-format}, of F' - \{A\} \ A]$ 
  by (auto simp add: insert-absorb)
qed
thus  $F' \in ?rhs$ 
  using  $\langle \text{card } A = \text{length } L \rangle \langle A \in F' \rangle \langle \bigcup F' \subseteq \{0..<n\} \rangle$ 
  by blast
qed
next
  show  $?rhs \subseteq ?lhs$ 
  proof
    let  $?F = L\text{-part-}P \ P \ n \ (L @ [k])$ 
    fix  $F'$ 
    assume  $F' \in ?rhs$ 
    then obtain  $A$  where  $\text{card } A = \text{length } L \ A \subseteq \{0..<n\} \ F' \in \text{mult-}P \ \text{Pinc}$ 
       $(L\text{-part-}P \ P \ n \ (L @ [k])) \ A$ 
    by auto
    then obtain  $F$  where  $\text{is-L-part } n \ (L @ [k]) \ F \ P \ F \ A \notin F \ \text{Pinc } A \ F \ F' = F$ 
       $\cup \{A\}$ 
    by auto
    hence  $\text{is-L-part } n \ (L @ [k + 1]) \ F'$ 
    using  $\text{is-L-part-insert-last } \langle \text{card } A = \text{length } L \rangle \langle A \subseteq \{0..<n\} \rangle$ 
    by auto
    moreover
    have  $P \ F'$ 
    using  $\langle F' = F \cup \{A\} \rangle \langle \text{Pinc } A \ F \rangle \langle P \ F \rangle \langle \text{card } A = \text{length } L \rangle \langle \text{is-L-part } n \ (L$ 
       $@ [k]) \ F \rangle$ 
    using  $\text{assms}[\text{rule-format}, of F \ A] \langle A \notin F \rangle$ 
    using  $\langle A \subseteq \{0..<n\} \rangle$   $\text{finite-subset}[of \bigcup F \cup A \ \{0..<n\}]$ 
    unfolding  $\text{is-L-part-def}$ 
    by force
    ultimately
    show  $F' \in ?lhs$ 

```

by simp
qed
qed

abbreviation *inj-preserved* **where**

inj-preserved $P \equiv \forall A F f. (inj-on f (\bigcup F \cup A) \wedge P A F) \longrightarrow P (f ' A) (op ' f ' F)$

lemma *L-part-mult-iso-representing-subset*:

assumes

incrementally-checks Pinc P inj-preserved Pinc

iso-representing-subset FFb (L-part-P P n (L @ [k])) (is iso-representing-subset - ?FFk)

length L ≤ n

FFb' = mult-all-P Pinc FFb n (length L)

shows *iso-representing-subset FFb' (L-part-P P n (L @ [k + 1])) (is iso-representing-subset ?FFb' ?FFk1)*

proof

have *FFb ⊆ ?FFk*

using *assms(3)*

by *auto*

thus *?FFb' ⊆ ?FFk1*

using *L-part-mult-P[of P Pinc, OF assms(1)]*

using *assms(5)*

by *auto*

next

have *iso-represents FFb ?FFk FFb ⊆ ?FFk*

using *assms(3)*

by *auto*

show *iso-represents ?FFb' ?FFk1*

unfolding *iso-represents-def*

proof

fix *F'*

assume *F' ∈ ?FFk1*

then obtain *A where card A = length L A ⊆ {0.. n }*

F' ∈ mult-P Pinc ?FFk A

by *(subst (asm) L-part-mult-P[of P Pinc n L k, OF assms(1)]) auto*

then obtain *F where is-L-part n (L @ [k]) F P F F' = F ∪ {A} A ∉ F Pinc A F*

by *auto*

then obtain *Fb where Fb ∈ FFb iso F Fb*

using *⟨iso-represents FFb ?FFk⟩*

unfolding *iso-represents-def*

by *auto metis*

then obtain *f where *: Fb = op ' f ' F and bij-betw f (⋃ F) (⋃ Fb)*

unfolding *iso-def*

by *auto*

have $\bigcup F \subseteq \{0.. n \} \cup Fb \subseteq \{0.. n \}$

```

using  $\langle is\text{-}L\text{-}part\ n\ (L\ @\ [k])\ F \rangle \langle Fb \in FFb \rangle \langle FFb \subseteq ?FFk \rangle$ 
unfolding  $is\text{-}L\text{-}part\text{-}def$ 
by  $blast+$ 

have  $\exists f'.\ inj\text{-}on\ f'\ (\bigcup F \cup A) \wedge f' \text{ ' } (\bigcup F \cup A) \subseteq \{0..<n\} \wedge (\forall x \in \bigcup F.\ f\ x = f'\ x)$ 
proof–
  obtain  $Bb$  where  $card\ (A - \bigcup F) = card\ (Bb - \bigcup Fb)\ Bb \subseteq \{0..<n\}$ 
    using  $bij\text{-}betw\text{-}complement[of\ \bigcup F\ \{0..<n\}\ \bigcup Fb\ A\ f]$ 
    using  $\langle \bigcup F \subseteq \{0..<n\} \rangle \langle \bigcup Fb \subseteq \{0..<n\} \rangle \langle A \subseteq \{0..<n\} \rangle \langle bij\text{-}betw\ f\ (\bigcup F)\ (\bigcup Fb) \rangle \langle card\ A = length\ L \rangle \langle length\ L \leq n \rangle$ 
    by  $auto$ 
  moreover
    have  $finite\ A$ 
    using  $\langle A \subseteq \{0..<n\} \rangle$ 
    by  $(auto\ simp\ add:\ finite\text{-}subset)$ 
  ultimately
    show  $?thesis$ 
    using  $bij\text{-}betw\text{-}extend[of\ f\ \bigcup F\ \bigcup Fb\ A - \bigcup F\ Bb - \bigcup Fb]$ 
    using  $\langle bij\text{-}betw\ f\ (\bigcup F)\ (\bigcup Fb) \rangle$ 
    using  $\langle Bb \subseteq \{0..<n\} \rangle \langle \bigcup Fb \subseteq \{0..<n\} \rangle$ 
    by  $(auto\ simp\ add:\ finite\text{-}subset\ bij\text{-}betw\text{-}def)\ (metis\ Un\text{-}least)$ 
  qed
  then obtain  $f'$  where  $**:\ inj\text{-}on\ f'\ (\bigcup F \cup A)\ f' \text{ ' } (\bigcup F \cup A) \subseteq \{0..<n\}$ 
 $Fb = op\ \text{ ' } f' \text{ ' } F$ 
    using  $*$ 
    by  $auto\ (metis\ (lifting)\ image\text{-}cong)$ 

show  $\exists Fb' \in ?FFb'.\ iso\ F'\ Fb'$ 
proof
  show  $iso\ F'\ (Fb \cup \{f' \text{ ' } A\})$ 
    using  $iso\text{-}insert(1)[of\ f'\ F\ A]$ 
    using  $**(1)\ **(\beta)\ \langle F' = F \cup \{A\} \rangle$ 
    by  $simp$ 
next
  show  $Fb \cup \{f' \text{ ' } A\} \in ?FFb'$ 
proof–
    have  $Pinc\ (f' \text{ ' } A)\ Fb$ 
    using  $**(1)\ **(\beta)$ 
    using  $\langle Pinc\ A\ F \rangle\ assms(2)[rule\text{-}format,\ of\ f'\ F\ A]$ 
    unfolding  $bij\text{-}betw\text{-}def$ 
    by  $simp$ 
  moreover
    have  $Fb \in FFb\ f' \text{ ' } A \notin Fb\ card\ (f' \text{ ' } A) = length\ L\ \text{and}\ f' \text{ ' } A \subseteq \{0..<n\}$ 
    using  $iso\text{-}insert[of\ f'\ F\ A]$ 
    using  $**$ 
    using  $\langle card\ A = length\ L \rangle \langle A \notin F \rangle \langle Fb \in FFb \rangle$ 
    unfolding  $bij\text{-}betw\text{-}def$ 
    by  $auto$ 

```

```

ultimately
show ?thesis
  using assms(5)
  by blast
qed
qed
qed
qed
end

```

16.1 Implementation by sorted and distinct lists

```

theory LPartitioningImpl
imports LPartitioning FamilyImpl NonIsomorphicFamiliesImpl
begin

```

```

definition mult-P-l where
  mult-P-l P F A =
    (let F' = [f ← F. A ∉ set f ∧ P A f] in
     map (λ x. A # x) F')

```

```

lemma mult-P-l-sorted-distinct:
  assumes sdff F sd A
  shows sdff (mult-P-l P F A)
using assms
unfolding mult-P-l-def
by auto

```

```

lemma mult-P-l-domain:
  assumes dmf FFb n set A ⊆ {0.. $n$ ::nat}
  shows dmf (mult-P-l P FFb A) n
using assms
unfolding mult-P-l-def
by auto

```

```

lemma mult-P-l-correctness:
  fixes FFb :: nat list list list
  assumes sdff FFb sd A dmf FFb n set A ⊆ {0.. $n$ }
  assumes ⋀ A F. [sd A; sdf F; dm (A # F) n] ⇒ P A F ⟷ P' (set A)
  (f-to-set-l F)
  shows fs-to-set-l (mult-P-l P FFb A) = mult-P P' (fs-to-set-l FFb) (set A) (is
  ?lhs = ?rhs)
proof
  show ?lhs ⊆ ?rhs
  proof
    fix x
    assume x ∈ ?lhs
    then obtain F where F ∈ set FFb A ∉ set F P A F x = f-to-set-l (A # F)

```



```

    unfolding mult-P-l-def Let-def
    by auto
  have  $x = \text{insert } (\text{set } A) (f\text{-to-set-l } F)$ 
    using  $\langle x = f\text{-to-set-l } (A \# F) \rangle$ 
    by auto
  moreover
  have  $f\text{-to-set-l } F \in f\text{-to-set-l ' set } FFb$ 
    using  $\langle F \in \text{set } FFb \rangle$ 
    by simp
  moreover
  have  $\text{set } A \notin f\text{-to-set-l } F$ 
    using  $\langle F \in \text{set } FFb \rangle \langle A \notin \text{set } F \rangle \text{SetImpl-lists.set-set}[of F A] \text{assms}(1)$ 
    assms(2)
    by auto
  moreover
  have  $P' (\text{set } A) (f\text{-to-set-l } F)$ 
    using  $\langle sd A \rangle \langle sdff FFb \rangle \langle F \in \text{set } FFb \rangle \langle \text{set } A \subseteq \{0..<n\} \rangle \langle dmf FFb n \rangle$ 
    using  $\langle P A F \rangle \text{assms}(5)[of A F]$ 
    by simp
  ultimately
  show  $x \in ?rhs$ 
    by auto
qed
next
show  $?rhs \subseteq ?lhs$ 
proof
  fix  $x$ 
  assume  $x \in ?rhs$ 
  then obtain  $F$  where  $F \in \text{set } FFb$   $\text{set } A \notin f\text{-to-set-l } F$   $P' (\text{set } A) (f\text{-to-set-l } F)$ 
     $x = f\text{-to-set-l } F \cup \{\text{set } A\}$ 
    by auto
  thus  $x \in ?lhs$ 
    using  $\text{assms}(5)[of A F] \langle sdff FFb \rangle \langle sd A \rangle \langle dmf FFb n \rangle \langle \text{set } A \subseteq \{0..<n\} \rangle$ 
    unfolding mult-P-l-def
    apply auto
    apply (rule image-eqI[where  $x = F$ ])
    by auto
qed
qed

```

definition *mult-all-P-l* **where**

$\text{mult-all-P-l } P F n m = \text{concat } (\text{map } (\lambda A. \text{mult-P-l } P F A) (\text{all-mn-subsets } n m))$

lemma *mult-all-P-l-sorted-distinct*:

assumes $sdff F$
 shows $sdff (\text{mult-all-P-l } P F n m)$
 unfolding *mult-all-P-l*-def

```

proof (simp)
  show  $\forall A \in \text{set } (\text{all-mn-subsets } n \ m). \text{sdff } (\text{mult-P-l } P \ F \ A)$ 
proof
  fix  $A$ 
  assume  $A \in \text{set } (\text{all-mn-subsets } n \ m)$ 
  hence  $sd \ A$ 
  by (rule all-mn-subsets-sorted-distinct)
  thus  $\text{sdff } (\text{mult-P-l } P \ F \ A)$ 
  using assms mult-P-l-sorted-distinct[of F A P]
  by blast
qed
qed

lemma mult-all-P-l-domain:
  assumes  $dmf \ FFb \ n$ 
  shows  $dmf \ (\text{mult-all-P-l } P \ FFb \ n \ m) \ n$ 
unfolding mult-all-P-l-def
proof(simp del: SetImpl-lists.f-to-set-def, rule ballI, rule ballI)
  fix  $A \ F$ 
  assume  $A \in \text{set } (\text{all-mn-subsets } n \ m) \ F \in \text{set } (\text{mult-P-l } P \ FFb \ A)$ 
  thus  $dm \ F \ n$ 
  using mult-P-l-domain[OF assms, of A P]
  using all-mn-subsets[of n m]
  by auto
qed

lemma mult-all-P-correctness:
  fixes  $FFb :: \text{nat list list list}$ 
  assumes  $\text{sdff } FFb \ dmf \ FFb \ n$ 
   $\bigwedge A \ F. \llbracket sd \ A; \text{sdff } F; dm \ (A \ \# \ F) \ n \rrbracket \implies P \ A \ F \longleftrightarrow P' \ (\text{set } A) \ (\text{f-to-set-l } F)$ 
  shows  $\text{fs-to-set-l } (\text{mult-all-P-l } P \ FFb \ n \ m) = \text{mult-all-P } P' \ (\text{fs-to-set-l } FFb) \ n \ m$ 
  (is  $?lhs = ?rhs$ )
proof
  show  $?lhs \subseteq ?rhs$ 
proof
  fix  $x$ 
  assume  $x \in ?lhs$ 
  then obtain  $A$  where  $A \in \text{set } (\text{all-mn-subsets } n \ m) \ x \in \text{fs-to-set-l } (\text{mult-P-l } P \ FFb \ A)$ 
  unfolding mult-all-P-l-def
  by auto
moreover
  have  $sd \ A$ 
  using  $\langle A \in \text{set } (\text{all-mn-subsets } n \ m) \rangle$ 
  by (rule all-mn-subsets-sorted-distinct)
  ultimately
  show  $x \in ?rhs$ 
  using all-mn-subsets[of n m] assms(1) assms(2)
  by (subst (asm) mult-P-l-correctness[where  $P=P$  and  $P'=P'$ , OF - - -
```

```

assms(3)) auto
qed
next
  show  $?rhs \subseteq ?lhs$ 
  proof
    fix  $x$ 
    assume  $x \in ?rhs$ 
    then obtain  $A$  where  $\text{card } A = m \wedge A \subseteq \{0..<n\} \wedge x \in \text{mult-}P \ P' \ (\text{fs-to-set-l}$ 
 $\text{FFb}) \ A$ 
    by auto
    obtain  $Al$  where  $Al \in \text{set } (all-mn-subsets \ n \ m) \wedge \text{set } Al = A \text{ sorted } Al \wedge \text{distinct}$ 
 $Al$ 
    using  $\langle \text{card } A = m \rangle \langle A \subseteq \{0..<n\} \rangle$ 
    using all-mn-subsets-completeness [of  $A \ m \ n$ ] all-mn-subsets-sorted-distinct
    by auto
    hence  $x \in \text{mult-}P \ P' \ (\text{fs-to-set-l} \ \text{FFb}) \ (\text{set } Al)$ 
    using  $\langle x \in \text{mult-}P \ P' \ (\text{fs-to-set-l} \ \text{FFb}) \ A \rangle$ 
    by simp
    thus  $x \in ?lhs$ 
    using  $\langle Al \in \text{set } (all-mn-subsets \ n \ m) \rangle \langle \text{assms}(1) \rangle \langle \text{sd } Al \rangle \langle \text{assms}(2) \rangle$ 
    unfolding mult-all-P-l-def
    by (subst (asm) mult-P-l-correctness[symmetric, of FFb  $Al \ n \ P \ P'$ , OF - - -
- assms(3)) auto
qed
qed

```

definition *mult-all-base-P-l* **where**

mult-all-base-P-l $P \ F \ n \ m \ \text{perms} = \text{non-isomorphic-families-l} \ \text{perms} \ (\text{mult-all-P-l}$
 $P \ F \ n \ m)$

lemma *mult-all-base-P-l-sorted-distinct*:

```

assumes sdff  $F$ 
shows sdff (mult-all-base-P-l  $P \ F \ n \ m \ \text{perms}$ )
proof–
  have sdff (mult-all-P-l  $P \ F \ n \ m$ )
  using assms
  by (rule mult-all-P-l-sorted-distinct)
  thus ?thesis
  unfolding mult-all-base-P-l-def
  using SetPermutations-lists.non-isomorphic-families-subset[of perms mult-all-P-l
 $P \ F \ n \ m$ ]
  by auto
qed

```

lemma *mult-all-base-P-l-domain*:

```

assumes dmf  $F \ n$ 
shows dmf (mult-all-base-P-l  $P \ F \ n \ m \ \text{perms}$ )  $n$ 
proof–

```

```

have dmf (mult-all-P-l P F n m) n
  using assms
  by (rule mult-all-P-l-domain)
thus ?thesis
  unfolding mult-all-base-P-l-def
  using SetPermutations-lists.non-isomorphic-families-subset[of perms mult-all-P-l
P F n m]
  by auto
qed

lemma mult-all-base-P-l-correctness:
  assumes  $\bigwedge A F. \llbracket sd\ A; sdf\ F; dm\ (A \# F)\ n \rrbracket \implies Pinc'\ A\ F \longleftrightarrow Pinc\ (set\ A)\ (f\text{-to-set-l}\ F)$ 
    incrementally-checks Pinc P inj-preserved Pinc
  assumes sdiff FFb dmf FFb n  $\forall p \in set\ perms. p <\sim\sim> [0..<n]$ 
  assumes iso-representing-subset (fs-to-set-l FFb) (L-part-P P n (L @ [k]))
    length L  $\leq n$ 
    FFb' = fs-to-set-l (mult-all-base-P-l Pinc' FFb n (length L) perms)
  shows iso-representing-subset FFb' (L-part-P P n (L @ [k+1]))
proof-
  have iso-representing-subset (fs-to-set-l (mult-all-P-l Pinc' FFb n (length L)))
    (L-part-P P n (L @ [k+1]))
  proof (rule L-part-mult-iso-representing-subset[where FFb'=fs-to-set-l (mult-all-P-l
Pinc' FFb n (length L)) and n=n and L=L and k=k and P=P and FFb=fs-to-set-l
FFb and Pinc=Pinc, OF assms(2)])
    show iso-representing-subset (fs-to-set-l FFb) (L-part-P P n (L @ [k])) length
    L  $\leq n$ 
    by fact+
  next
    show fs-to-set-l (mult-all-P-l Pinc' FFb n (length L)) = mult-all-P Pinc
    (fs-to-set-l FFb) n (length L)
    by (rule mult-all-P-correctness[of FFb n Pinc' Pinc, OF assms(4) assms(5)
assms(1)])
  next
    show inj-preserved Pinc
    by fact
qed
thus ?thesis
  apply (subst assms(9))+
  unfolding mult-all-base-P-l-def
  using iso-representing-subset-non-isomorphic-families-l[OF assms(6), of mult-all-P-l
Pinc' FFb n (length L)]
  using mult-all-P-l-domain[OF assms(5), of Pinc' length L]
  by auto
qed

```

16.2 Recursive enumeration procedure

For a given list L , find an iso-representing collection of all L -partitioned families that satisfy some predicate P .

abbreviation *dec-last* **where**

dec-last $L \equiv \text{butlast } L @ [\text{last } L - (1::\text{nat})]$

function *enum-rec* **where**

enum-rec $L \ v0 \ f =$
 (*if* $L = []$ *then* $v0$
 else if $\text{last } L = 0$ *then* *enum-rec* (*butlast* L) $v0 \ f$
 else $f \ (\text{enum-rec} \ (\text{dec-last } L) \ v0 \ f) \ L$)

by *pat-completeness auto*

termination

proof–

let $?R = \text{measure } (\lambda (L, -, -). \text{length } L + \text{sum-list } L)$

show *?thesis*

proof(*relation* $?R$)

fix $L :: \text{nat list}$ **and** $v0 :: 'a$ **and** $f :: 'a \Rightarrow \text{nat list} \Rightarrow 'a$

assume $L \neq []$ $\text{last } L = 0$

hence $\text{sum-list } L = \text{sum-list } (\text{butlast } L)$

by (*subst append-butlast-last-id*[*of* L , *symmetric*]) *auto*

thus $((\text{butlast } L, v0, f), L, v0, f) \in ?R$

using $\langle L \neq [] \rangle$

by *simp*

next

fix $L :: \text{nat list}$ **and** $v0 :: 'a$ **and** $f :: 'a \Rightarrow \text{nat list} \Rightarrow 'a$

assume $L \neq []$ $\text{last } L \neq 0$

hence $\text{sum-list } L = \text{sum-list } (\text{butlast } L) + \text{last } L$

using *sum-list-append*[*of* $\text{butlast } L$ [$\text{last } L$]]

by (*subst append-butlast-last-id*[*of* L , *symmetric*]) *auto*

thus $((\text{butlast } L @ [\text{last } L - 1], v0, f), L, v0, f) \in ?R$

using $\langle L \neq [] \rangle \langle \text{last } L \neq 0 \rangle$

by *simp*

qed *simp*

qed

declare *enum-rec.simps*[*simp del*]

lemma *enum-rec*:

assumes $P \ v0 \ []$

assumes $\bigwedge v \ L. \ P \ v \ L \implies P \ v \ (L @ [0])$

$\bigwedge v \ L' :: (\text{nat list}). \ \llbracket L' \neq []; \text{last } L' > 0; \text{length } L' \leq \text{length } L; P \ v \ (\text{dec-last } L') \rrbracket \implies P \ (f \ v \ L') \ L'$

shows $P \ (\text{enum-rec } L \ v0 \ f) \ L$

using *assms*

proof (*induct* $L \ v0 \ f$ *rule*: *enum-rec.induct*)

case ($1 \ L \ v0 \ f$)

show *?case*

proof (*cases* $L = []$)

```

    case True
    thus ?thesis
      using 1(3)
      by (simp add: enum-rec.simps)
  next
    case False
    show ?thesis
    proof (cases last L = 0)
      case True
      thus ?thesis
        using ⟨L ≠ []⟩
        using enum-rec.simps[of L v0 f]
        using 1(1)[OF ⟨L ≠ []⟩ ⟨last L = 0⟩ 1(3) 1(4) 1(5)] 1(4)[of enum-rec
(butlast L) v0 f butlast L]
        using append-butlast-last-id[of L]
        by auto
    next
      case False
      thus ?thesis
        using ⟨L ≠ []⟩
        using enum-rec.simps[of L v0 f]
        using 1(2)[OF ⟨L ≠ []⟩ ⟨last L ≠ 0⟩ 1(3) 1(4) 1(5)]
        using 1(5)
        by (smt append-butlast-last-id length-append-singleton neq0-conv order-refl)
    qed
  qed
qed

```

```

lemma enum-rec-iso-representing-subset:
  assumes  $\bigwedge A F. \llbracket sd\ A; sdf\ F; dm\ (A \# F)\ n \rrbracket \implies Pinc'\ A\ F \longleftrightarrow Pinc\ (set\ A)\ (f\text{-to-set-}l\ F)$  and
    incrementally-checks Pinc P and inj-preserved Pinc and
 $\forall p \in set\ perms. p <^{\sim\sim} [0..<n]$  and length L - 1 ≤ n
  assumes P {} Mult = (λ F L. mult-all-base-P-l Pinc' F n (length L - 1) perms)
  shows iso-representing-subset (fs-to-set-l(enum-rec L [] Mult)) (L-part-P P n L)
proof(rule enum-rec[of λ X L. iso-representing-subset (fs-to-set-l X) (L-part-P P n L) ∧ sdff X ∧ dmf X n, THEN conjunct1])
  show iso-representing-subset (fs-to-set-l []) (L-part-P P n []) ∧ sdff [] ∧ dmf [] n
  (is iso-representing-subset ?X ?Y ∧ -)
proof—
  have ?Y = {{} }
    using is-L-part-zero[of 6 [], symmetric] ⟨P {}⟩
    unfolding is-L-part-def
    by auto
  thus ?thesis
    by (auto simp add: iso-represents-def)
  qed
next

```

```

fix v L
assume iso-representing-subset (fs-to-set-l v) (L-part-P P n L)  $\wedge$  sdff v  $\wedge$  dmf v
n
thus iso-representing-subset (fs-to-set-l v) (L-part-P P n (L @ [0]))  $\wedge$  sdff v  $\wedge$ 
dmf v n
using is-L-part-zero
by auto
next
fix v and L'::nat list
assume L'  $\neq []$  last L' > 0 length L'  $\leq$  length L
hence **: length (butlast L')  $\leq$  n
using  $\langle \text{length } L - 1 \leq n \rangle$ 
by auto
assume iso-representing-subset (fs-to-set-l v) (L-part-P P n (dec-last L'))  $\wedge$  sdff
v  $\wedge$  dmf v n
hence *: sdff v dmf v n iso-representing-subset (fs-to-set-l v) (L-part-P P n
(dec-last L'))
by auto
have iso-representing-subset (fs-to-set-l (Mult v L')) (L-part-P P n (butlast L' @
[ $\text{last } L' - 1 + 1$ ]))
using  $\langle L' \neq [] \rangle$ 
by (subst assms(7))+ (rule mult-all-base-P-l-correctness[OF assms(1-3) * (1-2)
assms(4) * (3) ** (1)], auto)
hence iso-representing-subset (fs-to-set-l (Mult v L')) (L-part-P P n L') (is ?T1)
using  $\langle L' \neq [] \rangle$   $\langle \text{last } L' > 0 \rangle$ 
by simp
moreover
have sdff (Mult v L')  $\wedge$  dmf (Mult v L') n
using *(1) *(2)
using mult-all-base-P-l-sorted-distinct[of v Pinc' n length L' - 1 perms]
using mult-all-base-P-l-domain[of v n Pinc' length L' - 1 perms]
by (subst assms(7))+ blast
ultimately
show iso-representing-subset (fs-to-set-l (Mult v L')) (L-part-P P n L')  $\wedge$ 
sdff (Mult v L')  $\wedge$  dmf (Mult v L') n
by simp
qed

```

16.3 Dynamic programming enumeration procedure

definition pwge-impl **where**

pwge-impl L L' \longleftrightarrow list-all ($\lambda (x, y). x \geq y$) (zip L L')

lemma pwge-impl:

assumes length L = length L'

shows pwge-impl L L' \longleftrightarrow L \succeq L'

using assms

unfolding pwge-impl-def pwge-def

by (auto simp add: list-all-iff set-zip)

function *enum-dp* **where**
enum-dp *val* *res* *k* *L* *g* *stop* *maxL* = *foldl*
 (λ *r* *n*. *let* *l* = *inc-nth* *L* *n*
 in if *length* *l* ≠ *length* *maxL* ∨ *stop* *l* ∨ ¬ *pwge-impl* *maxL* *l* *then*
 r
 else
 enum-dp (*g* *val* *n*) *r* *n* *l* *g* *stop* *maxL*
)
 (*val* # *res*)
 [*k*..*length* *L*]
by *pat-completeness auto*

abbreviation *enum-dp-step* **where**
enum-dp-step *val* *res* *k* *g* *stop* *maxL* *L* ≡ *let* *l* = *inc-nth* *L* *k*
 in if *length* *l* ≠ *length* *maxL* ∨
 stop *l* ∨
 ¬ *pwge-impl* *maxL* *l*
 then *res* *else* *enum-dp* (*g* *val* *k*) *res* *k* *l* *g* *stop* *maxL*

definition *listdiff* **where**
listdiff *xs* *ys* = *sum-list* (*map* (λ (*a*, *b*). *a* − *b*) (*zip* *xs* *ys*))

termination

proof (*relation measure* (λ (*val*, *res*, *k*, *L*, *g*, *stop*, *maxL*). *listdiff* (*map* (*op*+1) *maxL*) *L*))
fix *val* *res* *k* *L* *g* *stop* *maxL* *r* *n* *l*
assume *l* = *inc-nth* *L* *n* *n* ∈ *set* [*k*..*length* *L*]
and *: ¬ (*length* *l* ≠ *length* *maxL* ∨ *stop* *l* ∨ ¬ *pwge-impl* *maxL* *l*)
have **: ∀ *x*. *x* < *length* *maxL* → *l* ! *x* ≤ *maxL* ! *x* *length* *l* = *length* *maxL* *n*
 < *length* *L*
using * ⟨*n* ∈ *set* [*k*..*length* *L*⟩
by (*auto simp add: pwge-impl-def list-all-iff set-zip*)
have ∀ *x* ∈ {0..*length* *maxL*}.
 Suc (*maxL* ! *x*) − *inc-nth* *L* *n* ! *x* ≤ *Suc* (*maxL* ! *x*) − *L* ! *x*
using ⟨*n* < *length* *L*⟩
by (*auto simp add: nth-list-update*)
moreover
have *Suc* (*maxL* ! *n*) − *inc-nth* *L* *n* ! *n* < *Suc* (*maxL* ! *n*) − *L* ! *n*
using ⟨*n* < *length* *L*⟩ ⟨*l* = *inc-nth* *L* *n*⟩ **
by (*auto simp add: nth-list-update split: if-split-asm*)
ultimately
have *listdiff* (*map* (*op*+1) *maxL*) *l* < *listdiff* (*map* (*op*+1) *maxL*) *L*
using ⟨*l* = *inc-nth* *L* *n*⟩ ⟨*length* *l* = *length* *maxL*⟩ ⟨*n* < *length* *L*⟩
unfolding *listdiff-def*
by (*auto simp add: list-ex-iff sum-list-sum-nth nth-list-update*)
 (*rule sum-strict-mono-ex1, auto*)
thus ((*g* *val* *n*, *r*, *n*, *l*, *g*, *stop*, *maxL*), (*val*, *res*, *k*, *L*, *g*, *stop*, *maxL*))
 ∈ *measure* (λ(*val*, *res*, *k*, *L*, *g*, *stop*, *maxL*). *listdiff* (*map* (*op*+1) *maxL*))


```

L)
  by simp
qed simp
declare enum-dp.simps[simp del]

lemma enum-dp-mono:
  assumes length L = length maxL
  shows set (val # res) ⊆ set (enum-dp val res k L g stop maxL)
  using assms
  proof (induct val res k L g stop maxL rule: enum-dp.induct)
    case (1 val res k L g stop maxL)
    show ?case
    proof (subst enum-dp.simps, subst foldl-conv-fold, rule fold-invariant[where
      Q=λ x. x ∈ set [k..

```

```

      P (g val k') (inc-nth L k')
    shows  $\forall L' \in X. \exists B \in \text{set } (\text{enum-dp val res } k \text{ } L \text{ } g \text{ stop maxL}). P \ B \ L'$ 
  using assms
  proof (induct val res k L g stop maxL arbitrary: X rule: enum-dp.induct)
    case (1 val res k L g stop maxL)
    show ?case
    proof
      fix L'
      assume L'  $\succeq$  L
      hence *: L'  $\succeq$  L maxL  $\succeq$  L' take k L' = take k L  $\neg$  ( $\exists Ls. Ls \succeq L \wedge \text{take } k \ L$ 
= take k Ls  $\wedge$  L'  $\succeq$  Ls  $\wedge$  stop Ls)
      using 1(4)
      by auto
    show  $\exists B \in \text{set } (\text{enum-dp val res } k \text{ } L \text{ } g \text{ stop maxL}). P \ B \ L'$ 
    proof (cases L = L')
      case True
      thus ?thesis
      using 1(2) 1(5)
      using enum-dp-mono[of L maxL val res k g stop]
      by auto
    next
      case False
      have  $\exists k'. k' \geq k \wedge k' < \text{length } L \wedge \text{take } k' \ L' = \text{take } k' \ L \wedge L' \succeq (\text{inc-nth}$ 
L k')  $\wedge$  ( $\forall k''. k'' > k' \wedge k'' < \text{length } L \longrightarrow L ! k'' = 0$ )
      using  $\langle \text{take } k \ L' = \text{take } k \ L \rangle$ 
      using 1(3) 1(6)
    proof (induct length L - k arbitrary: k)
      case 0
      thus ?case
      by simp
    next
      case (Suc p)
      show ?case
      proof (cases L'  $\succeq$  inc-nth L k)
        case True
        thus ?thesis
        using Suc(3) Suc(4) Suc(5)
        by (rule-tac x=k in exI) simp
      next
        case False
        hence L' ! k = L ! k length L = length L'
        using Suc(3)  $\langle L' \succeq L \rangle$ 
        unfolding pwge-def
        by auto (metis Suc-diff-Suc Suc-leI Suc-neq-Zero diff-is-0-eq linorder-cases
nth-list-update-eq nth-list-update-neq)
      moreover
      hence Suc k < length L
      using  $\langle k < \text{length } L \rangle$ 

```

```

    using  $\langle L \neq L' \rangle$   $Suc(3)$  list-eq-iff-nth-eq[of  $L\ L'$ ]
    by auto (metis less-linear less-trans-Suc nth-take)
ultimately
have  $take\ (Suc\ k)\ L' = take\ (Suc\ k)\ L$ 
  using  $Suc(3)$ 
  using take-Suc-conv-app-nth[of  $k\ L$ ]
  using take-Suc-conv-app-nth[of  $k\ L'$ ]
  by auto
moreover
have  $p = length\ L - Suc\ k$ 
  using  $Suc(2)$   $\langle Suc\ k < length\ L \rangle$ 
  by auto
moreover
have  $\forall k''. k'' > Suc\ k \wedge k'' < length\ L \longrightarrow L ! k'' = 0$ 
  using  $Suc(4)$ 
  by simp
ultimately
obtain  $k'$  where  $k' \geq Suc\ k\ k' < length\ L\ take\ k'\ L' = take\ k'\ L\ L' \succeq$ 
inc-nth  $L\ k' \forall k''. k'' > k' \wedge k'' < length\ L \longrightarrow L ! k'' = 0$ 
  using  $\langle Suc\ k < length\ L \rangle$ 
  using  $Suc(1)$ [of  $Suc\ k$ ]
  by auto
thus ?thesis
  by (rule-tac  $x=k'$  in exI) simp
qed
qed

then obtain  $k'$  where **:  $k' \geq k\ k' < length\ L\ take\ k'\ L' = take\ k'\ L\ L' \succeq$ 
(inc-nth  $L\ k') \forall k''. k'' > k' \wedge k'' < length\ L \longrightarrow L ! k'' = 0$ 
  by auto
let ?l = inc-nth  $L\ k'$ 
let ?X' =  $\{L'. L' \succeq ?l \wedge maxL \succeq L' \wedge take\ k'\ L' = take\ k'\ ?l \wedge \neg (\exists\ Ls. Ls \succeq ?l \wedge take\ k'\ ?l = take\ k'\ Ls \wedge L' \succeq Ls \wedge stop\ Ls)\}$ 

have  $?l \succeq L$ 
  using  $\langle k' < length\ L \rangle$ 
  unfolding pwge-def
  by (auto simp add: nth-list-update)
have  $take\ k\ L = take\ k\ ?l$ 
  using  $\langle k' < length\ L \rangle \langle k \leq k' \rangle$ 
  by simp

have  $\neg stop\ ?l$ 
  using *  $\langle k' \geq k \rangle \langle ?l \succeq L \rangle \langle take\ k\ L = take\ k\ ?l \rangle \langle L' \succeq (inc-nth\ L\ k') \rangle$ 
  by metis

let ?a = foldl  $(\lambda r\ n. enum-dp-step\ val\ r\ n\ g\ stop\ maxL\ L)\ (val\ \# res)\ [k..<k']$ 

have  $\exists B \in set\ (enum-dp\ (g\ val\ k')\ ?a\ k'\ (inc-nth\ L\ k')\ g\ stop\ maxL). P\ B$ 

```

```

L'
  proof (rule 1(1)[rule-format])
    show L' ∈ ?X'
  proof-
    have ¬ (∃ Ls. Ls ⊇ inc-nth L k' ∧ take k' (inc-nth L k') = take k' Ls ∧
L' ⊇ Ls ∧ stop Ls)
    proof (rule ccontr)
      assume ¬ ?thesis
      then obtain Ls where Ls ⊇ ?l take k' ?l = take k' Ls L' ⊇ Ls stop Ls
      by auto
      hence Ls ⊇ L ∧ take k L = take k Ls ∧ L' ⊇ Ls ∧ stop Ls
      using ⟨?l ⊇ L⟩ pwge-trans[of Ls ?l L] ⟨k' ≥ k⟩ ⟨k' < length L⟩
      using take-prefix[of k k' L Ls]
      by simp
      thus False
      using *(4)
      by auto
    qed
    thus ?thesis
    using * **
    by auto
  qed
next
show k' ∈ set [k..<length L]
  using **
  by simp
next
show ¬ (length (inc-nth L k') ≠ length maxL ∨
  stop (inc-nth L k') ∨
  ¬ pwge-impl maxL (inc-nth L k'))
  using 1(2) * ⟨¬ stop ?l⟩
  using *(4)
  using pwge-trans[of maxL L' ?l]
  by (auto simp add: pwge-impl)
next
show P (g val k') (inc-nth L k')
  using 1(5) 1(7) *(5) ⟨k' < length L⟩ ⟨length L = length maxL⟩
  by simp
next
show length (inc-nth L k') = length maxL
  using 1(2)
  by simp
next
show k' < length ?l
  using ⟨k' < length L⟩
  by simp
next
fix val L k'
assume P val L ∧ k''. k' < k'' ∧ k'' < length L ⟹ L ! k'' = 0 k' < length

```

```

L k' < length maxL
  thus P (g val k') (inc-nth L k')
    using 1(7)
    by simp
next
  fix k''
  assume k' < k'' ∧ k'' < length (inc-nth L k')
  thus inc-nth L k' ! k'' = 0
    using **(5)
    by auto
qed simp-all
hence ++: ∃ B ∈ set (enum-dp-step val ?a k' g stop maxL L). P B L'
  using ⟨length L = length maxL⟩ ⟨¬ stop ?l⟩
  using ⟨maxL ⋮ L'⟩ ⟨L' ⋮ ?l⟩
  using pwge-trans[of maxL L' ?l]
  by (auto simp add: Let-def pwge-impl)

have [k..

```

Incrementally build all generating subsets for L-partitioned lists upto the given bounds

definition *enum-dp-mult-P* **where**

```

enum-dp-mult-P P n perms stops maxL =
  enum-dp [[]] [] 0 (replicate (n+1) (0::nat))
    (λ FFb m. mult-all-base-P-l P FFb n m perms)
    (λ L. list-ex (λ L'. pwge-impl L L') stops)
    maxL

```

lemma *enum-dp-mult-P-correct*:

```

assumes  $\bigwedge A F. \llbracket sd\ A; sdf\ F; dm\ (A \# F)\ n \rrbracket \implies Pinc'\ A\ F \longleftrightarrow Pinc\ (set\ A)$ 
(f-to-set-l F)
  incrementally-checks Pinc P inj-preserved Pinc
assumes
   $X = \{L'.\ maxL \succeq L' \wedge \neg (\exists S \in set\ stops.\ L' \succeq S)\}$  (is - = ?lhs) and
   $P\ \{\}$  and
   $length\ maxL = n+1$  and
   $\forall S \in set\ stops.\ length\ S = n+1$  and
   $\forall p \in set\ perms.\ p < \sim \sim > [0..<n]$ 
shows  $\forall L' \in X.\ \exists B \in set\ (enum-dp-mult-P\ Pinc'\ n\ perms\ stops\ maxL).$ 
  iso-representing-subset (fs-to-set-l B) (L-part-P P n L')
proof–
  have  $\forall L' \in X.\ \exists B \in set\ (enum-dp-mult-P\ Pinc'\ n\ perms\ stops\ maxL).$ 
    iso-representing-subset (fs-to-set-l B) (L-part-P P n L') \wedge sdff\ B \wedge dmf\ B
  n
    unfolding enum-dp-mult-P-def
    proof (rule enum-dp-lemma)
      show  $length\ (replicate\ (n+1)\ (0::nat)) = length\ maxL$ 
        using  $\langle length\ maxL = n+1 \rangle$ 
        by simp
      next
        show  $\forall k''.\ 0 < k'' \wedge k'' < length\ (replicate\ (n+1)\ (0::nat)) \longrightarrow$ 
           $replicate\ (n+1)\ (0::nat)\ !\ k'' = 0$ 
          by auto
        next
          show  $0 < length\ (replicate\ (n+1)\ (0::nat))$ 
            by simp
          next
            show  $iso-representing-subset\ (fs-to-set-l\ [\ ])\ (L-part-P\ P\ n\ (replicate\ (n+1)\$ 
               $0)) \wedge sdff\ [\ ] \wedge dmf\ [\ ]\ n$ 
            proof–
              have  $*$ :  $L-part-P\ P\ n\ (replicate\ (n+1)\ 0) = \{\{\}\}$ 
                proof–
                  have  $*$ :  $L-part-P\ P\ n\ (replicate\ (n+1)\ 0) = L-part-P\ P\ n\ []$ 
                    using is-L-part-zeros-replicate[of n n+1]
                    by simp
                  show ?thesis
                  proof (subst *, rule)
                    show  $L-part-P\ P\ n\ [] \subseteq \{\{\}\}$ 
                      unfolding is-L-part-def
                      by auto
                    next
                      show  $\{\{\}\} \subseteq L-part-P\ P\ n\ []$ 
                        using  $\langle P\ \{\} \rangle$ 
                        by (simp add: is-L-part-def)
                      qed
                    qed
                  show ?thesis
                    by (subst *)+ (auto simp add: iso-represents-def)
            qed

```

```

qed
next
  fix val L k'
  assume *: iso-representing-subset (fs-to-set-l val) (L-part-P P n L) ∧ sdff val
  ∧ dmf val n and
    k' < length L ∀ k''. k'' > k' ∧ k'' < length L ⟶ L ! k'' = 0 k' < length maxL
  moreover
  have take (k'+1) L = take k' L @ [L ! k']
    using ⟨k' < length L⟩ take-Suc-conv-app-nth[of k' L]
    by simp
  ultimately
  have iso-representing-subset (fs-to-set-l val) (L-part-P P n (take k' L @ [L !
k']))
    using is-L-part-zeros[of k' L n]
    by auto
  have iso-representing-subset (fs-to-set-l (mult-all-base-P-l Pinc' val n k' perms))
(L-part-P P n (take k' L @ [(L ! k') + 1]))
    proof (rule mult-all-base-P-l-correctness[OF assms(1-2)])
      show iso-representing-subset (fs-to-set-l val) (L-part-P P n (take k' L @ [L !
k']))
        by fact
    next
      show fs-to-set-l (mult-all-base-P-l Pinc' val n k' perms) =
        fs-to-set-l (mult-all-base-P-l Pinc' val n (length (take k' L)) perms)
        using ⟨k' < length L⟩
        by (simp add: min-def)
    next
      show length (take k' L) ≤ n
        using ⟨length maxL = n + 1⟩ ⟨k' < length maxL⟩
        by simp
    next
      show sdff val dmf val n
        using *
        by simp-all
    next
      show ∀ p ∈ set perms. p <~> [0..<n]
        by fact
    next
      show inj-preserved Pinc
        by fact
  qed simp-all
  moreover
  have take (k'+1) (inc-nth L k') = take k' L @ [Suc (L ! k')]
    using ⟨k' < length L⟩ take-Suc-conv-app-nth[of k' inc-nth L k']
    by auto
  ultimately
  have iso-representing-subset (fs-to-set-l (mult-all-base-P-l Pinc' val n k' perms))
(L-part-P P n (take (k'+1) (inc-nth L k')))
    by simp

```

```

hence iso-representing-subset (fs-to-set-l (mult-all-base-P-l Pinc' val n k' perms))
(L-part-P P n (inc-nth L k')) (is ?T1)
  using is-L-part-zeros[of k' inc-nth L k' n]  $\langle \forall k''. k'' > k' \wedge k'' < \text{length } L \longrightarrow$ 
L ! k'' = 0  $\rangle$ 
  by auto
moreover
have sdff (mult-all-base-P-l Pinc' val n k' perms) (is ?T2)
  using *[THEN conjunct2, THEN conjunct1]
  by (rule mult-all-base-P-l-sorted-distinct)
moreover
have dmf (mult-all-base-P-l Pinc' val n k' perms) n (is ?T3)
  using *[THEN conjunct2, THEN conjunct2]
  by (rule mult-all-base-P-l-domain)
ultimately
show ?T1  $\wedge$  ?T2  $\wedge$  ?T3
  by simp
next
show X =
  {L'.
   L'  $\succeq$  replicate (n+1) 0  $\wedge$  maxL  $\succeq$  L'  $\wedge$  take 0 L' = take 0 (replicate (n+1)
0)  $\wedge$ 
    $\neg (\exists Ls. Ls \succeq \text{replicate } (n+1) 0 \wedge$ 
     take 0 (replicate (n+1) 0) = take 0 Ls  $\wedge$ 
     L'  $\succeq$  Ls  $\wedge$  list-ex (pwge-impl Ls) stops)} (is - = ?rhs)
proof (subst assms(4), rule)
  show ?lhs  $\subseteq$  ?rhs
proof
  fix L'
  assume L'  $\in$  ?lhs
  hence maxL  $\succeq$  L'  $\neg (\exists S \in \text{set stops}. L' \succeq S)$ 
  by auto
  hence length L' = n + 1
  using  $\langle \text{length maxL} = n + 1 \rangle$ 
  by (simp add: pwge-def)
  show L'  $\in$  ?rhs
proof-
  have L'  $\succeq$  replicate (n+1) 0
  using  $\langle \text{length } L' = n+1 \rangle$ 
  by (rule pwge-replicate-0)
moreover
  have  $\neg (\exists Ls. Ls \succeq \text{replicate } (n+1) 0 \wedge \text{take } 0 (\text{replicate } (n+1) 0) =$ 
take 0 Ls  $\wedge$  L'  $\succeq$  Ls  $\wedge$  list-ex (pwge-impl Ls) stops)
  proof (rule ccontr)
  assume  $\neg$  ?thesis
  then obtain Ls where L'  $\succeq$  Ls list-ex (pwge-impl Ls) stops
  by auto
moreover
  hence length Ls = n + 1
  using  $\langle \text{length } L' = n + 1 \rangle$ 

```



```

      unfolding pwge-def
    by simp
  ultimately
  obtain  $S$  where  $S \in \text{set stops } Ls \succeq S$ 
    using pwge-impl[of  $Ls$ ]  $\langle \forall S \in \text{set stops. length } S = n + 1 \rangle$ 
    by (auto simp add: list-ex-iff)
  thus False
    using  $\langle \neg (\exists S \in \text{set stops. } L' \succeq S) \rangle \langle L' \succeq Ls \rangle$  pwge-trans[of  $L' Ls S$ ]
    by auto
qed
ultimately
show ?thesis
  using  $\langle \text{maxL} \succeq L' \rangle$ 
  by auto
qed
qed
next
show ?rhs  $\subseteq$  ?lhs
proof
  fix  $L'$ 
  assume  $L' \in ?rhs$ 
  hence  $\text{maxL} \succeq L' \wedge (\exists Ls. L' \succeq Ls \wedge Ls \succeq \text{replicate } (n+1) 0 \wedge \text{list-ex } (pwge-impl Ls) \text{ stops})$ 
    by auto
  have  $\neg (\exists S \in \text{set stops. } L' \succeq S)$ 
  proof (rule ccontr)
    assume  $\neg ?thesis$ 
    then obtain  $S$  where  $S \in \text{set stops } L' \succeq S$ 
      by auto
    moreover
    hence  $\text{length } L' = n+1 \wedge \text{length } S = n+1$ 
      using  $\langle \text{maxL} \succeq L' \rangle \langle \text{length maxL} = n+1 \rangle$ 
      by (auto simp add: pwge-def)
    ultimately
    have  $L' \succeq S \wedge S \succeq \text{replicate } (n+1) 0 \wedge \text{list-ex } (pwge-impl S) \text{ stops}$ 
      using pwge-refl[of  $S$ ] pwge-impl[of  $S S$ ] pwge-replicate-0[of  $S n+1$ ]
      by (auto simp add: list-ex-iff)
    thus False
      using  $\langle \neg (\exists Ls. L' \succeq Ls \wedge Ls \succeq \text{replicate } (n+1) 0 \wedge \text{list-ex } (pwge-impl$ 
 $Ls) \text{ stops}) \rangle$ 
      by auto
  qed
  thus  $L' \in ?lhs$ 
    using  $\langle \text{maxL} \succeq L' \rangle$ 
    by simp
qed
qed
qed
thus ?thesis

```

```

    by auto
qed

end

```

16.4 Generating all irreducible families

```

theory LPartitioningIrreducible
imports LPartitioning IrreducibleFamilies
begin

```

lemma *irreducible-expressible*:

incrementally-checks $(\lambda A F. \neg \text{expressible } A F)$ *irreducible*

proof (*safe*)

fix $A :: 'a \text{ set}$ **and** $F :: 'a \text{ set set}$

assume $\neg \text{expressible } A F$ *irreducible* F *finite* $(\bigcup F)$ *finite* A

$\forall A' \in F. \text{card } A \geq \text{card } A' \text{ reducible } (F \cup \{A\})$

obtain $A' Fb$ **where** $A' \in F \cup \{A\}$ $Fb \neq \{\}$ $Fb \subseteq F \cup \{A\}$ $A' \notin Fb$ $A' = \bigcup Fb$

using $\langle \text{reducible } (F \cup \{A\}) \rangle$

unfolding *reducible-def*

by *auto*

have *finite* A'

using $\langle A' \in F \cup \{A\} \rangle$ $\langle \text{finite } (\bigcup F) \rangle$ $\langle \text{finite } A \rangle$

by (*auto simp add: finiteUn-iff*)

show *False*

proof (*cases* $A' = A$)

case *True*

thus *?thesis*

using $\langle \neg \text{expressible } A F \rangle$

unfolding *expressible-def*

using $\langle A' = \bigcup Fb \rangle$ $\langle Fb \subseteq F \cup \{A\} \rangle$ $\langle A' \notin Fb \rangle$ $\langle Fb \neq \{\} \rangle$

by *auto*

next

case *False*

hence $A' \in F$

using $\langle A' \in F \cup \{A\} \rangle$

by *simp*

have $A \in Fb$

using $\langle Fb \subseteq F \cup \{A\} \rangle$ $\langle A' \notin Fb \rangle$ $\langle A' = \bigcup Fb \rangle$ $\langle A' \in F \rangle$ $\langle Fb \neq \{\} \rangle$

using $\langle \text{irreducible } F \rangle$

unfolding *reducible-def*

by *blast*

hence $A \subseteq A'$

using $\langle A' = \bigcup Fb \rangle$

by *auto*

hence $\text{card } A' > \text{card } A$

using $\langle A' \neq A \rangle$

using *card-mono*[*of* $A' A$] *card-seteq*[*of* $A' A$] $\langle \text{finite } A' \rangle$

```

    by auto
  thus False
    using ⟨A' ∈ F⟩ ⟨∀ A' ∈ F. card A ≥ card A'⟩
    by auto
qed
qed (auto simp add: reducible-def expressible-def)

lemma not-expressible-iso:
  assumes ¬ expressible A F and inj-on f (⋃ F ∪ A)
  shows ¬ expressible (f ' A) (op ' f ' F)
  proof (rule ccontr)
    assume ¬ ?thesis
    then obtain Fb' where Fb' ⊆ op ' f ' F f ' A = ⋃ Fb' Fb' ≠ {}
      unfolding expressible-def
      by auto
    then obtain F' where F' ⊆ F and ++: op ' f ' F' = Fb'
      by (metis (no-types) subset-image-iff)
    have A = ⋃ F'
    proof
      show A ⊆ ⋃ F'
      proof
        fix x
        assume x ∈ A
        then obtain y where y ∈ Fb' f x ∈ y
          using ⟨image f A = ⋃ Fb'⟩
          by auto
        then obtain x' y' where y' ∈ F' x' ∈ y' f x' = f x
          using ++
          by auto
        hence x' = x
          using ⟨inj-on f (⋃ F ∪ A)⟩ ⟨x ∈ A⟩ ⟨y' ∈ F'⟩ ⟨F' ⊆ F⟩
          unfolding inj-on-def
          by auto
        thus x ∈ ⋃ F'
          using ⟨x' ∈ y'⟩ ⟨y' ∈ F'⟩
          by auto
      qed
    qed
  next
    show ⋃ F' ⊆ A
    proof
      fix x
      assume x ∈ ⋃ F'
      hence f x ∈ f ' A
        using ++
        using ⟨image f A = ⋃ Fb'⟩
        by auto
      then obtain x' where x' ∈ A f x = f x'
        by auto
      thus x ∈ A
    proof

```

```

    using ⟨ $x \in \bigcup F'$ ⟩ ⟨ $F' \subseteq F$ ⟩
    using ⟨inj-on  $f$  ( $\bigcup F \cup A$ )⟩
    unfolding inj-on-def
    by auto
  qed
qed
moreover
have  $F' \neq \{\}$ 
  using ++ ⟨ $Fb' \neq \{\}$ ⟩
  by auto
ultimately
show False
  using ⟨ $\neg$  expressible  $A$   $F$ ⟩ ⟨ $F' \subseteq F$ ⟩
  unfolding expressible-def
  by auto
qed

lemma inj-preserved-not-expressible:
  shows inj-preserved ( $\lambda A F. \neg$  expressible  $A$   $F$ )
  using not-expressible-iso
  by auto

abbreviation L-part-irreducible where
  L-part-irreducible  $\equiv$  L-part-P irreducible
abbreviation mult-irreducible where
  mult-irreducible  $\equiv$  mult-P ( $\lambda A F. \neg$  expressible  $A$   $F$ )
abbreviation mult-all-irreducible where
  mult-all-irreducible  $\equiv$  mult-all-P ( $\lambda A F. \neg$  expressible  $A$   $F$ )

lemma L-part-irreducible-mult-irreducible:
  L-part-irreducible  $n$  ( $L$  @  $[k+1]$ ) = mult-all-irreducible (L-part-irreducible  $n$  ( $L$ 
  @  $[k]$ ))  $n$  (length  $L$ )
  using L-part-mult-P[of irreducible  $\lambda A f. \neg$  expressible  $A$   $f$   $n$   $L$   $k$ , OF irreducible-expressible]
  by auto

lemmas L-part-irreducible-mult-irreducible-iso-representing-subset =
  L-part-mult-iso-representing-subset[of irreducible  $\lambda A F. \neg$  expressible  $A$   $F$ ,
  OF irreducible-expressible inj-preserved-not-expressible]

end

```

16.5 Generating all families that are not FCs covered by the given collection

```

theory LPartitioningNonFCsCovered
imports LPartitioning Covering
begin

```

```

lemma pwge-FCs-covered:

```

```

assumes  $\forall F \in L\text{-part } n \ L. \text{ FCs-covered } \mathcal{F} \ F \ L' \succeq L$ 
shows  $\forall F \in L\text{-part } n \ L'. \text{ FCs-covered } \mathcal{F} \ F$ 
proof (safe)
  fix  $F'$ 
  assume  $\text{is-}L\text{-part } n \ L' \ F'$ 
  then obtain  $F$  where  $F \subseteq F'$   $\text{is-}L\text{-part } n \ L \ F$ 
    using  $\langle L' \succeq L \rangle \text{is-}L\text{-part-subset}[\text{of } L' \ L \ n \ F]$ 
    by auto
  then obtain  $F_c$  where  $\text{FC-covered } F_c \ F \ F_c \in \mathcal{F}$ 
    using  $\text{assms}(1)[\text{rule-format, of } F]$ 
    by auto
  thus  $\text{FCs-covered } \mathcal{F} \ F'$ 
    using  $\text{FC-covered-mono}[\text{of } F \ F' \ F_c] \ \langle F \subseteq F' \rangle$ 
    by auto
qed

lemma notFCs-covered:
  incrementally-checks  $(\lambda A \ F. \neg \text{FCs-covered } \mathcal{F} \ (F \cup \{A\})) \ (\lambda F. \neg \text{FCs-covered } \mathcal{F} \ F)$ 
using FC-covered-mono
by blast

lemma notFCs-covered-iso:
  assumes  $\neg \text{FCs-covered } \mathcal{F} \ (F \cup \{A\})$  and  $\text{inj-on } f \ (\bigcup F \cup A)$ 
  shows  $\neg \text{FCs-covered } \mathcal{F} \ ((\text{op } 'f' \ F) \cup \{f' \ A\})$ 
proof–
  have  $\text{iso } (F \cup \{A\}) \ ((\text{op } 'f' \ F) \cup \{f' \ A\})$ 
    unfolding iso-def
    unfolding bij-betw-def
    using  $\langle \text{inj-on } f \ (\bigcup F \cup A) \rangle$ 
    by  $(\text{rule-tac } x=f \ \text{in } exI) \ (\text{auto simp add: Un-commute})$ 
  thus ?thesis
    using assms
    using  $\text{FC-covered-iso}[\text{of } (\text{op } 'f' \ F) \cup \{f' \ A\} \ F \cup \{A\}] \ \text{iso-sym}$ 
    by blast
qed

lemma inj-preserved-not-FCs-covered:
  inj-preserved  $(\lambda A \ F. \neg \text{FCs-covered } \mathcal{F} \ (F \cup \{A\}))$ 
using notFCs-covered-iso
by blast

abbreviation L-part-notFCs-covered where
   $L\text{-part-notFCs-covered } \mathcal{F} \equiv L\text{-part-}P \ (\lambda F. \neg \text{FCs-covered } \mathcal{F} \ F)$ 
abbreviation mult-all-notFCs-covered where
   $\text{mult-all-notFCs-covered } \mathcal{F} \equiv \text{mult-all-}P \ (\lambda A \ f. \neg \text{FCs-covered } \mathcal{F} \ (f \cup \{A\}))$ 

lemma L-part-mult-iso-representing-subset-notFC-covered:
  assumes

```

```

    iso-representing-subset FFb (L-part-notFCs-covered  $\mathcal{F}$  n (L @ [k]))
    FFb' = mult-all-notFCs-covered  $\mathcal{F}$  FFb n (length L)
    length L ≤ n
    shows iso-representing-subset FFb' (L-part-notFCs-covered  $\mathcal{F}$  n (L @ [k + 1]))
using assms
using L-part-mult-iso-representing-subset[of  $\lambda F. \neg$  FCs-covered  $\mathcal{F}$  F  $\lambda A f. \neg$ 
FCs-covered  $\mathcal{F}$  (f  $\cup$  {A})] FFb n L k, OF notFCs-covered inj-preserved-not-FCs-covered]
by auto

end

```

16.5.1 Implementation

```

theory LPartitioningNonFCsCoveredImpl
imports LPartitioningNonFCsCovered CoveringImpl LPartitioningImpl
begin

```

lemma notFCs-covered-l:

```

    assumes perms = permute [0.. $n$ ] dmf  $\mathcal{F}$  n sdff  $\mathcal{F}$  dm (A # F) n
    sd A sdf F
    shows
       $\neg$  FCs-covered-l  $\mathcal{F}$  (A # F) perms  $\longleftrightarrow$ 
       $\neg$  FCs-covered (fs-to-set-l  $\mathcal{F}$ ) (f-to-set-l F  $\cup$  {set A})
    using FC-covered-l-soundness[of perms n - A # F]
    using FC-covered-l-completeness[of A # F - n perms]
    using assms
    unfolding FCs-covered-l-def
    by (auto simp add: list-ex-iff isPermutation-permute)

```

definition mult-notFCs-covered-l **where**

```

    mult-notFCs-covered-l  $\mathcal{F}$  F A perms =
      (let F' = filter ( $\lambda f. A \notin$  set f) F;
       F'' = map ( $\lambda x. A \# x$ ) F'
       in filter ( $\lambda X. \neg$  FCs-covered-l  $\mathcal{F}$  X perms) F'')

```

definition mult-all-notFCs-covered-l **where**

```

    mult-all-notFCs-covered-l  $\mathcal{F}$  F n m perms = concat (map ( $\lambda A. mult-notFCs-covered-l$ 
 $\mathcal{F}$  F A perms) (all-mn-subsets n m))

```

definition mult-all-base-notFCs-covered-l **where**

```

    mult-all-base-notFCs-covered-l  $\mathcal{F}$  F n m perms = non-isomorphic-families-l perms
    (mult-all-notFCs-covered-l  $\mathcal{F}$  F n m perms)

```

lemma mult-all-base-notFCs-covered-l:

```

    mult-all-base-notFCs-covered-l  $\mathcal{F}$  F n m perms = mult-all-base-P-l ( $\lambda A F. \neg$ 
    FCs-covered-l  $\mathcal{F}$  (A # F) perms) F n m perms
    by (simp add: mult-all-base-P-l-def mult-all-base-notFCs-covered-l-def mult-all-P-l-def
    mult-all-notFCs-covered-l-def mult-P-l-def mult-notFCs-covered-l-def filter-map)

```

definition *FC-covered-l-opt* :: nat list list list \Rightarrow nat list list \Rightarrow bool **where**
 $FC\text{-covered-l-opt } Fc\text{-perms } F \longleftrightarrow list\text{-ex } (\lambda Fc'. set Fc' \subseteq set (close\text{-l } F))$

Fc-perms

definition *FCs-covered-l-opt* **where**

$FCs\text{-covered-l-opt } \mathcal{F}\text{-perms } F \longleftrightarrow list\text{-ex } (\lambda Fc. FC\text{-covered-l-opt } Fc F) \mathcal{F}\text{-perms}$

definition *FCs-covered-l-opt'* **where**

$FCs\text{-covered-l-opt'} \mathcal{F}\text{-perms } F \longleftrightarrow$

$(let\ clF = set (close\text{-l } F)$

$in\ list\text{-ex } (list\text{-ex } (\lambda Fc'. set Fc' \subseteq clF)) \mathcal{F}\text{-perms})$

lemma [*simp*]: $FCs\text{-covered-l-opt'} = FCs\text{-covered-l-opt}$

by (*rule ext*) + (*simp add: FCs-covered-l-opt'-def FCs-covered-l-opt-def FC-covered-l-opt-def*)

definition *mult-notFCs-covered-l-opt* **where**

$mult\text{-notFCs-covered-l-opt } \mathcal{F}\text{-perms } F A =$

$(let\ F' = filter (\lambda f. A \notin set f) F;$

$F'' = map (\lambda x. A \# x) F'$

$in\ filter (\lambda X. \neg FCs\text{-covered-l-opt'} \mathcal{F}\text{-perms } X) F'')$

definition *mult-all-notFCs-covered-l-opt* **where**

$mult\text{-all-notFCs-covered-l-opt } \mathcal{F}\text{-perms } F n m = concat (map (\lambda A. mult\text{-notFCs-covered-l-opt } \mathcal{F}\text{-perms } F A) (all\text{-mn-subsets } n m))$

definition *mult-all-base-notFCs-covered-l-opt* **where**

$mult\text{-all-base-notFCs-covered-l-opt } \mathcal{F}\text{-perms } F n m perms = non\text{-isomorphic-families-l } perms (mult\text{-all-notFCs-covered-l-opt } \mathcal{F}\text{-perms } F n m)$

lemma *mult-all-base-notFCs-covered-opt*:

assumes $\mathcal{F}\text{-perms} = map (\lambda F. map (\lambda p. permute\text{-family-l } p F) perms) \mathcal{F}$

shows $mult\text{-all-base-notFCs-covered-l-opt } \mathcal{F}\text{-perms } F n m perms =$

$mult\text{-all-base-notFCs-covered-l } \mathcal{F} F n m perms$

using *assms*

by (*auto simp add: list-ex-iff mult-all-base-notFCs-covered-l-def mult-all-base-notFCs-covered-l-opt-def*

mult-all-notFCs-covered-l-def mult-all-notFCs-covered-l-opt-def mult-notFCs-covered-l-def

mult-notFCs-covered-l-opt-def FCs-covered-l-opt-def FCs-covered-l-def FC-covered-l-opt-def

FC-covered-l-def)

abbreviation *enum-rec-notFCs-covered-l* **where**

$enum\text{-rec-notFCs-covered-l } \mathcal{F}\text{-perms } L n perms \equiv$

$enum\text{-rec } L [] (\lambda F L. mult\text{-all-base-notFCs-covered-l-opt } \mathcal{F}\text{-perms } F n (length L - 1) perms)$

lemma *enum-rec-notFCs-covered-l-iso-representing-subset*:

assumes $\mathcal{F}\text{-perms} = map (\lambda F. map (\lambda p. permute\text{-family-l } p F) (permute [0..<n])) \mathcal{F} \ \& \notin\ set \mathcal{F}$

$dmf \mathcal{F} n\ sdiff \mathcal{F} length L - 1 \leq n$

shows *iso-representing-subset* (*fs-to-set-l* (*enum-rec-notFCs-covered-l* $\mathcal{F}\text{-perms } L n (permute [0..<n]))) (*L-part-notFCs-covered* (*fs-to-set-l* \mathcal{F}) $n L$)$

```

using assms
proof –
  have  $\forall Fc \in \text{set } \mathcal{F}. \neg FC\text{-covered } (f\text{-to-set-}l\ Fc) \ \{\}$ 
    using  $\langle [] \notin \text{set } \mathcal{F} \rangle$ 
    unfolding FC-covered-def
    by (auto simp add: closure-def)
  thus ?thesis
    using  $\langle \text{sdff } \mathcal{F} \rangle$ 
    by (subst mult-all-base-notFCs-covered-opt [OF assms(1)]) +
      (rule enum-rec-iso-representing-subset [where n=n and perms=permute
        [0..<n], OF notFCs-covered-l [of permute [0..<n], OF - assms(3)] notFCs-covered
        inj-preserved-not-FCs-covered - (length L - 1 ≤ n), simp-all add: isPermutation-permute
        mult-all-base-notFCs-covered-l])
  qed

end

```

16.6 Generating all irreducible families that are not FCs covered by the given collection

```

theory LPartitioningIrreducibleNonFCsCovered
imports LPartitioningIrreducible LPartitioningNonFCsCovered IrreducibleFamilies
begin

```

```

lemma irreducible-expressible-notFCs-covered:
  incrementally-checks
    ( $\lambda A\ F. \neg \text{expressible } A\ F \wedge \neg FCs\text{-covered } \mathcal{F} (F \cup \{A\})$ )
    ( $\lambda F. \text{irreducible } F \wedge \neg FCs\text{-covered } \mathcal{F} F$ )
using irreducible-expressible
using notFCs-covered[of ]
by metis

lemma irreducible-expressible-notFCs-covered-iso:
assumes  $\neg \text{expressible } A\ F \neg FCs\text{-covered } \mathcal{F} (F \cup \{A\})$  inj-on f ( $\bigcup F \cup A$ )
  shows  $\neg \text{expressible } (f\ 'A) (op\ '\ 'f\ 'F) \wedge \neg FCs\text{-covered } \mathcal{F} (op\ '\ 'f\ 'F \cup \{f\ '\ 'A\})$ 
using assms
using not-expressible-iso[of A F f] notFCs-covered-iso[of ]
by auto

lemma inj-preserved-not-expressible-notFCs-covered:
  inj-preserved ( $\lambda A\ F. \neg \text{expressible } A\ F \wedge \neg FCs\text{-covered } \mathcal{F} (F \cup \{A\})$ )
using inj-preserved-not-expressible inj-preserved-not-FCs-covered
by blast

```

```

abbreviation L-part-irreducible-notFCs-covered where
  L-part-irreducible-notFCs-covered  $\mathcal{F} \equiv L\text{-part-}P\ (\lambda F. \text{irreducible } F \wedge \neg FCs\text{-covered } \mathcal{F} F)$ 

```


abbreviation *mult-all-irreducible-notFCs-covered* **where**

mult-all-irreducible-notFCs-covered $\mathcal{F} \equiv \text{mult-all-}P (\lambda A f. \neg \text{expressible } A f \wedge \neg \text{FCs-covered } \mathcal{F} (f \cup \{A\}))$

lemma *L-part-mult-iso-representing-subset-irreducible-notFC-covered*:

assumes

iso-representing-subset FFb (*L-part-irreducible-notFCs-covered* \mathcal{F} n ($L @ [k]$))

length $L \leq n$

$FFb' = \text{mult-all-irreducible-notFCs-covered } \mathcal{F} \text{ } FFb \text{ } n \text{ (length } L)$

shows *iso-representing-subset* FFb' (*L-part-irreducible-notFCs-covered* \mathcal{F} n ($L @ [k + 1]$))

using *assms*

using *L-part-mult-iso-representing-subset*[*of* $\lambda F. \text{irreducible } F \wedge \neg \text{FCs-covered } \mathcal{F}$

$F \wedge A f. \neg \text{expressible } A f \wedge \neg \text{FCs-covered } \mathcal{F} (f \cup \{A\})$ FFb , *OF irreducible-expressible-notFCs-covered* *inj-preserved-not-expressible-notFCs-covered*]

by *auto*

lemma *iso-represents-L-part-irreducible-notFCs-covered*:

fixes $\mathcal{F} \mathcal{N} FFb :: \text{nat set set set}$

assumes *iso-representing-subset* FFb (*L-part-irreducible-notFCs-covered* \mathcal{F} n L)

$\forall F \in FFb. \text{nonFCs-covered } \mathcal{N} F$

$\forall N \in \mathcal{N}. \text{finite } (\bigcup N)$

shows $\forall F \in \text{L-part-irreducible } n \text{ } L. \text{covered } \mathcal{F} \mathcal{N} F$

using *assms*

using *iso-represents-nonFCs-covered*[*of* $\{F \in \text{L-part } n \text{ } L. \text{irreducible } F \wedge \neg \text{FCs-covered } \mathcal{F} F\} \mathcal{N} FFb$]

using *is-L-part-finite*[*of* $n \text{ } L$]

unfolding *covered-def*

by *auto blast*

end

16.6.1 Implementation

theory *LPartitioningIrreducibleNonFCsCoveredImpl*

imports *LPartitioningIrreducibleNonFCsCovered IrreducibleFamiliesImpl CoveringImpl*

LPartitioningNonFCsCoveredImpl

begin

lemma *irreducible-expressible-notFCs-covered-l*:

assumes *perms* = *permute* $[0..<n]$ *dmf* \mathcal{F} n *sdff* \mathcal{F}

sd A *sdf* F *dm* $(A \# F)$ n

shows $\neg \text{expressible-l } A F \wedge \neg \text{FCs-covered-l } \mathcal{F} (A \# F) \text{ perms} \longleftrightarrow \neg \text{expressible } (\text{set } A) (f\text{-to-set-l } F) \wedge \neg \text{FCs-covered } (fs\text{-to-set-l } \mathcal{F}) (f\text{-to-set-l } F \cup \{\text{set } A\})$

using *assms*

using *expressible-l-soundness*[*of* $A F$]

using *expressible-l-completeness*[*OF* *assms*(4–5)]

using *notFCs-covered-l*[*OF* *assms*(1–3) *assms*(6)]

by *auto*

definition *mult-irreducible-notFCs-covered-l* **where**

mult-irreducible-notFCs-covered-l \mathcal{F} F A perms =
 (let $F' = \text{filter } (\lambda f. A \notin \text{set } f \wedge \neg \text{expressible-l } A f) F$;
 $F'' = \text{map } (\lambda x. A \# x) F'$
 in $\text{filter } (\lambda X. \neg \text{FCs-covered-l } \mathcal{F} X \text{ perms}) F''$)

definition *mult-all-irreducible-notFCs-covered-l* **where**

mult-all-irreducible-notFCs-covered-l \mathcal{F} F n m perms =
 $\text{concat } (\text{map } (\lambda A. \text{mult-irreducible-notFCs-covered-l } \mathcal{F} F A \text{ perms}) (\text{all-mn-subsets } n m))$

definition *mult-all-base-irreducible-notFCs-covered-l* **where**

mult-all-base-irreducible-notFCs-covered-l \mathcal{F} F n m perms =
 $\text{non-isomorphic-families-l perms } (\text{mult-all-irreducible-notFCs-covered-l } \mathcal{F} F n m \text{ perms})$

lemma *mult-all-base-irreducible-notFCs-covered*:

mult-all-base-irreducible-notFCs-covered-l \mathcal{F} F n m perms =
 $\text{mult-all-base-P-l } (\lambda A F. \neg \text{expressible-l } A F \wedge \neg \text{FCs-covered-l } \mathcal{F} (A \# F) \text{ perms}) F n m \text{ perms}$

unfolding *mult-all-base-irreducible-notFCs-covered-l-def* *mult-all-base-P-l-def* *mult-all-irreducible-notFCs-covered-l-def* *mult-P-l-def*

by (*simp add: filter-map Let-def*)

definition *mult-irreducible-notFCs-covered-l-opt* **where**

mult-irreducible-notFCs-covered-l-opt \mathcal{F} - perms F A =
 (let $F' = \text{filter } (\lambda f. A \notin \text{set } f \wedge \neg \text{expressible-l } A f) F$;
 $F'' = \text{map } (\lambda x. A \# x) F'$
 in $\text{filter } (\lambda X. \neg \text{FCs-covered-l-opt } \mathcal{F}\text{-perms } X) F''$)

definition *mult-all-irreducible-notFCs-covered-l-opt* **where**

mult-all-irreducible-notFCs-covered-l-opt \mathcal{F} - perms F n m =
 $\text{concat } (\text{map } (\lambda A. \text{mult-irreducible-notFCs-covered-l-opt } \mathcal{F}\text{-perms } F A) (\text{all-mn-subsets } n m))$

definition *mult-all-base-irreducible-notFCs-covered-l-opt* **where**

mult-all-base-irreducible-notFCs-covered-l-opt \mathcal{F} - perms F n m perms =
 $\text{non-isomorphic-families-l perms } (\text{mult-all-irreducible-notFCs-covered-l-opt } \mathcal{F}\text{-perms } F n m)$

lemma *mult-all-base-irreducible-notFCs-covered-opt*:

assumes $\mathcal{F}\text{-perms} = \text{map } (\lambda F. \text{map } (\lambda p. \text{permute-family-l } p F) \text{ perms}) \mathcal{F}$
shows *mult-all-base-irreducible-notFCs-covered-l-opt* \mathcal{F} - perms F n m perms =
 $\text{mult-all-base-irreducible-notFCs-covered-l } \mathcal{F} F n m \text{ perms}$

using *assms*

by (*auto simp add: list-ex-iff mult-all-base-irreducible-notFCs-covered-l-def mult-all-base-irreducible-notFCs-covered-l-opt-def mult-all-irreducible-notFCs-covered-l-def mult-all-irreducible-notFCs-covered-l-opt-def FCs-covered-l-opt-def FCs-covered-l-def FC-covered-l-opt-def FC-covered-l-def*)

abbreviation *enum-rec-irreducible-notFCs-covered-l* **where**

enum-rec-irreducible-notFCs-covered-l \mathcal{F} -perms L n perms \equiv
enum-rec L $[\Box]$ $(\lambda F L. \text{mult-all-base-irreducible-notFCs-covered-l-opt } \mathcal{F}\text{-perms } F$
 $n \text{ (length } L - 1) \text{ perms)})$

abbreviation *enum-dp-irreducible-notFCs-covered-l* **where**

enum-dp-irreducible-notFCs-covered-l \mathcal{F} -perms n perms stops $\text{max}L \equiv$
enum-dp $[\Box] \Box 0$ (*replicate* $(n+1)$ $(0::\text{nat})$)
 $(\lambda FFb m. \text{mult-all-base-irreducible-notFCs-covered-l-opt } \mathcal{F}\text{-perms } FFb \ n \ m$
 $\text{perms})$
 $(\lambda L. \text{list-ex } (\lambda L'. \text{pwge-impl } L \ L') \text{ stops})$
 $\text{max}L$

lemma *enum-dp-irreducible-notFCs-covered-l*:

assumes $\mathcal{F}\text{-perms} = \text{map } (\lambda F. \text{map } (\lambda p. \text{permute-family-l } p \ F) \text{ perms}) \ \mathcal{F}$

shows

enum-dp-irreducible-notFCs-covered-l \mathcal{F} -perms n perms stops $\text{max}L =$
enum-dp-mult-P $(\lambda A F. \neg \text{expressible-l } A \ F \wedge \neg \text{FCs-covered-l } \mathcal{F} \ (A \ \# \ F)$

$\text{perms}) \ n \text{ perms stops } \text{max}L$

unfolding *enum-dp-mult-P-def*

by (*subst mult-all-base-irreducible-notFCs-covered-opt*[*OF assms*], *subst mult-all-base-irreducible-notFCs-covered-simp*)

lemma *enum-dp-irreducible-notFCs-covered-l-iso-representing-subsets*:

assumes

$\mathcal{F}\text{-perms} = \text{map } (\lambda F. \text{map } (\lambda p. \text{permute-family-l } p \ F) \text{ perms}) \ \mathcal{F}$
 $X = \{L'. \text{max}L \succeq L' \wedge \neg (\exists S \in \text{set stops. } L' \succeq S)\}$ (**is** $- = ?\text{lhs}$)
 $\text{length } \text{max}L = n + 1 \ \forall S \in \text{set stops. } \text{length } S = n + 1$
 $\Box \notin \text{set } \mathcal{F} \text{ perms} = \text{permute } [0..<n] \text{ dmf } \mathcal{F} \ n \ \text{sdff } \mathcal{F}$

shows $\forall L' \in X. \exists B \in \text{set } (\text{enum-dp-irreducible-notFCs-covered-l } \mathcal{F}\text{-perms } n$
 $\text{perms stops } \text{max}L).$

iso-representing-subset (*fs-to-set-l* B) (*L-part-irreducible-notFCs-covered*
 $(\text{fs-to-set-l } \mathcal{F}) \ n \ L')$

proof–

have *irreducible* $\{\} \wedge \neg \text{FCs-covered } (\text{fs-to-set-l } \mathcal{F}) \ \{\}$

using $\langle \Box \notin \text{set } \mathcal{F} \rangle$

unfolding *FC-covered-def*

by (*auto simp add: closure-def reducible-def*)

thus *?thesis*

using *assms*

by (*subst enum-dp-irreducible-notFCs-covered-l*[*OF assms*(1)])

$(\text{rule } \text{enum-dp-mult-P-correct } [\textbf{where } \text{Pinc}' = \lambda A F. \neg \text{expressible-l } A \ F \wedge$
 $\neg \text{FCs-covered-l } \mathcal{F} \ (A \ \# \ F) \text{ perms, } \text{OF irreducible-expressible-notFCs-covered-l}[\text{OF}$
 $\text{assms}(6) \ \text{assms}(7)] \text{ irreducible-expressible-notFCs-covered inj-preserved-not-expressible-notFCs-covered}],$
 $\text{auto simp add: isPermutation-permute})$

qed

end

17 Minimal FC(6) families and maximal nonFC(6) families

```
theory FC6-Data
imports Main
begin
```

definition *FC6* :: *nat list list list* **where**

```
FC6 = [
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,4],[0,1,2,3,5],[1,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,5],[0,1,3,4,5],[0,2,3,4,5],[1,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,4],[1,2,3,4]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,4],[1,2,3,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,5],[0,2,3,5],[1,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,5],[0,3,4,5],[1,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,5],[1,2,3,5],[0,1,2,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,5],[0,2,4,5],[1,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,5],[1,2,4,5],[0,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,5],[2,3,4,5],[0,1,2,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,5],[2,3,4,5],[1,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,2,5],[0,3,4,5],[1,3,4,5],[0,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,5],[0,1,4,5],[2,3,4,5],[0,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,5],[0,2,4,5],[0,3,4,5],[1,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,5],[0,2,4,5],[1,3,4,5],[0,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,5],[2,3,4,5],[0,1,3,4,5],[0,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,5],[2,3,4,5],[0,2,3,4,5],[1,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,4],[0,1,2,5],[0,1,3,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,4],[0,1,2,5],[0,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,5],[0,1,3,5],[2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,5],[0,2,3,5],[1,2,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,5],[0,1,3,5],[0,1,4,5],[0,2,3,4,5]],
  [[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,5],[0,1,3,5],[0,2,4,5],[0,2,3,4,5]],
  [[0,1,2],[0,1,3,4],[0,2,3,4],[0,1,2,3,5],[1,2,3,4,5]],
  [[0,1,2],[0,1,3,4],[0,1,3,5],[0,2,3,4,5],[1,2,3,4,5]],
  [[0,1,2],[0,1,3,4],[0,2,3,5],[0,1,2,4,5],[1,2,3,4,5]],
  [[0,1,2],[0,1,3,4],[0,2,3,5],[0,1,3,4,5],[1,2,3,4,5]],
  [[0,1,2],[0,1,3,4],[0,3,4,5],[0,1,2,3,5],[0,1,2,4,5],[1,2,3,4,5]],
  [[0,1,2],[0,1,3,4],[2,3,4,5],[0,1,2,3,5],[0,1,2,4,5],[0,2,3,4,5],[1,2,3,4,5]],
  [[0,1,2],[0,1,2,3],[0,1,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5],[1,2,3,4,5]],
  [[0,1,2],[0,3,4,5],[1,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,2,3,4,5],[1,2,3,4,5]],
  [[0,1,2],[0,1,3,4],[0,2,3,4],[1,2,3,4]],
  [[0,1,2],[0,1,3,4],[0,2,3,4],[1,2,3,5]],
  [[0,1,2],[0,1,3,4],[0,2,3,5],[1,2,4,5]],
  [[0,1,2],[0,1,3,4],[0,2,3,5],[1,3,4,5]],
  [[0,1,2],[0,1,2,3],[0,1,3,4],[0,2,3,4],[1,2,3,4,5]],
  [[0,1,2],[0,1,2,3],[0,1,3,4],[0,2,3,5],[1,2,3,4,5]],
  [[0,1,2],[0,1,2,3],[0,1,3,4],[0,2,4,5],[1,2,3,4,5]],
  [[0,1,2],[0,1,2,3],[0,1,4,5],[0,2,4,5],[1,2,3,4,5]],
  [[0,1,2],[0,1,3,4],[0,2,3,4],[0,1,3,5],[0,1,2,4,5]]]
```

$[[0,1,2],[0,1,3,4],[0,2,3,4],[0,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,2,3,4],[1,3,4,5],[0,1,2,3,5]],$
 $[[0,1,2],[0,1,3,4],[0,2,3,4],[1,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,1,3,5],[0,1,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,1,3,5],[0,2,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,1,3,5],[0,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,1,3,5],[2,3,4,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,1,3,5],[2,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,3,4,5],[0,1,2,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,3,4,5],[1,3,4,5],[0,1,2,3,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,3,4,5],[1,3,4,5],[0,1,2,3,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,3,4,5],[2,3,4,5],[0,1,2,3,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,3,4,5],[2,3,4,5],[0,1,2,3,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,3,4,5],[0,1,2,3,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[2,3,4,5],[0,1,2,4,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,4,5],[0,3,4,5],[0,1,2,3,4],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,3,5],[0,1,3,4,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[2,3,4,5],[0,1,2,3,5],[0,1,3,4,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,4,5],[2,3,4,5],[0,1,2,3,4],[0,1,3,4,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,2,3,4],[0,1,3,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,2,3,4],[0,1,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,2,3,4],[1,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,4,5],[0,2,4,5],[1,3,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,2,3,4],[0,1,3,5],[0,2,3,5]],$
 $[[0,1,2],[0,1,3,4],[0,2,3,4],[0,1,3,5],[0,3,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,2,3,4],[0,3,4,5],[1,3,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,1,3,5],[0,1,4,5],[0,3,4,5]],$
 $[[0,1,2],[0,1,3,4],[0,1,3,5],[0,3,4,5],[1,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,2,3,4],[0,3,4,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,1,4,5],[0,2,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,3,4,5],[1,3,4,5],[0,1,2,3,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,3,4,5],[1,3,4,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,3,4,5],[1,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,3,4,5],[2,3,4,5],[0,1,2,3,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,3,4,5],[2,3,4,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,3,4,5],[2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,4,5],[0,3,4,5],[2,3,4,5],[0,1,2,3,4]],$
 $[[0,1,2],[0,1,2,3],[0,1,4,5],[0,3,4,5],[2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[2,3,4,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,5],[2,3,4,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,1,3,5],[0,3,4,5],[0,1,2,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,3,4],[0,1,4,5],[0,3,4,5],[0,1,2,3,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,3,4,5],[1,3,4,5],[2,3,4,5],[0,1,2,3,4],[0,1,2,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,5],[2,3,4,5],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,3,4,5],[1,3,4,5],[0,1,2,3,5],[0,2,3,4,5],[1,2,3,4,5]],$

$[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[2,3,4,5],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,4],[0,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,3,5],[0,2,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,3,5],[2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,5],[0,2,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,3,4,5],[1,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,3,4,5],[2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,5],[0,2,4,5],[0,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,3,4,5],[1,3,4,5],[2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,5],[2,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,3,5],[0,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,5],[0,3,4,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,5],[0,1,3,5],[0,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,2,3,4]],$
 $[[0,1,2],[0,3,4],[1,2,3,4]],$
 $[[0,1,2],[0,1,3],[0,1,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,2,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,3],[2,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,3,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,3,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[1,2,3,5],[0,1,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,2,4,5],[0,1,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[1,2,3,5],[0,1,2,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,2,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,2,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,3],[2,3,4,5],[0,1,2,3,4],[0,1,2,4,5],[0,1,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,5],[0,1,2,3,5],[0,1,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,5],[0,1,3,4,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,1,4,5],[0,2,4,5]],$
 $[[0,1,2],[0,1,3],[0,1,4,5],[2,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,2,4,5],[1,2,4,5]],$
 $[[0,1,2],[0,1,3],[0,2,4,5],[1,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[1,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,3,4],[0,2,3,5]],$
 $[[0,1,2],[0,1,3],[0,1,2,4],[0,2,4,5],[0,1,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,1,2,4],[0,2,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,1,2,4],[0,3,4,5],[0,1,2,3,5]],$
 $[[0,1,2],[0,1,3],[0,1,2,4],[0,3,4,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,1,3],[0,1,2,4],[0,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,1,2,4],[2,3,4,5],[0,1,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,4],[1,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,3,4],[1,2,3,4,5]],$

$[[0,1,2],[0,3,4],[0,1,2,3],[1,2,3,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[1,2,4,5],[0,1,2,3,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,5],[1,2,3,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,5],[1,2,3,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,5],[1,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,5],[1,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,3,4],[0,2,3,4],[0,1,2,3,5]],$
 $[[0,1,2],[3,4,5],[0,1,3,4],[0,2,3,4],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,1,2,4],[2,3,4,5],[0,1,2,3,5],[0,1,2,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,3,4],[0,1,2,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,3,4],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,4,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,1,2,3,5],[0,1,3,4,5],[0,2,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,1,2,4],[0,1,3,4],[0,2,4,5]],$
 $[[0,1,2],[0,1,3],[0,1,2,4],[0,1,3,4],[2,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,1,2,4],[0,1,2,5],[0,2,4,5]],$
 $[[0,1,2],[0,1,3],[0,1,2,4],[0,1,3,5],[2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,4],[1,2,3,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,3,4],[0,1,3,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,5],[0,1,3,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,5],[1,2,3,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,3,5],[0,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[1,2,3,5],[0,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[1,2,4,5],[0,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,5],[1,2,3,5],[0,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,3,4],[0,2,3,4]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,3,4],[0,1,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,4,5],[0,2,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,5],[0,3,4,5],[1,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,1,3,5],[0,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,3,4],[0,3,4,5],[0,1,2,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,3,4],[2,3,4,5],[0,1,2,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,3,4],[0,1,2,5],[0,1,2,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,3,4],[2,3,4,5],[0,1,2,3,5],[0,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,4,5],[2,3,4,5],[0,1,2,3,4],[0,2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,2,3,4]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,1,3,4],[2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,1,3,5],[0,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,1,3,5],[2,3,4,5]],$
 $[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,4],[0,1,2,5],[0,3,4,5],[0,1,3,4,5],[0,2,3,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,3,4,5],[1,3,4,5],[0,1,2,3,5],[0,1,2,4,5]],$
 $[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,3,4,5],[1,3,4,5],[0,1,2,3,5],[0,2,3,4,5]],$
 $[[0,1,2],[0,1,3],[0,2,3]],$
 $[[0,1,2],[0,1,3],[0,1,4]],$
 $[[0,1,2],[0,1,3],[0,2,4]],$
 $[[0,1,2],[0,1,3],[2,3,4]],$

]

$$nonFC6 = [$$

[illegible]

```

[[[0,1,2],[0,1,3],[0,1,2,3],[0,1,2,4],[2,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,3,4,5]],
[[0,1,2],[0,1,3],[0,1,2,3],[0,1,2,4],[2,3,4,5],[0,1,2,3,4],[0,1,2,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,5],[0,1,2,3,4],[0,1,2,3,5],[1,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,3,4],[0,1,2,3],[1,2,3,5],[0,1,2,3,4],[0,1,2,3,5],[1,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,3,4],[0,1,2,3],[0,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[1,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,3,4],[0,1,2,5],[0,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[1,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,4,5],[0,1,2,3,4],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,3,4],[0,1,2,3],[0,1,3,4],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,3,4],[0,1,2,5],[0,1,3,5],[0,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,3,4],[0,1,3,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,1,2,3,4],[0,1,3,4,5],[0,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,3,4],[2,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,3,4],[2,3,4,5],[0,1,2,3,4],[0,1,3,4,5],[0,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,4,5],[2,3,4,5],[0,1,2,3,4],[0,1,2,4,5],[0,1,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,4],[0,1,2,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,3,4],[0,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,3,4,5],[0,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,5],[0,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,3,4,5],[1,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,3,4,5],[0,1,2,3,4],[0,1,3,4,5],[0,2,3,4,5],[1,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,4],[0,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,1,2,5],[0,1,3,4],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,1,3,4],[0,3,4,5],[0,1,2,3,4],[0,1,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,1,3],[0,1,2,3],[0,1,2,4],[0,1,2,5],[0,1,3,4],[0,1,3,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,1,2,5],[0,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[1,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,3,4,5],[1,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[3,4,5],[0,1,2,3],[0,1,2,4],[0,1,2,5],[0,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,1,3],[0,1,2,3],[0,1,2,4],[0,1,2,5],[0,1,3,4],[0,1,3,5],[0,1,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,3,4],[0,1,2,3],[0,1,2,4],[0,1,2,5],[0,1,3,4],[0,3,4,5],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,3,4,5],[0,2,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,1,3],[2,4,5],[0,1,2,3],[0,1,2,4,5],[0,1,3,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,1,3],[2,4,5],[0,1,2,3],[0,1,2,3,4],[0,1,2,3,5],[0,1,2,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,1,3],[2,4,5],[0,1,2,3],[0,1,2,4],[0,1,2,3,4],[0,1,2,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,1,3],[2,4,5],[0,1,2,3],[0,1,3,4],[0,1,2,3,4],[0,1,2,4,5],[0,1,2,3,4,5]],
[[0,1,2],[0,1,3],[2,4,5],[0,1,2,3],[2,3,4,5],[0,1,2,4,5],[0,1,2,3,4,5]]
]

```

```

end
theory FranklImpl
imports Frankl FamilyImpl
begin

```

```

context SetImpl
begin

```

```

end

```

```

end

```

18 Tactics

```
theory FCTactics
imports FranklImpl SomeShareNegativeImpl
begin
```

ML-file *eval-tac.ML*

18.1 Tactics for verifying FC families

Tactic relies on applying the *SomeShareNegativeImpl.some-share-negative* function.

```
ML⟨
fun FC-family-uce-shares-nonneg-inst w =
  Thm.instantiate'
    [SOME @ {ctyp nat}]
    [SOME (Thm.ctrm-of @ {context} w)]
    @ {thm FC-family-uce-shares-nonneg};

fun SomeShareNegativeSound-nats-inst A A' w w' =
  Thm.instantiate'
    []
    (map (fn x => SOME (Thm.ctrm-of @ {context} x)) [A, A', w', w]) @ {thm
some-share-negative-soundness-nats};

fun fc-family-tac ctx A Al wl =
  let
    val term-zip = @ {term (zip::(nat list => nat list => (nat × nat) list)) [0..<6::nat]}
    val term-weights2map = @ {term weights2map::(nat × nat) list => nat => nat};
    val term-sls = @ {term sorted-list-of-set::nat set=>nat list};
    val term-tabulate = @ {term Mapping.tabulate::nat list => (nat => nat) => (nat,
nat) mapping};
    val term-sup = @ {term Sup::(nat set set=>nat set)};

    val w = term-weights2map $ (term-zip $ wl);
    val w' = list-comb (term-tabulate, [term-sls $ (term-sup $ A), w]);
  in
    resolve-tac ctx [FC-family-uce-shares-nonneg-inst w]
    THEN' asm-full-simp-tac (ctx addsimps [@ {thm weight-fun-def}, @ {thm weights2map-def},
@ {thm upt-def}])
    THEN' resolve-tac ctx [SomeShareNegativeSound-nats-inst A Al w w']
    THEN' asm-full-simp-tac ctx
    THEN' asm-full-simp-tac ctx
    THEN' asm-full-simp-tac ctx
    THEN' asm-full-simp-tac ctx
    THEN' asm-full-simp-tac ctx
  end
⟩
```

18.2 Tactics for verifying nonFC families

Tactic for checking if a family belongs to the union-closed extension of a given family

thm *SetUnionImpl-nats.union-closed-additional-set*

ML⟨⟨

```
(*
fun union-closed-additional-tac' ctx F I i =
  EqSubst.eqsubst-tac ctx [0] [f-to-set-l-thm F] i THEN
  EqSubst.eqsubst-tac ctx [0] [f-to-set-l-thm I] i THEN
  resolve-tac ctx [@{thm SetUnionImpl-lists.union-closed-additional-set}] i THEN
  eval-tac ctx i
*)

fun union-closed-additional-tac' ctx F I i =
  EqSubst.eqsubst-tac ctx [0] [f-to-set-n-thm F] i THEN
  EqSubst.eqsubst-tac ctx [0] [f-to-set-n-thm I] i THEN
  resolve-tac ctx [@{thm SetUnionImpl-nats.union-closed-additional-set}] i THEN
  eval-tac ctx i

fun union-closed-tac' ctx F i =
  EqSubst.eqsubst-tac ctx [0] [f-to-set-l-thm F] i THEN
  resolve-tac ctx [@{thm SetUnionImpl-lists.union-closed-set}] i THEN
  eval-tac ctx i

fun union-closed-additional-select-tac ctx (t, i) =
  case t of
    @{term Trueprop} $ t' => union-closed-additional-select-tac ctx (t', i)
  | @{term UnionClosed.union-closed-additional::nat set set => nat set set => bool}
    $ F $ I => union-closed-additional-tac' ctx F I i
  | - => raise Fail (select-tac: pattern not matched)

fun union-closed-additional-tac ctx i =
  SUBGOAL (union-closed-additional-select-tac ctx) i

fun union-closed-select-tac ctx (t, i) =
  case t of
    @{term Trueprop} $ t' => union-closed-select-tac ctx (t', i)
  | @{term UnionClosed.union-closed::nat set set => bool} $ F => union-closed-tac'
    ctx F i
  | - => raise Fail (select-tac: pattern not matched)

fun union-closed-tac ctx i =
  SUBGOAL (union-closed-select-tac ctx) i

fun union-closed-extension-tac ctx =
```

```

rewrite-goals-tac ctx [@{thm union-closed-extensions-def}] THEN
EqSubst.eqsubst-tac ctx [0] [@{thm mem-Collect-eq}] 1 THEN
resolve-tac ctx [@{thm conjI}] 1 THEN
union-closed-additional-tac ctx 2 THEN
auto-tac ctx
>>

```

Tactic for checking if the given coefficients c satisfy the given system of inequalities determined by the given families Fs

Lemma that allows to reformulate the statement as a problem over lists and gain faster executability.

lemma *nonFC-is-system-solution-lists*:

```

assumes  $Fs = \text{map } f\text{-to-set-l } Fs\text{-l set } Fc\text{-l} = \bigcup Fc \ \forall F \in \text{set } Fs\text{-l. distinct } F$ 
 $\forall F \in \text{set } Fs\text{-l. } \forall A \in \text{set } F. \text{sorted } A \wedge \text{distinct } A$ 
shows  $(\text{let } Fs' = Fs \text{ in } \forall a \in \bigcup Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } c$ 
 $(\text{map } (\text{Frankl.frankl-fun } a) Fs')))) < 0) \longleftrightarrow$ 
 $(\text{list-all } (\lambda a. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } c (\text{map } (\text{frankl-fun-l}$ 
 $a) Fs\text{-l}))) < 0) Fc\text{-l})$ 

```

proof–

```

have  $\forall a. \text{map } (\text{Frankl.frankl-fun } a) Fs = \text{map } (\text{frankl-fun-l } a) Fs\text{-l}$ 
using assms SetImpl-lists.frankl-fun-set
by (simp add: count-l-def)
thus ?thesis
using assms(2)
unfolding list-all-iff
by auto

```

qed

ML⟨⟨

```

fun nonFC-is-system-solution-tac' ctx FF Fc-l i =
let
  val thm = @{thm nonFC-is-system-solution-lists} OF [list-f-to-set-l-thm FF]
  val thm = Thm.instantiate' [] [SOME (Thm.ctrm-of @{context} Fc-l)] thm
in
  EqSubst.eqsubst-tac ctx [0] [thm] i
end

```

```

fun nonFC-is-system-solution-tac-select ctx Fc-l (t, i) =
case t of
  @{term Trueprop} $ t =>
    nonFC-is-system-solution-tac-select ctx Fc-l (t, i)
  | @{term HOL.Let :: nat set set list => (nat set set list => bool) => bool} $ FF
  $ - =>
    nonFC-is-system-solution-tac' ctx FF Fc-l i
  | - $ t => nonFC-is-system-solution-tac-select ctx Fc-l (t, i)
  | - => no-tac

```

```

fun nonFC-is-system-solution-tac ctx Fc-l i =
  SUBGOAL (nonFC-is-system-solution-tac-select ctx Fc-l) i THEN
  eval-tac ctx i THEN
  eval-tac ctx i THEN
  eval-tac ctx i THEN
  eval-tac ctx i
>>

end

```

19 FC status proofs

```

theory FC6-Status
imports FC6-Data SomeShareNegativeImpl WeightsShares-NotFCFamily FCTactics
begin

```

```

lemma [simp]:  $\neg$  FC-family ( $\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,2,3,4\}, \{0,1,2,3,4\}, \{1,2,3,4,5\}, \{0,1,2,3,4,5\}\}$ )
  (is  $\neg$  FC-family ?Fc)
proof-
  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
  set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
  set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,1,2,3,4\}\}$ 
  set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,1,2,3,4\}\}$ 
  set set
  let ?Fs5 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,1,2,3,4\}\}$ 
  set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
  set set
  let ?Fs = [ $?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6$ ]
  let ?c = [29, 29, 29, 29, 37, 4] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\text{?Fc}\}$ 
proof-
  have  $?Fs1 \in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have  $?Fs2 \in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have  $?Fs3 \in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have  $?Fs4 \in \{\text{?Fc}\}$ 

```

```

    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs5 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs6 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
  show ?thesis
    by (simp del: union-closed-extensions-def)
qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,3,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\})$ (is $\neg FC\text{-family } ?Fc$)

proof—

```

  let ?Fs1 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,4}, {0,1,3,4}, {0,1,3,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,3,4,5}, {0,1,2,3,4,5} }
set set
  let ?Fs2 = { {}, {1}, {2}, {1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {0,1,3,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,3,4,5}, {0,1,2,3,4,5} }
set set
  let ?Fs3 = { {}, {0}, {2}, {0,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {0,1,3,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,3,4,5}, {0,1,2,3,4,5} }
set set
  let ?Fs4 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,3,4}, {0,1,3,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,3,4,5}, {0,1,2,3,4,5} }
set set
  let ?Fs5 = { {}, {0}, {1}, {0,1}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {0,1,3,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,3,4,5}, {0,1,2,3,4,5} }
set set
  let ?Fs6 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,4}, {0,1,3,4}, {0,1,3,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,3,4,5}, {0,1,2,3,4,5} }
set set
  let ?Fs7 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,4}, {0,1,3,4}, {0,1,3,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,3,4,5}, {0,1,2,3,4,5} }
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7]

```

```

let ?c = [21, 37, 37, 30, 21, 1, 12] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs7  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next
  show finite ( $\bigcup ?Fc$ )
  by auto
next
  show union-closed ?Fc
  by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```



```

lemma [simp]:  $\neg FC\text{-family}$  ( $\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,5\}, \{0,1,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ )
(is  $\neg FC\text{-family}$  ?Fc)
proof -
  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
  set set
  let ?Fs2 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,1,2,3\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,1,2,3\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,1,2,3\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,1,2,3\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,1,2,3\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [11, 19, 19, 11, 11, 11] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof -
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 

```

```

      by auto
    next
      show length ?c = length ?Fs
      by auto
    next
      show finite (⋃ ?Fc)
      by auto
    next
      show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,5\}, \{0,2,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$ (is $\neg FC\text{-family} ?Fc$)

proof–

```

  let ?Fs1 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}}
  set set

```

```

  let ?Fs2 = {\{\}, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,2,3,4\}}
  set set

```

```

  let ?Fs3 = {\{\}, \{0\}, \{2\}, \{0,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,2,3,4\}}
  set set

```

```

  let ?Fs4 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,2,3,4\}}
  set set

```

```

  let ?Fs5 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}}
  set set

```

```

  let ?Fs6 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}}
  set set

```

```

  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]

```

```

  let ?c = [11, 19, 15, 15, 11, 11] :: nat list

```

show ?thesis

proof (rule nonFC[**where** $Fs = ?Fs$ and $c = ?c$])

```

  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 

```

proof–

```

  have ?Fs1 ∈ {\{?Fc\}}

```

```

    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)

```

moreover

```

  have ?Fs2 ∈ {\{?Fc\}}

```

```

    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)

```

moreover

```

  have ?Fs3 ∈ {\{?Fc\}}

```

```

    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)

```

moreover

```

  have ?Fs4 ∈ {\{?Fc\}}

```

```

    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)

```

moreover

```

  have ?Fs5 ∈ {\{?Fc\}}

```

```

    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)

```

moreover

```

    have ?Fs6 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,2,3,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$ (is $\neg FC\text{-family } ?Fc$)

proof–

let ?Fs1 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$

let ?Fs2 = $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,1,2,4\}\}$

let ?Fs3 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,1,2,4\}\}$

let ?Fs4 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$

let ?Fs5 = $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,1,2,4\}\}$

let ?Fs6 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$

let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]

let ?c = [20, 57, 45, 45, 46, 33] :: nat list

show ?thesis

proof (rule nonFC[where Fs = ?Fs and c = ?c])

show $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$

proof–

have ?Fs1 ∈ {?Fc}

by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)

```

moreover
have ?Fs2 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs3 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs6 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
ultimately
show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
  by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
next
show ∃ wj ∈ List.set ?c. 0 < wj
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite (⋃ ?Fc)
  by auto
next
show union-closed ?Fc
  by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

```

lemma [simp]: ¬ FC-family ({}, {0,1,2,3}, {0,1,2,4}, {0,1,3,5}, {2,3,4,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}) (is ¬ FC-family ?Fc)

proof—

```

let ?Fs1 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
let ?Fs2 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,2,4,5}, {0,1,2,4,5,6}}
let ?Fs3 = {{}, {1}, {2}, {1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {3,4}, {0,1,2,4,5}, {0,1,2,4,5,6}}
let ?Fs4 = {{}, {0}, {2}, {0,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {3,4}, {0,1,2,4,5}, {0,1,2,4,5,6}}

```

```

set set
let ?Fs5 = {{},{0},{1},{0,1},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4},{3,
set set
let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [5, 7, 7, 7, 7, 5] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof-
    have ?Fs1  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{\context} @{\term [0..<6]::nat list} 1 >>)
next
show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite ( $\bigcup ?Fc$ )
  by auto
next
show union-closed ?Fc
  by (tactic << union-closed-tac @{\context} 1 >>)

```

qed
qed

lemma [simp]: $\neg FC\text{-family } (\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,3,4,5\}, \{1,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$
 $set\ set) \text{ (is } \neg FC\text{-family } ?Fc)$

proof–

let $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
 $set\ set$

let $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$
 $set\ set$

let $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
 $set\ set$

let $?Fs4 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,1,2,4\}, \{0,1,2,3,4\}\}$
 $set\ set$

let $?Fs5 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,0,2,4\}, \{0,1,2,3,4\}\}$
 $set\ set$

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,0,1,4\}, \{0,1,2,3,4\}\}$
 $set\ set$

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [3, 4, 4, 4, 4, 3] :: nat\ list$

show $?thesis$

proof (rule nonFC[**where** $Fs = ?Fs$ **and** $c = ?c$])

show $\forall F \in set\ ?Fs. F \in \{\ ?Fc \}$

proof–

have $?Fs1 \in \{\ ?Fc \}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs2 \in \{\ ?Fc \}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs3 \in \{\ ?Fc \}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs4 \in \{\ ?Fc \}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs5 \in \{\ ?Fc \}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs6 \in \{\ ?Fc \}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

ultimately

show $?thesis$

by (simp del: union-closed-extensions-def)

qed

next

show let $Fs = ?Fs$ in $\forall a \in \bigcup ?Fc. sum\text{-list } (map (\lambda (x, y). int\ x * y) (zip\ ?c (map (frankl\text{-fun } a) Fs))) < 0$

by (tactic $\ll nonFC\text{-is-system-solution-tac } @\{context\} @\{term\ } [0..<6]::nat$

```

list} 1 >>))
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show  $\text{length } ?c = \text{length } ?Fs$ 
  by auto
next
  show  $\text{finite } (\bigcup ?Fc)$ 
  by auto
next
  show  $\text{union-closed } ?Fc$ 
  by (tactic << union-closed-tac @{context} 1 >>))
qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$ (is $\neg \text{FC-family } ?Fc$)

proof–

```

let ?Fs1 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}}
set set

```

```

let ?Fs2 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,2,3,4\}}
set set

```

```

let ?Fs3 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,3,4\}}
set set

```

```

let ?Fs4 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}}
set set

```

```

let ?Fs5 = {\{\}, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,2,3,4\}}
set set

```

```

let ?Fs6 = {\{\}, \{0\}, \{2\}, \{0,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,2,3,4\}}
set set

```

```

let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]

```

```

let ?c = [8, 17, 17, 17, 18, 18] :: nat list

```

show $?thesis$

proof (rule nonFC[where $Fs = ?Fs$ and $c = ?c$])

```

  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 

```

proof–

```

  have ?Fs1  $\in \{\{?Fc\}\}$ 

```

```

    by (tactic << union-closed-extension-tac @{context} >>))

```

moreover

```

  have ?Fs2  $\in \{\{?Fc\}\}$ 

```

```

    by (tactic << union-closed-extension-tac @{context} >>))

```

moreover

```

  have ?Fs3  $\in \{\{?Fc\}\}$ 

```

```

    by (tactic << union-closed-extension-tac @{context} >>))

```

moreover

```

  have ?Fs4  $\in \{\{?Fc\}\}$ 

```

```

    by (tactic << union-closed-extension-tac @{context} >>))

```

moreover

```

    have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

```

lemma [simp]: ¬ FC-family ({}, {0,1,2,3}, {0,1,2,4}, {0,1,3,4}, {0,2,3,5}, {1,2,3,5}, {0,1,2,3,4}, {0,1,2,3,5},
set set) (is ¬ FC-family ?Fc)
proof–
  let ?Fs1 = ({}, {0}, {1}, {0,1}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {3,4},
set set
  let ?Fs2 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4},
set set
  let ?Fs3 = ({}, {1}, {2}, {1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {3,4},
set set
  let ?Fs4 = ({}, {0}, {2}, {0,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {3,4},
set set
  let ?Fs5 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3},
set set
  let ?Fs6 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3},
set set
  let ?Fs7 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3},
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7]
  let ?c = [8, 8, 8, 8, 3, 2, 2] :: nat list
  show ?thesis

```



```

proof (rule nonFC[where  $Fs = ?Fs$  and  $c = ?c$ ])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof–
    have  $?Fs1 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have  $?Fs2 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have  $?Fs3 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have  $?Fs4 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have  $?Fs5 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have  $?Fs6 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have  $?Fs7 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    ultimately
    show  $?thesis$ 
    by (simp del: union-closed-extensions-def)
  qed
next
  show let  $Fs = ?Fs$  in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show  $\text{length } ?c = \text{length } ?Fs$ 
  by auto
next
  show  $\text{finite } (\bigcup ?Fc)$ 
  by auto
next
  show  $\text{union-closed } ?Fc$ 
  by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed
qed

```

```

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,2,3,5\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$ 
set set (is  $\neg \text{FC-family } ?Fc$ )

```

```

proof–
  let ?Fs1 = { {}, {0}, {1}, {0,1}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {3,4}, {0,3,4}, {1,3,4}, {0,1,3,4}, {0,1,2,3,4} }
  set set
  let ?Fs2 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {1,2,4}, {0,1,2,4}, {0,1,2,3,4} }
  set set
  let ?Fs3 = { {}, {1}, {2}, {1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {3,4}, {1,3,4}, {2,3,4}, {1,2,3,4} }
  set set
  let ?Fs4 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {1,2,4}, {0,1,2,4}, {0,1,2,3,4} }
  set set
  let ?Fs5 = { {}, {0}, {2}, {0,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {3,4}, {1,3,4}, {2,3,4}, {0,1,2,3,4} }
  set set
  let ?Fs6 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {1,2,4}, {0,1,2,4}, {0,1,2,3,4} }
  set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [23, 23, 24, 17, 18, 11] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{ ?Fc \}$ 
    proof–
      have ?Fs1  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      moreover
      have ?Fs2  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      moreover
      have ?Fs3  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      moreover
      have ?Fs4  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      moreover
      have ?Fs5  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      moreover
      have ?Fs6  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      ultimately
      show ?thesis
      by (simp del: union-closed-extensions-def)
    qed
  next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{ \text{context} \} @\{ \text{term } [0..<6]::\text{nat list} \} 1 \gg$ )
  next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
  next

```

```

    show length ?c = length ?Fs
    by auto
  next
    show finite (⋃ ?Fc)
    by auto
  next
    show union-closed ?Fc
    by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

lemma [simp]: ¬ FC-family ({ {}, {0,1,2,3}, {0,1,2,4}, {0,1,3,5}, {0,1,4,5}, {2,3,4,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5} }) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3} }
  set set
  let ?Fs2 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3} }
  set set
  let ?Fs3 = { {}, {0}, {1}, {0,1}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {3,4}, {0,1,2,3,4} }
  set set
  let ?Fs4 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,3,4} }
  set set
  let ?Fs5 = { {}, {2}, {3}, {2,3}, {0,1,2,3}, {4}, {2,4}, {0,1,2,4}, {3,4}, {2,3,4}, {0,1,2,3,4}, {5}, {2,5}, {3,5}, {0,1,2,3,4,5} }
  set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
  let ?c = [1, 1, 1, 1, 3] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show ∀ F ∈ set ?Fs. F ∈ { ?Fc }
  proof-
    have ?Fs1 ∈ { ?Fc }
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs2 ∈ { ?Fc }
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs3 ∈ { ?Fc }
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs4 ∈ { ?Fc }
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs5 ∈ { ?Fc }
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next

```

```

show let  $Fs = ?Fs$  in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{context\} @\{term [0..<6]::nat \text{list}\} 1 \gg$ )
next
  show  $\exists wj \in List.set ?c. 0 < wj$ 
  by auto
next
  show  $\text{length } ?c = \text{length } ?Fs$ 
  by auto
next
  show  $\text{finite } (\bigcup ?Fc)$ 
  by auto
next
  show  $\text{union-closed } ?Fc$ 
  by (tactic  $\ll \text{union-closed-tac } @\{context\} 1 \gg$ )
qed
qed

```

lemma [*simp*]: $\neg FC\text{-family } (\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,5\}, \{0,2,4,5\}, \{1,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$ (is $\neg FC\text{-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
set set

let $?Fs2 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,1,2,4\}\}$
set set

let $?Fs3 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,0,2,4\}\}$
set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,0,1,4\}\}$
set set

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}\}$
set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [13, 18, 18, 13, 13, 13] :: nat \text{ list}$

show *?thesis*

proof (*rule nonFC[where $Fs = ?Fs$ and $c = ?c$]*)

show $\forall F \in set ?Fs. F \in \{\{?Fc\}\}$

proof–

have $?Fs1 \in \{\{?Fc\}\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs2 \in \{\{?Fc\}\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs3 \in \{\{?Fc\}\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{context\} \gg$)

moreover

```

    have ?Fs4 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,3,5\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$ (is $\neg FC\text{-family } ?Fc$)

proof—

```

  let ?Fs1 = { { }, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5} }
set set
  let ?Fs2 = { { }, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5} }
set set
  let ?Fs3 = { { }, {1}, {2}, {1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {3,4}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5} }
set set
  let ?Fs4 = { { }, {0}, {1}, {0,1}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {3,4}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5} }
set set
  let ?Fs5 = { { }, {0}, {2}, {0,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {3,4}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5} }
set set
  let ?Fs6 = { { }, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5} }
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [37, 52, 56, 37, 53, 24] :: nat list

```

```

show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof-
    have ?Fs1  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
  show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y)) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite ( $\bigcup ?Fc$ )
  by auto
next
show union-closed ?Fc
  by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,2,3,5\}, \{1,2,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$ 
  (is  $\neg \text{FC-family } ?Fc$ )
proof-
  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 

```

```

set set
let ?Fs2 = {{},{0},{1},{0,1},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4},{3,4}}
set set
let ?Fs3 = {{},{1},{2},{1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,4},{3,4}}
set set
let ?Fs4 = {{},{0},{2},{0,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4},{3,4}}
set set
let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4},{0,1,2,4}}
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [14, 29, 31, 31, 22, 22] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
show length ?c = length ?Fs
  by auto

```

```

next
  show finite ( $\bigcup$  ?Fc)
  by auto
next
  show union-closed ?Fc
  by (tactic  $\ll$  union-closed-tac @{context} 1  $\gg$ )
qed
qed

lemma [simp]:  $\neg$  FC-family ( $\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,3,5\}, \{0,2,3,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ )
  (is  $\neg$  FC-family ?Fc)
proof-
  let ?Fs1 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
  set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$ 
  set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
  set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [44, 16, 32, 23, 15, 15] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\text{?Fc}\}$ 
proof-
  have ?Fs1  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs2  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs3  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs4  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs5  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs6  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  ultimately
  show ?thesis

```



```

    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
      by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
      by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
      by auto
  next
    show  $\text{union-closed } ?Fc$ 
      by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,3,5\}, \{0,2,3,5\}, \{0,2,4,5\}, \{0,1,2,3,4\}\}$
 $\text{set set})$ (is $\neg \text{FC-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}\}$
 set set

let $?Fs2 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}\}$
 set set

let $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$
 set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}\}$
 set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
 set set

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
 set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [704124, 378530, 378530, 368378, 247096, 247096] :: \text{nat list}$

show $?thesis$

proof (rule $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$)

show $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}$

proof–

have $?Fs1 \in \{\{?Fc\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \{\{?Fc\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

```

    have ?Fs3 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs4 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

```

lemma [simp]: ¬ FC-family ({}, {0,1,2,3}, {0,1,2,4}, {0,1,3,4}, {0,2,3,5}, {0,2,4,5}, {0,3,4,5}, {0,1,2,3,4},
set set) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = {{}, {1}, {2}, {1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {3,4},
set set
  let ?Fs2 = {{}, {0}, {1}, {0,1}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {3,4},
set set
  let ?Fs3 = {{}, {0}, {2}, {0,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {3,4},
set set
  let ?Fs4 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3},
set set
  let ?Fs5 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,4},
set set
  let ?Fs6 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3},

```

```

set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [143, 75, 52, 52, 75, 75] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof-
    have ?Fs1  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{\context} @{\term [0..<6]::nat list} 1 >>)
next
show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite ( $\bigcup ?Fc$ )
  by auto
next
show union-closed ?Fc
  by (tactic << union-closed-tac @{\context} 1 >>)
qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0, 1, 2\}, \{0, 1, 3, 4\}, \{0, 2, 3, 5\}, \{0, 1, 2, 3, 4\}, \{0, 1, 2, 3, 5\}, \{1, 2, 3, 4, 5\}, \{0, 1, 2,$

```

set set) (is  $\neg$  FC-family ?Fc)
proof-
  let ?Fs1 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4},{5},{0,1,2,3,4},{0,1,2,3,4,5}}
set set
  let ?Fs2 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4},{1,2,4},{0,1,2,4},{0,1,2,3,4},{0,1,2,3,4,5}}
set set
  let ?Fs3 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,4},{0,1,2,3,4},{0,1,2,3,4,5}}
set set
  let ?Fs4 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4},{0,1,2,3,4},{0,1,2,3,4,5}}
set set
  let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3},{0,1,2,3,4},{0,1,2,3,4,5}}
set set
  let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3},{0,1,2,3,4},{0,1,2,3,4,5}}
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [32, 19, 38, 32, 11, 11] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
proof-
  have ?Fs1  $\in \{?Fc\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs2  $\in \{?Fc\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs3  $\in \{?Fc\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs4  $\in \{?Fc\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs5  $\in \{?Fc\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs6  $\in \{?Fc\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  ultimately
  show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll$  nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1  $\gg$ )
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto

```

```

next
  show  $\text{length } ?c = \text{length } ?Fs$ 
  by auto
next
  show  $\text{finite } (\bigcup ?Fc)$ 
  by auto
next
  show  $\text{union-closed } ?Fc$ 
  by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,3,4\}, \{0,2,3,4\}, \{0,1,2,3,4\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}, \{1,2,3,4,5\}\}$ 
 $\text{set set})$  (is  $\neg \text{FC-family } ?Fc$ )
proof-
  let  $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let  $?Fs2 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let  $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let  $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,3,4\}, \{0,2,3,4\}, \{0,1,2,3,4\}, \{5\}, \{0,5\}, \{1,5\}, \{0,1,2,3,4,5\}\}$ 
  set set
  let  $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3,4\}\}$ 
  set set
  let  $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]$ 
  let  $?c = [58, 47, 47, 60, 6] :: \text{nat list}$ 
  show  $?thesis$ 
proof (rule nonFC[where  $Fs = ?Fs$  and  $c = ?c$ ])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
proof-
  have  $?Fs1 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have  $?Fs2 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have  $?Fs3 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have  $?Fs4 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have  $?Fs5 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
  show  $?thesis$ 
  by (simp del: union-closed-extensions-def)
qed

```

```

next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show  $\text{length } ?c = \text{length } ?Fs$ 
  by auto
next
  show  $\text{finite } (\bigcup ?Fc)$ 
  by auto
next
  show  $\text{union-closed } ?Fc$ 
  by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,3,4\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}\})$ (is $\neg \text{FC-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs2 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [32, 27, 18, 21, 16, 10] :: \text{nat list}$

show $?thesis$

proof (rule $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$)

show $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$

proof–

have $?Fs1 \in \{\{?Fc\}\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \{\{?Fc\}\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs3 \in \{\{?Fc\}\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

```

moreover
have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs6 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
ultimately
show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
  by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
next
show ∃ wj ∈ List.set ?c. 0 < wj
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite (⋃ ?Fc)
  by auto
next
show union-closed ?Fc
  by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,3,4\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}\})$ (is $\neg FC\text{-family } ?Fc$)

proof—

```

let ?Fs1 = { { }, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3} }
set set
let ?Fs2 = { { }, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4} }
set set
let ?Fs3 = { { }, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4} }
set set
let ?Fs4 = { { }, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4} }
set set
let ?Fs5 = { { }, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4} }
set set
let ?Fs6 = { { }, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3} }
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]

```

```

let ?c = [14, 32, 32, 26, 14, 7] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof-
    have ?Fs1  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next
  show finite ( $\bigcup ?Fc$ )
  by auto
next
  show union-closed ?Fc
  by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0, 1, 2\}, \{0, 1, 3, 4\}, \{2, 3, 4, 5\}, \{0, 1, 2, 3, 4\}, \{0, 1, 2, 3, 5\}, \{0, 1, 3, 4, 5\}, \{0, 2, 3, 4, 5\}\})$ 
set set) (is  $\neg \text{FC-family } ?Fc$ )
proof-

```



```

    let ?Fs1 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,3,4}}
  set set
    let ?Fs2 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,3,4}}
  set set
    let ?Fs3 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4},{1,2,4},{0,1,2,3,4}}
  set set
    let ?Fs4 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,3,4}}
  set set
    let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3,4}}
  set set
    let ?Fs6 = {{},{0},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3,4}}
  set set
    let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
    let ?c = [46, 45, 30, 44, 27, 17] :: nat list
    show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
  next
    show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show length ?c = length ?Fs

```

```

    by auto
  next
    show finite (⋃ ?Fc)
    by auto
  next
    show union-closed ?Fc
    by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,2,3}, {0,1,4,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,4,5}, {0,1,3,4,5}, {0,1,2,3,4,5})
  (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}}
  set set
  let ?Fs2 = {{}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {0,1,3,4}, {0,1,2,3,4}}
  set set
  let ?Fs3 = {{}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,3,4}, {0,1,2,3,4}}
  set set
  let ?Fs4 = {{}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {0,1,3,4}, {0,1,2,3,4}}
  set set
  let ?Fs5 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,4}, {0,1,3,4}, {0,1,2,3,4}}
  set set
  let ?Fs6 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,3,4}, {0,1,2,3,4}}
  set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [27, 64, 64, 52, 27, 16] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show ∀ F ∈ set ?Fs. F ∈ {?Fc}
proof-
  have ?Fs1 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs2 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs3 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs6 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  ultimately

```

```

    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
  next
    show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,3,4\}, \{0,2,3,4\}, \{0,2,3,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,3,4,5\}\}, \text{set set})$ (is $\neg \text{FC-family } ?Fc$)

proof—

let $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,3,4\}\}$ set set

let $?Fs2 = \{\{3,5\}, \{0,3,5\}, \{0,2,3,5\}, \{0,1,2,3,5\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}, \{0,1,2,3,4,5\}\}::\text{nat set set}$

let $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}\}$ set set

let $?Fs4 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}\}$ set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,3,4\}\}$ set set

let $?Fs6 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,3,4\}\}$ set set

let $?Fs7 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs8 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7, ?Fs8]$

let $?c = [61, 1, 44, 34, 27, 2, 5, 16]::\text{nat list}$

show ?thesis

proof (rule nonFC[where $Fs = ?Fs$ and $c = ?c$])

show $\forall F \in \text{set } ?Fs. F \in \llbracket ?Fc \rrbracket$

proof—

have $?Fs1 \in \llbracket ?Fc \rrbracket$

```

      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs7 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs8 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,3,4}, {0,2,3,4}, {1,3,4,5}, {0,1,2,3,4}, {0,1,3,4,5}, {0,2,3,4,
set set) (is ¬ FC-family ?Fc)
proof-

```

```

    let ?Fs1 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,3,4}}
  set set
    let ?Fs2 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,3,4}}
  set set
    let ?Fs3 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,3,4},{0,2,3,4},{0,1,2,3,4},{5},{0,5},{1,5},{0,1,2,3,4,5}}
  set set
    let ?Fs4 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,3,4}}
  set set
    let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3,4}}
  set set
    let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
    let ?c = [35, 32, 44, 26, 6] :: nat list
    show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof-
    have ?Fs1  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next
  show finite ( $\bigcup ?Fc$ )
  by auto
next

```

```

    show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
  qed
qed

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,3,4}, {0,3,4,5}, {2,3,4,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,3,4,5},
set set) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = ({}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,3,4},
set set
  let ?Fs2 = ({}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,3,4},
set set
  let ?Fs3 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {1,2,4}, {0,1,2,3,4},
set set
  let ?Fs4 = ({}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,3,4},
set set
  let ?Fs5 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3,4},
set set
  let ?Fs6 = ({}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {0,1,3,4}, {0,1,2,3,4}, {5}, {0,1,2,3,4,5},
set set
  let ?Fs7 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3,4},
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7]
  let ?c = [29, 21, 20, 21, 12, 2, 17] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show ∀ F ∈ set ?Fs. F ∈ {?Fc}
    proof-
      have ?Fs1 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
      moreover
      have ?Fs2 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
      moreover
      have ?Fs3 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
      moreover
      have ?Fs4 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
      moreover
      have ?Fs5 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
      moreover
      have ?Fs6 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
      moreover
      have ?Fs7 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    ultimately

```

```

    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
  next
    show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,3,4\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}\}, \text{set set})$ (is $\neg \text{FC-family } ?Fc$)

proof—

let $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$
 set set

let $?Fs2 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$
 set set

let $?Fs3 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$
 set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}\}$
 set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
 set set

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
 set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [32, 58, 49, 38, 28, 16] :: \text{nat list}$

show ?thesis

proof (rule nonFC[where $Fs = ?Fs$ and $c = ?c$])

show $\forall F \in \text{set } ?Fs. F \in \ll ?Fc \gg$

proof—

have $?Fs1 \in \ll ?Fc \gg$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \ll ?Fc \gg$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

```

moreover
have ?Fs3 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs6 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
ultimately
show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
  by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
next
show ∃ wj ∈ List.set ?c. 0 < wj
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite (⋃ ?Fc)
  by auto
next
show union-closed ?Fc
  by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,4,5\}, \{0,3,4,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}\})$
 $\text{set set} \text{ (is } \neg FC\text{-family } ?Fc)$
proof–
let ?Fs1 = $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$
 set set
let ?Fs2 = $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$
 set set
let ?Fs3 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$
 set set
let ?Fs4 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}\}$
 set set
let ?Fs5 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
 set set


```

let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
let ?c = [26, 22, 8, 17, 27] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof –
    have ?Fs1  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y)) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll$  nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1  $\gg$ )
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next
  show finite ( $\bigcup ?Fc$ )
  by auto
next
  show union-closed ?Fc
  by (tactic  $\ll$  union-closed-tac @{context} 1  $\gg$ )
qed
qed

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,3,4\}, \{0,1,3,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}\})$ 
set set (is  $\neg \text{FC-family } ?Fc$ )
proof –
  let ?Fs1 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,3,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}\}$ 
  set set

```

```

    let ?Fs2 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4}}
  set set
    let ?Fs3 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4}}
  set set
    let ?Fs4 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
  set set
    let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
  set set
    let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4},{1,2,4},{0,1,2,4}}
  set set
    let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
    let ?c = [79636, 57824, 31844, 22436, 22436, 34534] :: nat list
    show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next

```

```

    show finite (⋃ ?Fc)
    by auto
next
    show union-closed ?Fc
    by (tactic ⟨⟨ union-closed-tac @ {context} 1 ⟩⟩)
qed
qed

lemma [simp]: ¬ FC-family ({{},{0,1,2},{0,1,2,3},{0,1,2,4},{0,1,3,4},{0,1,2,3,4},{0,1,2,3,5},{0,2,3,4},
set set) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{
set set
  let ?Fs2 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,4},
set set
  let ?Fs3 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4},
set set
  let ?Fs4 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4},
set set
  let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4},
set set
  let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [30252, 74746, 74746, 61390, 31264, 14168] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show ∀ F ∈ set ?Fs. F ∈ {?Fc}
proof-
  have ?Fs1 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
  moreover
  have ?Fs2 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
  moreover
  have ?Fs3 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
  moreover
  have ?Fs4 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
  moreover
  have ?Fs5 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
  moreover
  have ?Fs6 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
  ultimately
  show ?thesis
    by (simp del: union-closed-extensions-def)

```

```

    qed
  next
    show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
  next
    show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}, \{0,1,3,4,5\}, \{0,2,3,4\}, \{0,2,3,4,5\}\})$ (is $\neg \text{FC-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
 set set

let $?Fs2 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs3 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [9240, 22078, 22078, 17794, 9240, 4172] :: \text{nat list}$

show $?thesis$

proof (rule $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$)

show $\forall F \in \text{set } ?Fs. F \in \llbracket ?Fc \rrbracket$

proof–

have $?Fs1 \in \llbracket ?Fc \rrbracket$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \llbracket ?Fc \rrbracket$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs3 \in \llbracket ?Fc \rrbracket$

```

      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic << union-closed-tac @{context} 1 >>)
  qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\})$ (is $\neg FC\text{-family} ?Fc$)

proof—

```

  let ?Fs1 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}}
  let ?Fs2 = {{}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {0,1,2,3,4}}
  let ?Fs3 = {{}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {0,1,2,3,4}}
  let ?Fs4 = {{}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {0,1,2,3,4}}
  let ?Fs5 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {3,4}, {0,1,3,4}, {0,1,2,3,4}}
  let ?Fs6 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}}

```

```

let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [57, 145, 145, 120, 61, 32] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof-
    have ?Fs1  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
  show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite ( $\bigcup ?Fc$ )
  by auto
next
show union-closed ?Fc
  by (tactic << union-closed-tac @{context} 1 >>)
qed
qed
lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,3,4,5\}\})$ 
  (is  $\neg \text{FC-family } ?Fc$ )

```

```

proof–
  let ?Fs1 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3} }
  set set
  let ?Fs2 = { {}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4} }
  set set
  let ?Fs3 = { {}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4} }
  set set
  let ?Fs4 = { {}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4} }
  set set
  let ?Fs5 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4} }
  set set
  let ?Fs6 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3} }
  set set
  let ?Fs = [ ?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6 ]
  let ?c = [ 8, 36, 36, 29, 16, 14 ] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{ ?Fc \}$ 
    proof–
      have ?Fs1  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      moreover
      have ?Fs2  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      moreover
      have ?Fs3  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      moreover
      have ?Fs4  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      moreover
      have ?Fs5  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      moreover
      have ?Fs6  $\in \{ ?Fc \}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{ \text{context} \} \gg$ )
      ultimately
      show ?thesis
      by (simp del: union-closed-extensions-def)
    qed
  next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{ \text{context} \} @\{ \text{term } [0..<6]::\text{nat list} \} 1 \gg$ )
  next
  show  $\exists w_j \in \text{List.set } ?c. 0 < w_j$ 
  by auto
  next

```

```

    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @ {context} 1 ⟩⟩)
qed
qed

lemma [simp]: ¬ FC-family ({ {}, {0,1,2}, {0,1,2,3}, {0,1,3,4}, {2,3,4,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,2,3,4,5},
set set) (is ¬ FC-family ?Fc)
proof -
  let ?Fs1 = ({ {}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {0,1,3,4}, {2,3,4,5} })
set set
  let ?Fs2 = ({ {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,3,4}, {2,3,4,5} })
set set
  let ?Fs3 = ({ {}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {0,1,3,4}, {2,3,4,5} })
set set
  let ?Fs4 = ({ {}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,3,4}, {2,3,4,5} })
set set
  let ?Fs5 = ({ {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,3,4}, {2,3,4,5} })
set set
  let ?Fs6 = ({ {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,3,4}, {2,3,4,5} })
set set
  let ?Fs7 = ({ {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,3,4}, {2,3,4,5} })
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7]
  let ?c = [37, 24, 38, 38, 20, 4, 8] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show ∀ F ∈ set ?Fs. F ∈ { ?Fc }
proof -
  have ?Fs1 ∈ { ?Fc }
    by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
  moreover
  have ?Fs2 ∈ { ?Fc }
    by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
  moreover
  have ?Fs3 ∈ { ?Fc }
    by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
  moreover
  have ?Fs4 ∈ { ?Fc }
    by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
  moreover
  have ?Fs5 ∈ { ?Fc }
    by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
  moreover

```



```

    have ?Fs6 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs7 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,3,4\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}\})$ (is $\neg FC\text{-family } ?Fc$)

proof–

```

  let ?Fs1 = { { }, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,3,4}, {0,2,3,4}, {0,1,2,3,4} }
  let ?Fs2 = { { }, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {0,1,3,4}, {0,2,3,4}, {0,1,2,3,4} }
  let ?Fs3 = { { }, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,3,4}, {0,2,3,4}, {0,1,2,3,4} }
  let ?Fs4 = { { }, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {0,1,3,4}, {0,2,3,4}, {0,1,2,3,4} }
  let ?Fs5 = { { }, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4} }
  let ?Fs6 = { { }, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4} }
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [12, 19, 19, 18, 11, 6] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show ∀ F ∈ set ?Fs. F ∈ {?Fc}
  end

```

```

proof–
  have ?Fs1 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs2 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs3 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs4 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs5 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs6 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  ultimately
  show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,2,3}, {0,1,4,5}, {2,3,4,5}, {0,1,2,3,4}, {0,1,2,4,5}, {0,2,3,4,5}, {0,1,2,3,4,5}) (is ¬ FC-family ?Fc)

proof–

```

  let ?Fs1 = {{0,1,2,3},{0,1,2,3,4},{0,1,2,3,4,5}}::nat set set
  let ?Fs2 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}::nat set set
  let ?Fs3 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,4}}::nat set set

```

```

set set
let ?Fs4 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4}}
set set
let ?Fs5 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4}}
set set
let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
let ?Fs7 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4}}
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7]
let ?c = [1, 29, 50, 50, 49, 27, 18] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs7  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
  show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
by auto
next
show length ?c = length ?Fs

```



```

    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs5 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs6 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
  show ?thesis
    by (simp del: union-closed-extensions-def)
qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0, 1, 2\}, \{0, 1, 2, 3\}, \{0, 1, 4, 5\}, \{2, 3, 4, 5\}, \{0, 1, 2, 3, 4\}, \{0, 1, 2, 3, 5\}, \{0, 1, 2, 4, 5\}, \{0, 1, 3, 4, 5\}, \{0, 1, 2, 3, 4, 5\}\})$ (is $\neg FC\text{-family} ?Fc$)

proof—

```

  let ?Fs1 = {{}, {0}, {1}, {0, 1}, {2}, {0, 2}, {1, 2}, {0, 1, 2}, {3}, {0, 3}, {1, 3}, {0, 1, 3}, {2, 3}, {0, 2, 3}, {1, 2, 3}, {0, 1, 2, 3}}
set set
  let ?Fs2 = {{}, {0}, {1}, {0, 1}, {2}, {0, 2}, {1, 2}, {0, 1, 2}, {3}, {0, 3}, {1, 3}, {0, 1, 3}, {2, 3}, {0, 2, 3}, {1, 2, 3}, {0, 1, 2, 3}, {4}}
set set
  let ?Fs3 = {{}, {1}, {2}, {1, 2}, {0, 1, 2}, {3}, {1, 3}, {2, 3}, {1, 2, 3}, {0, 1, 2, 3}, {4}, {1, 4}, {2, 4}, {1, 2, 4}, {0, 1, 2, 4}, {3, 4}}
set set
  let ?Fs4 = {{}, {0}, {1}, {0, 1}, {0, 1, 2}, {3}, {0, 3}, {1, 3}, {0, 1, 3}, {0, 1, 2, 3}, {4}, {0, 4}, {1, 4}, {0, 1, 4}, {0, 1, 2, 4}, {3, 4}}
set set
  let ?Fs5 = {{}, {0}, {1}, {0, 1}, {2}, {0, 2}, {1, 2}, {0, 1, 2}, {0, 1, 2, 3}, {4}, {0, 4}, {1, 4}, {0, 1, 4}, {2, 4}, {0, 2, 4}, {1, 2, 4}, {0, 1, 2, 4}}
set set
  let ?Fs6 = {{}, {0}, {2}, {0, 2}, {0, 1, 2}, {3}, {0, 3}, {2, 3}, {0, 2, 3}, {0, 1, 2, 3}, {4}, {0, 4}, {2, 4}, {0, 2, 4}, {0, 1, 2, 4}, {3, 4}}
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [20, 20, 41, 36, 13, 38] :: nat list
  show ?thesis

```

```

proof (rule nonFC[where  $Fs = ?Fs$  and  $c = ?c$ ])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof–
    have  $?Fs1 \in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have  $?Fs2 \in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have  $?Fs3 \in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have  $?Fs4 \in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have  $?Fs5 \in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have  $?Fs6 \in \{?Fc\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    ultimately
    show  $?thesis$ 
    by (simp del: union-closed-extensions-def)
  qed
next
  show let  $Fs = ?Fs$  in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show  $\text{length } ?c = \text{length } ?Fs$ 
  by auto
next
  show  $\text{finite } (\bigcup ?Fc)$ 
  by auto
next
  show  $\text{union-closed } ?Fc$ 
  by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed
qed

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,3,4,5\}, \{1,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}\})$ 
set set (is  $\neg \text{FC-family } ?Fc$ )
proof–
  let  $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set

```

```

    let ?Fs2 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4}}
  set set
    let ?Fs3 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,4}}
  set set
    let ?Fs4 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,4}}
  set set
    let ?Fs5 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4}}
  set set
    let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
  set set
    let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
    let ?c = [17, 17, 27, 27, 22, 16] :: nat list
    show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next

```



```

    show finite (⋃ ?Fc)
    by auto
next
    show union-closed ?Fc
    by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,2,3}, {0,1,3,4}, {0,1,3,5}, {0,2,4,5}, {0,1,2,3,4}, {0,1,2,3,5},
set set) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = ({}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,3,4},
set set
  let ?Fs2 = ({}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,3,4},
set set
  let ?Fs3 = ({}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,3,4},
set set
  let ?Fs4 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3,4},
set set
  let ?Fs5 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,3,4},
set set
  let ?Fs6 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3,4},
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [12100, 9034, 6824, 2830, 3222, 2830] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show ∀ F ∈ set ?Fs. F ∈ {?Fc}
proof-
  have ?Fs1 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs2 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs3 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs6 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  ultimately
  show ?thesis
  by (simp del: union-closed-extensions-def)

```

```

    qed
  next
    show let  $Fs = ?Fs$  in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
  next
    show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

lemma $[\text{simp}]$: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,3,4\}, \{0,1,3,5\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}\} \text{ set set})$ (is $\neg \text{FC-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}\}$ set set

let $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$ set set

let $?Fs3 = \{\{\}, \{2\}, \{0,1,2\}, \{3\}, \{2,3\}, \{0,1,2,3\}, \{4\}, \{2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$ set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]$

let $?c = [18, 15, 53, 8, 8] :: \text{nat list}$

show $?thesis$

proof (rule $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$)

show $\forall F \in \text{set } ?Fs. F \in \llbracket ?Fc \rrbracket$

proof–

have $?Fs1 \in \llbracket ?Fc \rrbracket$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \llbracket ?Fc \rrbracket$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs3 \in \llbracket ?Fc \rrbracket$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

```

    have ?Fs4 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

```

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,2,3}, {0,1,3,4}, {0,2,3,5}, {0,2,4,5}, {0,1,2,3,4}, {0,1,2,3,5})
set set) (is ¬ FC-family ?Fc)
proof–
  let ?Fs1 = ({}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,3,4}, {0,1,2,3,5})
set set
  let ?Fs2 = ({}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,3,4}, {0,1,2,3,5})
set set
  let ?Fs3 = ({}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,3,4}, {0,1,2,3,5})
set set
  let ?Fs4 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,3,4}, {0,1,2,3,5})
set set
  let ?Fs5 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}, {0,1,2,3,5})
set set
  let ?Fs6 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}, {0,1,2,3,5})
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [932597, 693189, 527149, 236278, 219240, 231261] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show ∀ F ∈ set ?Fs. F ∈ {?Fc}

```

```

proof–
  have ?Fs1 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs2 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs3 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs4 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs5 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs6 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  ultimately
  show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: ¬ FC-family ({ {}, {0,1,2}, {0,1,2,3}, {0,1,3,4}, {0,1,4,5}, {0,2,4,5}, {0,1,2,3,4}, {0,1,2,4,5},
set set) (is ¬ FC-family ?Fc)

proof–

```

  let ?Fs1 = { {}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4},  
set set
  let ?Fs2 = { {}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4},  
set set

```

```

    let ?Fs3 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,3,4}},
    set set
    let ?Fs4 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}},
    set set
    let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}},
    set set
    let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4}},
    set set
    let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
    let ?c = [37927, 27507, 20582, 15435, 9416, 4017] :: nat list
    show ?thesis
    proof (rule nonFC[where Fs = ?Fs and c = ?c])
      show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
      proof-
        have ?Fs1  $\in \{?Fc\}$ 
        by (tactic << union-closed-extension-tac @{context} >>))
      moreover
      have ?Fs2  $\in \{?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
      moreover
      have ?Fs3  $\in \{?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
      moreover
      have ?Fs4  $\in \{?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
      moreover
      have ?Fs5  $\in \{?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
      moreover
      have ?Fs6  $\in \{?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
      ultimately
      show ?thesis
      by (simp del: union-closed-extensions-def)
    qed
  next
    show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>))
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show length ?c = length ?Fs
    by auto
  next
    show finite ( $\bigcup ?Fc$ )
    by auto

```

```

next
  show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,2,3}, {0,1,3,4}, {0,1,4,5}, {2,3,4,5}, {0,1,2,3,4}, {0,1,2,4,5}, {0,1,3,4,5}, {0,1,4,5,6}, {2,3,4,5,6}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6})
set set) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}}
set set
  let ?Fs2 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,4}}
set set
  let ?Fs3 = {{}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {0,1,2,3,4}}
set set
  let ?Fs4 = {{}, {2}, {0,1,2}, {3}, {2,3}, {0,1,2,3}, {4}, {2,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {2,3,4}, {0,1,2,3,4}, {0,1,2,3,4,5}}
set set
  let ?Fs5 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}}
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
  let ?c = [17, 11, 22, 65, 10] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show ∀ F ∈ set ?Fs. F ∈ {?Fc}
proof-
  have ?Fs1 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs2 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs3 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs4 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs5 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
  show ?thesis
    by (simp del: union-closed-extensions-def)
qed
qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>)
next

```

```

    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
      by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
      by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
      by auto
  next
    show  $\text{union-closed } ?Fc$ 
      by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed
qed

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,3,4\}, \{0,3,4,5\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,3,4,5\}\}$ 
 $\text{set set})$  (is  $\neg \text{FC-family } ?Fc$ )
proof-
  let  $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$ 
 $\text{set set}$ 
  let  $?Fs2 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
 $\text{set set}$ 
  let  $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}\}$ 
 $\text{set set}$ 
  let  $?Fs4 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$ 
 $\text{set set}$ 
  let  $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$ 
 $\text{set set}$ 
  let  $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$ 
 $\text{set set}$ 
  let  $?Fs7 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$ 
 $\text{set set}$ 
  let  $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7]$ 
  let  $?c = [27, 40, 29, 31, 1, 23, 14] :: \text{nat list}$ 
  show  $?thesis$ 
  proof (rule nonFC[where  $Fs = ?Fs$  and  $c = ?c$ ])
    show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
    proof-
      have  $?Fs1 \in \{\{?Fc\}\}$ 
        by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs2 \in \{\{?Fc\}\}$ 
        by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs3 \in \{\{?Fc\}\}$ 
        by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs4 \in \{\{?Fc\}\}$ 
        by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
    end
  end

```

```

    have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs7 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,4,5\}, \{0,3,4,5\}, \{1,3,4,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}\}$
 $set\ set)$ (is $\neg FC\text{-family } ?Fc$)

proof—

```

  let ?Fs1 = {\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,4,5\}}
set set
  let ?Fs2 = {\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,4,5\}}
set set
  let ?Fs3 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,3,4\}, \{0,1,2,3,4,5\}}
set set
  let ?Fs4 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{0,1,2,3,4\}, \{0,1,2,3,4,5\}}
set set
  let ?Fs5 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{0,1,4,5\}, \{0,1,2,4,5\}, \{3,4,5\}, \{0,1,2,3,4,5\}}
set set
  let ?Fs6 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{5\}, \{0,5\}, \{1,5\}, \{0,1,5\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}}
set set
  let ?Fs7 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}, \{0,1,2,3,4,5\}}
set set

```



```

let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7]
let ?c = [14, 14, 6, 16, 5, 4, 4] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs7  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next
  show finite ( $\bigcup ?Fc$ )
  by auto
next
  show union-closed ?Fc
  by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```

```

lemma [simp]:  $\neg FC\text{-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,4,5\}, \{0,3,4,5\}, \{2,3,4,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}\})$ 
set set (is  $\neg FC\text{-family } ?Fc$ )
proof -
  let  $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
set set
  let  $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}\}$ 
set set
  let  $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$ 
set set
  let  $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
  let  $?Fs5 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$ 
set set
  let  $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]$ 
  let  $?c = [22, 16, 9, 27, 17] :: \text{nat list}$ 
  show  $?thesis$ 
proof (rule nonFC[where  $Fs = ?Fs$  and  $c = ?c$ ])
    show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
    proof -
      have  $?Fs1 \in \{?Fc\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs2 \in \{?Fc\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs3 \in \{?Fc\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs4 \in \{?Fc\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs5 \in \{?Fc\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      ultimately
      show  $?thesis$ 
      by (simp del: union-closed-extensions-def)
    qed
  next
    show let  $Fs = ?Fs$  in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto

```

```

next
  show finite ( $\bigcup$  ?Fc)
  by auto
next
  show union-closed ?Fc
  by (tactic  $\ll$  union-closed-tac @{context} 1  $\gg$ )
qed
qed

lemma [simp]:  $\neg$  FC-family ( $\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}\}$ 
set set) (is  $\neg$  FC-family ?Fc)
proof-
  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
  let ?Fs2 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [78355, 295203, 264999, 147297, 196138, 128904] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\text{?Fc}\}$ 
proof-
  have ?Fs1  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs2  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs3  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs4  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs5  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs6  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  ultimately
  show ?thesis

```

```

    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic « nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 »)
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite ( $\bigcup ?Fc$ )
      by auto
  next
    show union-closed ?Fc
      by (tactic « union-closed-tac @{context} 1 »)
  qed
qed

```

lemma [simp]: $\neg FC\text{-family } (\{\{\}, \{0, 1, 2\}, \{0, 1, 2, 3\}, \{0, 1, 3, 4\}, \{0, 1, 3, 5\}, \{0, 3, 4, 5\}, \{0, 1, 2, 3, 4\}, \{0, 1, 2, 3, 5\}\} \text{ set set})$ (is $\neg FC\text{-family } ?Fc$)

proof–

```

  let ?Fs1 =  $\{\{\}, \{1\}, \{2\}, \{1, 2\}, \{0, 1, 2\}, \{3\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}, \{0, 1, 2, 3\}, \{4\}, \{1, 4\}, \{2, 4\}, \{1, 2, 4\}, \{0, 1, 2, 4\}\}$ 
  set set
  let ?Fs2 =  $\{\{1, 4, 5\}, \{0, 1, 4, 5\}, \{0, 1, 2, 4, 5\}, \{0, 1, 3, 4, 5\}, \{0, 1, 2, 3, 4, 5\}\} :: \text{nat set}$ 
  set
  let ?Fs3 =  $\{\{\}, \{0\}, \{1\}, \{0, 1\}, \{0, 1, 2\}, \{3\}, \{0, 3\}, \{1, 3\}, \{0, 1, 3\}, \{0, 1, 2, 3\}, \{4\}, \{0, 4\}, \{1, 4\}, \{0, 1, 4\}, \{0, 1, 2, 4\}\}$ 
  set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{2\}, \{0, 2\}, \{0, 1, 2\}, \{3\}, \{0, 3\}, \{2, 3\}, \{0, 2, 3\}, \{0, 1, 2, 3\}, \{4\}, \{0, 4\}, \{2, 4\}, \{0, 2, 4\}, \{0, 1, 2, 4\}\}$ 
  set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0, 1\}, \{2\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\}, \{0, 1, 2, 3\}, \{4\}, \{0, 4\}, \{1, 4\}, \{0, 1, 4\}, \{2, 4\}, \{0, 2, 4\}\}$ 
  set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{0, 1\}, \{2\}, \{0, 2\}, \{0, 1, 2\}, \{3\}, \{0, 3\}, \{0, 1, 3\}, \{2, 3\}, \{0, 2, 3\}, \{0, 1, 2, 3\}, \{0, 1, 3, 4\}, \{0, 1, 2, 3, 4\}\}$ 
  set set
  let ?Fs7 =  $\{\{\}, \{0\}, \{1\}, \{0, 1\}, \{2\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\}, \{3\}, \{0, 3\}, \{1, 3\}, \{0, 1, 3\}, \{2, 3\}, \{0, 2, 3\}, \{1, 2, 3\}, \{0, 1, 2, 3\}\}$ 
  set set
  let ?Fs8 =  $\{\{\}, \{0\}, \{1\}, \{0, 1\}, \{2\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\}, \{3\}, \{0, 3\}, \{1, 3\}, \{0, 1, 3\}, \{2, 3\}, \{0, 2, 3\}, \{1, 2, 3\}, \{0, 1, 2, 3\}\}$ 
  set set
  let ?Fs = [ $?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7, ?Fs8$ ]
  let ?c = [28, 1, 11, 20, 13, 1, 7, 6] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \mathbb{N} ?Fc$ 
    proof–
      have  $?Fs1 \in \mathbb{N} ?Fc$ 
      by (tactic « union-closed-extension-tac @{context} »)
    qed
  qed

```

```

moreover
have ?Fs2 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs3 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs6 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs7 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs8 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
ultimately
show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
  by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
next
show ∃ wj ∈ List.set ?c. 0 < wj
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite (⋃ ?Fc)
  by auto
next
show union-closed ?Fc
  by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

```

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,2,3}, {0,1,3,4}, {0,2,3,5}, {0,3,4,5}, {0,1,2,3,4}, {0,1,2,3,5},
set set) (is ¬ FC-family ?Fc)

proof—

let ?Fs1 = {{}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}}

```

set set
  let ?Fs2 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4}}
set set
  let ?Fs3 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4}}
set set
  let ?Fs4 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4}}
set set
  let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
  let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [117, 68, 68, 50, 29, 29] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
    proof -
      have ?Fs1  $\in \{?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
    moreover
      have ?Fs2  $\in \{?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
    moreover
      have ?Fs3  $\in \{?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
    moreover
      have ?Fs4  $\in \{?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
    moreover
      have ?Fs5  $\in \{?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
    moreover
      have ?Fs6  $\in \{?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
    ultimately
      show ?thesis
      by (simp del: union-closed-extensions-def)
    qed
  next
    show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>))
  next
    show  $\exists w_j \in \text{List.set } ?c. 0 < w_j$ 
    by auto
  next
    show length ?c = length ?Fs
    by auto

```

```

next
  show finite ( $\bigcup$  ?Fc)
  by auto
next
  show union-closed ?Fc
  by (tactic  $\ll$  union-closed-tac @{context} 1  $\gg$ )
qed
qed

lemma [simp]:  $\neg$  FC-family ( $\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,3,4\}, \{0,1,4,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\}$ 
set set) (is  $\neg$  FC-family ?Fc)
proof-
  let ?Fs1 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [165436, 122596, 67260, 41682, 45304, 71336] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\text{?Fc}\}$ 
proof-
  have ?Fs1  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs2  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs3  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs4  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs5  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs6  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  ultimately
  show ?thesis

```

```

    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
      by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
      by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
      by auto
  next
    show  $\text{union-closed } ?Fc$ 
      by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,3,4\}, \{0,2,4,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}\} \text{ set set})$ (is $\neg \text{FC-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ set set

let $?Fs2 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$ set set

let $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}\}$ set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$ set set

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs7 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7]$

let $?c = [50, 29, 29, 21, 14, 2, 11] :: \text{nat list}$

show $?thesis$

proof (rule $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$)

show $\forall F \in \text{set } ?Fs. F \in \llbracket ?Fc \rrbracket$

proof–

have $?Fs1 \in \llbracket ?Fc \rrbracket$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \llbracket ?Fc \rrbracket$


```

      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs7 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```

```

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,2,3}, {0,1,2,4}, {0,1,3,5}, {2,3,4,5}, {0,1,2,3,4}, {0,1,2,3,5},
set set) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {
set set
  let ?Fs2 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {
set set
  let ?Fs3 = ({}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {
set set

```



```

next
  show finite ( $\bigcup$  ?Fc)
  by auto
next
  show union-closed ?Fc
  by (tactic  $\ll$  union-closed-tac @{context} 1  $\gg$ )
qed
qed

lemma [simp]:  $\neg$  FC-family ( $\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,5\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}\}$ 
set set) (is  $\neg$  FC-family ?Fc)
proof-
  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs3 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{0,1\}, \{2\}, \{0,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs7 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7]
  let ?c = [19, 65, 73, 39, 66, 1, 34] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{\text{?Fc}\}$ 
    proof-
      have ?Fs1  $\in \{\text{?Fc}\}$ 
      by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
      moreover
      have ?Fs2  $\in \{\text{?Fc}\}$ 
      by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
      moreover
      have ?Fs3  $\in \{\text{?Fc}\}$ 
      by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
      moreover
      have ?Fs4  $\in \{\text{?Fc}\}$ 
      by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
      moreover
      have ?Fs5  $\in \{\text{?Fc}\}$ 
      by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
      moreover
      have ?Fs6  $\in \{\text{?Fc}\}$ 
      by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
    end
  end

```

```

moreover
have  $?Fs7 \in \{\{?Fc\}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac  $@\{context\}$   $\gg$ )
ultimately
show  $?thesis$ 
  by (simp del: union-closed-extensions-def)
qed
next
show let  $Fs = ?Fs$  in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip}$ 
 $?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll$  nonFC-is-system-solution-tac  $@\{context\}$   $@\{term [0..<6]::nat$ 
 $list\}$   $1$   $\gg$ )
next
show  $\exists wj \in List.set ?c. 0 < wj$ 
  by auto
next
show  $\text{length } ?c = \text{length } ?Fs$ 
  by auto
next
show finite  $(\bigcup ?Fc)$ 
  by auto
next
show union-closed  $?Fc$ 
  by (tactic  $\ll$  union-closed-tac  $@\{context\}$   $1$   $\gg$ )
qed
qed

```

```

lemma [simp]:  $\neg FC\text{-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,3,4,5\}, \{1,3,4,5\}, \{0,1,2,3,4\}, \{0,1,3,4,5\}\}$ 
 $set\ set)$  (is  $\neg FC\text{-family } ?Fc$ )
proof–
  let  $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$ 
 $set\ set$ 
  let  $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
 $set\ set$ 
  let  $?Fs3 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
 $set\ set$ 
  let  $?Fs4 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$ 
 $set\ set$ 
  let  $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}\}$ 
 $set\ set$ 
  let  $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
 $set\ set$ 
  let  $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$ 
  let  $?c = [10, 10, 16, 16, 13, 9] :: nat\ list$ 
show  $?thesis$ 
proof (rule nonFC [where  $Fs = ?Fs$  and  $c = ?c$ ])
  show  $\forall F \in set\ ?Fs. F \in \{\{?Fc\}\}$ 
proof–
  have  $?Fs1 \in \{\{?Fc\}\}$ 

```

```

      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,4,5\}, \{0,2,4,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}\}$
 $set\ set)$ (is $\neg FC\text{-family } ?Fc$)

proof–

```

  let ?Fs1 = {\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}, \{0,1,2,3,4,5\}}
  set set
  let ?Fs2 = {\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}, \{0,1,2,3,4,5\}}
  set set
  let ?Fs3 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}, \{0,1,2,3,4,5\}}
  set set

```

```

    let ?Fs4 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{
set set
    let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4},{
set set
    let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{
set set
    let ?Fs7 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4},{
set set
    let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7]
    let ?c = [98, 56, 56, 42, 8, 42, 3] :: nat list
    show ?thesis
    proof (rule nonFC[where Fs = ?Fs and c = ?c])
      show  $\forall F \in \text{set } ?Fs. F \in \{\?Fc\}$ 
      proof-
        have ?Fs1  $\in \{\?Fc\}$ 
          by (tactic << union-closed-extension-tac @{context} >>)
        moreover
        have ?Fs2  $\in \{\?Fc\}$ 
          by (tactic << union-closed-extension-tac @{context} >>)
        moreover
        have ?Fs3  $\in \{\?Fc\}$ 
          by (tactic << union-closed-extension-tac @{context} >>)
        moreover
        have ?Fs4  $\in \{\?Fc\}$ 
          by (tactic << union-closed-extension-tac @{context} >>)
        moreover
        have ?Fs5  $\in \{\?Fc\}$ 
          by (tactic << union-closed-extension-tac @{context} >>)
        moreover
        have ?Fs6  $\in \{\?Fc\}$ 
          by (tactic << union-closed-extension-tac @{context} >>)
        moreover
        have ?Fs7  $\in \{\?Fc\}$ 
          by (tactic << union-closed-extension-tac @{context} >>)
        ultimately
        show ?thesis
          by (simp del: union-closed-extensions-def)
      qed
    next
      show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
        by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
    next
      show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
        by auto
    next
      show length ?c = length ?Fs
        by auto

```

```

next
  show finite ( $\bigcup$  ?Fc)
  by auto
next
  show union-closed ?Fc
  by (tactic  $\ll$  union-closed-tac @{context} 1  $\gg$ )
qed
qed

lemma [simp]:  $\neg$  FC-family ( $\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}\}$ 
set set) (is  $\neg$  FC-family ?Fc)
proof-
  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs4 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [73763, 73763, 138832, 153866, 144226, 38988] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\text{?Fc}\}$ 
proof-
  have ?Fs1  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs2  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs3  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs4  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs5  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs6  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  ultimately
  show ?thesis

```

```

    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
next
  show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
next
  show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
next
  show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}\}$
 $\text{set set})$ (is $\neg \text{FC-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
 set set

let $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$
 set set

let $?Fs3 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs5 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [214769, 223347, 447215, 395473, 419030, 113672] :: \text{nat list}$

show $?thesis$

proof (rule $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$)

show $\forall F \in \text{set } ?Fs. F \in \ll ?Fc \gg$

proof–

have $?Fs1 \in \ll ?Fc \gg$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \ll ?Fc \gg$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover


```

    have ?Fs3 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs4 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}\}$
 $set\ set)$ (is $\neg FC\text{-family } ?Fc$)

proof—

```

  let ?Fs1 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}}
  set set
  let ?Fs2 = {\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}}
  set set
  let ?Fs3 = {\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}}
  set set
  let ?Fs4 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}}
  set set
  let ?Fs5 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}}
  set set
  let ?Fs6 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}}

```

```

set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [11, 27, 23, 23, 11, 11] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\?Fc\}$ 
  proof-
    have ?Fs1  $\in \{\?Fc\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs2  $\in \{\?Fc\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs3  $\in \{\?Fc\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs4  $\in \{\?Fc\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs5  $\in \{\?Fc\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    moreover
    have ?Fs6  $\in \{\?Fc\}$ 
    by (tactic << union-closed-extension-tac @{\context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{\context} @{\term [0..<6]::nat list} 1 >>)
next
show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite ( $\bigcup ?Fc$ )
  by auto
next
show union-closed ?Fc
  by (tactic << union-closed-tac @{\context} 1 >>)
qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0, 1, 2\}, \{0, 1, 2, 3\}, \{0, 1, 2, 4\}, \{0, 3, 4, 5\}, \{1, 3, 4, 5\}, \{0, 1, 2, 3, 4\}, \{0, 1, 2, 3, 5\}$

```

set set) (is  $\neg$  FC-family ?Fc)
proof-
  let ?Fs1 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
  let ?Fs2 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4},{0,1,2,4}}
set set
  let ?Fs3 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
  let ?Fs4 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,4},{0,1,2,3,4}}
set set
  let ?Fs5 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4},{0,1,2,3,4}}
set set
  let ?Fs6 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4},{0,1,2,3,4}}
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [27, 29, 29, 54, 51, 42] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
proof-
  have ?Fs1  $\in \{?Fc\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs2  $\in \{?Fc\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs3  $\in \{?Fc\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs4  $\in \{?Fc\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs5  $\in \{?Fc\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs6  $\in \{?Fc\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
  show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto

```

```

next
  show  $\text{length } ?c = \text{length } ?Fs$ 
  by auto
next
  show  $\text{finite } (\bigcup ?Fc)$ 
  by auto
next
  show  $\text{union-closed } ?Fc$ 
  by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,2,3,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$ 
  (is  $\neg \text{FC-family } ?Fc$ )
proof-
  let ?Fs1 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
  set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
  set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
  set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
  set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
  set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
  set set
  let ?Fs = [ $?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6$ ]
  let ?c = [57, 33, 24, 33, 13, 14] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
proof-
  have  $?Fs1 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have  $?Fs2 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have  $?Fs3 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have  $?Fs4 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have  $?Fs5 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have  $?Fs6 \in \{\{?Fc\}\}$ 

```

```

      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
      by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,5\}, \{0,3,4,5\}, \{1,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,3,4,5\}, \{1,3,4,5\}, \{0,1,2,3,4,5\}\})$ (is $\neg FC\text{-family} ?Fc$)

proof–

let ?Fs1 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$

let ?Fs2 = $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$

let ?Fs3 = $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$

let ?Fs4 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$

let ?Fs5 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$

let ?Fs6 = $\{\{\}, \{1\}, \{0,1,2\}, \{3\}, \{1,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{0,1,2,3,4\}, \{5\}, \{1,2,3,4,5\}, \{0,1,2,3,4,5\}\}$

let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]

let ?c = [32, 19, 62, 32, 32, 86] :: nat list

show ?thesis

proof (rule nonFC[where Fs = ?Fs and c = ?c])

show $\forall F \in \text{set } ?Fs. F \in \llbracket ?Fc \rrbracket$

proof–

have ?Fs1 ∈ $\llbracket ?Fc \rrbracket$

by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)

moreover

```

    have ?Fs2 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs3 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs4 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,5\}, \{0,1,4,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$
 $set\ set) \text{ (is } \neg FC\text{-family } ?Fc)$

proof—

```

  let ?Fs1 = {\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}}
  set set
  let ?Fs2 = {\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}}
  set set
  let ?Fs3 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}}
  set set
  let ?Fs4 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}}
  set set

```

```

let ?Fs5 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3} }
set set
let ?Fs6 = { {}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {1,2,4}, {0,1,2,4} }
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [204402, 158745, 86596, 82121, 52537, 52479] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof–
    have ?Fs1  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite ( $\bigcup ?Fc$ )
    by auto
  next
  show union-closed ?Fc
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed

```

qed

```

lemma [simp]:  $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,5\}, \{0,1,3,4\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$ 
set set (is  $\neg FC\text{-family } ?Fc$ )
proof -
  let ?Fs1 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [256637, 76996, 189326, 145602, 81687, 50367] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
    proof -
      have ?Fs1  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have ?Fs2  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have ?Fs3  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have ?Fs4  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have ?Fs5  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have ?Fs6  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      ultimately
      show ?thesis
      by (simp del: union-closed-extensions-def)
    qed
  next
    show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )

```



```

next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show  $\text{length } ?c = \text{length } ?Fs$ 
  by auto
next
  show  $\text{finite } (\bigcup ?Fc)$ 
  by auto
next
  show  $\text{union-closed } ?Fc$ 
  by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed
qed

```

```

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,5\}, \{0,1,3,4\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{2,3,4,5\}\})$ 
  (is  $\neg \text{FC-family } ?Fc$ )
proof-
  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
  set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
  set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
  set set
  let ?Fs5 =  $\{\{\}, \{2\}, \{0,1,2\}, \{3\}, \{2,3\}, \{0,1,2,3\}, \{4\}, \{2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs = [ $?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5$ ]
  let ?c = [3, 10, 5, 5, 23] :: nat list
  show ?thesis
proof (rule nonFC[where  $Fs = ?Fs$  and  $c = ?c$ ])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
proof-
  have  $?Fs1 \in \{\{?Fc\}\}$ 
  by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs2 \in \{\{?Fc\}\}$ 
  by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs3 \in \{\{?Fc\}\}$ 
  by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs4 \in \{\{?Fc\}\}$ 
  by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs5 \in \{\{?Fc\}\}$ 
  by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  ultimately

```

```

    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
  next
    show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,1,3,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$ (is $\neg \text{FC-family } ?Fc$)

proof—

let $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs2 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [459409, 352236, 190460, 188023, 101088, 108864] :: \text{nat list}$

show ?thesis

proof (rule nonFC[where $Fs = ?Fs$ and $c = ?c$])

show $\forall F \in \text{set } ?Fs. F \in \llbracket ?Fc \rrbracket$

proof—

have $?Fs1 \in \llbracket ?Fc \rrbracket$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \llbracket ?Fc \rrbracket$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

```

moreover
have ?Fs3 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs6 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
ultimately
show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
  by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
next
show ∃ wj ∈ List.set ?c. 0 < wj
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite (⋃ ?Fc)
  by auto
next
show union-closed ?Fc
  by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

```

```

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,2,3}, {0,1,2,4}, {0,1,2,5}, {0,1,3,4}, {0,2,3,4}, {0,1,2,3,4}, {
set set) (is ¬ FC-family ?Fc)
proof -
  let ?Fs1 = {{1,2,3,4}, {0,1,2,3,4}, {0,1,2,3,4,5}}::nat set set
  let ?Fs2 = {{}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {
set set
  let ?Fs3 = {{}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {
set set
  let ?Fs4 = {{}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {
set set
  let ?Fs5 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {
set set
  let ?Fs6 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {

```

```

set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [1, 17, 12, 12, 6, 6] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
  show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
by auto
next
show length ?c = length ?Fs
by auto
next
show finite ( $\bigcup ?Fc$ )
by auto
next
show union-closed ?Fc
by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,5\}, \{0,1,3,4\}, \{0,2,3,5\}, \{0,1,2,3,4\}, \{$


```

      by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>))
    next
      show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
      by auto
    next
      show  $\text{length } ?c = \text{length } ?Fs$ 
      by auto
    next
      show  $\text{finite } (\bigcup ?Fc)$ 
      by auto
    next
      show  $\text{union-closed } ?Fc$ 
      by (tactic << union-closed-tac @{context} 1 >>))
  qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,3,4,5\}, \{1,2,3,4,5\}, \{0,1,2,3,4,5\}\})$
 set set (is $\neg \text{FC-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs2 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [62, 38, 38, 35, 35, 8] :: \text{nat list}$

show $?thesis$

proof (rule nonFC[where $Fs = ?Fs$ and $c = ?c$])

show $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$

proof–

have $?Fs1 \in \{\{?Fc\}\}$

by (tactic << union-closed-extension-tac @{context} >>))

moreover

have $?Fs2 \in \{\{?Fc\}\}$

by (tactic << union-closed-extension-tac @{context} >>))

moreover

have $?Fs3 \in \{\{?Fc\}\}$

by (tactic << union-closed-extension-tac @{context} >>))

moreover

have $?Fs4 \in \{\{?Fc\}\}$

by (tactic << union-closed-extension-tac @{context} >>))

```

moreover
have ?Fs5 ∈ { ?Fc }
  by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
moreover
have ?Fs6 ∈ { ?Fc }
  by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
ultimately
show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
  by (tactic ⟨⟨ nonFC-is-system-solution-tac @ {context} @ {term [0..<6]::nat
list} 1 ⟩⟩)
next
show ∃ wj ∈ List.set ?c. 0 < wj
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite (⋃ ?Fc)
  by auto
next
show union-closed ?Fc
  by (tactic ⟨⟨ union-closed-tac @ {context} 1 ⟩⟩)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,3,4\}, \{1,2,3,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,2,3,4,5\}\})$ (is $\neg FC\text{-family} ?Fc$)

proof—

let ?Fs1 = $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$

let ?Fs2 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$

let ?Fs3 = $\{\{\}, \{0\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{0,1,2,3,4\}, \{5\}, \{0,1,2,3,5\}\}$

let ?Fs4 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$

let ?Fs5 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$

let ?Fs6 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$

let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]

let ?c = [137, 84, 213, 63, 1, 27] :: nat list

show ?thesis

proof (rule nonFC[where Fs = ?Fs and c = ?c])

```

show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
proof–
  have  $?Fs1 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs2 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs3 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs4 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs5 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs6 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  ultimately
  show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let  $Fs = ?Fs$  in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
  show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
  show finite  $(\bigcup ?Fc)$ 
    by auto
  next
  show union-closed  $?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

```

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{0,2,4,5\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}\}$ 
set set) (is  $\neg \text{FC-family } ?Fc$ )
proof–
  let  $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$ 
set set
  let  $?Fs2 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$ 

```



```

set set
let ?Fs3 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,3,4}}
set set
let ?Fs4 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,1,2,3,4}}
set set
let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,1,2,3,4}}
set set
let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,1,2,3,4}}
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [97755, 76615, 44765, 30765, 12600, 13440] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>))
  moreover
  have ?Fs2  $\in \{?Fc\}$ 
  by (tactic << union-closed-extension-tac @{context} >>))
  moreover
  have ?Fs3  $\in \{?Fc\}$ 
  by (tactic << union-closed-extension-tac @{context} >>))
  moreover
  have ?Fs4  $\in \{?Fc\}$ 
  by (tactic << union-closed-extension-tac @{context} >>))
  moreover
  have ?Fs5  $\in \{?Fc\}$ 
  by (tactic << union-closed-extension-tac @{context} >>))
  moreover
  have ?Fs6  $\in \{?Fc\}$ 
  by (tactic << union-closed-extension-tac @{context} >>))
  ultimately
  show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>))
next
show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
by auto
next
show length ?c = length ?Fs
by auto
next
show finite ( $\bigcup ?Fc$ )

```

```

    by auto
  next
    show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
  qed
qed

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,3}, {0,1,2,3}, {0,2,4,5}, {0,1,2,3,4}, {0,1,2,4,5}, {0,2,3,4,5},
set set) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = ({}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {0,1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4},
set set
  let ?Fs2 = ({}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4},
set set
  let ?Fs3 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4},
set set
  let ?Fs4 = ({}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {0,1,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4},
set set
  let ?Fs5 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3},
set set
  let ?Fs6 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3},
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [99652, 47993, 31140, 75844, 14050, 13168] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show ∀ F ∈ set ?Fs. F ∈ {?Fc}
proof-
  have ?Fs1 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs2 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs3 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs4 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs5 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs6 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
  show ?thesis
    by (simp del: union-closed-extensions-def)
qed

```

```

next
  show let  $Fs = ?Fs$  in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
next
  show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
next
  show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
next
  show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed
qed

```

lemma $[\text{simp}] : \neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{2,3,4,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\})$ (is $\neg \text{FC-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}\}$
 set set

let $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{4\}\}$
 set set

let $?Fs4 = \{\{\}, \{2\}, \{0,1,2\}, \{3\}, \{0,1,3\}, \{2,3\}, \{0,1,2,3\}, \{4\}, \{2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$
 set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4]$

let $?c = [13, 13, 8, 50] :: \text{nat list}$

show $?thesis$

proof (rule $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$)

show $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$

proof–

have $?Fs1 \in \{\{?Fc\}\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \{\{?Fc\}\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs3 \in \{\{?Fc\}\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs4 \in \{\{?Fc\}\}$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

ultimately

```

    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
  next
    show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}\} \text{ set set})$ (is $\neg \text{FC-family } ?Fc$)

proof—

let $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}\}$ set set

let $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}\}$ set set

let $?Fs3 = \{\{\}, \{2\}, \{0,1,2\}, \{3\}, \{0,1,3\}, \{2,3\}, \{0,1,2,3\}, \{4\}, \{2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$ set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [42, 44, 163, 12, 1, 11] :: \text{nat list}$

show ?thesis

proof (rule nonFC[where $Fs = ?Fs$ and $c = ?c$])

show $\forall F \in \text{set } ?Fs. F \in \ll ?Fc \gg$

proof—

have $?Fs1 \in \ll ?Fc \gg$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \ll ?Fc \gg$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

```

moreover
have  $?Fs3 \in \{\{?Fc\}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac  $@\{context\}$   $\gg$ )
moreover
have  $?Fs4 \in \{\{?Fc\}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac  $@\{context\}$   $\gg$ )
moreover
have  $?Fs5 \in \{\{?Fc\}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac  $@\{context\}$   $\gg$ )
moreover
have  $?Fs6 \in \{\{?Fc\}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac  $@\{context\}$   $\gg$ )
ultimately
show  $?thesis$ 
  by (simp del: union-closed-extensions-def)
qed
next
show  $let\ Fs = ?Fs\ in\ \forall\ a \in \bigcup\ ?Fc.\ sum\_list\ (map\ (\lambda\ (x,\ y).\ int\ x * y)\ (zip\ ?c\ (map\ (frankl\_fun\ a)\ Fs))) < 0$ 
  by (tactic  $\ll$  nonFC-is-system-solution-tac  $@\{context\}$   $@\{term\ [0..<6]::nat\ list\}$   $1\ \gg$ )
next
show  $\exists\ wj \in List.set\ ?c.\ 0 < wj$ 
  by auto
next
show  $length\ ?c = length\ ?Fs$ 
  by auto
next
show  $finite\ (\bigcup\ ?Fc)$ 
  by auto
next
show union-closed  $?Fc$ 
  by (tactic  $\ll$  union-closed-tac  $@\{context\}\ 1\ \gg$ )
qed
qed

```

lemma [*simp*]: $\neg FC_family\ (\{\{\}, \{0,1,2\}, \{0,3,4\}, \{0,1,2,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{1,2,3,4\}, \{1,2,3,5\}, \{1,2,4,5\}\})$ (**is** $\neg FC_family\ ?Fc$)

proof–

```

let  $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
set set
let  $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
set set
let  $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
let  $?Fs4 = \{\{\}, \{0\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{0,1,2,3,4\}, \{5\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}, \{0,1,2,3,4,5\}\}$ 
set set
let  $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set

```

```

let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
let ?c = [15, 8, 8, 19, 2] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof-
    have ?Fs1  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y)) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next
  show finite ( $\bigcup ?Fc$ )
  by auto
next
  show union-closed ?Fc
  by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0, 1, 2\}, \{3, 4, 5\}, \{0, 1, 2, 3\}, \{0, 1, 2, 3, 4\}, \{0, 1, 2, 4, 5\}, \{0, 1, 3, 4, 5\}, \{0, 2, 3, 4\}, \{0, 2, 3, 5\}, \{0, 2, 4, 5\}, \{0, 3, 4, 5\}\})$ 
set set (is  $\neg \text{FC-family } ?Fc$ )
proof-
  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0, 1\}, \{2\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\}, \{3\}, \{0, 3\}, \{1, 3\}, \{0, 1, 3\}, \{2, 3\}, \{0, 2, 3\}, \{1, 2, 3\}, \{0, 1, 2, 3\}\}$ 
  set set

```

```

    let ?Fs2 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4}}
  set set
    let ?Fs3 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,4}}
  set set
    let ?Fs4 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4}}
  set set
    let ?Fs5 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4}}
  set set
    let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
  set set
    let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
    let ?c = [12, 12, 12, 12, 12, 11] :: nat list
    show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next

```

```

    show finite (⋃ ?Fc)
    by auto
next
    show union-closed ?Fc
    by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,3}, {0,1,2,3}, {0,1,2,4}, {0,3,4,5}, {0,1,2,3,4}, {0,1,3,4,5}, {0,1,2,3,4,5},
set set) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = ({}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {0,1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,3,4},
set set
  let ?Fs2 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3},
set set
  let ?Fs3 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3},
set set
  let ?Fs4 = ({}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,3,4},
set set
  let ?Fs5 = ({}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {0,1,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,3,4},
set set
  let ?Fs6 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,1,2,3,4},
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [39709, 5528, 4837, 12536, 31167, 18165] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show ∀ F ∈ set ?Fs. F ∈ {?Fc}
proof-
  have ?Fs1 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
  have ?Fs2 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
  have ?Fs3 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
  have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
  have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
  have ?Fs6 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
ultimately
  show ?thesis
  by (simp del: union-closed-extensions-def)

```



```

    qed
  next
    show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
  next
    show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,3,4\}, \{0,1,2,3\}, \{0,1,3,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,3,4,5\}\} \text{ set set})$ (is $\neg \text{FC-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ set set

let $?Fs2 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$ set set

let $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$ set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}\}$ set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [69366, 37435, 36332, 25335, 6449, 22026] :: \text{nat list}$

show $?thesis$

proof (rule $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$)

show $\forall F \in \text{set } ?Fs. F \in \ll ?Fc \gg$

proof–

have $?Fs1 \in \ll ?Fc \gg$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \ll ?Fc \gg$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs3 \in \ll ?Fc \gg$

```

      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,3,4\}, \{0,1,2,3\}, \{1,2,4,5\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,2,3,4,5\}\}, \{0,1,2,3,4,5\})$ (is $\neg FC\text{-family } ?Fc$)

proof—

```

  let ?Fs1 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
  let ?Fs2 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,4}}
set set
  let ?Fs3 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4},{0,1,2,4}}
set set
  let ?Fs4 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4},{0,1,2,3,4}}
set set
  let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3},{0,1,2,3,4}}
set set
  let ?Fs6 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4},{0,1,2,3,4}}
set set

```

```

let ?Fs7 = {{},{0},{1},{0,1},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{1,2,3},{0,1,2,3},{4},{0,4},{1,4}}
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6, ?Fs7]
let ?c = [13, 25, 11, 10, 9, 47, 37] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof -
    have ?Fs1  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs7  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite ( $\bigcup ?Fc$ )
    by auto
  next
  show union-closed ?Fc
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )

```

qed
qed

lemma [simp]: $\neg FC\text{-family } (\{\{\}, \{0,1,2\}, \{0,3,4\}, \{0,1,2,5\}, \{1,3,4,5\}, \{0,1,2,3,4\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\})$
 $set\ set$ (is $\neg FC\text{-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,3,4\}\}$
 $set\ set$

let $?Fs2 = \{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,3,4\}\}$
 $set\ set$

let $?Fs3 = \{\{3,4,5\}, \{0,3,4,5\}, \{1,3,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\} :: nat\ set$
 set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,3,4\}, \{0,1,3,4\}, \{0,2,3,4\}, \{0,1,2,3,4\}, \{5\}, \{0,1,2,3,4,5\}\}$
 $set\ set$

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}\}$
 $set\ set$

let $?Fs6 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
 $set\ set$

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]$

let $?c = [34, 23, 1, 50, 16, 7] :: nat\ list$

show $?thesis$

proof (rule nonFC[where $Fs = ?Fs$ and $c = ?c$])

show $\forall F \in set\ ?Fs. F \in \{?Fc\}$

proof–

have $?Fs1 \in \{?Fc\}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs2 \in \{?Fc\}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs3 \in \{?Fc\}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs4 \in \{?Fc\}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs5 \in \{?Fc\}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

moreover

have $?Fs6 \in \{?Fc\}$

by (tactic $\ll union\text{-closed-extension-tac } @\{context\} \gg$)

ultimately

show $?thesis$

by (simp del: union-closed-extensions-def)

qed

next

show let $Fs = ?Fs$ in $\forall a \in \bigcup ?Fc. sum\text{-list } (map (\lambda (x, y). int\ x * y) (zip\ ?c (map (frankl\text{-fun } a) Fs))) < 0$

by (tactic $\ll nonFC\text{-is-system-solution-tac } @\{context\} @\{term\ } [0..<6] :: nat$

```

list} 1 >>))
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show  $\text{length } ?c = \text{length } ?Fs$ 
  by auto
next
  show  $\text{finite } (\bigcup ?Fc)$ 
  by auto
next
  show  $\text{union-closed } ?Fc$ 
  by (tactic << union-closed-tac @{context} 1 >>))
qed
qed

```

```

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,2,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \dots\}$ 
set set) (is  $\neg \text{FC-family } ?Fc$ )
proof-
  let ?Fs1 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \dots\}$ 
set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \dots\}$ 
set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \dots\}$ 
set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \dots\}$ 
set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \dots\}$ 
set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \dots\}$ 
set set
  let ?Fs = [ $?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6$ ]
  let ?c = [73661, 57777, 35199, 8792, 22307, 9420] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
proof-
  have  $?Fs1 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>))
  moreover
  have  $?Fs2 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>))
  moreover
  have  $?Fs3 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>))
  moreover
  have  $?Fs4 \in \{\{?Fc\}\}$ 
  by (tactic << union-closed-extension-tac @{context} >>))
  moreover

```

```

    have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs6 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

```

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,3}, {0,1,2,3}, {0,1,2,4}, {2,3,4,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5})
set set (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = ({}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5})
set set
  let ?Fs2 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5})
set set
  let ?Fs3 = ({}, {2}, {0,1,2}, {3}, {0,1,3}, {2,3}, {0,1,2,3}, {4}, {2,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {2,3,4}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5})
set set
  let ?Fs4 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5})
set set
  let ?Fs5 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5})
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
  let ?c = [39, 37, 144, 10, 11] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show ∀ F ∈ set ?Fs. F ∈ {?Fc}
    proof-
      have ?Fs1 ∈ {?Fc}

```

```

      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{0,1,2,4\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{2,3,4,5,6\}\})$ (is $\neg FC\text{-family } ?Fc$)

proof—

```

  let ?Fs1 = {{}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {2,3,4}, {0,2,3,4}, {1,2,3,4}, {0,1,2,3,4}, {0,1,2,4,5}, {0,1,3,4,5}, {2,3,4,5,6}}
  set set
  let ?Fs2 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {1,2,4}, {0,1,2,4}, {0,1,3,4}, {0,1,2,3,4}, {0,1,2,4,5}, {0,1,3,4,5}, {2,3,4,5,6}}
  set set
  let ?Fs3 = {{}, {2}, {0,1,2}, {3}, {0,1,3}, {2,3}, {0,1,2,3}, {4}, {2,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {2,3,4}, {0,1,2,3,4}, {0,1,2,4,5}, {0,1,3,4,5}, {2,3,4,5,6}}
  set set
  let ?Fs4 = {{}, {1}, {0,1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {0,1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {0,1,2,4}, {2,3,4}, {0,1,2,3,4}, {0,1,2,4,5}, {0,1,3,4,5}, {2,3,4,5,6}}
  set set
  let ?Fs5 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,4}, {0,1,3,4}, {0,1,2,3,4}, {0,1,2,4,5}, {0,1,3,4,5}, {2,3,4,5,6}}

```

```

set set
let ?Fs6 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{0,1,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4},{0,1,2,3,4}}
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [42, 39, 147, 14, 12, 8] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof-
    have ?Fs1  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
by auto
next
show length ?c = length ?Fs
by auto
next
show finite ( $\bigcup ?Fc$ )
by auto
next
show union-closed ?Fc
by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```



```

lemma [simp]:  $\neg FC\text{-family } (\{\{\}, \{0,1,2\}, \{0,3,4\}, \{0,1,2,3\}, \{0,1,2,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{1,2,3,4,5\}\})$ 
set set (is  $\neg FC\text{-family } ?Fc$ )
proof –
  let  $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
set set
  let  $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$ 
set set
  let  $?Fs3 = \{\{\}, \{0\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{0,1,2,3,4\}, \{5\}, \{0,1,2,3,5\}\}$ 
set set
  let  $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
  let  $?Fs5 = \{\{\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{0,1,2,3,4\}\}$ 
set set
  let  $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]$ 
  let  $?c = [5950, 3438, 7803, 2511, 984] :: \text{nat list}$ 
  show  $?thesis$ 
proof (rule  $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$ )
    show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
    proof –
      have  $?Fs1 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs2 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs3 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs4 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs5 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      ultimately
      show  $?thesis$ 
      by (simp del:  $\text{union-closed-extensions-def}$ )
    qed
  next
    show  $\text{let } Fs = ?Fs \text{ in } \forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto

```

```

next
  show finite ( $\bigcup$  ?Fc)
  by auto
next
  show union-closed ?Fc
  by (tactic  $\ll$  union-closed-tac @{context} 1  $\gg$ )
qed
qed

lemma [simp]:  $\neg$  FC-family ( $\{\{\}, \{0,1,2\}, \{0,3,4\}, \{0,1,2,3\}, \{1,2,3,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{1,2,3,4,5\}\}$ 
set set) (is  $\neg$  FC-family ?Fc)
proof-
  let ?Fs1 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$ 
set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$ 
set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}\}$ 
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [56, 32, 23, 24, 146, 146] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\text{?Fc}\}$ 
proof-
  have ?Fs1  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs2  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs3  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs4  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs5  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs6  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  ultimately
  show ?thesis

```

```

    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
      by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
      by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
      by auto
  next
    show  $\text{union-closed } ?Fc$ 
      by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,3,4\}, \{0,1,2,3\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{1,2,3,4,5\}\} \text{ set set})$ (is $\neg \text{FC-family } ?Fc$)

proof–

let $?Fs1 = \{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ set set

let $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}\}$ set set

let $?Fs3 = \{\{\}, \{0\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{0,1,2,3,4\}, \{5\}, \{0,1,2,3,5\}\}$ set set

let $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ set set

let $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]$

let $?c = [50, 30, 61, 26, 5] :: \text{nat list}$

show $?thesis$

proof (rule $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$)

show $\forall F \in \text{set } ?Fs. F \in \llbracket ?Fc \rrbracket$

proof–

have $?Fs1 \in \llbracket ?Fc \rrbracket$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs2 \in \llbracket ?Fc \rrbracket$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have $?Fs3 \in \llbracket ?Fc \rrbracket$

by (tactic $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

```

moreover
have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
ultimately
show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
  by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
next
show ∃ wj ∈ List.set ?c. 0 < wj
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite (⋃ ?Fc)
  by auto
next
show union-closed ?Fc
  by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,3,4\}, \{0,1,2,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{1,2,3,4,5\}\}$
 $set\ set)$ (**is** $\neg FC\text{-family } ?Fc$)

proof—

let ?Fs1 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
 $set\ set$

let ?Fs2 = $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{1,2,3,4\}\}$
 $set\ set$

let ?Fs3 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$
 $set\ set$

let ?Fs4 = $\{\{\}, \{0\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{0,1,2,3,4\}, \{5\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\}$
 $set\ set$

let ?Fs5 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,1,2,3,4\}, \{5\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\}$
 $set\ set$

let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]

let ?c = [3, 20, 12, 24, 11] :: nat list

show ?thesis

proof (rule nonFC[**where** Fs = ?Fs **and** c = ?c])

show ∀ F ∈ set ?Fs. F ∈ {?Fc}

proof—

```

    have ?Fs1 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs2 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs3 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs4 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{3,4,5\}, \{0,1,2,3\}, \{0,1,4,5\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}\}$
 $set\ set)$ (**is** $\neg FC\text{-family } ?Fc$)

proof—

```

  let ?Fs1 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{
set set
  let ?Fs2 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{
set set
  let ?Fs3 = {\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{
set set
  let ?Fs4 = {\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{
set set

```

```

let ?Fs5 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4}}
set set
let ?Fs6 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4}}
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [30, 21, 36, 28, 33, 24] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof–
    have ?Fs1  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite ( $\bigcup ?Fc$ )
    by auto
  next
  show union-closed ?Fc
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed

```

qed

```

lemma [simp]: ¬ FC-family ({{},{0,1,2},{0,3,4},{0,1,2,3},{0,1,3,4},{0,1,2,3,4},{0,1,2,3,5},{0,1,2,4,5}}
set set) (is ¬ FC-family ?Fc)
proof –
  let ?Fs1 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,4}}
set set
  let ?Fs2 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4}}
set set
  let ?Fs3 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
  let ?Fs4 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4}}
set set
  let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4},{0,1,2,4}}
set set
  let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [178250, 81120, 80616, 86520, 86394, 5796] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show ∀ F ∈ set ?Fs. F ∈ {?Fc}
    proof –
      have ?Fs1 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs2 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs3 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs4 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs6 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      ultimately
      show ?thesis
      by (simp del: union-closed-extensions-def)
    qed
  next
    show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @ {context} @ {term [0..<6]::nat
list} 1 ⟩⟩)

```



```

    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
  have ?Fs6 ∈ {?Fc}
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
  show ?thesis
    by (simp del: union-closed-extensions-def)
qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```

```

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {3,4,5}, {0,1,2,3}, {0,1,3,4}, {0,1,3,5}, {0,1,2,3,4}, {0,1,2,3,5}, {
set set) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {
set set
  let ?Fs2 = ({}, {2}, {0,1,2}, {3}, {2,3}, {0,1,2,3}, {4}, {2,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {2,3,4}, {0,1,2,3,4}, {
set set
  let ?Fs3 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {
set set
  let ?Fs4 = ({}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {
set set
  let ?Fs5 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
  let ?c = [22, 52, 13, 12, 13] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show ∀ F ∈ set ?Fs. F ∈ {?Fc}
    proof-
      have ?Fs1 ∈ {?Fc}
        by (tactic << union-closed-extension-tac @{context} >>)

```

```

moreover
have ?Fs2 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs3 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
ultimately
show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
  by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
next
show ∃ wj ∈ List.set ?c. 0 < wj
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite (⋃ ?Fc)
  by auto
next
show union-closed ?Fc
  by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

```

```

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {3,4,5}, {0,1,2,3}, {0,1,2,4}, {0,1,3,4}, {0,1,2,3,4}, {0,1,3,4,5}, {
set set) (is ¬ FC-family ?Fc)
proof–
  let ?Fs1 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {
set set
  let ?Fs2 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {
set set
  let ?Fs3 = ({}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,
set set
  let ?Fs4 = ({}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,
set set
  let ?Fs5 = ({}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,
set set

```

```

let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [16266, 16266, 21609, 19355, 14455, 9636] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof –
    have ?Fs1  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite ( $\bigcup ?Fc$ )
    by auto
  next
  show union-closed ?Fc
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

```

lemma [simp]:  $\neg FC\text{-family } (\{\{\}, \{0,1,2\}, \{3,4,5\}, \{0,1,2,3\}, \{0,1,3,4\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \dots\})$ 
set set (is  $\neg FC\text{-family } ?Fc$ )
proof –
  let  $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \dots\}$ 
set set
  let  $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \dots\}$ 
set set
  let  $?Fs3 = \{\{\}, \{2\}, \{0,1,2\}, \{3\}, \{2,3\}, \{0,1,2,3\}, \{4\}, \{2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}, \dots\}$ 
set set
  let  $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \dots\}$ 
set set
  let  $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \dots\}$ 
set set
  let  $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]$ 
  let  $?c = [8, 12, 22, 10, 7] :: \text{nat list}$ 
  show  $?thesis$ 
proof (rule  $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$ )
    show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
    proof –
      have  $?Fs1 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs2 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs3 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs4 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      moreover
      have  $?Fs5 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
      ultimately
      show  $?thesis$ 
      by (simp  $\text{del: union-closed-extensions-def}$ )
    qed
  next
    show  $\text{let } Fs = ?Fs \text{ in } \forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next

```

```

    show finite (⋃ ?Fc)
    by auto
  next
    show union-closed ?Fc
    by (tactic ⟨⟨ union-closed-tac @ {context} 1 ⟩⟩)
  qed
qed

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {3,4,5}, {0,1,2,3}, {0,1,3,4}, {2,3,4,5}, {0,1,2,3,4}, {0,1,3,4,5}, {2,3,4,5,6}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {1,2,4}, {0,1,2,4}, {3,4}, {0,3,4}, {1,3,4}, {2,3,4}, {0,1,3,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
  set set
  let ?Fs2 = {{}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {3,4}, {0,3,4}, {1,3,4}, {2,3,4}, {0,1,3,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
  set set
  let ?Fs3 = {{}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {3,4}, {0,3,4}, {1,3,4}, {2,3,4}, {0,1,3,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
  set set
  let ?Fs4 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {3,4}, {0,3,4}, {1,3,4}, {2,3,4}, {0,1,3,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
  set set
  let ?Fs5 = {{}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {3,4}, {0,3,4}, {2,3,4}, {0,1,3,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
  set set
  let ?Fs6 = {{}, {0}, {0,1}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {0,1,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {0,1,4}, {0,1,2,4}, {3,4}, {0,3,4}, {2,3,4}, {0,1,3,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
  set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [197232, 224123, 125619, 172578, 112046, 156791] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show ∀ F ∈ set ?Fs. F ∈ {?Fc}
    proof-
      have ?Fs1 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs2 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs3 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs4 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs6 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      ultimately
      show ?thesis
      by (simp del: union-closed-extensions-def)
    end
  end
end

```

```

    qed
  next
    show let  $Fs = ?Fs$  in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
  next
    show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

```

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{3,4,5\}, \{0,1,2,3\}, \{0,1,4,5\}, \{2,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}\})$ 
set set (is  $\neg \text{FC-family } ?Fc$ )
proof-
  let  $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
  set set
  let  $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}\}$ 
  set set
  let  $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let  $?Fs4 = \{\{\}, \{2\}, \{0,1,2\}, \{3\}, \{2,3\}, \{0,1,2,3\}, \{4\}, \{2,4\}, \{0,1,2,4\}, \{3,4\}, \{2,3,4\}, \{0,1,2,3,4\}, \{5\}, \{2,5\}, \{0,1,2,5\}, \{0,1,3,5\}, \{0,1,4,5\}, \{0,1,2,5\}, \{0,1,3,5\}, \{0,1,4,5\}, \{0,1,2,3,5\}, \{0,1,3,4,5\}, \{0,1,4,5\}, \{0,1,2,3,4,5\}\}$ 
  set set
  let  $?Fs5 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
  set set
  let  $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]$ 
  let  $?c = [28, 20, 21, 58, 26] :: \text{nat list}$ 
  show ?thesis
  proof (rule nonFC[where  $Fs = ?Fs$  and  $c = ?c$ ])
    show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
    proof-
      have  $?Fs1 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
      have  $?Fs2 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
      have  $?Fs3 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
      have  $?Fs4 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
      have  $?Fs5 \in \{\{?Fc\}\}$ 
      by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover

```

```

    have ?Fs4 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    moreover
    have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
      by auto
  next
    show length ?c = length ?Fs
      by auto
  next
    show finite (⋃ ?Fc)
      by auto
  next
    show union-closed ?Fc
      by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

```

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,3,4}, {0,1,2,3}, {0,1,2,4}, {0,1,2,5}, {0,1,2,3,4}, {0,1,2,3,5}, {
set set) (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = ({}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {
set set
  let ?Fs2 = ({}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {
set set
  let ?Fs3 = ({}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {
set set
  let ?Fs4 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {
set set
  let ?Fs5 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {
set set
  let ?Fs6 = ({}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [337, 167, 167, 149, 149, 13] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show ∀ F ∈ set ?Fs. F ∈ {?Fc}
  end

```

```

proof–
  have ?Fs1 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs2 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs3 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs4 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs5 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  moreover
  have ?Fs6 ∈ {?Fc}
    by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
  ultimately
  show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
  next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
  show length ?c = length ?Fs
    by auto
  next
  show finite (⋃ ?Fc)
    by auto
  next
  show union-closed ?Fc
    by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
  qed
qed

```

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {3,4,5}, {0,1,2,3}, {0,1,3,4}, {0,3,4,5}, {0,1,2,3,4}, {0,1,2,3,5}, {set set}) (is ¬ FC-family ?Fc)

proof–

```

  let ?Fs1 = {{}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,4},
set set
  let ?Fs2 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4},
set set

```



```

    let ?Fs3 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,3,4}}
  set set
    let ?Fs4 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,3,4}}
  set set
    let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3,4}}
  set set
    let ?Fs6 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3,4}}
  set set
    let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
    let ?c = [77671, 77671, 46046, 65159, 65159, 46046] :: nat list
    show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>))
  moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>))
  moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>))
  moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>))
  moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>))
  moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>))
  ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>))
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next
  show finite ( $\bigcup ?Fc$ )
  by auto

```

```

next
  show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,3,4}, {0,1,2,3}, {0,1,2,5}, {0,3,4,5}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6})
set set (is ¬ FC-family ?Fc)
proof-
  let ?Fs1 = {{}, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
set set
  let ?Fs2 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
set set
  let ?Fs3 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
set set
  let ?Fs4 = {{}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
set set
  let ?Fs5 = {{}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
set set
  let ?Fs6 = {{}, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [41997, 19668, 18337, 20271, 17963, 4377] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show ∀ F ∈ set ?Fs. F ∈ {?Fc}
    proof-
      have ?Fs1 ∈ {?Fc}
        by (tactic << union-closed-extension-tac @{context} >>)
      moreover
      have ?Fs2 ∈ {?Fc}
        by (tactic << union-closed-extension-tac @{context} >>)
      moreover
      have ?Fs3 ∈ {?Fc}
        by (tactic << union-closed-extension-tac @{context} >>)
      moreover
      have ?Fs4 ∈ {?Fc}
        by (tactic << union-closed-extension-tac @{context} >>)
      moreover
      have ?Fs5 ∈ {?Fc}
        by (tactic << union-closed-extension-tac @{context} >>)
      moreover
      have ?Fs6 ∈ {?Fc}
        by (tactic << union-closed-extension-tac @{context} >>)
      ultimately
      show ?thesis
        by (simp del: union-closed-extensions-def)
    qed
  qed
next

```

```

    show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
  next
    show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{3,4,5\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \dots\}$
 $\text{set set})$ (is $\neg \text{FC-family } ?Fc$)

proof–

```

  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \dots\}$ 
  set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \dots\}$ 
  set set
  let ?Fs3 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \dots\}$ 
  set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \dots\}$ 
  set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \dots\}$ 
  set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \dots\}$ 
  set set

```

```

  let ?Fs = [ $?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6$ ]

```

```

  let ?c = [14, 15, 16, 16, 13, 15] :: nat list

```

show $?thesis$

proof (rule $\text{nonFC}[\text{where } Fs = ?Fs \text{ and } c = ?c]$)

```

  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 

```

proof–

```

  have  $?Fs1 \in \{\{?Fc\}\}$ 

```

```

    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )

```

moreover

```

  have  $?Fs2 \in \{\{?Fc\}\}$ 

```

```

    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )

```

moreover

```

  have  $?Fs3 \in \{\{?Fc\}\}$ 

```

```

    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )

```

moreover


```

show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof-
    have ?Fs1  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y)) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next
  show finite ( $\bigcup ?Fc$ )
  by auto
next
  show union-closed ?Fc
  by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed
qed

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,3,4\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\})$ 
set set) (is  $\neg \text{FC-family } ?Fc$ )
proof-
  let ?Fs1 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,3,4\}\}$ 

```

```

set set
let ?Fs2 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4}}
set set
let ?Fs3 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
let ?Fs4 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,3,4}}
set set
let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
let ?Fs6 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,3,4}}
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [20851, 9303, 9303, 10251, 789, 10251] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    moreover
    have ?Fs6  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
show  $\exists w_j \in \text{List.set } ?c. 0 < w_j$ 
by auto
next
show length ?c = length ?Fs
by auto

```

```

next
  show finite ( $\bigcup$  ?Fc)
  by auto
next
  show union-closed ?Fc
  by (tactic  $\ll$  union-closed-tac @{context} 1  $\gg$ )
qed
qed

lemma [simp]:  $\neg$  FC-family ( $\{\{\}, \{0,1,2\}, \{3,4,5\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,5\}, \{0,1,3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
set set) (is  $\neg$  FC-family ?Fc)
proof-
  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}\}$ 
set set
  let ?Fs4 =  $\{\{\}, \{2\}, \{0,1,2\}, \{3\}, \{2,3\}, \{0,1,2,3\}, \{4\}, \{2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}\}$ 
set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}\}$ 
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
  let ?c = [3, 5, 5, 15, 5] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\text{?Fc}\}$ 
proof-
  have ?Fs1  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs2  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs3  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs4  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  moreover
  have ?Fs5  $\in \{\text{?Fc}\}$ 
  by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  ultimately
  show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 

```

```

      by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 >>))
    next
      show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
      by auto
    next
      show  $\text{length } ?c = \text{length } ?Fs$ 
      by auto
    next
      show  $\text{finite } (\bigcup ?Fc)$ 
      by auto
    next
      show  $\text{union-closed } ?Fc$ 
      by (tactic << union-closed-tac @{context} 1 >>))
  qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{3,4,5\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,3,4\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\})$
 $\text{set set} \text{ (is } \neg \text{FC-family } ?Fc)$

```

proof–
  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{1,3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{1,3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs3 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{1,3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,2,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{2,3,4\}, \{0,2,3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{1,3,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{1,3,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs = [ $?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6$ ]
  let ?c = [212, 212, 264, 231, 31, 205] :: nat list
  show ?thesis
proof (rule nonFC[where  $Fs = ?Fs$  and  $c = ?c$ ])
    show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
    proof–
      have  $?Fs1 \in \{\{?Fc\}\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
    moreover
      have  $?Fs2 \in \{\{?Fc\}\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
    moreover
      have  $?Fs3 \in \{\{?Fc\}\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))
    moreover
      have  $?Fs4 \in \{\{?Fc\}\}$ 
      by (tactic << union-closed-extension-tac @{context} >>))

```



```

moreover
have ?Fs5 ∈ { ?Fc }
  by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
moreover
have ?Fs6 ∈ { ?Fc }
  by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
ultimately
show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
  by (tactic ⟨⟨ nonFC-is-system-solution-tac @ {context} @ {term [0..<6]::nat
list} 1 ⟩⟩)
next
show ∃ wj ∈ List.set ?c. 0 < wj
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite (⋃ ?Fc)
  by auto
next
show union-closed ?Fc
  by (tactic ⟨⟨ union-closed-tac @ {context} 1 ⟩⟩)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,5\}, \{0,1,3,4\}, \{0,1,3,5\}, \{0,1,4,5\}, \{0,2,3,4\}, \{0,2,3,5\}, \{0,2,4,5\}, \{0,3,4,5\}\})$ (is $\neg FC\text{-family} ?Fc$)

proof—

let ?Fs1 = $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}\}$

let ?Fs2 = $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}\}$

let ?Fs3 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}\}$

let ?Fs4 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,2,4\}\}$

let ?Fs5 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$

let ?Fs6 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$

let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]

let ?c = [8420, 7380, 3448, 3448, 662, 662] :: nat list

show ?thesis

proof (rule nonFC[**where** Fs = ?Fs and c = ?c])

```

show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
proof–
  have  $?Fs1 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs2 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs3 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs4 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs5 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  moreover
  have  $?Fs6 \in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
  ultimately
  show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let  $Fs = ?Fs$  in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
  show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
  show finite  $(\bigcup ?Fc)$ 
    by auto
  next
  show union-closed  $?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{3,4,5\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,3,4,5\}\})$ 
  proof–
  let  $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
  set set
  let  $?Fs2 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 

```

```

set set
  let ?Fs3 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,3,4}}
set set
  let ?Fs4 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
set set
  let ?Fs5 = {{},{0},{0,1,2},{3},{0,3},{0,1,2,3},{4},{0,4},{0,1,2,4},{3,4},{0,3,4},{0,1,2,3,4},{5},{0,1,2,3,4,5}}
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
  let ?c = [3, 3, 4, 3, 8] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{\?Fc\}$ 
    proof-
      have ?Fs1  $\in \{\?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
      have ?Fs2  $\in \{\?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
      have ?Fs3  $\in \{\?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
      have ?Fs4  $\in \{\?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>)
    moreover
      have ?Fs5  $\in \{\?Fc\}$ 
      by (tactic << union-closed-extension-tac @{context} >>)
    ultimately
      show ?thesis
      by (simp del: union-closed-extensions-def)
    qed
  next
    show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show length ?c = length ?Fs
    by auto
  next
    show finite ( $\bigcup ?Fc$ )
    by auto
  next
    show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
  qed

```

qed

```

lemma [simp]: ¬ FC-family ({{},{0,1,2},{3,4,5},{0,1,2,3},{0,1,2,4},{0,3,4,5},{1,3,4,5},{0,1,2,3,4},{0,1,2,3,5},{0,1,2,4,5},{0,1,3,4,5},{0,1,2,3,4,5}}) (is ¬ FC-family ?Fc)
proof -
  let ?Fs1 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{3},{0,3},{1,3},{0,1,3},{2,3},{0,2,3},{1,2,3},{0,1,2,3}}
  set set
  let ?Fs2 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4},{0,2,4},{0,1,2,4}}
  set set
  let ?Fs3 = {{},{1},{2},{1,2},{0,1,2},{3},{1,3},{2,3},{1,2,3},{0,1,2,3},{4},{1,4},{2,4},{1,2,4},{0,1,2,4},{0,1,2,3,4}}
  set set
  let ?Fs4 = {{},{0},{2},{0,2},{0,1,2},{3},{0,3},{2,3},{0,2,3},{0,1,2,3},{4},{0,4},{2,4},{0,2,4},{0,1,2,4},{0,1,2,3,4}}
  set set
  let ?Fs5 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,4},{0,1,2,3,4}}
  set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
  let ?c = [44, 49, 50, 50, 70] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show ∀ F ∈ set ?Fs. F ∈ {?Fc}
    proof -
      have ?Fs1 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs2 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs3 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs4 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      moreover
      have ?Fs5 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @ {context} ⟩⟩)
      ultimately
      show ?thesis
      by (simp del: union-closed-extensions-def)
    qed
  next
    show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip ?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @ {context} @ {term [0..<6]::nat list} 1 ⟩⟩)
  next
    show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
  next
    show length ?c = length ?Fs

```

```

    by auto
  next
    show finite ( $\bigcup$  ?Fc)
    by auto
  next
    show union-closed ?Fc
    by (tactic  $\ll$  union-closed-tac @{context} 1  $\gg$ )
qed
qed

lemma [simp]:  $\neg$  FC-family ( $\{\{\}, \{0,1,2\}, \{3,4,5\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,5\}, \{0,3,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,3,4,5\}\}$ 
set set) (is  $\neg$  FC-family ?Fc)
proof-
  let ?Fs1 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
set set
  let ?Fs3 =  $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs4 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,2,4\}, \{1,2,4\}, \{0,1,2,4\}\}$ 
set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,4\}, \{0,1,2,4\}, \{0,1,2,3,4\}\}$ 
set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
  let ?c = [18, 18, 27, 18, 22, 22] :: nat list
  show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\text{?Fc}\}$ 
proof-
    have ?Fs1  $\in \{\text{?Fc}\}$ 
    by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
    moreover
    have ?Fs2  $\in \{\text{?Fc}\}$ 
    by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
    moreover
    have ?Fs3  $\in \{\text{?Fc}\}$ 
    by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
    moreover
    have ?Fs4  $\in \{\text{?Fc}\}$ 
    by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
    moreover
    have ?Fs5  $\in \{\text{?Fc}\}$ 
    by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
    moreover
    have ?Fs6  $\in \{\text{?Fc}\}$ 
    by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
  ultimately

```

```

    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show  $\text{length } ?c = \text{length } ?Fs$ 
    by auto
  next
    show  $\text{finite } (\bigcup ?Fc)$ 
    by auto
  next
    show  $\text{union-closed } ?Fc$ 
    by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
  qed
qed

```

lemma [simp]: $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,5\}, \{0,1,3,4\}, \{0,1,3,5\}, \{0,1,4,5\}, \{0,2,3,4\}, \{0,2,3,5\}, \{0,2,4,5\}, \{0,3,4,5\}\})$ (is $\neg \text{FC-family } ?Fc$)

proof—

```

  let ?Fs1 =  $\{\{\}, \{2\}, \{0,1,2\}, \{3\}, \{0,1,3\}, \{2,3\}, \{0,1,2,3\}, \{4\}, \{2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs2 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
  set set
  let ?Fs3 =  $\{\{\}, \{0\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{0,1,2,4\}, \{3,4\}, \{0,3,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs4 =  $\{\{\}, \{1\}, \{0,1\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs5 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}\}$ 
  set set
  let ?Fs6 =  $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,1,2,3,4\}\}$ 
  set set

```

```

  let ?Fs = [ $?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6$ ]

```

```

  let ?c = [118, 6, 1, 6, 15, 16] :: nat list

```

show ?thesis

proof (rule nonFC[**where** $Fs = ?Fs$ and $c = ?c$])

```

  show  $\forall F \in \text{set } ?Fs. F \in \llbracket ?Fc \rrbracket$ 

```

proof—

```

  have  $?Fs1 \in \llbracket ?Fc \rrbracket$ 

```

```

    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )

```

moreover

```

  have  $?Fs2 \in \llbracket ?Fc \rrbracket$ 

```

```

    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )

```

```

moreover
have ?Fs3 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs6 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
ultimately
show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
  by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
next
show ∃ wj ∈ List.set ?c. 0 < wj
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite (⋃ ?Fc)
  by auto
next
show union-closed ?Fc
  by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,3,4\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,5\}, \{0,1,3,4\}, \{0,3,4,5\}, \{0,1,4,5\}, \{0,2,3,4\}, \{0,2,3,5\}, \{0,2,4,5\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,2,3,4,5\}\})$

proof—

```

let ?Fs1 = { { }, {1}, {2}, {1,2}, {0,1,2}, {3}, {1,3}, {2,3}, {1,2,3}, {0,1,2,3}, {4}, {1,4}, {2,4}, {1,2,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,4,5}, {0,1,3,4,5}, {0,2,3,4,5} }
let ?Fs2 = { {2,3,4}, {0,2,3,4}, {0,1,2,3,4}, {0,2,3,4,5}, {0,1,2,3,4,5} } :: nat set
let ?Fs3 = { { }, {0}, {2}, {0,2}, {0,1,2}, {3}, {0,3}, {2,3}, {0,2,3}, {0,1,2,3}, {4}, {0,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,4,5}, {0,1,3,4,5}, {0,2,3,4,5} }
let ?Fs4 = { { }, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4}, {0,2,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,4,5}, {0,1,3,4,5}, {0,2,3,4,5} }
let ?Fs5 = { { }, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}, {0,1,2,3,5}, {0,1,2,4,5}, {0,1,3,4,5}, {0,2,3,4,5} }

```

```

let ?Fs6 = {{},{0},{1},{0,1},{0,1,2},{3},{0,3},{1,3},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{0,1,2,3,4}}
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]
let ?c = [65, 1, 34, 29, 29, 30] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{\{?Fc\}\}$ 
  proof -
    have ?Fs1  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs2  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs3  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs4  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs5  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    moreover
    have ?Fs6  $\in \{\{?Fc\}\}$ 
    by (tactic  $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$ )
    ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
  by (tactic  $\ll \text{nonFC-is-system-solution-tac } @\{\text{context}\} @\{\text{term } [0..<6]::\text{nat list}\} 1 \gg$ )
next
  show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
  by auto
next
  show length ?c = length ?Fs
  by auto
next
  show finite ( $\bigcup ?Fc$ )
  by auto
next
  show union-closed ?Fc
  by (tactic  $\ll \text{union-closed-tac } @\{\text{context}\} 1 \gg$ )
qed
qed

```



```

lemma [simp]:  $\neg FC\text{-family } (\{\{\}, \{0,1,2\}, \{0,1,3\}, \{2,4,5\}, \{0,1,2,3\}, \{0,1,2,4,5\}, \{0,1,3,4,5\}, \{0,1,2,3,4,5\}\}$ 
 $set\ set)$  (is  $\neg FC\text{-family } ?Fc$ )
proof –
  let  $?Fs1 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}\}$ 
 $set\ set$ 
  let  $?Fs2 = \{\{\}, \{2\}, \{0,1,2\}, \{3\}, \{0,1,3\}, \{2,3\}, \{0,1,2,3\}, \{4\}, \{2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$ 
 $set\ set$ 
  let  $?Fs3 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$ 
 $set\ set$ 
  let  $?Fs4 = \{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,1,2,3,4\}\}$ 
 $set\ set$ 
  let  $?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4]$ 
  let  $?c = [8, 22, 7, 4] :: nat\ list$ 
  show  $?thesis$ 
  proof (rule nonFC[where  $Fs = ?Fs$  and  $c = ?c$ ])
    show  $\forall F \in set\ ?Fs. F \in \{?Fc\}$ 
    proof –
      have  $?Fs1 \in \{?Fc\}$ 
      by (tactic  $\ll union\text{-closed-extension-tac } @\{context\} \gg$ )
      moreover
      have  $?Fs2 \in \{?Fc\}$ 
      by (tactic  $\ll union\text{-closed-extension-tac } @\{context\} \gg$ )
      moreover
      have  $?Fs3 \in \{?Fc\}$ 
      by (tactic  $\ll union\text{-closed-extension-tac } @\{context\} \gg$ )
      moreover
      have  $?Fs4 \in \{?Fc\}$ 
      by (tactic  $\ll union\text{-closed-extension-tac } @\{context\} \gg$ )
      ultimately
      show  $?thesis$ 
      by (simp del: union-closed-extensions-def)
    qed
  next
    show let  $Fs = ?Fs$  in  $\forall a \in \bigcup ?Fc. sum\text{-list } (map\ (\lambda (x, y). int\ x * y) (zip\ ?c\ (map\ (frankl\text{-fun } a)\ Fs))) < 0$ 
    by (tactic  $\ll nonFC\text{-is-system-solution-tac } @\{context\} @\{term\ } [0..<6]::nat\ list\} 1 \gg$ )
  next
    show  $\exists wj \in List.set\ ?c. 0 < wj$ 
    by auto
  next
    show  $length\ ?c = length\ ?Fs$ 
    by auto
  next
    show finite  $(\bigcup ?Fc)$ 
    by auto
  next
    show union-closed  $?Fc$ 
    by (tactic  $\ll union\text{-closed-tac } @\{context\} 1 \gg$ )

```

qed
qed

lemma *[simp]*: $\neg FC\text{-family } (\{\{\}, \{0,1,2\}, \{0,1,3\}, \{2,4,5\}, \{0,1,2,3\}, \{0,1,2,3,4\}, \{0,1,2,3,5\}, \{0,1,2,4,5\}, \{0,1,2,3,4,5\}\})$
set set (**is** $\neg FC\text{-family } ?Fc$)

proof–

let *?Fs1* = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,3,4\}\}$
set set

let *?Fs2* = $\{\{\}, \{2\}, \{0,1,2\}, \{3\}, \{0,1,3\}, \{2,3\}, \{0,1,2,3\}, \{4\}, \{2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$
set set

let *?Fs3* = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
set set

let *?Fs4* = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
set set

let *?Fs5* = $\{\{\}, \{1\}, \{2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{1,2,3\}, \{0,1,2,3\}, \{4\}, \{1,4\}, \{2,4\}, \{1,2,3,4\}\}$
set set

let *?Fs6* = $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{3,4\}, \{0,3,4\}, \{0,1,3,4\}\}$
set set

let *?Fs7* = $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,2,3,4\}\}$
set set

let *?Fs8* = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}\}$
set set

let *?Fs* = [*?Fs1*, *?Fs2*, *?Fs3*, *?Fs4*, *?Fs5*, *?Fs6*, *?Fs7*, *?Fs8*]

let *?c* = [34, 77, 13, 13, 8, 1, 7, 17] :: *nat list*

show *?thesis*

proof (*rule nonFC*[**where** *Fs* = *?Fs* **and** *c* = *?c*])

show $\forall F \in \text{set } ?Fs. F \in \{\text{?Fc}\}$

proof–

have *?Fs1* $\in \{\text{?Fc}\}$

by (*tactic* $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have *?Fs2* $\in \{\text{?Fc}\}$

by (*tactic* $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have *?Fs3* $\in \{\text{?Fc}\}$

by (*tactic* $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have *?Fs4* $\in \{\text{?Fc}\}$

by (*tactic* $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have *?Fs5* $\in \{\text{?Fc}\}$

by (*tactic* $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have *?Fs6* $\in \{\text{?Fc}\}$

by (*tactic* $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

have *?Fs7* $\in \{\text{?Fc}\}$

by (*tactic* $\ll \text{union-closed-extension-tac } @\{\text{context}\} \gg$)

moreover

```

    have ?Fs8 ∈ {?Fc}
      by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
    ultimately
    show ?thesis
      by (simp del: union-closed-extensions-def)
  qed
next
  show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
    by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
next
  show ∃ wj ∈ List.set ?c. 0 < wj
    by auto
next
  show length ?c = length ?Fs
    by auto
next
  show finite (⋃ ?Fc)
    by auto
next
  show union-closed ?Fc
    by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

```

lemma [simp]: $\neg FC\text{-family} (\{\{\}, \{0,1,2\}, \{0,1,3\}, \{2,4,5\}, \{0,1,2,3\}, \{0,1,2,4\}, \{0,1,2,3,4\}, \{0,1,2,4,5\}, \{0,1,2,3,4,5\}\}, \text{set set})$ (is $\neg FC\text{-family } ?Fc$)

proof–

let ?Fs1 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{0,1,2,4\}\}$
set set

let ?Fs2 = $\{\{\}, \{2\}, \{0,1,2\}, \{3\}, \{0,1,3\}, \{2,3\}, \{0,1,2,3\}, \{4\}, \{2,4\}, \{0,1,2,4\}, \{3,4\}, \{0,1,3,4\}, \{2,3,4\}, \{0,1,2,3,4\}\}$
set set

let ?Fs3 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{1,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{1,2,3\}, \{0,1,2,3\}\}$
set set

let ?Fs4 = $\{\{\}, \{0\}, \{2\}, \{0,2\}, \{0,1,2\}, \{3\}, \{0,3\}, \{0,1,3\}, \{2,3\}, \{0,2,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{2,4\}, \{0,1,2,4\}\}$
set set

let ?Fs5 = $\{\{\}, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}, \{0,1,3\}, \{0,1,2,3\}, \{4\}, \{0,4\}, \{1,4\}, \{0,1,4\}, \{2,4\}, \{0,1,2,4\}\}$
set set

let ?Fs6 = $\{\{5\}, \{1,5\}, \{2,5\}, \{1,2,5\}, \{0,1,2,5\}, \{0,1,3,5\}, \{0,1,2,3,5\}, \{4,5\}, \{1,4,5\}, \{2,4,5\}, \{1,2,4,5\}, \{0,1,2,4,5\}\}$
set set

let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5, ?Fs6]

let ?c = [59, 151, 25, 1, 39, 4] :: nat list

show ?thesis

proof (rule nonFC[where Fs = ?Fs and c = ?c])

show $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$

proof–

have ?Fs1 ∈ {?Fc}

by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)

```

moreover
have ?Fs2 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs3 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs4 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs5 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
moreover
have ?Fs6 ∈ {?Fc}
  by (tactic ⟨⟨ union-closed-extension-tac @{context} ⟩⟩)
ultimately
show ?thesis
  by (simp del: union-closed-extensions-def)
qed
next
show let Fs = ?Fs in ∀ a ∈ ⋃ ?Fc. sum-list (map (λ (x, y). int x * y) (zip
?c (map (frankl-fun a) Fs))) < 0
  by (tactic ⟨⟨ nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat
list} 1 ⟩⟩)
next
show ∃ wj ∈ List.set ?c. 0 < wj
  by auto
next
show length ?c = length ?Fs
  by auto
next
show finite (⋃ ?Fc)
  by auto
next
show union-closed ?Fc
  by (tactic ⟨⟨ union-closed-tac @{context} 1 ⟩⟩)
qed
qed

```

lemma [simp]: ¬ FC-family ({}, {0,1,2}, {0,1,3}, {2,4,5}, {0,1,2,3}, {0,1,3,4}, {0,1,2,3,4}, {0,1,2,4,5}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}) (is ¬ FC-family ?Fc)

proof—

```

let ?Fs1 = {{}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
set set
let ?Fs2 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
set set
let ?Fs3 = {{}, {2}, {0,1,2}, {3}, {0,1,3}, {2,3}, {0,1,2,3}, {4}, {2,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {2,3,4}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}
set set
let ?Fs4 = {{}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {2,3}, {0,2,3}, {1,2,3}, {0,1,2,3}, {0,1,2,3,4}, {0,1,2,3,4,5}, {0,1,2,3,4,5,6}}

```

```

set set
let ?Fs5 = {{},{0},{1},{0,1},{2},{0,2},{1,2},{0,1,2},{0,1,3},{0,1,2,3},{4},{0,4},{1,4},{0,1,4},{2,4}}
set set
let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
let ?c = [37, 16, 110, 14, 21] :: nat list
show ?thesis
proof (rule nonFC[where Fs = ?Fs and c = ?c])
  show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
  proof-
    have ?Fs1  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs2  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs3  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs4  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  moreover
    have ?Fs5  $\in \{?Fc\}$ 
    by (tactic << union-closed-extension-tac @{context} >>)
  ultimately
    show ?thesis
    by (simp del: union-closed-extensions-def)
  qed
next
show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
by (tactic << nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1 >>)
next
show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
by auto
next
show length ?c = length ?Fs
by auto
next
show finite ( $\bigcup ?Fc$ )
by auto
next
show union-closed ?Fc
by (tactic << union-closed-tac @{context} 1 >>)
qed
qed
lemma [simp]:  $\neg \text{FC-family } (\{\{\}, \{0,1,2\}, \{0,1,3\}, \{2,4,5\}, \{0,1,2,3\}, \{2,3,4,5\}, \{0,1,2,4,5\}, \{0,1,2,3,4,5\}\} : \text{set set})$  (is  $\neg \text{FC-family } ?Fc$ )

```

```

proof–
  let ?Fs1 = { {}, {0}, {1}, {0,1}, {0,1,2}, {3}, {0,3}, {1,3}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {0,1,2,3,4} }
  set set
  let ?Fs2 = { {}, {0}, {1}, {0,1}, {2}, {0,2}, {1,2}, {0,1,2}, {0,1,3}, {0,1,2,3}, {4}, {0,4}, {1,4}, {0,1,4}, {2,4} }
  set set
  let ?Fs3 = { {}, {2}, {0,1,2}, {3}, {0,1,3}, {2,3}, {0,1,2,3}, {4}, {2,4}, {0,1,2,4}, {3,4}, {0,1,3,4}, {2,3,4}, {0,1,2,3,4} }
  set set
  let ?Fs4 = { {}, {2}, {0,1,2}, {3}, {0,1,3}, {2,3}, {0,1,2,3}, {2,4,5}, {0,1,2,4,5}, {2,3,4,5}, {0,1,2,3,4,5} } :: nat list
  set set
  let ?Fs5 = { {}, {0,1}, {2}, {0,1,2}, {3}, {0,1,3}, {2,3}, {0,1,2,3}, {2,4,5}, {0,1,2,4,5}, {2,3,4,5}, {0,1,2,3,4,5} } :: nat list
  set set
  let ?Fs = [?Fs1, ?Fs2, ?Fs3, ?Fs4, ?Fs5]
  let ?c = [16, 7, 36, 2, 28] :: nat list
  show ?thesis
  proof (rule nonFC[where Fs = ?Fs and c = ?c])
    show  $\forall F \in \text{set } ?Fs. F \in \{?Fc\}$ 
    proof–
      have ?Fs1  $\in \{?Fc\}$ 
      by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
      moreover
      have ?Fs2  $\in \{?Fc\}$ 
      by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
      moreover
      have ?Fs3  $\in \{?Fc\}$ 
      by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
      moreover
      have ?Fs4  $\in \{?Fc\}$ 
      by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
      moreover
      have ?Fs5  $\in \{?Fc\}$ 
      by (tactic  $\ll$  union-closed-extension-tac @{context}  $\gg$ )
      ultimately
      show ?thesis
      by (simp del: union-closed-extensions-def)
    qed
  next
    show let Fs = ?Fs in  $\forall a \in \bigcup ?Fc. \text{sum-list } (\text{map } (\lambda (x, y). \text{int } x * y) (\text{zip } ?c (\text{map } (\text{frankl-fun } a) Fs))) < 0$ 
    by (tactic  $\ll$  nonFC-is-system-solution-tac @{context} @{term [0..<6]::nat list} 1  $\gg$ )
  next
    show  $\exists wj \in \text{List.set } ?c. 0 < wj$ 
    by auto
  next
    show length ?c = length ?Fs
    by auto
  next
    show finite ( $\bigcup ?Fc$ )
    by auto

```

```

next
  show union-closed ?Fc
    by (tactic << union-closed-tac @{context} 1 >>)
qed
qed

```

```

theorem  $\forall F \in \text{set } \text{nonFC6}. \neg \text{FC-family } (f\text{-to-set-}l\ F)$ 
unfolding nonFC6-def
by (simp del: One-nat-def)

end

```

20 Proof that all families are covered

```

theory FC6
imports FC6-Data LPartitioningIrreducibleNonFCsCoveredImpl
begin

```

```

definition FC6-perms where
  FC6-perms = map ( $\lambda F. \text{map } (\lambda p. \text{permute-family-}l\ p\ F) (\text{permute } [0..<6])$ )
FC6

```

```

definition nonFC6-perms where
  nonFC6-perms = map ( $\lambda Nc. \text{map } (\lambda p. \text{close-insert-empty-}l\ (\text{permute-family-}l\ p\ Nc)) (\text{permute } [0..<6])$ ) nonFC6

```

lemma *L-part-6*:

```

assumes  $\bigcup F \subseteq \{0..<6::nat\}$ 
shows  $\exists n0\ n1\ n2\ n3\ n4\ n5\ n6. \quad$ 
  is-L-part 6 [n0, n1, n2, n3, n4, n5, n6] F  $\wedge$ 
   $n0 \leq 1 \wedge n1 \leq 6 \wedge n2 \leq 15 \wedge n3 \leq 20 \wedge n4 \leq 15 \wedge n5 \leq 6 \wedge n6 \leq 1$ 
using assms
proof-
{
  fix A
  assume  $A \in F$ 
  hence  $\text{card } A \leq \text{card } (\bigcup F)$ 
    using card-mono[of  $\bigcup F\ A$ ] finite-subset[of  $\bigcup F\ \{0..<6::nat\}$ ] assms
  by (smt Union-upper card-atLeastLessThan card-mono finite-atLeastLessThan)
  hence  $\text{card } A < 7$ 
    using assms card-mono[of  $\{0..<6::nat\} \cup F$ ]

```

```

    by auto
  }
  moreover
  have 6 choose 2 = 15 6 choose 3 = 20 6 choose 4 = 15 6 choose 5 = 6
    by eval+
  ultimately
  show ?thesis
    unfolding is-L-part-def
    apply (rule-tac x=card {A ∈ F. card A = 0} in exI)
    apply (rule-tac x=card {A ∈ F. card A = 1} in exI)
    apply (rule-tac x=card {A ∈ F. card A = 2} in exI)
    apply (rule-tac x=card {A ∈ F. card A = 3} in exI)
    apply (rule-tac x=card {A ∈ F. card A = 4} in exI)
    apply (rule-tac x=card {A ∈ F. card A = 5} in exI)
    apply (rule-tac x=card {A ∈ F. card A = 6} in exI)
    using assms
    apply auto
    apply (subgoal-tac Suc (Suc (Suc (Suc (Suc (Suc (Suc 0))))))) = 7)
    apply metis
    apply simp
    apply (case-tac n=0, simp)
    apply (case-tac n=1, simp)
    apply (case-tac n=2, simp)
    apply (case-tac n=3, simp)
    apply (case-tac n=4, simp)
    apply (case-tac n=5, simp)
    apply (case-tac n=6, simp)
    apply simp
    using n-subsets-ub[of F 6 0]
    apply simp
    using n-subsets-ub[of F 6 1]
    apply simp
    using n-subsets-ub[of F 6 2]
    apply simp
    using n-subsets-ub[of F 6 3]
    apply simp
    using n-subsets-ub[of F 6 4]
    apply simp
    using n-subsets-ub[of F 6 5]
    apply simp
    using n-subsets-ub[of F 6 6]
    apply simp
  done
qed

```

theorem *enum-rec-notFCs-covered-l:*

assumes *enum-rec-notFCs-covered-l FC6-perms* $[0, n1, n2, n3, n4, n5, n6]$ 6
(permute $[0..<6]) = []$

shows $\forall F \in L\text{-part } 6 [0, n1, n2, n3, n4, n5, n6]. FCs\text{-covered } (fs\text{-to-set-}l FC6) F$

proof –

have $\square \notin set FC6 \text{ dmf } FC6 \ 6 \text{ sdff } FC6$
by *eval* +
thus *?thesis*
using *assms*
using *enum-rec-notFCs-covered-l-iso-representing-subset*[*of FC6-perms 6 FC6*
 $[0, n1, n2, n3, n4, n5, n6]$]
unfolding *FC6-perms-def iso-represents-def*
by *simp*
qed

lemma *FC-0000560*:

shows $\forall F \in L\text{-part } 6 [0, 0, 0, 0, 5, 6, 0]. FCs\text{-covered } (fs\text{-to-set-}l FC6) F$
proof (*rule enum-rec-notFCs-covered-l*)
show *enum-rec-notFCs-covered-l FC6-perms* $[0, 0, 0, 0, 5, 6, 0] \ 6$ (*permute*
 $[0..<6]$) = \square
by *eval*
qed

lemma *FC-0000700*:

shows $\forall F \in L\text{-part } 6 [0, 0, 0, 0, 7, 0, 0]. FCs\text{-covered } (fs\text{-to-set-}l FC6) F$
proof (*rule enum-rec-notFCs-covered-l*)
show *enum-rec-notFCs-covered-l FC6-perms* $[0, 0, 0, 0, 7, 0, 0] \ 6$ (*permute*
 $[0..<6]$) = \square
by *eval*
qed

lemma *FC-0001650*:

shows $\forall F \in L\text{-part } 6 [0, 0, 0, 1, 6, 5, 0]. FCs\text{-covered } (fs\text{-to-set-}l FC6) F$
proof (*rule enum-rec-notFCs-covered-l*)
show *enum-rec-notFCs-covered-l FC6-perms* $[0, 0, 0, 1, 6, 5, 0] \ 6$ (*permute*
 $[0..<6]$) = \square
by *eval*
qed

lemma *FC-0002060*:

shows $\forall F \in L\text{-part } 6 [0, 0, 0, 2, 0, 6, 0]. FCs\text{-covered } (fs\text{-to-set-}l FC6) F$
proof (*rule enum-rec-notFCs-covered-l*)
show *enum-rec-notFCs-covered-l FC6-perms* $[0, 0, 0, 2, 0, 6, 0] \ 6$ (*permute*
 $[0..<6]$) = \square
by *eval*
qed

lemma *FC-0003040*:

shows $\forall F \in L\text{-part } 6 [0, 0, 0, 3, 0, 4, 0]. FCs\text{-covered } (fs\text{-to-set-}l FC6) F$
proof (*rule enum-rec-notFCs-covered-l*)

show *enum-rec-notFCs-covered-l FC6-perms* $[0, 0, 0, 3, 0, 4, 0]$ 6 (*permute* $[0..<6]$) = []
by *eval*
qed

lemma *FC-0003230*:

shows $\forall F \in L\text{-part } 6 \ [0, 0, 0, 3, 2, 3, 0]$. *FCs-covered* (*fs-to-set-l FC6*) *F*
proof (*rule enum-rec-notFCs-covered-l*)
show *enum-rec-notFCs-covered-l FC6-perms* $[0, 0, 0, 3, 2, 3, 0]$ 6 (*permute* $[0..<6]$) = []
by *eval*
qed

lemma *FC-0003300*:

shows $\forall F \in L\text{-part } 6 \ [0, 0, 0, 3, 3, 0, 0]$. *FCs-covered* (*fs-to-set-l FC6*) *F*
proof (*rule enum-rec-notFCs-covered-l*)
show *enum-rec-notFCs-covered-l FC6-perms* $[0, 0, 0, 3, 3, 0, 0]$ 6 (*permute* $[0..<6]$) = []
by *eval*
qed

lemma *FC-0004000*:

shows $\forall F \in L\text{-part } 6 \ [0, 0, 0, 4, 0, 0, 0]$. *FCs-covered* (*fs-to-set-l FC6*) *F*
proof (*rule enum-rec-notFCs-covered-l*)
show *enum-rec-notFCs-covered-l FC6-perms* $[0, 0, 0, 4, 0, 0, 0]$ 6 (*permute* $[0..<6]$) = []
by *eval*
qed

lemma *FC-0010000*:

shows $\forall F \in L\text{-part } 6 \ [0, 0, 1, 0, 0, 0, 0]$. *FCs-covered* (*fs-to-set-l FC6*) *F*
proof (*rule enum-rec-notFCs-covered-l*)
show *enum-rec-notFCs-covered-l FC6-perms* $[0, 0, 1, 0, 0, 0, 0]$ 6 (*permute* $[0..<6]$) = []
by *eval*
qed

lemma *FC-0100000*:

shows $\forall F \in L\text{-part } 6 \ [0, 1, 0, 0, 0, 0, 0]$. *FCs-covered* (*fs-to-set-l FC6*) *F*
proof (*rule enum-rec-notFCs-covered-l*)
show *enum-rec-notFCs-covered-l FC6-perms* $[0, 1, 0, 0, 0, 0, 0]$ 6 (*permute* $[0..<6]$) = []
by *eval*
qed

definition *all-FC-partitions* :: *nat list list* **where**

all-FC-partitions =
[

```

[0, 0, 0, 0, 5, 6, 0],
[0, 0, 0, 0, 7, 0, 0],
[0, 0, 0, 1, 6, 5, 0],
[0, 0, 0, 2, 0, 6, 0],
[0, 0, 0, 3, 0, 4, 0],
[0, 0, 0, 3, 2, 3, 0],
[0, 0, 0, 3, 3, 0, 0],
[0, 0, 0, 4, 0, 0, 0],
[0, 1, 0, 0, 0, 0, 0],
[0, 0, 1, 0, 0, 0, 0]
]

```

definition *notAllFC-partitions* **where**

notAllFC-partitions = $\{L'. [1, 6, 15, 20, 15, 6, 1] \succeq L' \wedge \neg (\exists S \in \text{set all-FC-partitions}. L' \succeq S)\}$

lemma *notAllFC-partitions-remove-empty*:

assumes $[1, n1, n2, n3, n4, n5, n6] \in \text{notAllFC-partitions}$ (**is** ?*L* \in -)
shows $[0, n1, n2, n3, n4, n5, n6] \in \text{notAllFC-partitions}$ (**is** ?*L'* \in -)
using *assms*
unfolding *notAllFC-partitions-def all-FC-partitions-def*
by (*auto simp add: pwge-Cons pwge-Nil*)

theorem *notAllFC-partitions*:

assumes $\forall L F. L \in \text{notAllFC-partitions} \wedge \text{hd } L = 0 \wedge F \in L\text{-part-irreducible}$
 $6 L \longrightarrow \text{covered } \mathcal{F} \mathcal{N} F$
 $\mathcal{F} = \text{fs-to-set-l FC6}$
shows $\forall F. \bigcup F \subseteq \{0..<6::\text{nat}\} \longrightarrow \text{covered } \mathcal{F} \mathcal{N} F$

proof(*rule all-irreducible-covered-all-covered*)

show $\forall F. \text{irreducible } F \wedge \bigcup F \subseteq \{0..<6::\text{nat}\} \longrightarrow \text{covered } \mathcal{F} \mathcal{N} F$

proof (*safe*)

fix *F*

assume $\bigcup F \subseteq \{0..<6::\text{nat}\}$ *irreducible F*

then obtain *n0 n1 n2 n3 n4 n5 n6* **where** *is-L-part 6* [*n0, n1, n2, n3, n4, n5, n6*] *F* **and** *: $n0 \leq 1 \ n1 \leq 6 \ n2 \leq 15 \ n3 \leq 20 \ n4 \leq 15 \ n5 \leq 6 \ n6 \leq 1$

using *L-part-6[of F]*

by *auto*

let ?*L* = [*n0, n1, n2, n3, n4, n5, n6*]

have **: $[1, 6, 15, 20, 15, 6, 1] \succeq [n0, n1, n2, n3, n4, n5, n6]$

using *

by (*simp add: pwge-Cons*)

show *covered F N F*

proof (*cases* $\exists L' \in \text{set all-FC-partitions}. ?L \succeq L'$)

case *False*

show ?*thesis*

```

proof (cases n0 = 0)
  case True
  thus ?thesis
    using assms(1)[rule-format, of ?L F] ⟨irreducible F⟩ ⟨is-L-part 6 ?L F⟩ ⟨¬
(∃ L' ∈ set all-FC-partitions. ?L ⋃ L')⟩ **
    unfolding notAllFC-partitions-def
    by auto
next
  case False
  hence n0 = 1
    using ⟨n0 ≤ 1⟩
    by auto
  show ?thesis
  proof (subst covered-remove-empty)
    show finite F
      using ⟨is-L-part 6 ?L F⟩
      by (simp add: is-L-part-finite)
    next
    let ?L' = [0, n1, n2, n3, n4, n5, n6]
    have ?L ∈ notAllFC-partitions
      using ** ⟨¬ (∃ L' ∈ set all-FC-partitions. ?L ⋃ L')⟩ ⟨n0 = 1⟩
      unfolding notAllFC-partitions-def
      by simp
    hence ?L' ∈ notAllFC-partitions
      by (subst (asm) ⟨n0 = 1⟩) (rule notAllFC-partitions-remove-empty)
    moreover
    have hd ?L' = 0
      by simp
    moreover
    have is-L-part 6 ?L' (F - {{{}}})
      using ⟨is-L-part 6 ?L F⟩
      using is-L-part-remove[of {} F 6 ?L] ⟨n0 = 1⟩
      using is-L-part-empty-mem[of 6 ?L F]
      by auto
    moreover
    have irreducible (F - {{{}}})
      using ⟨irreducible F⟩
      by (auto simp add: reducible-def)
    ultimately
    show covered  $\mathcal{F} \mathcal{N}$  (F - {{{}}})
      using assms(1)[rule-format, of ?L]
      by simp
    qed simp
  qed
next
  case True
  then obtain L where L ∈ set all-FC-partitions ?L ⋃ L
    unfolding all-FC-partitions-def
    by auto

```

```

show ?thesis
proof-
{
  assume ?L  $\succeq$  [0, 0, 0, 0, 5, 6, 0]
  hence covered  $\mathcal{F} \mathcal{N} F$ 
    using  $\langle is-L-part\ 6\ ?L\ F \rangle\ assms(2)$ 
    unfolding covered-def
    using FC-0000560
    using pwge-FCs-covered[of 6 -  $\mathcal{F}$  ?L]
    by auto
}
moreover
{
  assume ?L  $\succeq$  [0, 0, 0, 0, 7, 0, 0]
  hence covered  $\mathcal{F} \mathcal{N} F$ 
    using  $\langle is-L-part\ 6\ ?L\ F \rangle\ assms(2)$ 
    unfolding covered-def
    using FC-0000700
    using pwge-FCs-covered[of 6 -  $\mathcal{F}$  ?L]
    by auto
}
moreover
{
  assume ?L  $\succeq$  [0, 0, 0, 1, 6, 5, 0]
  hence covered  $\mathcal{F} \mathcal{N} F$ 
    using  $\langle is-L-part\ 6\ ?L\ F \rangle\ assms(2)$ 
    unfolding covered-def
    using FC-0001650
    using pwge-FCs-covered[of 6 -  $\mathcal{F}$  ?L]
    by auto
}
moreover
{
  assume ?L  $\succeq$  [0, 0, 0, 2, 0, 6, 0]
  hence covered  $\mathcal{F} \mathcal{N} F$ 
    using  $\langle is-L-part\ 6\ ?L\ F \rangle\ assms(2)$ 
    unfolding covered-def
    using FC-0002060
    using pwge-FCs-covered[of 6 -  $\mathcal{F}$  ?L]
    by auto
}
moreover
{
  assume ?L  $\succeq$  [0, 0, 0, 3, 0, 4, 0]
  hence covered  $\mathcal{F} \mathcal{N} F$ 
    using  $\langle is-L-part\ 6\ ?L\ F \rangle\ assms(2)$ 
    unfolding covered-def
    using FC-0003040
    using pwge-FCs-covered[of 6 -  $\mathcal{F}$  ?L]

```

```

    by auto
  }
  moreover
  {
    assume ?L  $\succeq$  [0, 0, 0, 3, 2, 3, 0]
    hence covered  $\mathcal{F} \mathcal{N} F$ 
      using  $\langle is-L-part\ 6\ ?L\ F \rangle\ assms(2)$ 
      unfolding covered-def
      using FC-0003230
      using pwge-FCs-covered[of 6 -  $\mathcal{F}$  ?L]
      by auto
  }
  moreover
  {
    assume ?L  $\succeq$  [0, 0, 0, 3, 3, 0, 0]
    hence covered  $\mathcal{F} \mathcal{N} F$ 
      using  $\langle is-L-part\ 6\ ?L\ F \rangle\ assms(2)$ 
      unfolding covered-def
      using FC-0003300
      using pwge-FCs-covered[of 6 -  $\mathcal{F}$  ?L]
      by auto
  }
  moreover
  {
    assume ?L  $\succeq$  [0, 0, 0, 4, 0, 0, 0]
    hence covered  $\mathcal{F} \mathcal{N} F$ 
      using  $\langle is-L-part\ 6\ ?L\ F \rangle\ assms(2)$ 
      unfolding covered-def
      using FC-0004000
      using pwge-FCs-covered[of 6 -  $\mathcal{F}$  ?L]
      by auto
  }
  moreover
  {
    assume ?L  $\succeq$  [0, 1, 0, 0, 0, 0, 0]
    hence covered  $\mathcal{F} \mathcal{N} F$ 
      using  $\langle is-L-part\ 6\ ?L\ F \rangle\ assms(2)$ 
      unfolding covered-def
      using FC-0100000
      using pwge-FCs-covered[of 6 -  $\mathcal{F}$  ?L]
      by auto
  }
  moreover
  {
    assume ?L  $\succeq$  [0, 0, 1, 0, 0, 0, 0]
    hence covered  $\mathcal{F} \mathcal{N} F$ 
      using  $\langle is-L-part\ 6\ ?L\ F \rangle\ assms(2)$ 
      unfolding covered-def
      using FC-0010000
  }

```

```

      using pwge-FCs-covered[of 6 -  $\mathcal{F}$  ?L]
      by auto
    }
  ultimately
  show ?thesis
    using  $\langle L \in \text{set all-FC-partitions} \rangle \langle ?L \succeq L \rangle$ 
    unfolding all-FC-partitions-def
    by auto
qed
qed
qed
qed

```

theorem

shows $\forall F. \bigcup F \subseteq \{0..<6::\text{nat}\} \longrightarrow \text{covered } (\text{fs-to-set-l FC6}) (\text{fs-to-set-l nonFC6}) F$

proof (rule notAllFC-partitions)

show $\forall L F. L \in \text{notAllFC-partitions} \wedge$
 $\text{hd } L = 0 \wedge F \in L\text{-part-irreducible } 6 L \longrightarrow$
 $\text{covered } (\text{fs-to-set-l FC6}) (\text{fs-to-set-l nonFC6}) F$

proof (safe)

fix L **and** $F::\text{nat set set}$

assume $L \in \text{notAllFC-partitions}$ $\text{hd } L = 0$ *irreducible* F *is-L-part* $6 L F$

let $?maxL = [0, 6, 15, 20, 15, 6, 1]$

let $?stops = \text{all-FC-partitions}$

let $?X = \{L'. ?maxL \succeq L' \wedge \neg (\exists S \in \text{set } ?stops. L' \succeq S)\}$

let $?perms = \text{permute } [0..<6]$

from $\langle L \in \text{notAllFC-partitions} \rangle \langle \text{hd } L = 0 \rangle$

have $L \in ?X$

unfolding notAllFC-partitions-def

by (cases L) (simp add: pwge-def, auto simp add: pwge-Cons)

moreover

have $\forall S \in \text{set all-FC-partitions}. \text{length } S = 7 \implies S \notin \text{set FC6}$ *dmf FC6 6 sdff FC6*

unfolding all-FC-partitions-def

by eval+

ultimately

obtain B **where**

$B \in \text{set } (\text{enum-dp-irreducible-notFCs-covered-l FC6-perms } 6 ?perms ?stops$
 $?maxL)$
 $\text{iso-representing-subset } (\text{fs-to-set-l } B) (L\text{-part-irreducible-notFCs-covered } (\text{fs-to-set-l FC6}) 6 L)$

using enum-dp-irreducible-notFCs-covered-l-iso-representing-subsets[of FC6-perms
 $\text{permute } [0..<6] \text{ FC6 } ?X ?maxL ?stops 6]$

by (auto simp add: FC6-perms-def)

show $\text{covered } (\text{fs-to-set-l FC6}) (\text{fs-to-set-l nonFC6}) F$

```

proof (rule iso-represents-L-part-irreducible-notFCs-covered[rule-format])
  show  $F \in L\text{-part-irreducible } 6 \ L$ 
    using  $\langle is\text{-}L\text{-part } 6 \ L \ F \rangle \langle irreducible \ F \rangle$ 
    by simp
next
  fix  $N$ 
  assume  $N \in fs\text{-to-set-l } nonFC6$ 
  thus finite  $(\bigcup N)$ 
    by auto
next
  show iso-representing-subset  $(fs\text{-to-set-l } B) (L\text{-part-irreducible-notFCs-covered } (fs\text{-to-set-l } FC6) \ 6 \ L)$ 
    by fact
next
  fix  $F$ 
  assume  $F \in fs\text{-to-set-l } B$ 
  moreover
    have *:  $\forall B \in set \ (enum\text{-dp-irreducible-notFCs-covered-l } FC6\text{-perms } 6 \ (permute \ [0..<6]) \ all\text{-FC-partitions } [0, 6, 15, 20, 15, 6, 1]). \forall F \in set \ B. nonFCs\text{-covered-l-opt } nonFC6\text{-perms } F$ 
      by eval
    have dmf  $nonFC6 \ 6$ 
      by eval
    hence  $\forall B \in set \ (enum\text{-dp-irreducible-notFCs-covered-l } FC6\text{-perms } 6 \ (permute \ [0..<6]) \ all\text{-FC-partitions } [0, 6, 15, 20, 15, 6, 1]). \forall F \in fs\text{-to-set-l } B. nonFCs\text{-covered } (fs\text{-to-set-l } nonFC6) \ F$ 
      using *
      using nonFCs-covered-l-soundness[of permute [0..<6] 6 ]
      by (subst (asm) nonFCs-covered-l-opt[of nonFC6-perms permute [0..<6] nonFC6]) (auto simp add: list-ex-iff isPermutation-permute nonFC6-perms-def)
    ultimately
    show nonFCs-covered  $(fs\text{-to-set-l } nonFC6) \ F$ 
      using  $\langle B \in set \ (enum\text{-dp-irreducible-notFCs-covered-l } FC6\text{-perms } 6 \ ?perms \ ?stops \ ?maxL) \rangle$ 
      by auto
    qed
  qed
qed simp

```

definition all-nonFC-partitions **where**

```

all-nonFC-partitions =
[
  [0, 0, 0, 0, 3, 6, 1],
  [0, 0, 0, 0, 4, 1, 1],
  [0, 0, 0, 1, 1, 6, 1],
  [0, 0, 0, 1, 2, 1, 1],
  [0, 0, 0, 2, 0, 1, 1]
]

```



```
]
end
```