Chapter 1

Library p_algebra_Matrix_product_h_1

```
Require Import BuiltIn.
Require BuiltIn.
Require HighOrd.
Require int.Int.
Require map.Map.

Axiom t : Type.
Parameter t_WhyType : WhyType t.

Parameter tzero: t.

Parameter tone: t.

Parameter infix_asdt: t → t → t.

Parameter infix_pldt: t → t → t.

Parameter infix_lseqdt: t → t → Prop.

Parameter prefix_mn: t → t.

Axiom Assoc :
  ∀ (x:t) (y:t) (z:t),
  ((infix_pldt (infix_pldt x y) z) = (infix_pldt x (infix_pldt y z))).

Axiom Unit_def_l : ∀ (x:t), ((infix_pldt tzero x) = x).

Axiom Unit_def_r : ∀ (x:t), ((infix_pldt x tzero) = x).

Axiom Inv_def_l : ∀ (x:t), ((infix_pldt (prefix_mn x) x) = tzero).

Axiom Inv_def_r : ∀ (x:t), ((infix_pldt x (prefix_mn x)) = tzero).

Axiom Comm : ∀ (x:t) (y:t), ((infix_pldt x y) = (infix_pldt y x)).

Axiom Assoc1 :
  ∀ (x:t) (y:t) (z:t),
  ((infix_asdt (infix_asdt x y) z) = (infix_asdt x (infix_asdt y z))).
```

```
Axiom Mul_distr_l :
  ∀ (x:t) (y:t) (z:t),
  ((infix_asdt x (infix_pldt y z)) =
   (infix_pldt (infix_asdt x y) (infix_asdt x z))).

Axiom Mul_distr_r :
  ∀ (x:t) (y:t) (z:t),
  ((infix_asdt (infix_pldt y z) x) =
   (infix_pldt (infix_asdt y x) (infix_asdt z x))).

Axiom Comm1 : ∀ (x:t) (y:t), ((infix_asdt x y) = (infix_asdt y x)).

Axiom Unitary : ∀ (x:t), ((infix_asdt tone x) = x).

Axiom NonTrivialRing : ¬ (tzero = tone).

Axiom Refl : ∀ (x:t), infix_lseqdt x x.

Axiom Trans :
  ∀ (x:t) (y:t) (z:t), (infix_lseqdt x y) → (infix_lseqdt y z) →
  infix_lseqdt x z.

Axiom Antisymm :
  ∀ (x:t) (y:t), (infix_lseqdt x y) → (infix_lseqdt y x) → (x = y).

Parameter infix_mndt: t → t → t.

Axiom infix_mndt_def :
  ∀ (x:t) (y:t), ((infix_mndt x y) = (infix_pldt x (prefix_mn y))).

Parameter prefix_mndt: t → t.

Axiom prefix_mndt_def : ∀ (x:t), ((prefix_mndt x) = (prefix_mn x)).

Parameter infix_lsdt: t → t → Prop.

Axiom infix_lsdt_def :
  ∀ (i:t) (j:t), (infix_lsdt i j) ↔ ((infix_lseqdt i j) ∧ ¬ (i = j)).

Parameter infix_gtdt: t → t → Prop.

Axiom infix_gtdt_def :
  ∀ (i:t) (j:t), (infix_gtdt i j) ↔ (infix_lsdt j i).

Parameter infix_gteqdt: t → t → Prop.

Axiom infix_gteqdt_def :
  ∀ (i:t) (j:t), (infix_gteqdt i j) ↔ (infix_lseqdt j i).

Parameter requal: t → t → bool.

Axiom requal_spec : ∀ (a:t) (b:t), ((requal a b) = true) ↔ (a = b).

Parameter ttwo: t.

Axiom ttwo_def : (ttwo = (infix_pldt tone tone)).

Axiom ZeroLessOne : infix_lsdt tzero tone.
```

```
Axiom absorbing_zero : ∀ (i:t), ((infix_asdt i tzero) = tzero).

Parameter comparable: t → t → Prop.

Axiom comparable_def :
  ∀ (a:t) (b:t),
  (comparable a b) ↔ ((infix_lseqdt a b) ∨ (infix_lseqdt b a)).

Axiom Transitive_comparable :
  ∀ (x:t) (y:t) (z:t), (comparable x y) → (comparable y z) →
  comparable x z.

Axiom Unitary_comparable :
  ∀ (x:t) (y:t), (comparable x y) →
  (comparable x tzero) ∧ (comparable x tone).

Axiom CompatStrictOrderAdd :
  ∀ (x:t) (y:t) (z:t), (infix_lsdt x y) → (comparable x z) →
  infix_lsdt (infix_pldt x z) (infix_pldt y z).

Axiom notZeroAdd :
  ∀ (x:t) (y:t), ¬ (x = tzero) → ¬ ((infix_pldt x y) = y).

Axiom CompatOrderMult :
  ∀ (x:t) (y:t) (z:t), (infix_lsdt x y) → (infix_lsdt tzero z) →
  infix_lsdt (infix_asdt x z) (infix_asdt y z).

Axiom compatStrictOrderMultComm :
  ∀ (x:t) (y:t) (z:t), (infix_lsdt x y) → (infix_lsdt tzero z) →
  infix_lsdt (infix_asdt z x) (infix_asdt z y).

Axiom compatOrderMult :
  ∀ (x:t) (y:t) (z:t), (infix_lseqdt x y) → (infix_lseqdt tzero z) →
  infix_lseqdt (infix_asdt x z) (infix_asdt y z).

Axiom compatOrderMultComm :
  ∀ (x:t) (y:t) (z:t), (infix_lseqdt x y) → (infix_lseqdt tzero z) →
  infix_lseqdt (infix_asdt z x) (infix_asdt z y).

Parameter inv: t → t.

Axiom Inverse :
  ∀ (x:t), ¬ (x = tzero) → ((infix_asdt x (inv x)) = tone).

Parameter infix_sldt: t → t → t.

Axiom infix_sldt_def :
  ∀ (x:t) (y:t), ((infix_sldt x y) = (infix_asdt x (inv y))).

Axiom add_div :
  ∀ (x:t) (y:t) (z:t), ¬ (z = tzero) →
  ((infix_sldt (infix_pldt x y) z) =
   (infix_pldt (infix_sldt x z) (infix_sldt y z))).
```

Axiom *sub_div* :
  $\forall\ (x{:}t)\ (y{:}t)\ (z{:}t),\ \neg\ (z = tzero) \to$
  $((infix\_sldt\ (infix\_mndt\ x\ y)\ z) =$
   $(infix\_mndt\ (infix\_sldt\ x\ z)\ (infix\_sldt\ y\ z)))$.

Axiom *neg_div* :
  $\forall\ (x{:}t)\ (y{:}t),\ \neg\ (y = tzero) \to$
  $((infix\_sldt\ (prefix\_mndt\ x)\ y) = (prefix\_mndt\ (infix\_sldt\ x\ y)))$.

Axiom *assoc_mul_div* :
  $\forall\ (x{:}t)\ (y{:}t)\ (z{:}t),\ \neg\ (z = tzero) \to$
  $((infix\_sldt\ (infix\_asdt\ x\ y)\ z) = (infix\_asdt\ x\ (infix\_sldt\ y\ z)))$.

Axiom *assoc_div_mul* :
  $\forall\ (x{:}t)\ (y{:}t)\ (z{:}t),\ (\neg\ (y = tzero) \land \neg\ (z = tzero)) \to$
  $((infix\_sldt\ (infix\_sldt\ x\ y)\ z) = (infix\_sldt\ x\ (infix\_asdt\ y\ z)))$.

Axiom *assoc_div_div* :
  $\forall\ (x{:}t)\ (y{:}t)\ (z{:}t),\ (\neg\ (y = tzero) \land \neg\ (z = tzero)) \to$
  $((infix\_sldt\ x\ (infix\_sldt\ y\ z)) = (infix\_sldt\ (infix\_asdt\ x\ z)\ y))$.

Axiom *inv_mult* :
  $\forall\ (x{:}t)\ (y{:}t),\ \neg\ (x = tzero) \to \neg\ (y = tzero) \to$
  $((infix\_asdt\ (infix\_sldt\ tone\ x)\ (infix\_sldt\ tone\ y)) =$
   $(infix\_sldt\ tone\ (infix\_asdt\ x\ y)))$.

Axiom *set* : $\forall\ (a{:}\mathtt{Type}),\ \mathtt{Type}$.
Parameter *set_WhyType* : $\forall\ (a{:}\mathtt{Type})\ \{a\_WT{:}\mathbf{WhyType}\ a\},\ \mathbf{WhyType}\ (set\ a)$.

Parameter *mem*: $\forall\ \{a{:}\mathtt{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\},\ a \to (set\ a) \to \mathtt{Prop}$.

Parameter *infix_eqeq*:
  $\forall\ \{a{:}\mathtt{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\},\ (set\ a) \to (set\ a) \to \mathtt{Prop}$.

Axiom *infix_eqeq_spec* :
  $\forall\ \{a{:}\mathtt{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\},$
  $\forall\ (s1{:}set\ a)\ (s2{:}set\ a),$
  $(infix\_eqeq\ s1\ s2) \leftrightarrow \forall\ (x{:}a),\ (mem\ x\ s1) \leftrightarrow (mem\ x\ s2)$.

Axiom *extensionality* :
  $\forall\ \{a{:}\mathtt{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\},$
  $\forall\ (s1{:}set\ a)\ (s2{:}set\ a),\ (infix\_eqeq\ s1\ s2) \to (s1 = s2)$.

Parameter *subset*:
  $\forall\ \{a{:}\mathtt{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\},\ (set\ a) \to (set\ a) \to \mathtt{Prop}$.

Axiom *subset_spec* :
  $\forall\ \{a{:}\mathtt{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\},$
  $\forall\ (s1{:}set\ a)\ (s2{:}set\ a),$
  $(subset\ s1\ s2) \leftrightarrow \forall\ (x{:}a),\ (mem\ x\ s1) \to mem\ x\ s2$.

Axiom *subset_refl* :

$\forall$ {a:Type} {a_WT:**WhyType** a}, $\forall$ (s:*set* a), *subset* s s.

`Axiom` *subset_trans* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ ($s1$:*set* a) ($s2$:*set* a) ($s3$:*set* a), (*subset* $s1$ $s2$) $\rightarrow$
(*subset* $s2$ $s3$) $\rightarrow$ *subset* $s1$ $s3$.

`Parameter` *is_empty*: $\forall$ {a:Type} {a_WT:**WhyType** a}, (*set* a) $\rightarrow$ `Prop`.

`Axiom` *is_empty_spec* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (s:*set* a), (*is_empty* s) $\leftrightarrow$ $\forall$ (x:a), $\neg$ (*mem* x s).

`Parameter` *empty*: $\forall$ {a:Type} {a_WT:**WhyType** a}, *set* a.

`Axiom` *empty_def* : $\forall$ {a:Type} {a_WT:**WhyType** a}, *is_empty* (*empty* : *set* a).

`Parameter` *add*: $\forall$ {a:Type} {a_WT:**WhyType** a}, a $\rightarrow$ (*set* a) $\rightarrow$ *set* a.

`Axiom` *add_spec* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (x:a) (s:*set* a), $\forall$ (y:a),
(*mem* y (*add* x s)) $\leftrightarrow$ ((y = x) $\vee$ (*mem* y s)).

`Parameter` *singleton*: $\forall$ {a:Type} {a_WT:**WhyType** a}, a $\rightarrow$ *set* a.

`Axiom` *singleton_def* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (x:a), ((*singleton* x) = (*add* x (*empty* : *set* a))).

`Parameter` *remove*: $\forall$ {a:Type} {a_WT:**WhyType** a}, a $\rightarrow$ (*set* a) $\rightarrow$ *set* a.

`Axiom` *remove_spec* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (x:a) (s:*set* a), $\forall$ (y:a),
(*mem* y (*remove* x s)) $\leftrightarrow$ ($\neg$ (y = x) $\wedge$ (*mem* y s)).

`Axiom` *add_remove* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (x:a) (s:*set* a), (*mem* x s) $\rightarrow$ ((*add* x (*remove* x s)) = s).

`Axiom` *remove_add* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (x:a) (s:*set* a), ((*remove* x (*add* x s)) = (*remove* x s)).

`Axiom` *subset_remove* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (x:a) (s:*set* a), *subset* (*remove* x s) s.

`Parameter` *union*:
$\forall$ {a:Type} {a_WT:**WhyType** a}, (*set* a) $\rightarrow$ (*set* a) $\rightarrow$ *set* a.

`Axiom` *union_spec* :
$\forall$ {a:Type} {a_WT:**WhyType** a},

$\forall$ ($s1$:*set* a) ($s2$:*set* a), $\forall$ (x:a),
(*mem* x (*union* $s1$ $s2$)) $\leftrightarrow$ ((*mem* x $s1$) $\vee$ (*mem* x $s2$)).

`Parameter` *inter*:
$\forall$ {a:Type} {a_WT:**WhyType** a}, (*set* a) $\rightarrow$ (*set* a) $\rightarrow$ *set* a.

`Axiom` *inter_spec* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ ($s1$:*set* a) ($s2$:*set* a), $\forall$ (x:a),
(*mem* x (*inter* $s1$ $s2$)) $\leftrightarrow$ ((*mem* x $s1$) $\wedge$ (*mem* x $s2$)).

`Parameter` *diff*:
$\forall$ {a:Type} {a_WT:**WhyType** a}, (*set* a) $\rightarrow$ (*set* a) $\rightarrow$ *set* a.

`Axiom` *diff_spec* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ ($s1$:*set* a) ($s2$:*set* a), $\forall$ (x:a),
(*mem* x (*diff* $s1$ $s2$)) $\leftrightarrow$ ((*mem* x $s1$) $\wedge$ $\neg$ (*mem* x $s2$)).

`Axiom` *subset_diff* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ ($s1$:*set* a) ($s2$:*set* a), *subset* (*diff* $s1$ $s2$) $s1$.

`Parameter` *choose*: $\forall$ {a:Type} {a_WT:**WhyType** a}, (*set* a) $\rightarrow$ a.

`Axiom` *choose_spec* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (s:*set* a), $\neg$ (*is_empty* s) $\rightarrow$ *mem* (*choose* s) s.

`Parameter` *cardinal*: $\forall$ {a:Type} {a_WT:**WhyType** a}, (*set* a) $\rightarrow$ **Z**.

`Axiom` *cardinal_nonneg* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (s:*set* a), ((*cardinal* s) $\geq$ $0\%Z$)$\%Z$.

`Axiom` *cardinal_empty* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (s:*set* a), ((*cardinal* s) = $0\%Z$) $\leftrightarrow$ (*is_empty* s).

`Axiom` *cardinal_add* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (x:a), $\forall$ (s:*set* a), $\neg$ (*mem* x s) $\rightarrow$
((*cardinal* (*add* x s)) = ($1\%Z$ + (*cardinal* s))$\%Z$).

`Axiom` *cardinal_remove* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ (x:a), $\forall$ (s:*set* a), (*mem* x s) $\rightarrow$
((*cardinal* s) = ($1\%Z$ + (*cardinal* (*remove* x s)))$\%Z$).

`Axiom` *cardinal_subset* :
$\forall$ {a:Type} {a_WT:**WhyType** a},
$\forall$ ($s1$:*set* a) ($s2$:*set* a), (*subset* $s1$ $s2$) $\rightarrow$

```
  ((cardinal s1) ≤ (cardinal s2))%Z.
Axiom subset_eq :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s1:set a) (s2:set a), (subset s1 s2) →
  ((cardinal s1) = (cardinal s2)) → infix_eqeq s1 s2.
Axiom cardinal1 :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a), ((cardinal s) = 1%Z) → ∀ (x:a), (mem x s) →
  (x = (choose s)).
Parameter op: ∀ {im:Type} {im_WT:WhyType im}, im → im → im.
Parameter po: ∀ {im:Type} {im_WT:WhyType im}, im → im → im.
Parameter inver: ∀ {im:Type} {im_WT:WhyType im}, im → im.
Inductive ref (a:Type) :=
  | mk_ref : a → ref a.
Axiom ref_WhyType : ∀ (a:Type) {a_WT:WhyType a}, WhyType (ref a).
Definition contents {a:Type} {a_WT:WhyType a} (v:ref a) : a :=
  match v with
  | mk_ref x ⇒ x
  end.
Parameter prefix_ex: ∀ {a:Type} {a_WT:WhyType a}, (ref a) → a.
Axiom prefix_ex_def :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (x:ref a), ((prefix_ex x) = (contents x)).
Axiom union_exchange :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (s':set a), ¬ (is_empty s') →
  ((union (add (choose s') s) (remove (choose s') s')) = (union s s')).
Axiom inter_empty :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (s':set a), (is_empty s) → is_empty (inter s s').
Axiom inter_empty_comm :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (s':set a), (is_empty s') → is_empty (inter s s').
Axiom union_empty :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (s':set a), (is_empty s) → ((union s s') = s').
Axiom union_comm :
  ∀ {a:Type} {a_WT:WhyType a},
```

```
  ∀ (s:set a) (s':set a), (is_empty s') → ((union s s') = s).
Axiom union_empty_comm :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (s':set a), (is_empty s') → ((union s s') = s).
Axiom union_add :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (s':set a) (x:a), ¬ (mem x s') →
  ((union s (add x s')) = (add x (union s s'))).
Axiom union_add_comm :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (s':set a) (x:a), ¬ (mem x s') →
  ((add x (union s s')) = (union s (add x s'))).
Axiom remove_add1 :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (x:a), ¬ (mem x s) → ((remove x (add x s)) = s).
Axiom add_remove1 :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (x:a), (mem x s) → ((add x (remove x s)) = s).
Parameter injective:
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b}, (set a) →
  (a → b) → Prop.
Axiom injective_def :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s:set a) (f:a → b),
  (injective s f) ↔
  ∀ (a1:a) (b1:a), ¬ (a1 = b1) → ¬ ((f a1) = (f b1)).
Parameter apply:
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b}, (set a) →
  (a → b) → set b.
Axiom apply_def :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s:set a) (f:a → b),
  ((is_empty s) → ((apply s f) = (empty : set b))) ∧
  (¬ (is_empty s) →
   ((apply s f) = (add (f (choose s)) (apply (remove (choose s) s) f)))).
Axiom apply_spec :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s:set a) (f:a → b), (injective s f) → ∀ (a1:a),
  (mem a1 s) ↔ (mem (f a1) (apply s f)).
Axiom apply_choose :
```

```
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s:set a), ∀ (f:a → b),
  ((choose (apply s f)) = (f (choose s))).
Axiom apply_remove_choose :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s:set a) (f:a → b), (injective s f) →
  ((apply (remove (choose s) s) f) =
   (remove (choose (apply s f)) (apply s f))).
Parameter right_injections:
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b}, a → (set b) →
  set (a× b)%type.
Axiom right_injections_def :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (a1:a) (s:set b),
  ((is_empty s) → ((right_injections a1 s) = (empty : set (a× b)%type))) ∧
  (¬ (is_empty s) →
   ((right_injections a1 s) =
    (add (a1 , choose s) (right_injections a1 (remove (choose s) s))))).
Axiom right_injections_spec :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (a1:a) (s:set b),
  ((cardinal (right_injections a1 s)) = (cardinal s)) ∧
  ((∀ (a':a), ∀ (b1:b),
    (mem (a', b1) (right_injections a1 s)) ↔ ((a' = a1) ∧ (mem b1 s))) ∧
   ((right_injections a1 s) = (apply s (fun (b1:b) ⇒ (a1 , b1))))).
Parameter left_injections:
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b}, (set a) → b →
  set (a× b)%type.
Axiom left_injections_def :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s:set a) (b1:b),
  ((is_empty s) → ((left_injections s b1) = (empty : set (a× b)%type))) ∧
  (¬ (is_empty s) →
   ((left_injections s b1) =
    (add (choose s , b1) (left_injections (remove (choose s) s) b1)))).
Axiom left_injections_spec :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s:set a) (b1:b),
  (∀ (a1:a), ∀ (b':b),
   (mem (a1 , b') (left_injections s b1)) ↔ ((mem a1 s) ∧ (b' = b1))) ∧
  (((cardinal (left_injections s b1)) = (cardinal s)) ∧
```

```
  ((left_injections s b1) = (apply s (fun (a1:a) ⇒ (a1, b1))))).

Axiom right_injections_l :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (a1:a) (s:set b),
  ((cardinal (right_injections a1 s)) = (cardinal s)) ∧
  ((∀ (a':a), ∀ (b1:b),
    (mem (a', b1) (right_injections a1 s)) ↔ ((a' = a1) ∧ (mem b1 s))) ∧
   ((right_injections a1 s) = (apply s (fun (b1:b) ⇒ (a1, b1))))).

Axiom left_injections_l :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s:set a) (b1:b),
  (∀ (a1:a), ∀ (b':b),
   (mem (a1, b') (left_injections s b1)) ↔ ((mem a1 s) ∧ (b' = b1))) ∧
  (((cardinal (left_injections s b1)) = (cardinal s)) ∧
   ((left_injections s b1) = (apply s (fun (a1:a) ⇒ (a1, b1))))).

Axiom disjoint_injections :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s:set a) (b1:b) (c:b), ¬ (b1 = c) →
  (is_empty (inter (right_injections b1 s) (right_injections c s))) ∧
  (is_empty (inter (left_injections s b1) (left_injections s c))).

Axiom induction :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (p:(set a) → bool) (t1:set a),
  (∀ (s:set a), (is_empty s) → ((p s) = true)) →
  (∀ (s:set a), ((p s) = true) → ∀ (t2:a), ¬ (mem t2 s) →
   ((p (add t2 s)) = true)) →
  ((p t1) = true).

Axiom cardinal_sum :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (s':set a),
  ((cardinal (union s s')) =
   (((cardinal s) + (cardinal s'))%Z - (cardinal (inter s s')))%Z).

Axiom cardinal_sum_empty_inter :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (s':set a), ((inter s s') = (empty : set a)) →
  ((cardinal (union s s')) = ((cardinal s) + (cardinal s'))%Z).

Parameter cartesian_product:
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b}, (set a) →
  (set b) → set (a× b)%type.

Axiom cartesian_product_spec :
```

```
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s1:set a) (s2:set b),
  ((cardinal (cartesian_product s1 s2)) = ((cardinal s1) × (cardinal s2))%Z) ∧
  ((∀ (a1:a), ∀ (b1:b),
    (mem (a1, b1) (cartesian_product s1 s2)) ↔ ((mem a1 s1) ∧ (mem b1 s2))) ∧
   ∀ (o:(a× b)%type),
   (mem o (cartesian_product s1 s2)) ↔
   match o with
   | (a1, b1) ⇒ (mem a1 s1) ∧ (mem b1 s2)
   end).

Parameter commute:
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b}, (a× b)%type →
  (b× a)%type.

Axiom commute_def :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (o:(a× b)%type),
  match o with
  | (a1, b1) ⇒ ((commute o) = (b1, a1))
  end.

Axiom commute_inj :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (a1:a) (a':a) (b1:b) (b':b), ((a1 = a') → ¬ (b1 = b')) →
  ¬ ((commute (a1, b1)) = (commute (a', b'))).

Axiom commute_inj_gen :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s1:set a) (s2:set b),
  injective (cartesian_product s1 s2) (fun (y0:(a× b)%type) ⇒ (commute y0)).

Parameter commute_product:
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b}, (set a) →
  (set b) → set (b× a)%type.

Axiom commute_product_def :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s1:set a) (s2:set b),
  ((commute_product s1 s2) =
   (apply (cartesian_product s1 s2) (fun (y0:(a× b)%type) ⇒ (commute y0)))).

Axiom commute_product_spec :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s1:set a) (s2:set b),
  ((commute_product s1 s2) = (cartesian_product s2 s1)).

Parameter commute_product_el:
```

```
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b}, (set a) →
  (set b) → set (b× a)%type.

Axiom commute_product_el_def :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s1:set a) (s2:set b),
  ((commute_product_el s1 s2) =
   (apply (cartesian_product s1 s2) (fun (y0:(a× b)%type) ⇒ (commute y0)))).

Axiom commute_product_el_spec :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s1:set a) (s2:set b), ∀ (o:(a× b)%type),
  match o with
  | (a1 , b1) ⇒
      ((mem o (cartesian_product s1 s2)) ↔ ((mem a1 s1) ∧ (mem b1 s2))) ∧
      (((mem a1 s1) ∧ (mem b1 s2)) ↔
       (mem (b1 , a1) (commute_product_el s1 s2)))
  end.

Axiom cartesian_product_union :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s1:set a) (s2:set b) (s3:set b),
  ((cartesian_product s1 (union s2 s3)) =
   (union (cartesian_product s1 s2) (cartesian_product s1 s3))).

Axiom cartesian_union_product :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s1:set a) (s2:set a) (s3:set b),
  ((cartesian_product (union s1 s2) s3) =
   (union (cartesian_product s1 s3) (cartesian_product s2 s3))).

Parameter id: ∀ {a:Type} {a_WT:WhyType a}, a → a.

Axiom id_def : ∀ {a:Type} {a_WT:WhyType a}, ∀ (e:a), ((id e) = e).

Axiom cartesian_product_cardone_r :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s1:set a) (s2:set b), ((cardinal s1) = 1%Z) →
  (infix_eqeq (cartesian_product s1 s2) (right_injections (choose s1) s2)) ∧
  (infix_eqeq (cartesian_product s1 s2)
   (apply s2 (fun (e2:b) ⇒ (choose s1 , e2)))).

Axiom cartesian_product_cardone_l :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (s1:set a) (s2:set b), ((cardinal s2) = 1%Z) →
  (infix_eqeq (cartesian_product s1 s2) (left_injections s1 (choose s2))) ∧
  (infix_eqeq (cartesian_product s1 s2)
   (apply s1 (fun (e1:a) ⇒ (e1 , choose s2)))).
```

```
Parameter op_neutral_left:
  ∀ {im:Type} {im_WT:WhyType im}, (im → im → im) → im → Prop.
Axiom op_neutral_left_def :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (neutral:im),
  (op_neutral_left op1 neutral) ↔ ∀ (e:im), (((op1 neutral) e) = e).
Parameter op_neutral_right:
  ∀ {im:Type} {im_WT:WhyType im}, (im → im → im) → im → Prop.
Axiom op_neutral_right_def :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (neutral:im),
  (op_neutral_right op1 neutral) ↔ ∀ (e:im), (((op1 e) neutral) = e).
Parameter op_assoc:
  ∀ {im:Type} {im_WT:WhyType im}, (im → im → im) → Prop.
Axiom op_assoc_def :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im),
  (op_assoc op1) ↔
  ∀ (a:im) (b:im) (c:im),
  (((op1 ((op1 a) b)) c) = ((op1 a) ((op1 b) c))).
Parameter op_neutral_left_c:
  ∀ {im:Type} {im_WT:WhyType im}, (im → im → im) → im → Prop.
Axiom op_neutral_left_c_def :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (neutral:im),
  (op_neutral_left_c op1 neutral) ↔
  ∀ (a:im), (∀ (b:im), (((op1 a) b) = b)) → (a = neutral).
Parameter op_refl:
  ∀ {im:Type} {im_WT:WhyType im}, (im → im → im) → Prop.
Axiom op_refl_def :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im),
  (op_refl op1) ↔ ∀ (a:im) (b:im), (((op1 a) b) = ((op1 b) a)).
Parameter assoc:
  ∀ {im:Type} {im_WT:WhyType im}, (im → im → im) → Prop.
Axiom assoc_def :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im),
  (assoc op1) ↔
```

$\forall$ $(a{:}im)$ $(b{:}im)$ $(c{:}im)$,
$(((op1\ ((op1\ a)\ b))\ c) = ((op1\ a)\ ((op1\ b)\ c)))$.

`Parameter` *opposite_n*:
$\forall$ $\{im{:}\mathtt{Type}\}$ $\{im\_WT{:}\textbf{WhyType}\ im\}$, $(im \to im \to im) \to$
$(im \to im \to im) \to im \to \mathtt{Prop}$.

`Axiom` *opposite_n_def* :
$\forall$ $\{im{:}\mathtt{Type}\}$ $\{im\_WT{:}\textbf{WhyType}\ im\}$,
$\forall$ $(op1{:}im \to im \to im)$ $(po1{:}im \to im \to im)$ $(neutral{:}im)$,
$(\mathsf{opposite\_n}\ op1\ po1\ neutral) \leftrightarrow \forall\ (a{:}im), (((po1\ a)\ a) = neutral)$.

`Parameter` *inverse*:
$\forall$ $\{im{:}\mathtt{Type}\}$ $\{im\_WT{:}\textbf{WhyType}\ im\}$, $(im \to im \to im) \to$
$(im \to im \to im) \to (im \to im) \to \mathtt{Prop}$.

`Axiom` *inverse_def* :
$\forall$ $\{im{:}\mathtt{Type}\}$ $\{im\_WT{:}\textbf{WhyType}\ im\}$,
$\forall$ $(op1{:}im \to im \to im)$ $(po1{:}im \to im \to im)$ $(inver1{:}im \to im)$,
$(\mathsf{inverse}\ op1\ po1\ inver1) \leftrightarrow$
$\forall$ $(a{:}im)$ $(b{:}im)$, $(((po1\ a)\ b) = ((op1\ a)\ (inver1\ b)))$.

`Parameter` *opposite*:
$\forall$ $\{im{:}\mathtt{Type}\}$ $\{im\_WT{:}\textbf{WhyType}\ im\}$, $(im \to im \to im) \to$
$(im \to im \to im) \to \mathtt{Prop}$.

`Axiom` *opposite_def* :
$\forall$ $\{im{:}\mathtt{Type}\}$ $\{im\_WT{:}\textbf{WhyType}\ im\}$,
$\forall$ $(op1{:}im \to im \to im)$ $(po1{:}im \to im \to im)$,
$(\mathsf{opposite}\ op1\ po1) \leftrightarrow \forall\ (a{:}im)\ (b{:}im), (((op1\ ((po1\ a)\ b))\ b) = a)$.

`Parameter` *opposite_com*:
$\forall$ $\{im{:}\mathtt{Type}\}$ $\{im\_WT{:}\textbf{WhyType}\ im\}$, $(im \to im \to im) \to$
$(im \to im \to im) \to \mathtt{Prop}$.

`Axiom` *opposite_com_def* :
$\forall$ $\{im{:}\mathtt{Type}\}$ $\{im\_WT{:}\textbf{WhyType}\ im\}$,
$\forall$ $(op1{:}im \to im \to im)$ $(po1{:}im \to im \to im)$,
$(\mathsf{opposite\_com}\ op1\ po1) \leftrightarrow$
$\forall$ $(a{:}im)$ $(b{:}im)$, $(((po1\ ((op1\ a)\ b))\ b) = a)$.

`Axiom` *refl* :
$\forall$ $\{im{:}\mathtt{Type}\}$ $\{im\_WT{:}\textbf{WhyType}\ im\}$,
$\forall$ $(op1{:}im \to im \to im)$ $(a{:}im)$ $(b{:}im)$, $(\mathsf{op\_refl}\ op1) \to$
$(((op1\ a)\ b) = ((op1\ b)\ a))$.

`Parameter` *neutral*:
$\forall$ $\{im{:}\mathtt{Type}\}$ $\{im\_WT{:}\textbf{WhyType}\ im\}$, $(im \to im \to im) \to im \to \mathtt{Prop}$.

`Axiom` *neutral_def* :

```
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (neutral1:im),
  (neutral op1 neutral1) ↔
  ((op_neutral_left op1 neutral1) ∧
   ((op_neutral_right op1 neutral1) ∧
    ((op_assoc op1) ∧ (op_neutral_left_c op1 neutral1)))).

Parameter iterates:
  ∀ {im:Type} {im_WT:WhyType im}, (im → im → im) → im → Prop.

Axiom iterates_def :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (neutral1:im),
  (iterates op1 neutral1) ↔
  ((op_neutral_left op1 neutral1) ∧
   ((op_neutral_right op1 neutral1) ∧
    ((op_assoc op1) ∧ ((op_neutral_left_c op1 neutral1) ∧ (op_refl op1))))).

Parameter iterable:
  ∀ {im:Type} {im_WT:WhyType im}, (im → im → im) → Prop.

Axiom iterable_def :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im), (iterable op1) ↔ ∃ e:im, iterates op1 e.

Parameter neutral_elt:
  ∀ {im:Type} {im_WT:WhyType im}, (im → im → im) → im.

Axiom neutral_elt_spec :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im), (iterable op1) →
  iterates op1 (neutral_elt op1).

Parameter inverse_tuple:
  ∀ {im:Type} {im_WT:WhyType im}, (im → im → im) →
  (im → im → im) → im → Prop.

Axiom inverse_tuple_def :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (po1:im → im → im) (neutral1:im),
  (inverse_tuple op1 po1 neutral1) ↔
  ((opposite_n op1 po1 neutral1) ∧
   ((opposite op1 po1) ∧ (opposite_com op1 po1))).

Parameter iterate:
  ∀ {a:Type} {a_WT:WhyType a} {im:Type} {im_WT:WhyType im},
  (im → im → im) → (set a) → (a → im) → im.

Axiom Iterate_def_empty :
  ∀ {a:Type} {a_WT:WhyType a} {im:Type} {im_WT:WhyType im},
```

$\forall\ (op1{:}im \rightarrow im \rightarrow im),\ \forall\ (s{:}\mathsf{set}\ a),\ \forall\ (f{:}a \rightarrow im),$
$(\mathsf{is\_empty}\ s) \rightarrow ((\mathsf{iterate}\ op1\ (\mathsf{empty} : \mathsf{set}\ a)\ f) = (\mathsf{neutral\_elt}\ op1)).$

`Axiom` *Iterate_add* :
$\forall\ \{a{:}\mathtt{Type}\}\ \{a\_WT{:}\textbf{WhyType}\ a\}\ \{im{:}\mathtt{Type}\}\ \{im\_WT{:}\textbf{WhyType}\ im\},$
$\forall\ (op1{:}im \rightarrow im \rightarrow im),\ \forall\ (s{:}\mathsf{set}\ a),\ \forall\ (f{:}a \rightarrow im),$
$\forall\ (x{:}a),\ \neg\ (\mathsf{mem}\ x\ s) \rightarrow$
$((\mathsf{iterate}\ op1\ (\mathsf{add}\ x\ s)\ f) = ((op1\ (f\ x))\ (\mathsf{iterate}\ op1\ s\ f))).$

`Axiom` *minus_zero* :
$\forall\ \{im{:}\mathtt{Type}\}\ \{im\_WT{:}\textbf{WhyType}\ im\},$
$\forall\ (op1{:}im \rightarrow im \rightarrow im)\ (po1{:}im \rightarrow im \rightarrow im)\ (a{:}im),\ (\mathsf{iterable}\ op1) \rightarrow$
$(\mathsf{inverse\_tuple}\ op1\ po1\ (\mathsf{neutral\_elt}\ op1)) \rightarrow$
$(((po1\ a)\ (\mathsf{neutral\_elt}\ op1)) = a).$

`Axiom` *unic* :
$\forall\ \{im{:}\mathtt{Type}\}\ \{im\_WT{:}\textbf{WhyType}\ im\},$
$\forall\ (op1{:}im \rightarrow im \rightarrow im)\ (po1{:}im \rightarrow im \rightarrow im)\ (a{:}im)\ (b{:}im)\ (c{:}im),$
$(\mathsf{iterable}\ op1) \rightarrow (((op1\ a)\ b) = ((op1\ a)\ c)) \rightarrow$
$(\mathsf{inverse\_tuple}\ op1\ po1\ (\mathsf{neutral\_elt}\ op1)) \rightarrow (b = c).$

`Axiom` *substract_comm* :
$\forall\ \{im{:}\mathtt{Type}\}\ \{im\_WT{:}\textbf{WhyType}\ im\},$
$\forall\ (op1{:}im \rightarrow im \rightarrow im)\ (po1{:}im \rightarrow im \rightarrow im)\ (a{:}im)\ (b{:}im)\ (c{:}im),$
$(\mathsf{iterable}\ op1) \rightarrow (\mathsf{inverse\_tuple}\ op1\ po1\ (\mathsf{neutral\_elt}\ op1)) \rightarrow$
$(((po1\ ((op1\ a)\ b))\ a) = b) \wedge$
$((((po1\ ((op1\ b)\ a))\ a) = b) \wedge$
$((((po1\ a)\ ((po1\ b)\ c)) = ((op1\ ((po1\ a)\ b))\ c)) \wedge$
$(((po1\ ((op1\ a)\ b))\ c) = ((op1\ a)\ ((po1\ b)\ c))))).$

`Parameter` *int_iterate*:
$\forall\ \{im{:}\mathtt{Type}\}\ \{im\_WT{:}\textbf{WhyType}\ im\},\ (im \rightarrow im \rightarrow im) \rightarrow (\mathbf{Z} \rightarrow im) \rightarrow$
$\mathbf{Z} \rightarrow \mathbf{Z} \rightarrow im.$

`Axiom` *int_iterate_def* :
$\forall\ \{im{:}\mathtt{Type}\}\ \{im\_WT{:}\textbf{WhyType}\ im\},$
$\forall\ (op1{:}im \rightarrow im \rightarrow im)\ (f{:}\mathbf{Z} \rightarrow im)\ (i{:}\mathbf{Z})\ (j{:}\mathbf{Z}),$
$((j \leq i)\%Z \rightarrow ((\mathsf{int\_iterate}\ op1\ f\ i\ j) = (\mathsf{neutral\_elt}\ op1))) \wedge$
$(\neg\ (j \leq i)\%Z \rightarrow$
$((\mathsf{int\_iterate}\ op1\ f\ i\ j) =$
$((op1\ (f\ i))\ (\mathsf{int\_iterate}\ op1\ f\ (i + 1\%Z)\%Z\ j)))).$

`Axiom` *int_iterate_spec* :
$\forall\ \{im{:}\mathtt{Type}\}\ \{im\_WT{:}\textbf{WhyType}\ im\},$
$\forall\ (op1{:}im \rightarrow im \rightarrow im)\ (f{:}\mathbf{Z} \rightarrow im)\ (i{:}\mathbf{Z})\ (j{:}\mathbf{Z}),\ (j \leq i)\%Z \rightarrow$
$((\mathsf{int\_iterate}\ op1\ f\ i\ j) = (\mathsf{neutral\_elt}\ op1)).$

`Parameter` *int_int_iterate*:

$\forall \, \{im{:}\mathtt{Type}\} \; \{im\_WT{:}\textbf{WhyType} \; im\}, \; (im \to im \to im) \to (\mathbf{Z} \to \mathbf{Z} \to im) \to$
$\mathbf{Z} \to \mathbf{Z} \to \mathbf{Z} \to \mathbf{Z} \to im.$

`Axiom` *int_int_iterate_def* :
$\forall \, \{im{:}\mathtt{Type}\} \; \{im\_WT{:}\textbf{WhyType} \; im\},$
$\forall \, (op1{:}im \to im \to im) \; (f{:}\mathbf{Z} \to \mathbf{Z} \to im) \; (i{:}\mathbf{Z}) \; (j{:}\mathbf{Z}) \; (k{:}\mathbf{Z}) \; (l{:}\mathbf{Z}),$
$((j \le i)\%Z \to ((\mathit{int\_int\_iterate} \; op1 \; f \; i \; j \; k \; l) = (\mathit{neutral\_elt} \; op1))) \wedge$
$(\neg \, (j \le i)\%Z \to$
$((\mathit{int\_int\_iterate} \; op1 \; f \; i \; j \; k \; l) =$
$((op1 \; (\mathit{int\_iterate} \; op1 \; ((\mathtt{fun} \; (y0{:}\mathbf{Z} \to im) \; (y1{:}\mathbf{Z}) \Rightarrow (y0 \; y1)) \; (f \; i)) \; k \; l))$
$(\mathit{int\_int\_iterate} \; op1 \; f \; (i + 1\%Z)\%Z \; j \; k \; l)))).$

`Axiom` *iterate_empty* :
$\forall \, \{a{:}\mathtt{Type}\} \; \{a\_WT{:}\textbf{WhyType} \; a\} \; \{im{:}\mathtt{Type}\} \; \{im\_WT{:}\textbf{WhyType} \; im\},$
$\forall \, (op1{:}im \to im \to im) \; (s{:}\mathit{set} \; a) \; (t1{:}a \to im), \; (\mathit{is\_empty} \; s) \to$
$((\mathit{iterate} \; op1 \; s \; t1) = (\mathit{neutral\_elt} \; op1)).$

`Axiom` *iterate_add* :
$\forall \, \{a{:}\mathtt{Type}\} \; \{a\_WT{:}\textbf{WhyType} \; a\} \; \{im{:}\mathtt{Type}\} \; \{im\_WT{:}\textbf{WhyType} \; im\},$
$\forall \, (op1{:}im \to im \to im) \; (s{:}\mathit{set} \; a) \; (f{:}a \to im) \; (x{:}a), \; \neg \, (\mathit{mem} \; x \; s) \to$
$((\mathit{iterate} \; op1 \; (\mathit{add} \; x \; s) \; f) = ((op1 \; (f \; x)) \; (\mathit{iterate} \; op1 \; s \; f))).$

`Axiom` *iterate_remove* :
$\forall \, \{a{:}\mathtt{Type}\} \; \{a\_WT{:}\textbf{WhyType} \; a\} \; \{im{:}\mathtt{Type}\} \; \{im\_WT{:}\textbf{WhyType} \; im\},$
$\forall \, (op1{:}im \to im \to im) \; (s{:}\mathit{set} \; a) \; (f{:}a \to im) \; (x{:}a), \; (\mathit{iterable} \; op1) \to$
$(\mathit{inverse\_tuple} \; op1 \; (\mathtt{fun} \; (y0{:}im) \; (y1{:}im) \Rightarrow (\mathit{po} \; y0 \; y1)) \; (\mathit{neutral\_elt} \; op1)) \to$
$(\mathit{mem} \; x \; s) \to ((\mathit{iterate} \; op1 \; (\mathit{remove} \; x \; s) \; f) = (\mathit{po} \; (\mathit{iterate} \; op1 \; s \; f) \; (f \; x))).$

`Axiom` *iterate_def_choose* :
$\forall \, \{a{:}\mathtt{Type}\} \; \{a\_WT{:}\textbf{WhyType} \; a\} \; \{im{:}\mathtt{Type}\} \; \{im\_WT{:}\textbf{WhyType} \; im\},$
$\forall \, (op1{:}im \to im \to im) \; (s{:}\mathit{set} \; a) \; (f{:}a \to im), \; (\mathit{iterable} \; op1) \to$
$\neg \, (\mathit{is\_empty} \; s) \to$
$((\mathit{iterate} \; op1 \; s \; f) =$
$((op1 \; (f \; (\mathit{choose} \; s))) \; (\mathit{iterate} \; op1 \; (\mathit{remove} \; (\mathit{choose} \; s) \; s) \; f))).$

`Axiom` *choose_any* :
$\forall \, \{a{:}\mathtt{Type}\} \; \{a\_WT{:}\textbf{WhyType} \; a\} \; \{im{:}\mathtt{Type}\} \; \{im\_WT{:}\textbf{WhyType} \; im\},$
$\forall \, (op1{:}im \to im \to im) \; (s{:}\mathit{set} \; a) \; (f{:}a \to im) \; (t1{:}a), \; (\mathit{iterable} \; op1) \to$
$(\mathit{mem} \; t1 \; s) \to$
$((\mathit{iterate} \; op1 \; s \; f) = ((op1 \; (f \; t1)) \; (\mathit{iterate} \; op1 \; (\mathit{remove} \; t1 \; s) \; f))).$

`Axiom` *iterate_comp_iterate* :
$\forall \, \{a{:}\mathtt{Type}\} \; \{a\_WT{:}\textbf{WhyType} \; a\} \; \{im{:}\mathtt{Type}\} \; \{im\_WT{:}\textbf{WhyType} \; im\},$
$\forall \, (op1{:}im \to im \to im) \; (s1{:}\mathit{set} \; a) \; (f{:}a \to im) \; (g{:}a \to im),$
$(\mathit{iterable} \; op1) \to$
$((\mathit{iterate} \; op1 \; s1 \; (\mathtt{fun} \; (k{:}a) \Rightarrow ((op1 \; (f \; k)) \; (g \; k)))) =$
$((op1 \; (\mathit{iterate} \; op1 \; s1 \; (\mathtt{fun} \; (k{:}a) \Rightarrow (f \; k))))$
$(\mathit{iterate} \; op1 \; s1 \; (\mathtt{fun} \; (k{:}a) \Rightarrow (g \; k))))).$

Axiom *iterate_comp_iterate_com* :
$\forall$ $\{a:\text{Type}\}$ $\{a\_WT:$**WhyType** $a\}$ $\{im:\text{Type}\}$ $\{im\_WT:$**WhyType** $im\}$,
$\forall$ $(op1 : im \to im \to im)$ $(s1 : set\ a)$ $(f : a \to im)$ $(g : a \to im)$,
(*iterable* $op1$) $\to$
$(((op1\ (iterate\ op1\ s1\ (\texttt{fun}\ (k:a) \Rightarrow (f\ k))))$
$(iterate\ op1\ s1\ (\texttt{fun}\ (k:a) \Rightarrow (g\ k))))$
$= (iterate\ op1\ s1\ (\texttt{fun}\ (k:a) \Rightarrow ((op1\ (f\ k))\ (g\ k)))))$.

Axiom *iterate_transitivity* :
$\forall$ $\{a:\text{Type}\}$ $\{a\_WT:$**WhyType** $a\}$ $\{im:\text{Type}\}$ $\{im\_WT:$**WhyType** $im\}$,
$\forall$ $(op1 : im \to im \to im)$ $(po1 : im \to im \to im)$ $(s1 : set\ a)$ $(s2 : set\ a)$
$(f : a \to im)$,
$(iterable\ op1) \to (inverse\_tuple\ op1\ po1\ (neutral\_elt\ op1)) \to$
$((iterate\ op1\ (union\ s1\ s2)\ f) =$
$((po1\ ((op1\ (iterate\ op1\ s1\ f))\ (iterate\ op1\ s2\ f)))$
$(iterate\ op1\ (inter\ s1\ s2)\ f)))$.

Axiom *iterate_disjoint_transitivity* :
$\forall$ $\{a:\text{Type}\}$ $\{a\_WT:$**WhyType** $a\}$ $\{im:\text{Type}\}$ $\{im\_WT:$**WhyType** $im\}$,
$\forall$ $(op1 : im \to im \to im)$ $(s1 : set\ a)$ $(s2 : set\ a)$ $(t1 : a \to im)$,
$(iterable\ op1) \to ((inter\ s1\ s2) = (empty : set\ a)) \to$
$((iterate\ op1\ (union\ s1\ s2)\ t1) =$
$((op1\ (iterate\ op1\ s1\ t1))\ (iterate\ op1\ s2\ t1)))$.

Axiom *iterate_eq* :
$\forall$ $\{a:\text{Type}\}$ $\{a\_WT:$**WhyType** $a\}$ $\{im:\text{Type}\}$ $\{im\_WT:$**WhyType** $im\}$,
$\forall$ $(op1 : im \to im \to im)$ $(s : set\ a)$ $(f : a \to im)$ $(g : a \to im)$,
$(\forall\ (x:a), (mem\ x\ s) \to ((f\ x) = (g\ x))) \to$
$((iterate\ op1\ s\ f) = (iterate\ op1\ s\ g))$.

Axiom *iterate_apply* :
$\forall$ $\{a:\text{Type}\}$ $\{a\_WT:$**WhyType** $a\}$ $\{b:\text{Type}\}$ $\{b\_WT:$**WhyType** $b\}$
$\{im:\text{Type}\}$ $\{im\_WT:$**WhyType** $im\}$,
$\forall$ $(op1 : im \to im \to im)$ $(s : set\ b)$ $(f : b \to a)$ $(t1 : a \to im)$,
$(iterable\ op1) \to (injective\ s\ f) \to$
$((iterate\ op1\ (apply\ s\ f)\ t1) =$
$(iterate\ op1\ s\ (\texttt{fun}\ (b1:b) \Rightarrow (t1\ (f\ b1)))))$.

Parameter *to_fset*: $\mathbf{Z} \to \mathbf{Z} \to set\ \mathbf{Z}$.

Axiom *to_fset_spec* :
$\forall$ $(i:\mathbf{Z})$ $(j:\mathbf{Z})$,
$((i < j)\%Z \to ((cardinal\ (to\_fset\ i\ j)) = (j - i)\%Z)) \wedge$
$(((j \leq i)\%Z \to is\_empty\ (to\_fset\ i\ j)) \wedge$
$\forall\ (k:\mathbf{Z}), (mem\ k\ (to\_fset\ i\ j)) \leftrightarrow ((i \leq k)\%Z \wedge (k < j)\%Z))$.

Axiom *choose_to_fset* :
$\forall$ $(i:\mathbf{Z})$ $(j:\mathbf{Z})$, $(i \leq j)\%Z \to ((choose\ (to\_fset\ i\ j)) = i)$.

Axiom *to_fset_unit_ext* :
$\forall\ (i{:}\mathbf{Z})\ (j{:}\mathbf{Z}),\ (i < j)\%Z \rightarrow$
$((\mathit{to\_fset}\ i\ j) = (\mathit{add}\ i\ (\mathit{to\_fset}\ (i + 1\%Z)\%Z\ j))).$

Axiom *to_fset_ext* :
$\forall\ (i{:}\mathbf{Z})\ (i'{:}\mathbf{Z})\ (j{:}\mathbf{Z}),\ ((i \leq i')\%Z \wedge (i' \leq j)\%Z) \rightarrow$
$((\mathit{to\_fset}\ i\ j) = (\mathit{union}\ (\mathit{to\_fset}\ i\ i')\ (\mathit{to\_fset}\ i'\ j))).$

Parameter *filter*:
$\forall\ \{a{:}\mathrm{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\},\ (a \rightarrow \mathbf{bool}) \rightarrow (\mathit{set}\ a) \rightarrow \mathit{set}\ a.$

Axiom *filter_def* :
$\forall\ \{a{:}\mathrm{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\},$
$\forall\ (p{:}a \rightarrow \mathbf{bool})\ (u{:}\mathit{set}\ a),\ \forall\ (x{:}a),$
$(\mathit{mem}\ x\ (\mathit{filter}\ p\ u)) \leftrightarrow (((p\ x) = \mathrm{true}) \wedge (\mathit{mem}\ x\ u)).$

Axiom *filter_cardinal* :
$\forall\ \{a{:}\mathrm{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\},$
$\forall\ (p{:}a \rightarrow \mathbf{bool})\ (u{:}\mathit{set}\ a),\ ((\mathit{cardinal}\ (\mathit{filter}\ p\ u)) \leq (\mathit{cardinal}\ u))\%Z.$

Parameter *map*:
$\forall\ \{a{:}\mathrm{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\}\ \{b{:}\mathrm{Type}\}\ \{b\_WT{:}\mathbf{WhyType}\ b\},\ (a \rightarrow b) \rightarrow$
$(\mathit{set}\ a) \rightarrow \mathit{set}\ b.$

Axiom *map_def1* :
$\forall\ \{a{:}\mathrm{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\}\ \{b{:}\mathrm{Type}\}\ \{b\_WT{:}\mathbf{WhyType}\ b\},$
$\forall\ (f{:}a \rightarrow b)\ (u{:}\mathit{set}\ a),\ \forall\ (y{:}b),$
$(\mathit{mem}\ y\ (\mathit{map}\ f\ u)) \leftrightarrow \exists\ x{:}a,\ (\mathit{mem}\ x\ u) \wedge (y = (f\ x)).$

Axiom *map_def2* :
$\forall\ \{a{:}\mathrm{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\}\ \{b{:}\mathrm{Type}\}\ \{b\_WT{:}\mathbf{WhyType}\ b\},$
$\forall\ (f{:}a \rightarrow b)\ (u{:}\mathit{set}\ a),\ \forall\ (x{:}a),\ (\mathit{mem}\ x\ u) \rightarrow \mathit{mem}\ (f\ x)\ (\mathit{map}\ f\ u).$

Axiom *map_cardinal* :
$\forall\ \{a{:}\mathrm{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\}\ \{b{:}\mathrm{Type}\}\ \{b\_WT{:}\mathbf{WhyType}\ b\},$
$\forall\ (f{:}a \rightarrow b)\ (u{:}\mathit{set}\ a),\ ((\mathit{cardinal}\ (\mathit{map}\ f\ u)) \leq (\mathit{cardinal}\ u))\%Z.$

Parameter *nonnull_part*:
$\forall\ \{a{:}\mathrm{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\}\ \{im{:}\mathrm{Type}\}\ \{im\_WT{:}\mathbf{WhyType}\ im\},$
$(im \rightarrow im \rightarrow im) \rightarrow (\mathit{set}\ a) \rightarrow (a \rightarrow im) \rightarrow \mathit{set}\ a.$

Parameter *result*:
$\forall\ \{a{:}\mathrm{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\}\ \{im{:}\mathrm{Type}\}\ \{im\_WT{:}\mathbf{WhyType}\ im\},$
$(im \rightarrow im \rightarrow im) \rightarrow (a \rightarrow im) \rightarrow a \rightarrow \mathbf{bool}.$

Axiom *result_def* :
$\forall\ \{a{:}\mathrm{Type}\}\ \{a\_WT{:}\mathbf{WhyType}\ a\}\ \{im{:}\mathrm{Type}\}\ \{im\_WT{:}\mathbf{WhyType}\ im\},$
$\forall\ (op1{:}im \rightarrow im \rightarrow im)\ (t1{:}a \rightarrow im)\ (a1{:}a),$
$(((\mathit{result}\ op1\ t1)\ a1) = \mathrm{true}) \leftrightarrow \neg\ ((t1\ a1) = (\mathit{neutral\_elt}\ op1)).$

Axiom *nonnull_part_def* :

∀ {a:Type} {a_WT:**WhyType** a} {im:Type} {im_WT:**WhyType** im},
∀ ($op1$:im → im → im) (s:set a) ($t1$:a → im), (iterable $op1$) →
((nonnull_part $op1$ s $t1$) = (filter (result $op1$ $t1$) s)).

`Axiom` *nonnull_part_spec* :
∀ {a:Type} {a_WT:**WhyType** a} {im:Type} {im_WT:**WhyType** im},
∀ ($op1$:im → im → im) (s:set a) ($t1$:a → im), (iterable $op1$) →
∀ ($a1$:a),
(mem $a1$ (nonnull_part $op1$ s $t1$)) ↔
((mem $a1$ s) ∧ ¬ (($t1$ $a1$) = (neutral_elt $op1$))).

`Parameter` *null_part*:
∀ {a:Type} {a_WT:**WhyType** a} {im:Type} {im_WT:**WhyType** im},
(im → im → im) → (set a) → (a → im) → set a.

`Parameter` *result1*:
∀ {a:Type} {a_WT:**WhyType** a} {im:Type} {im_WT:**WhyType** im},
(im → im → im) → (a → im) → a → **bool**.

`Axiom` *result_def1* :
∀ {a:Type} {a_WT:**WhyType** a} {im:Type} {im_WT:**WhyType** im},
∀ ($op1$:im → im → im) ($t1$:a → im) ($a1$:a),
(((result1 $op1$ $t1$) $a1$) = true) ↔ (($t1$ $a1$) = (neutral_elt $op1$)).

`Axiom` *null_part_def* :
∀ {a:Type} {a_WT:**WhyType** a} {im:Type} {im_WT:**WhyType** im},
∀ ($op1$:im → im → im) (s:set a) ($t1$:a → im), (iterable $op1$) →
((null_part $op1$ s $t1$) = (filter (result1 $op1$ $t1$) s)).

`Axiom` *null_part_spec* :
∀ {a:Type} {a_WT:**WhyType** a} {im:Type} {im_WT:**WhyType** im},
∀ ($op1$:im → im → im) (s:set a) ($t1$:a → im), (iterable $op1$) →
∀ ($a1$:a),
(mem $a1$ (null_part $op1$ s $t1$)) ↔
((mem $a1$ s) ∧ (($t1$ $a1$) = (neutral_elt $op1$))).

`Axiom` *nullity_partition* :
∀ {a:Type} {a_WT:**WhyType** a} {im:Type} {im_WT:**WhyType** im},
∀ ($op1$:im → im → im) (s:set a) ($t1$:a → im), (iterable $op1$) →
(s = (union (nonnull_part $op1$ s $t1$) (null_part $op1$ s $t1$))) ∧
((inter (nonnull_part $op1$ s $t1$) (null_part $op1$ s $t1$)) = (empty : set a)).

`Axiom` *iterate_neutral* :
∀ {a:Type} {a_WT:**WhyType** a} {im:Type} {im_WT:**WhyType** im},
∀ ($op1$:im → im → im) (s:set a) ($t1$:a → im), (iterable $op1$) →
(∀ ($a1$:a), (mem $a1$ s) → (($t1$ $a1$) = (neutral_elt $op1$))) →
((iterate $op1$ s $t1$) = (neutral_elt $op1$)).

`Axiom` *iterate_nullity_partition* :

```
  ∀ {a:Type} {a_WT:WhyType a} {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (s:set a) (t1:a → im), (iterable op1) →
  ((iterate op1 s t1) = (iterate op1 (nonnull_part op1 s t1) t1)).

Parameter element: ∀ {a:Type} {a_WT:WhyType a}, (set a) → a.

Axiom element_def :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a), ((cardinal s) = 1%Z) → ((element s) = (choose s)).

Axiom iterate_cardone :
  ∀ {a:Type} {a_WT:WhyType a} {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (s:set a) (t1:a → im), (iterable op1) →
  ((cardinal s) = 1%Z) → ((iterate op1 s t1) = (t1 (element s))).

Axiom iterate_cardzero :
  ∀ {a:Type} {a_WT:WhyType a} {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (s:set a) (t1:a → im), (iterable op1) →
  ((cardinal s) = 0%Z) → ((iterate op1 s t1) = (neutral_elt op1)).

Axiom cardone :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (a1:a), (∀ (b:a), (mem b s) ↔ (b = a1)) →
  ((cardinal s) = 1%Z) ∧ ((element s) = a1).

Parameter sum: ∀ {a:Type} {a_WT:WhyType a}, (set a) → (a → t) → t.

Axiom sum_def :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (t1:a → t),
  ((sum s t1) = (iterate (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1)) s t1)).

Axiom sum_iter : iterates (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1)) tzero.

Axiom sum_iter__ : iterable (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1)).

Axiom sum_iter_ :
  (opposite_n (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1))
   (fun (y0:t) (y1:t) ⇒ (infix_mndt y0 y1)) tzero) ∧
  ((opposite (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1))
     (fun (y0:t) (y1:t) ⇒ (infix_mndt y0 y1))) ∧
   ((opposite_com (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1))
      (fun (y0:t) (y1:t) ⇒ (infix_mndt y0 y1))) ∧
    (inverse_tuple (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1))
      (fun (y0:t) (y1:t) ⇒ (infix_mndt y0 y1)) tzero))).

Axiom neutral_zero :
  ((neutral_elt (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1))) = tzero).

Parameter ind_sum: (Z → t) → Z → Z → t.

Axiom ind_sum_def :
```

```
  ∀ (f:Z → t) (i:Z) (j:Z),
  ((ind_sum f i j) =
   (int_iterate (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1)) f i j)).

Parameter product:
  ∀ {a:Type} {a_WT:WhyType a}, (set a) → (a → t) → t.

Axiom product_def :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (t1:a → t),
  ((product s t1) = (iterate (fun (y0:t) (y1:t) ⇒ (infix_asdt y0 y1)) s t1)).

Axiom product_iter :
  (op_neutral_left (fun (y0:t) (y1:t) ⇒ (infix_asdt y0 y1)) tone) ∧
  (((op_neutral_right (fun (y0:t) (y1:t) ⇒ (infix_asdt y0 y1)) tone) ∧
    (op_assoc (fun (y0:t) (y1:t) ⇒ (infix_asdt y0 y1)))) ∧
   ((op_neutral_left_c (fun (y0:t) (y1:t) ⇒ (infix_asdt y0 y1)) tone) ∧
    ((op_refl (fun (y0:t) (y1:t) ⇒ (infix_asdt y0 y1))) ∧
     (iterates (fun (y0:t) (y1:t) ⇒ (infix_asdt y0 y1)) tone)))).

Axiom product_iter_ : iterable (fun (y0:t) (y1:t) ⇒ (infix_asdt y0 y1)).

Axiom neutral_one :
  ((neutral_elt (fun (y0:t) (y1:t) ⇒ (infix_asdt y0 y1))) = tone).

Parameter ind_product: (Z → t) → Z → Z → t.

Axiom ind_product_def :
  ∀ (f:Z → t) (i:Z) (j:Z),
  ((ind_product f i j) =
   (int_iterate (fun (y0:t) (y1:t) ⇒ (infix_asdt y0 y1)) f i j)).

Definition matrix (a:Type) := Z → Z → a.

Parameter rows: ∀ {a:Type} {a_WT:WhyType a}, (Z → Z → a) → Z.

Axiom rows_spec :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (m:Z → Z → a), (0%Z < (rows m))%Z.

Parameter columns: ∀ {a:Type} {a_WT:WhyType a}, (Z → Z → a) → Z.

Axiom columns_spec :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (m:Z → Z → a), (0%Z < (columns m))%Z.

Parameter valid_index:
  ∀ {a:Type} {a_WT:WhyType a}, (Z → Z → a) → Z → Z → Prop.

Axiom valid_index_def :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (a1:Z → Z → a) (r:Z) (c:Z),
```

```
  (valid_index a1 r c) ↔
  (((0%Z ≤ r)%Z ∧ (r < (rows a1))%Z) ∧
   ((0%Z ≤ c)%Z ∧ (c < (columns a1))%Z)).

Parameter get:
  ∀ {a:Type} {a_WT:WhyType a}, (Z → Z → a) → Z → Z → a.

Axiom get_spec :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (a1:Z → Z → a) (r:Z) (c:Z), (valid_index a1 r c) →
  ((get a1 r c) = ((a1 r) c)).

Parameter make: ∀ {a:Type} {a_WT:WhyType a}, Z → Z → a → Z → Z → a.

Axiom make_spec :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (r:Z) (c:Z) (v:a), ((r ≥ 0%Z)%Z ∧ (c ≥ 0%Z)%Z) →
  ((rows (make r c v)) = r) ∧
  (((columns (make r c v)) = c) ∧
   ∀ (i:Z) (j:Z),
   (((0%Z ≤ i)%Z ∧ (i < r)%Z) ∧ ((0%Z ≤ j)%Z ∧ (j < c)%Z)) →
   ((((make r c v) i) j) = v)).

Parameter set1:
  ∀ {a:Type} {a_WT:WhyType a}, (Z → Z → a) → Z → Z → a →
  Z → Z → a.

Axiom set_def :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (a1:Z → Z → a) (r:Z) (c:Z) (v:a), (valid_index a1 r c) →
  ((set1 a1 r c v) =
   (map.Map.mixfix_lblsmnrb a1 r (map.Map.mixfix_lblsmnrb (a1 r) c v))).

Axiom set_spec :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (a1:Z → Z → a) (r:Z) (c:Z) (v:a), (valid_index a1 r c) →
  ((rows (set1 a1 r c v)) = (rows a1)) ∧
  (((columns (set1 a1 r c v)) = (columns a1)) ∧
   ((∀ (i:Z) (j:Z), (valid_index a1 i j) →
      (((i = r) ∧ (j = c)) → ((((set1 a1 r c v) i) j) = v)) ∧
      (~ ((i = r) ∧ (j = c)) → ((((set1 a1 r c v) i) j) = ((a1 i) j)))) ∧
    (((((set1 a1 r c v) r) c) = v) ∧
     ((∀ (i:Z) (j:Z), (valid_index (set1 a1 r c v) i j) → ¬ (i = r) →
        ((((set1 a1 r c v) i) j) = ((a1 i) j))) ∧
      ∀ (i:Z) (j:Z), (valid_index (set1 a1 r c v) i j) → ¬ (j = c) →
      ((((set1 a1 r c v) i) j) = ((a1 i) j)))))).

Parameter make_f:
```

$\forall$ {a:Type} {a_WT:$WhyType$ a}, $Z \to Z \to (Z \to Z \to a) \to Z \to Z \to a$.

Axiom *make_f_spec* :
$\forall$ {a:Type} {a_WT:$WhyType$ a},
$\forall$ (r:Z) (c:Z) (f:$Z \to Z \to a$),
(($rows$ ($make\_f$ r c f)) = r) $\wedge$
((($columns$ ($make\_f$ r c f)) = c) $\wedge$
$\forall$ (i:Z) (j:Z), ($valid\_index$ ($make\_f$ r c f) i j) $\to$
(((($make\_f$ r c f) i) j) = ((f i) j))).

Parameter *equal*:
$\forall$ {a:Type} {a_WT:$WhyType$ a}, ($Z \to Z \to a$) $\to$ ($Z \to Z \to a$) $\to$ Prop.

Axiom *equal_mat* :
$\forall$ {a:Type} {a_WT:$WhyType$ a},
$\forall$ (m:$Z \to Z \to a$) (n:$Z \to Z \to a$),
($equal$ m n) $\leftrightarrow$
(((($rows$ m) = ($rows$ n)) $\wedge$ (($columns$ m) = ($columns$ n))) $\wedge$
$\forall$ (i:Z) (j:Z), ($valid\_index$ m i j) $\to$ (((m i) j) = ((n i) j))).

Parameter *square*: $\forall$ {a:Type} {a_WT:$WhyType$ a}, ($Z \to Z \to a$) $\to$ Prop.

Axiom *square_mat* :
$\forall$ {a:Type} {a_WT:$WhyType$ a},
$\forall$ (m:$Z \to Z \to a$), ($square$ m) $\leftrightarrow$ (($rows$ m) = ($columns$ m)).

Axiom *equal_sym* :
$\forall$ {a:Type} {a_WT:$WhyType$ a},
$\forall$ (m:$Z \to Z \to a$) (n:$Z \to Z \to a$), ($equal$ m n) $\leftrightarrow$ ($equal$ n m).

Axiom *equal_rex* :
$\forall$ {a:Type} {a_WT:$WhyType$ a}, $\forall$ (m:$Z \to Z \to a$), $equal$ m m.

Axiom *equal_trans* :
$\forall$ {a:Type} {a_WT:$WhyType$ a},
$\forall$ (m:$Z \to Z \to a$) (n:$Z \to Z \to a$) (o:$Z \to Z \to a$), ($equal$ m n) $\to$
($equal$ n o) $\to$ $equal$ m o.

Parameter *equal_funct*:
$\forall$ {a:Type} {a_WT:$WhyType$ a}, ($Z \to Z \to Z \to a$) $\to$
($Z \to Z \to Z \to a$) $\to$ Prop.

Axiom *equal_mat_funct* :
$\forall$ {a:Type} {a_WT:$WhyType$ a},
$\forall$ (f:$Z \to Z \to Z \to a$) (g:$Z \to Z \to Z \to a$), $\forall$ (i:Z) (j:Z),
((0%Z $\leq$ i)%Z $\wedge$ (i < j)%Z) $\to$
($equal\_funct$ f g) $\leftrightarrow$
$\forall$ (k:Z), ((i $\leq$ k)%Z $\wedge$ (k < j)%Z) $\to$ $equal$ (f k) (g k).

Parameter *nonnull_part1*:

$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
($im \to im \to im$) $\to$ (set a) $\to$ ($a \to im$) $\to$ set a.

Parameter *result2*:
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
($im \to im \to im$) $\to$ ($a \to im$) $\to$ $a \to bool$.

Axiom *result_def2* :
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
$\forall$ ($op1$:$im \to im \to im$) ($t1$:$a \to im$) ($a1$:a),
$(((result2\ op1\ t1)\ a1) = true) \leftrightarrow \neg\ ((t1\ a1) = (neutral\_elt\ op1))$.

Axiom *nonnull_part_def1* :
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
$\forall$ ($op1$:$im \to im \to im$) (s:set a) ($t1$:$a \to im$), ($iterable\ op1$) $\to$
$((nonnull\_part1\ op1\ s\ t1) = (filter\ (result2\ op1\ t1)\ s))$.

Axiom *nonnull_part_spec1* :
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
$\forall$ ($op1$:$im \to im \to im$) (s:set a) ($t1$:$a \to im$), ($iterable\ op1$) $\to$
$\forall$ ($a1$:a),
$(mem\ a1\ (nonnull\_part1\ op1\ s\ t1)) \leftrightarrow$
$((mem\ a1\ s) \land \neg\ ((t1\ a1) = (neutral\_elt\ op1)))$.

Parameter *null_part1*:
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
($im \to im \to im$) $\to$ (set a) $\to$ ($a \to im$) $\to$ set a.

Parameter *result3*:
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
($im \to im \to im$) $\to$ ($a \to im$) $\to$ $a \to bool$.

Axiom *result_def3* :
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
$\forall$ ($op1$:$im \to im \to im$) ($t1$:$a \to im$) ($a1$:a),
$(((result3\ op1\ t1)\ a1) = true) \leftrightarrow ((t1\ a1) = (neutral\_elt\ op1))$.

Axiom *null_part_def1* :
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
$\forall$ ($op1$:$im \to im \to im$) (s:set a) ($t1$:$a \to im$), ($iterable\ op1$) $\to$
$((null\_part1\ op1\ s\ t1) = (filter\ (result3\ op1\ t1)\ s))$.

Axiom *null_part_spec1* :
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
$\forall$ ($op1$:$im \to im \to im$) (s:set a) ($t1$:$a \to im$), ($iterable\ op1$) $\to$
$\forall$ ($a1$:a),
$(mem\ a1\ (null\_part1\ op1\ s\ t1)) \leftrightarrow$
$((mem\ a1\ s) \land ((t1\ a1) = (neutral\_elt\ op1)))$.

Axiom *nullity_partition1* :

$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
$\forall$ ($op1$:$im \to im \to im$) (s:set a) ($t1$:$a \to im$), ($iterable$ $op1$) $\to$
(s = ($union$ ($nonnull\_part1$ $op1$ s $t1$) ($null\_part1$ $op1$ s $t1$))) $\land$
(($inter$ ($nonnull\_part1$ $op1$ s $t1$) ($null\_part1$ $op1$ s $t1$)) = ($empty$: set a)).

Axiom $iterate\_neutral1$:
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
$\forall$ ($op1$:$im \to im \to im$) (s:set a) ($t1$:$a \to im$), ($iterable$ $op1$) $\to$
($\forall$ ($a1$:a), (mem $a1$ s) $\to$ (($t1$ $a1$) = ($neutral\_elt$ $op1$))) $\to$
(($iterate$ $op1$ s $t1$) = ($neutral\_elt$ $op1$)).

Axiom $iterate\_nullity\_partition1$:
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
$\forall$ ($op1$:$im \to im \to im$) (s:set a) ($t1$:$a \to im$), ($iterable$ $op1$) $\to$
(($iterate$ $op1$ s $t1$) = ($iterate$ $op1$ ($nonnull\_part1$ $op1$ s $t1$) $t1$)).

Parameter $element1$: $\forall$ {a:Type} {a_WT:$WhyType$ a}, (set a) $\to$ a.

Axiom $element\_def1$:
$\forall$ {a:Type} {a_WT:$WhyType$ a},
$\forall$ (s:set a), (($cardinal$ s) = 1%Z) $\to$ (($element1$ s) = ($choose$ s)).

Axiom $iterate\_cardone1$:
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
$\forall$ ($op1$:$im \to im \to im$) (s:set a) ($t1$:$a \to im$), ($iterable$ $op1$) $\to$
(($cardinal$ s) = 1%Z) $\to$ (($iterate$ $op1$ s $t1$) = ($t1$ ($element1$ s))).

Axiom $iterate\_cardzero1$:
$\forall$ {a:Type} {a_WT:$WhyType$ a} {im:Type} {im_WT:$WhyType$ im},
$\forall$ ($op1$:$im \to im \to im$) (s:set a) ($t1$:$a \to im$), ($iterable$ $op1$) $\to$
(($cardinal$ s) = 0%Z) $\to$ (($iterate$ $op1$ s $t1$) = ($neutral\_elt$ $op1$)).

Axiom $cardone1$:
$\forall$ {a:Type} {a_WT:$WhyType$ a},
$\forall$ (s:set a) ($a1$:a), ($\forall$ (b:a), (mem b s) $\leftrightarrow$ (b = $a1$)) $\to$
(($cardinal$ s) = 1%Z) $\land$ (($element1$ s) = $a1$).

Parameter $indic$: $\forall$ {a:Type} {a_WT:$WhyType$ a}, $a \to a \to t$.

Axiom $indic\_def$:
$\forall$ {a:Type} {a_WT:$WhyType$ a},
$\forall$ ($a1$:a) (a':a),
(($a1$ = a') $\to$ (($indic$ $a1$ a') = $tone$)) $\land$
(~ ($a1$ = a') $\to$ (($indic$ $a1$ a') = $tzero$)).

Axiom $indic\_comm$:
$\forall$ {a:Type} {a_WT:$WhyType$ a},
$\forall$ ($a1$:a) (a':a), (($indic$ $a1$ a') = ($indic$ a' $a1$)).

Parameter $indic\_2$:

```
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b}, a → a → b →
  b → t.

Axiom indic_2_def :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (a1:a) (a':a) (b1:b) (b':b),
  (((a1 = a') ∧ (b1 = b')) → ((indic_2 a1 a' b1 b') = tone)) ∧
  (~ ((a1 = a') ∧ (b1 = b')) → ((indic_2 a1 a' b1 b') = tzero)).

Axiom indic_2_spec :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (a1:a) (a':a) (b1:b) (b':b),
  ((indic_2 a1 a' b1 b') = (infix_asdt (indic a1 a') (indic b1 b'))) ∧
  ((indic_2 a1 a' b1 b') = (indic (a1, b1) (a', b'))).

Axiom indic_2_comm :
  ∀ {a:Type} {a_WT:WhyType a} {b:Type} {b_WT:WhyType b},
  ∀ (a1:a) (a':a) (b1:b) (b':b),
  ((indic_2 a1 a' b1 b') = (indic_2 a' a1 b1 b')) ∧
  (((indic_2 a1 a' b1 b') = (indic_2 a1 a' b' b1)) ∧
   ((indic_2 a1 a' b1 b') = (indic_2 a' a1 b' b1))).

Parameter sum_indic:
  ∀ {a:Type} {a_WT:WhyType a}, (set a) → (a → t) → a → t.

Axiom sum_indic_def :
  ∀ {a:Type} {a_WT:WhyType a},
  ∀ (s:set a) (t1:a → t) (i:a),
  ((sum_indic s t1 i) =
   (sum s (fun (e:a) ⇒ (infix_asdt (t1 e) (indic i e))))).

Axiom int_iterate_def_empty :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (f:Z → im) (i:Z) (j:Z), (j ≤ i)%Z →
  ((int_iterate op1 f i j) = (neutral_elt op1)).

Axiom int_iterate_def_plus_one :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (f:Z → im) (i:Z) (j:Z), (i < j)%Z →
  ((int_iterate op1 f i j) = ((op1 (f i)) (int_iterate op1 f (i + 1%Z)%Z j))).

Axiom int_iterate_to_iterate :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (f:Z → im) (i:Z) (j:Z),
  ((int_iterate op1 f i j) = (iterate op1 (to_fset i j) f)).

Axiom int_iterate_right_extension :
  ∀ {im:Type} {im_WT:WhyType im},
  ∀ (op1:im → im → im) (f:Z → im) (i:Z) (j:Z), (assoc op1) →
```

$(i < j)\%Z \rightarrow$
$((int\_iterate\ op1\ f\ i\ j) =$
$((op1\ (int\_iterate\ op1\ f\ i\ (j - 1\%Z)\%Z))\ (f\ (j - 1\%Z)\%Z)))$.

`Axiom` $int\_iterate\_transitivity$:
$\forall\ \{im{:}\texttt{Type}\}\ \{im\_WT{:}WhyType\ im\}$,
$\forall\ (op1{:}im \rightarrow im \rightarrow im)\ (f{:}Z \rightarrow im)\ (i{:}Z)\ (k{:}Z)\ (j{:}Z),\ (assoc\ op1) \rightarrow$
$(neutral\ op1\ (neutral\_elt\ op1)) \rightarrow ((i \leq k)\%Z \wedge (k \leq j)\%Z) \rightarrow$
$((int\_iterate\ op1\ f\ i\ j) =$
$((op1\ (int\_iterate\ op1\ f\ i\ k))\ (int\_iterate\ op1\ f\ k\ j)))$.

`Axiom` $int\_iterate\_attr$:
$\forall\ \{im1{:}\texttt{Type}\}\ \{im1\_WT{:}WhyType\ im1\}\ \{im2{:}\texttt{Type}\}\ \{im2\_WT{:}WhyType\ im2\}$,
$\forall\ (op1{:}im1 \rightarrow im1 \rightarrow im1)\ (op2{:}im2 \rightarrow im2 \rightarrow im2)\ (t1{:}Z \rightarrow im1)$
$(f{:}im1 \rightarrow im2)\ (i{:}Z)\ (j{:}Z)$,
$(\forall\ (x{:}Z),\ \forall\ (y{:}im1)$,
$(((op2\ (f\ (t1\ x)))\ (f\ y)) = (f\ ((op1\ (t1\ x))\ y)))) \rightarrow$
$((f\ (neutral\_elt\ op1)) = (neutral\_elt\ op2)) \rightarrow (assoc\ op1) \rightarrow$
$(assoc\ op2) \rightarrow$
$((int\_iterate\ op2\ (\texttt{fun}\ (e{:}Z) \Rightarrow (f\ (t1\ e)))\ i\ j) =$
$(f\ (int\_iterate\ op1\ t1\ i\ j)))$.

`Parameter` $isum$: $\forall\ \{a{:}\texttt{Type}\}\ \{a\_WT{:}WhyType\ a\},\ (\texttt{set}\ a) \rightarrow (a \rightarrow Z) \rightarrow Z$.

`Axiom` $isum\_def$:
$\forall\ \{a{:}\texttt{Type}\}\ \{a\_WT{:}WhyType\ a\}$,
$\forall\ (s{:}\texttt{set}\ a)\ (t1{:}a \rightarrow Z)$,
$((isum\ s\ t1) = (iterate\ (\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 + y1)\%Z)\ s\ t1))$.

`Axiom` $isum\_iter$: $iterates\ (\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 + y1)\%Z)\ 0\%Z$.

`Axiom` $isum\_iter\_\_$: $iterable\ (\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 + y1)\%Z)$.

`Axiom` $isum\_iter\_$:
$(opposite\_n\ (\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 + y1)\%Z)$
$(\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 - y1)\%Z)\ 0\%Z) \wedge$
$((opposite\ (\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 + y1)\%Z)$
$(\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 - y1)\%Z)) \wedge$
$((opposite\_com\ (\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 + y1)\%Z)$
$(\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 - y1)\%Z)) \wedge$
$(inverse\_tuple\ (\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 + y1)\%Z)$
$(\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 - y1)\%Z)\ 0\%Z)))$.

`Axiom` $neutral\_zero1$:
$((neutral\_elt\ (\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0 + y1)\%Z)) = 0\%Z)$.

`Parameter` ind_isum: $(Z \rightarrow Z) \rightarrow Z \rightarrow Z \rightarrow Z$.

`Axiom` ind_isum_def :
$\forall\ (f{:}Z \rightarrow Z)\ (i{:}Z)\ (j{:}Z)$,

$((ind\_isum\ f\ \ i\ j) = (int\_iterate\ (\texttt{fun}\ (y0{:}Z)\ (y1{:}Z) \Rightarrow (y0\ +\ y1)\%Z)\ f\ \ i\ j)).$

`Parameter` m: $Z \rightarrow Z \rightarrow t$.

`Parameter` n: $Z \rightarrow Z \rightarrow t$.

`Axiom` H : $((columns\ \ m) = (rows\ \ n))$.

`Parameter` o: $Z \rightarrow Z \rightarrow t$.

`Axiom` o_def :

$$
\begin{array}{l}
(o = \\
\quad (\texttt{fun}\ (i{:}Z)\ (j{:}Z) \Rightarrow \\
\quad\quad (int\_iterate\ (\texttt{fun}\ (y0{:}t)\ (y1{:}t) \Rightarrow (infix\_pldt\ \ y0\ \ y1)) \\
\quad\quad\quad (\texttt{fun}\ (k{:}Z) \Rightarrow (infix\_pldt\ ((m\ \ i)\ \ k)\ ((n\ \ k)\ \ j)))\ 0\%Z\ (columns\ \ m)))).
\end{array}
$$

`Parameter` $o1$: Z.

`Axiom` o_def1 : $(o1\ = (columns\ \ n))$.

`Axiom` $H1$: $(0\%Z\ <\ o1)\%Z$.

`Parameter` $o2$: Z.

`Axiom` o_def2 : $(o2\ = (rows\ \ m))$.

`Axiom` $H2$: $(0\%Z\ <\ o2)\%Z$.

`Parameter` $result4$: $Z \rightarrow Z \rightarrow t$.

`Axiom` $result\_def4$: $(result4\ = (make\_f\ \ o2\ \ o1\ \ o))$.

`Axiom` $H3$: $((rows\ \ result4) = o2)$.

`Axiom` $H4$: $((columns\ \ result4) = o1)$.

`Axiom` $H5$:

$$
\begin{array}{l}
\forall\ (i{:}Z)\ (j{:}Z),\ (valid\_index\ \ result4\ \ i\ j) \rightarrow \\
(((result4\ \ i)\ j) = \\
\quad (((\texttt{fun}\ (i1{:}Z)\ (j1{:}Z) \Rightarrow \\
\quad\quad\quad (int\_iterate\ (\texttt{fun}\ (y0{:}t)\ (y1{:}t) \Rightarrow (infix\_pldt\ \ y0\ \ y1)) \\
\quad\quad\quad\quad (\texttt{fun}\ (k{:}Z) \Rightarrow (infix\_pldt\ ((m\ \ i1)\ \ k)\ ((n\ \ k)\ \ j1)))\ 0\%Z\ (columns\ \ m))) \\
\quad\quad i) \\
\quad j)).
\end{array}
$$

`Axiom` $H6$: $((rows\ \ result4) = (rows\ \ m))$.

`Axiom` $H7$: $((columns\ \ result4) = (columns\ \ n))$.

`Parameter` i: Z.

`Parameter` j: Z.

`Axiom` $H8$: $valid\_index\ \ result4\ \ i\ \ j$.

`Axiom` $Hinst$:

$$
\begin{array}{l}
(((result4\ \ i)\ j) = \\
\quad (((\texttt{fun}\ (i1{:}Z)\ (j1{:}Z) \Rightarrow
\end{array}
$$

```
        (int_iterate (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1))
          (fun (k:Z) ⇒ (infix_pldt ((m i1) k) ((n k) j1))) 0%Z (columns m)))
      i)
     j)).

Theorem h :
  ((((fun (i1:Z) (j1:Z) ⇒
        (int_iterate (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1))
          (fun (k:Z) ⇒ (infix_pldt ((m i1) k) ((n k) j1))) 0%Z (columns m)))
      i)
     j)
   =
   (int_iterate (fun (y0:t) (y1:t) ⇒ (infix_pldt y0 y1))
    (fun (k:Z) ⇒ (infix_pldt ((m i) k) ((n k) j))) 0%Z (columns m))).
```