

Tensor Product of Matrices

T.V.H. Prathamesh

February 6, 2026

Abstract

In this work, the Kronecker tensor product of matrices and the proofs of some of its properties are formalized. Properties which have been formalized include associativity of the tensor product and the mixed-product property. This formalization of tensor product of matrices relies on the formalization of matrices by Christian Sternagel and Rene Thiemann under the title ‘Executable Matrix Operations on Matrices of Arbitrary Dimensions’.

Contents

1	Tensor Product of Matrices	1
1.1	Defining the Tensor Product	1
1.2	Associativity and Distributive properties	59

We define Tensor Product of Matrices and prove properties such as associativity and mixed product property(distributivity) of the tensor product.

1 Tensor Product of Matrices

```
theory Matrix-Tensor
imports Matrix.Utility Matrix.Matrix-Legacy
begin
```

1.1 Defining the Tensor Product

We define a multiplicative locale here - *mult*, where the multiplication satisfies commutativity, associativity and contains a left and right identity

```
locale mult =
  fixes id::'a
  fixes f:: 'a ⇒ 'a ⇒ 'a (infixl ⟨*⟩ 60)
  assumes comm: f a b = f b a
  assumes assoc: (f (f a b) c) = (f a (f b c))
  assumes left-id: f id x = x
  assumes right-id: f x id = x
```

context *mult*

begin

$\text{times } a \ v$, gives us the product of the vector v with multiplied pointwise with a

primrec *times*:: 'a \Rightarrow 'a *vec* \Rightarrow 'a *vec*

where

times $n \ [] = []$

times $n \ (y\#ys) = (f \ n \ y)\#(\text{times } n \ ys)$

lemma *times-scalar-id*: *times id v = v*

by(*induction v*)(*auto simp add:left-id*)

lemma *times-vector-id*: *times v [id] = [v]*

by(*simp add:right-id*)

lemma *preserving-length*: *length (times n y) = (length y)*

by(*induction y*)(*auto*)

vec_vec_Tensor is the tensor product of two vectors. It is illustrated by the following relation

$\text{vec_vec_Tensor}(v_1, v_2, \dots, v_n)(w_1, w_2, \dots, w_m) = (v_1 \cdot w_1, \dots, v_1 \cdot w_m, \dots, v_n \cdot w_1, \dots, v_n \cdot w_m)$

primrec *vec-vec-Tensor*:: 'a *vec* \Rightarrow 'a *vec* \Rightarrow 'a *vec*

where

vec-vec-Tensor $[] \ ys = []$

vec-vec-Tensor $(x\#xs) \ ys = (\text{times } x \ ys)\@(\text{vec-vec-Tensor } xs \ ys)$

lemma *vec-vec-Tensor-left-id*: *vec-vec-Tensor [id] v = v*

by(*induction v*)(*auto simp add:left-id*)

lemma *vec-vec-Tensor-right-id*: *vec-vec-Tensor v [id] = v*

by(*induction v*)(*auto simp add:right-id*)

theorem *vec-vec-Tensor-length* :

$(\text{length}(\text{vec-vec-Tensor } x \ y)) = (\text{length } x) * (\text{length } y)$

by(*induction x*)(*auto simp add: preserving-length*)

theorem *vec-length*: **assumes** *vec m x* **and** *vec n y*

shows *vec (m*n) (vec-vec-Tensor x y)*

apply(*simp add:vec-def*)

apply(*simp add:vec-vec-Tensor-length*)

apply (*metis assms(1) assms(2) vec-def*)

done

vec_mat_Tensor is the tensor product of two vectors. It is illustrated by the following relation

$\text{vec_mat_Tensor } (v_1, v_2, \dots, v_n)(C_1, C_2, \dots, C_m) = (v_1 \cdot C_1, \dots, v_n \cdot C_1, \dots, v_1 \cdot C_m, \dots, v_n \cdot C_m)$

primrec *vec-mat-Tensor*:: 'a vec \Rightarrow 'a mat \Rightarrow 'a mat

where

vec-mat-Tensor $\square = \square$

vec-mat-Tensor $xs (ys\#yss) = (\text{vec-vec-Tensor } xs \ ys)\#(\text{vec-mat-Tensor } xs \ yss)$

lemma *vec-mat-Tensor-vector-id*: *vec-mat-Tensor* [id] $v = v$

by(*induction v*)(*auto simp add: times-scalar-id*)

lemma *vec-mat-Tensor-matrix-id*: *vec-mat-Tensor* $v [[id]] = [v]$

by(*induction v*)(*auto simp add: right-id*)

theorem *vec-mat-Tensor-length*:

length(*vec-mat-Tensor* $xs \ ys$) = *length* ys

by(*induction ys*)(*auto*)

theorem *length-matrix*:

assumes *mat nr nc* ($y\#ys$) **and** *length v = k*

and (*vec-mat-Tensor* $v (y\#ys) = x\#xs$)

shows (*vec* ($nr*k$) x)

proof –

have *vec-mat-Tensor* $v (y\#ys) = (\text{vec-vec-Tensor } v \ y)\#(\text{vec-mat-Tensor } v \ ys)$

using *vec-mat-Tensor-def* **assms** **by** *auto*

also have (*vec-vec-Tensor* $v \ y$) = x **using** *assms* **by** *auto*

also have *length y = nr*

using *assms* **by** (*simp add: mat-def vec-def*)

from *this*

have *length* (*vec-vec-Tensor* $v \ y$) = $nr*k$

using *assms* *vec-vec-Tensor-length* **by** *auto*

from *this*

have *length x = nr*k* **by** (*simp add: <vec-vec-Tensor v y = x>*)

from *this*

have *vec* ($nr*k$) x **using** *vec-def* **by** *auto*

from *this*

show *?thesis* **by** *auto*

qed

lemma *matrix-set-list*:

assumes *mat nr nc M*

and *length v = k*

and $x \in \text{set } M$

shows $\exists ys. \exists zs. (ys@x\#zs = M)$

using *assms* *set-def in-set-conv-decomp* **by** *metis*

primrec *reduct* :: 'a mat \Rightarrow 'a mat

where

reduct $\square = \square$

| $reduct (x\#xs) = xs$

lemma *length-reduct*:

assumes $m \neq []$

shows $length (reduct m) + 1 = (length m)$

apply(*auto*)

by (*metis One-nat-def Suc-eq-plus1 assms list.size(4) neq-Nil-conv reduct.simps(2)*)

lemma *mat-empty-column-length*: **assumes** $mat\ nr\ nc\ M$ **and** $M = []$

shows $nc = 0$

proof –

have ($length\ M = nc$) **using** *mat-def assms* **by** *metis*

from *this*

have $nc = 0$ **using** *assms* **by** *auto*

from *this*

show *?thesis* **by** *simp*

qed

lemma *vec-uniqueness*:

assumes $vec\ m\ v$

and $vec\ n\ v$

shows $m = n$

using *vec-def assms(1) assms(2)* **by** *metis*

lemma *mat-uniqueness*:

assumes $mat\ nr1\ nc\ M$

and $mat\ nr2\ nc\ M$ **and** $z = hd\ M$ **and** $M \neq []$

shows $(\forall x \in (set\ M). (nr1 = nr2))$

proof –

have $A: z \in set\ M$ **using** *assms(1) assms(3) assms(4) set-def mat-def*

by (*metis hd-in-set*)

have $Ball (set\ M) (vec\ nr1)$ **using** *mat-def assms(1)* **by** *auto*

then have *step1*: $((x \in (set\ M)) \longrightarrow (vec\ nr1\ x))$ **using** *Ball-def assms* **by** *auto*

have $Ball (set\ M) (vec\ nr2)$ **using** *mat-def assms(2)* **by** *auto*

then have *step2*: $((x \in (set\ M)) \longrightarrow (vec\ nr2\ x))$ **using** *Ball-def assms* **by** *auto*
from *step1 and step2*

have *step3*: $\forall x. ((x \in (set\ M)) \longrightarrow ((vec\ nr1\ x) \wedge (vec\ nr2\ x)))$

by (*metis* $\langle Ball (set\ M) (vec\ nr1) \rangle \langle Ball (set\ M) (vec\ nr2) \rangle$)

have $((vec\ nr1\ x) \wedge (vec\ nr2\ x)) \longrightarrow (nr1 = nr2)$ **using** *vec-uniqueness* **by** *auto*
with *step3*

have $(\forall x. ((x \in (set\ M)) \longrightarrow ((nr1 = nr2))))$ **by** (*metis vec-uniqueness*)

then

have $(\forall x \in (set\ M). (nr1 = nr2))$ **by** *auto*

then

show *?thesis* **by** *auto*

qed

lemma *mat-empty-row-length*: **assumes** $mat\ nr\ nc\ M$ **and** $M = []$

shows $mat\ 0\ nc\ M$

proof –

have $set\ M = \{\}$ **using** $mat-def\ assms\ empty-set$ **by** $auto$
then have $Ball\ (set\ M)\ (vec\ 0)$ **using** $Ball-def$ **by** $auto$
then have $mat\ 0\ nc\ M$
using $assms(1)\ assms(2)$ **by** $(simp\ add:\ mat-def)$
then show $?thesis$ **by** $auto$
qed

abbreviation $null-matrix::'a\ list\ list$

where

$null-matrix \equiv [Nil]$

lemma $null-mat:null-matrix = [[]]$

by $auto$

lemma $zero-matrix: mat\ 0\ 0\ []$ **using** $mat-def\ in-set-insert\ insert-Nil\ list.size(3)$
 $not-Cons-self2$

by $(metis\ (full-types))$

row_length gives the length of the first row of a matrix. For a ‘valid’ matrix, it is equal to the number of rows

definition $row-length::'a\ mat \Rightarrow nat$

where

$row-length\ xs \equiv if\ (xs = [])\ then\ 0\ else\ (length\ (hd\ xs))$

lemma $row-length-Nil:$

$row-length\ [] = 0$

using $row-length-def$ **by** $(metis)$

lemma $row-length-Null:$

$row-length\ [[]] = 0$

using $row-length-def$ **by** $auto$

lemma $row-length-vect-mat:$

$row-length\ (vec-mat-Tensor\ v\ m) = length\ v*(row-length\ m)$

proof($induct\ m$)

case Nil

have $row-length\ [] = 0$

using $row-length-Nil$ **by** $simp$

moreover have $vec-mat-Tensor\ v\ [] = []$

using $vec-mat-Tensor.simps(1)$ **by** $auto$

ultimately have

$row-length\ (vec-mat-Tensor\ v\ []) = length\ v*(row-length\ [])$

using $mult-0-right$ **by** $(metis)$

then show $?case$ **by** $metis$

next

fix $a\ m$

assume $A:row-length\ (vec-mat-Tensor\ v\ m) = length\ v * row-length\ m$

```

let ?case =
  row-length (vec-mat-Tensor v (a#m)) = (length v)*(row-length (a#m))
have A:row-length (a # m) = length a
  using row-length-def list.distinct(1)
  by auto
have (vec-mat-Tensor v (a#m)) = (vec-vec-Tensor v a)#(vec-mat-Tensor v m)

  using vec-mat-Tensor-def vec-mat-Tensor.simps(2)
  by auto
from this have
  row-length (vec-mat-Tensor v (a#m)) = length (vec-vec-Tensor v a)
  using row-length-def list.distinct(1) vec-mat-Tensor.simps(2)
  by auto
from this and vec-vec-Tensor-length have
  row-length (vec-mat-Tensor v (a#m)) = (length v)*(length a)
  by auto
from this and A have
  row-length (vec-mat-Tensor v (a#m)) = (length v)*(row-length (a#m))
  by auto
from this show ?case by auto
qed

```

Tensor is the tensor product of matrices

```

primrec Tensor:: 'a mat  $\Rightarrow$  'a mat  $\Rightarrow$  'a mat (infixl  $\langle \otimes \rangle$  63)
where
  Tensor [] xs = []
  Tensor (x#xs) ys = (vec-mat-Tensor x ys)@(Tensor xs ys)

```

```

lemma Tensor-null: xs  $\otimes$  [] = []
by(induction xs)(auto)

```

Tensor commutes with left and right identity

```

lemma Tensor-left-id: [[id]]  $\otimes$  xs = xs
by(induction xs)(auto simp add:times-scalar-id)

```

```

lemma Tensor-right-id: xs  $\otimes$  [[id]] = xs
by(induction xs)(auto simp add:vec-vec-Tensor-right-id)

```

row_length of tensor product of matrices is the product of their respective row lengths

```

lemma row-length-mat:
  (row-length (m1  $\otimes$  m2)) = (row-length m1)*(row-length m2)
proof(induct m1)
case Nil
  have row-length ([]  $\otimes$  m2) = 0
  using Tensor.simps(1) row-length-def
  by metis
from this
  have row-length ([]  $\otimes$  m2) = (row-length [])*(row-length m2)

```

```

    using row-length-Nil
    by auto
  then show ?case by metis
next
fix a m1
assume row-length (m1  $\otimes$  m2) = row-length m1 * row-length m2
let ?case =
  row-length ((a # m1)  $\otimes$  m2) = row-length (a # m1) * row-length m2
have B: row-length (a#m1) = length a
  using row-length-def list.distinct(1)
  by auto
have row-length ((a # m1)  $\otimes$  m2) = row-length (a # m1) * row-length m2
proof(induct m2)
case Nil
  show ?case using Tensor-null row-length-def mult-0-right by (metis)
next
fix aa m2
assume row-length (a # m1  $\otimes$  m2) = row-length (a # m1) * row-length m2
let ?case=
  row-length (a # m1  $\otimes$  aa # m2)
    = row-length (a # m1) * row-length (aa # m2)
have aa#m2  $\neq$  []
  by auto
from this have non-zero:(vec-mat-Tensor a (aa#m2))  $\neq$  []
  using vec-mat-Tensor-def by auto
from this have
  hd ((vec-mat-Tensor a (aa#m2))@(m1 $\otimes$ m2))
    = hd (vec-mat-Tensor a (aa#m2))
  by auto
from this have
  hd ((a#m1) $\otimes$ (aa#m2)) = hd (vec-mat-Tensor a (aa#m2))
  using Tensor.simps(2) by auto
from this have s1: row-length ((a#m1) $\otimes$ (aa#m2))
  = row-length (vec-mat-Tensor a (aa#m2))
  using row-length-def Nil-is-append-conv non-zero Tensor.simps(2)
  by auto
have row-length (vec-mat-Tensor a (aa#m2))
  = (length a)*row-length(aa#m2)
  using row-length-vect-mat by metis
from this and s1
have row-length (vec-mat-Tensor a (aa#m2))
  = (length a)*row-length(aa#m2)
  by auto
from this and B
  have row-length (vec-mat-Tensor a (aa#m2))
    = (row-length (a#m1))*row-length(aa#m2)
  by auto
from this and s1 show ?case by auto
qed

```

from *this* **show** *?case* **by** *auto*
qed

lemma *hd-set:assms* $x \in \text{set } (a\#M)$ **shows** $(x = a) \vee (x \in (\text{set } M))$
using *set-def assms set-ConsD* **by** *auto*

for every valid matrix can also be written in the following form

theorem *matrix-row-length*:
assumes *mat nr nc M*
shows *mat (row-length M) (length M) M*
proof(*cases M*)
case *Nil*
have *row-length M = 0*
using *row-length-def* **by** (*metis Nil*)
moreover **have** *length M = 0*
by (*metis Nil list.size(3)*)
moreover **have** *mat 0 0 M*
using *zero-matrix Nil* **by** *auto*
ultimately **show** *?thesis*
using *mat-empty-row-length row-length-def mat-def* **by** *metis*
next
case (*Cons a N*)
have *1: mat nr nc (a\#N)*
using *assms Cons* **by** *auto*
from *this* **have** $(x \in \text{set } (a\#N)) \longrightarrow (x = a) \vee (x \in (\text{set } N))$
using *hd-set* **by** *auto*
from *this* **and** *1* **have** *2:vec nr a*
by (*simp add: mat-def*)
have *row-length (a\#N) = length a*
using *row-length-def Cons list.distinct(1)* **by** *auto*
from *this* **have** *vec (row-length (a\#N)) a*
using *vec-def* **by** *auto*
from *this* **and** *2* **have** *3:(row-length M) = nr*
using *vec-uniqueness Cons* **by** *auto*
have *nc = (length M)*
using *1 and mat-def and assms* **by** *metis*
with *3*
have *mat (row-length M) (length M) M*
using *assms* **by** *auto*
from *this* **show** *?thesis* **by** *auto*
qed

lemma *reduct-matrix*:
assumes *mat (row-length (a\#M)) (length (a\#M)) (a\#M)*
shows *mat (row-length M) (length M) M*
proof(*cases M*)
case *Nil*

```

show ?thesis
  using row-length-def zero-matrix Nil list.size(3) by (metis)
next
case (Cons b N)
fix x
have 1: b ∈ (set M)
  using set-def Cons ListMem-iff elem by auto
have mat (row-length (a#M)) (length (a#M)) (a#M)
  using assms by auto
then have (x ∈ (set (a#M))) → ((x = a) ∨ (x ∈ set M))
  by auto
then have (x ∈ (set (a#M))) → (vec (row-length (a#M)) x)
  using mat-def Ball-def assms
  by metis
then have (x ∈ (set (a#M))) → (vec (length a) x)
  using row-length-def list.distinct(1)
  by auto
then have 2:x ∈ (set M) → (vec (length a) x)
  by auto
with 1 have 3:(vec (length a) b)
  using assms by (simp add: mat-def vec-def)
have 5: (vec (length b) b)
  using vec-def by auto
with 3 have (length a) = (length b)
  using vec-uniqueness by auto
with 2 have 4: x ∈ (set M) → (vec (length b) x)
  by auto
have 6: row-length M = (length b)
  using row-length-def Cons list.distinct(1)
  by auto
with 4 have x ∈ (set M) → (vec (row-length M) x)
  by auto
then have (∀ x. (x ∈ (set M) → (vec (row-length M) x)))
  using Cons 5 6 assms by (auto simp add: mat-def dest: vec-uniqueness)
then have Ball (set M) (vec (row-length M))
  using Ball-def by auto
then have (mat (row-length M) (length M) M)
  using mat-def by auto
then show ?thesis by auto
qed

```

theorem well-defined-vec-mat-Tensor:
 $(\text{mat } (\text{row-length } M) (\text{length } M) M) \implies$
 $(\text{mat } ((\text{row-length } M) * (\text{length } v)) (\text{length } M) (\text{vec-mat-Tensor } v M))$
proof(*induct* M)

```

case Nil
have (vec-mat-Tensor v []) = []
  using vec-mat-Tensor.simps(1) Nil
  by simp
moreover have (row-length [] = 0)
  using row-length-def Nil
  by metis
moreover have (length []) = 0
  using Nil by simp
ultimately have
  mat ((row-length [])*(length v)) (length []) (vec-mat-Tensor v [])
  using zero-matrix by (metis mult-zero-left)
then show ?case by simp
next
fix a M
assume hyp :
  (mat (row-length M) (length M) M
    $\implies$  mat (row-length M * length v) (length M) (vec-mat-Tensor v M))
  mat (row-length (a#M)) (length (a#M)) (a#M)
let ?case =
  mat (row-length (a#M) * length v) (length (a#M)) (vec-mat-Tensor v (a#M))
have step1: mat (row-length M) (length M) M
  using hyp(2) reduct-matrix by auto
then have step2:
  mat (row-length M * length v) (length M) (vec-mat-Tensor v M)
  using hyp(1) by auto
have
  mat
  (row-length (a#M) * length v)
  (length (a#M))
  (vec-mat-Tensor v (a#M))
proof (cases M)
case Nil
fix x
have 1:(vec-mat-Tensor v (a#M)) = [vec-vec-Tensor v a]
  using vec-mat-Tensor.simps Nil by auto
have (x  $\in$  (set [vec-vec-Tensor v a]))  $\longrightarrow$  x = (vec-vec-Tensor v a)
  using set-def by auto
then have 2:
  (x  $\in$  (set [vec-vec-Tensor v a]))
   $\longrightarrow$  (vec (length (vec-vec-Tensor v a)) x)
  using vec-def by metis
have 3:length (vec-vec-Tensor v a) = (length v)*(length a)
  using vec-vec-Tensor-length by auto
then have 4:
  length (vec-vec-Tensor v a) = (length v)*(row-length (a#M))
  using row-length-def list.distinct(1)
  by auto
have 6: length (vec-mat-Tensor v (a#M)) = (length (a#M))

```

```

    using vec-mat-Tensor-length by auto
  hence mat (length (vec-vec-Tensor v a)) (length (a # M)) [vec-vec-Tensor v a]
    by (simp add: Nil mat-def vec-def)
  hence
    mat (row-length (a#M) * length v)
      (length (vec-mat-Tensor v (a#M)))
      (vec-mat-Tensor v (a#M))
    using 1 4 6 by (simp add: mult.commute)
  then show ?thesis using 6 by auto
next
case (Cons b L)
fix x
have 1: x ∈ (set (a#M)) → ((x=a) ∨ (x ∈ (set M)))
  using hd-set by auto
have mat (row-length (a#M)) (length (a#M)) (a#M)
  using hyp by auto
then have x ∈ (set (a#M)) → (vec (row-length (a#M)) x)
  using mat-def Ball-def by metis
then have x ∈ (set (a#M)) → (vec (length a) x)
  using row-length-def list.distinct(1)
  by auto
with 1 have x ∈ (set M) → (vec (length a) x)
  by auto
moreover have b ∈ (set M)
  using Cons by auto
ultimately have vec (length a) b
  using hyp(2) by (simp add: mat-def vec-def)
then have (length b) = (length a)
  using vec-def vec-uniqueness by auto
then have 2: row-length M = (length a)
  using row-length-def Cons list.distinct(1) by auto
have mat (row-length M * length v) (length M) (vec-mat-Tensor v M)
  using step2 by auto
then have 3:
  Ball (set (vec-mat-Tensor v M)) (vec ((row-length M)*(length v)))
  using mat-def by auto
then have (x ∈ set (vec-mat-Tensor v M))
  → (vec ((row-length M)*(length v)) x)
  using mat-def Ball-def by auto
then have 4: (x ∈ set (vec-mat-Tensor v M))
  → (vec ((length a)*(length v)) x)
  using 2 by auto
have 5: length (vec-vec-Tensor v a) = (length a)*(length v)
  using vec-vec-Tensor-length by auto
then have 6: vec ((length a)*(length v)) (vec-vec-Tensor v a)
  using vec-vec-Tensor-length vec-def by (metis (full-types))
have 7: (length a) = (row-length (a#M))
  using row-length-def list.distinct(1) by auto
have vec-mat-Tensor v (a#M)

```

```

      = (vec-vec-Tensor v a)#(vec-mat-Tensor v M)
    using vec-mat-Tensor.simps(2) by auto
  then have (x ∈ set (vec-mat-Tensor v (a#M)))
    → ((x = (vec-vec-Tensor v a))
      ∨ (x ∈ (set (vec-mat-Tensor v M))))
    using hd-set by auto
  with 4 6 have (x ∈ set (vec-mat-Tensor v (a#M)))
    → vec ((length a)*(length v)) x
    by auto
  with 7 have (x ∈ set (vec-mat-Tensor v (a#M)))
    → vec ((row-length (a#M))*(length v)) x
    by auto
  then have ∀x.((x ∈ set (vec-mat-Tensor v (a#M)))
    → vec ((row-length (a#M))*(length v)) x)
    using 2 3 6 7 hd-set vec-mat-Tensor.simps(2) by auto
  then have 7:
    Ball
      (set (vec-mat-Tensor v (a#M)))
      (vec ((row-length (a#M))*(length v)))
    using Ball-def by auto
  have 8: length (vec-mat-Tensor v (a#M)) = length (a#M)
    using vec-mat-Tensor-length by auto
  with 6 7 have
    mat
      ((row-length (a#M))*(length v))
      (length (a#M))
      (vec-mat-Tensor v (a#M))
    using mat-def 5
    by (simp add: mat-def)
  then show ?thesis by auto
  qed
  with hyp show ?case by auto
qed

```

The following theorem gives length of tensor product of two matrices

lemma *length-Tensor*: $(\text{length } (M1 \otimes M2)) = (\text{length } M1) * (\text{length } M2)$

proof(*induct M1*)

case Nil

show ?case by auto

next

case (Cons a M1)

have $((a \# M1) \otimes M2) = (\text{vec-mat-Tensor } a M2) @ (M1 \otimes M2)$

using *Tensor.simps(2)* by auto

then have 1:

$\text{length } ((a \# M1) \otimes M2) = \text{length } ((\text{vec-mat-Tensor } a M2) @ (M1 \otimes M2))$

by auto

have 2: $\text{length } ((\text{vec-mat-Tensor } a M2) @ (M1 \otimes M2))$

$= \text{length } (\text{vec-mat-Tensor } a M2) + \text{length } (M1 \otimes M2)$

using *append-def*

```

      by auto
    have 3:(length (vec-mat-Tensor a M2)) = length M2
      using vec-mat-Tensor-length by (auto)
    have 4:length (M1 ⊗ M2) = (length M1)*(length M2)
      using Cons.hyps by auto
    with 2 3 have length ((vec-mat-Tensor a M2)@(M1 ⊗ M2))
      = (length M2) + (length M1)*(length M2)
      by auto
    then have 5:
      length ((vec-mat-Tensor a M2)@(M1 ⊗ M2)) = (1 + (length M1))*(length
M2)
      by auto
    with 1 have length ((a # M1) ⊗ M2) = ((length (a # M1)) * (length M2))
      by auto
    then show ?case by auto
  qed

```

lemma *append-reduct-matrix*:

```

(mat (row-length (M1 @ M2)) (length (M1 @ M2)) (M1 @ M2))
⇒(mat (row-length M2) (length M2) M2)

```

proof(*induct M1*)

case *Nil*

show ?thesis **using** *Nil append.simps(1)* **by** *auto*

next

case (*Cons a M1*)

have *mat (row-length (M1 @ M2)) (length (M1 @ M2)) (M1 @ M2)*

using *reduct-matrix Cons.premis append-Cons* **by** *metis*

from *this* **have** *(mat (row-length M2) (length M2) M2)*

using *Cons.hyps* **by** *auto*

from *this* **show** ?thesis **by** *simp*

qed

The following theorem proves that tensor product of two valid matrices is a valid matrix

theorem *well-defined-Tensor*:

```

(mat (row-length M1) (length M1) M1)
∧ (mat (row-length M2) (length M2) M2)
⇒(mat ((row-length M1)*(row-length M2)) ((length M1)*(length M2)) (M1 ⊗ M2))

```

proof(*induct M1*)

case *Nil*

have *(row-length []) * (row-length M2) = 0*

using *row-length-def mult-zero-left* **by** (*metis*)

moreover **have** *(length []) * (length M2) = 0*

using *mult-zero-left list.size(3)* **by** *auto*

moreover **have** *[] ⊗ M2 = []*

using *Tensor.simps(1)* **by** *auto*

ultimately **have**

```

      mat (row-length []*row-length M2) (length []*length M2) ([]  $\otimes$  M2)
      using zero-matrix by metis
    then show ?case by simp
  next
  case (Cons a M1)
  have step1: mat (row-length (a # M1)) (length (a # M1)) (a # M1)
    using Cons.premis by auto
  then have mat (row-length (M1)) (length (M1)) (M1)
    using reduct-matrix by auto
  moreover have mat (row-length (M2)) (length (M2)) (M2)
    using Cons.premis by auto
  ultimately have step2:
    mat (row-length M1 * row-length M2) (length M1 * length M2) (M1  $\otimes$  M2)
    using Cons.hyps by auto
  have 0:row-length (a#M1) = length a
    using row-length-def list.distinct(1) by auto
  have mat
    (row-length (a # M1)*row-length M2)
    (length (a # M1)*length M2)
    (a # M1  $\otimes$  M2)
  proof(cases M1)
  case Nil
  have (mat ((row-length M2)*(length a)) (length M2) (vec-mat-Tensor a M2))
    using Cons.premis well-defined-vec-mat-Tensor by auto
  moreover have (length (a # M1)) * (length M2) = length M2
    using Nil by auto
  moreover have (a#M1) $\otimes$ M2 = (vec-mat-Tensor a M2)
    using Nil Tensor.simps append.simps(1) by auto
  ultimately have
    (mat
      ((row-length M2)*(row-length (a#M1)))
      ((length (a # M1)) * (length M2))
      ((a#M1) $\otimes$ M2))
    using 0 by auto
  then show ?thesis by (simp add: mult.commute)
  next
  case (Cons b N1)
  fix x
  have 1:x  $\in$  (set (a#M1))  $\longrightarrow$  ((x=a)  $\vee$  (x  $\in$  (set M1)))
    using hd-set by auto
  have mat (row-length (a#M1)) (length (a#M1)) (a#M1)
    using Cons.premis by auto
  then have x  $\in$  (set (a#M1))  $\longrightarrow$  (vec (row-length (a#M1)) x)
    using mat-def Ball-def by metis
  then have x  $\in$  (set (a#M1)) $\longrightarrow$  (vec (length a) x)
    using row-length-def list.distinct(1)
    by auto
  with 1 have x  $\in$  (set M1) $\longrightarrow$  (vec (length a) x)
    by auto

```

moreover have $b \in (\text{set } M1)$
using *Cons* **by** *auto*
ultimately have $\text{vec } (\text{length } a) b$
using *Cons.prem*s **by** (*simp add: vec-def mat-def*)
then have $(\text{length } b) = (\text{length } a)$
using *vec-def vec-uniqueness* **by** *auto*
then have $2:\text{row-length } M1 = (\text{length } a)$
using *row-length-def Cons* **by** *auto*
then have *mat*
 $((\text{length } a) * \text{row-length } M2)$
 $(\text{length } M1 * \text{length } M2)$
 $(M1 \otimes M2)$
using *step2* **by** *auto*
then have *Ball* (*set* $(M1 \otimes M2)$) (*vec* $((\text{length } a) * (\text{row-length } M2))$)
using *mat-def* **by** *auto*
from this have 3:
 $\forall x. x \in (\text{set } (M1 \otimes M2)) \longrightarrow (\text{vec } ((\text{length } a) * (\text{row-length } M2))) x$
using *Ball-def* **by** *auto*
have *mat*
 $((\text{row-length } M2) * (\text{length } a))$
 $(\text{length } M2)$
 $(\text{vec-mat-Tensor } a M2)$
using *well-defined-vec-mat-Tensor Cons.prem*s
by *auto*
then have *Ball*
 $(\text{set } (\text{vec-mat-Tensor } a M2))$
 $(\text{vec } ((\text{row-length } M2) * (\text{length } a)))$
using *mat-def*
by *auto*
then have 4:
 $\forall x. x \in (\text{set } (\text{vec-mat-Tensor } a M2))$
 $\longrightarrow (\text{vec } ((\text{length } a) * (\text{row-length } M2))) x$
using *mult.commute* **by** *metis*

with 3 **have** 5: $\forall x. (x \in (\text{set } (\text{vec-mat-Tensor } a M2)))$
 $\vee (x \in (\text{set } (M1 \otimes M2)))$
 $\longrightarrow (\text{vec } ((\text{length } a) * (\text{row-length } M2))) x$
by *auto*
have 6: $(a \# M1 \otimes M2) = (\text{vec-mat-Tensor } a M2) @ (M1 \otimes M2)$
using *Tensor.simps(2)* **by** *auto*
then have $x \in (\text{set } (a \# M1 \otimes M2))$
 $\longrightarrow (x \in (\text{set } (\text{vec-mat-Tensor } a M2))) \vee (x \in (\text{set } (M1 \otimes M2)))$
using *set-def append-def* **by** *auto*
with 5 **have** 7: $\forall x. (x \in (\text{set } (a \# M1 \otimes M2)))$
 $\longrightarrow (\text{vec } ((\text{length } a) * (\text{row-length } M2))) x$
by *auto*
then have 8:
 $\text{Ball } (\text{set } (a \# M1 \otimes M2)) (\text{vec } ((\text{row-length } (a \# M1)) * (\text{row-length } M2)))$
using *Ball-def 0* **by** *auto*

```

have (length ((a#M1)⊗M2)) = (length (a#M1))*(length M2)
  using length-Tensor by metis
with 7 8
  have mat
    (row-length (a # M1) * row-length M2)
    (length (a # M1) * length M2)
    (a # M1 ⊗ M2)
    using mat-def by (metis 0 length-Tensor)
  then show ?thesis by auto
qed
then show ?case by auto
qed

```

```

theorem effective-well-defined-Tensor:
assumes (mat (row-length M1) (length M1) M1)
  and (mat (row-length M2) (length M2) M2)
shows mat
  ((row-length M1)*(row-length M2))
  ((length M1)*(length M2))
  (M1⊗M2)
using well-defined-Tensor assms by auto

```

```

definition natmod::nat ⇒ nat ⇒ nat (infixl ⟨nmod⟩ 50)
where
  natmod x y = nat ((int x) mod (int y))

```

```

theorem times-elements:
  ∀ i. ((i < (length v)) → (times a v)!i = f a (v!i))
apply (rule allI)
proof (induct v)
case Nil
  have (length [] = 0)
    by auto
  then have i < (length []) ⇒ False
    by auto
  moreover have (times a []) = []
    using times.simps(1) by auto
  ultimately have (i < (length [])) → (times a [])!i = f a ([]!i)
    by auto
  then have ∀ i. ((i < (length [])) → (times a [])!i = f a ([]!i))
    by auto
  then show ?case by auto
next
case (Cons x xs)
  have ∀ i. ((x#xs)!i+1 = (xs)!i)
    by auto
  have 0:((i < length (x#xs)) → ((i < (length xs)) ∨ (i = (length xs))))
    by auto

```

```

have 1: ((i < length xs) → ((times a xs)!i = f a (xs!i)))
  by (metis Cons.hyps)
have ∀ i. ((x#xs)!(i+1) = (xs)!i) by auto
have ((i < length (x#xs)) → (times a (x#xs))!i = f a ((x#xs)!i))
proof(cases i)
  case 0
    have ((times a (x#xs))!i = f a x
      using 0 times.simps(2) by auto
    then have (times a (x#xs))!i = f a ((x#xs)!i)
      using 0 by auto
    then show ?thesis by auto
  next
  case (Suc j)
    have 1:(times a (x#xs))!i = ((f a x)#(times a xs))!i
      using times.simps(2) by auto
    have 2:((f a x)#(times a xs))!i = (times a xs)!j
      using Suc by auto
    have 3:(i < length (x#xs)) → (j < length xs)
      using One-nat-def Suc Suc-eq-plus1 list.size(4) not-less-eq
      by metis
    have 4:(j < length xs) → ((times a xs)!j = (f a (xs!j)))
      using 1 by (metis Cons.hyps)
    have 5:(x#xs)!i = (xs!j)
      using Suc by (metis nth-Cons-Suc)
    with 1 2 4 have (j < length xs)
      → ((times a (x#xs))!i = (f a ((x#xs)!i)))
      by auto
    with 3 have (i < length (x#xs))
      → ((times a (x#xs))!i = (f a ((x#xs)!i)))
      by auto
    then show ?thesis by auto
  qed
then show ?case by auto
qed

```

lemma *simpl-times-elements*:

```

assumes (i < length xs)
shows ((i < (length v)) → (times a v)!i = f a (v!i))
using times-elements by auto

```

lemma *append-simpl*: $i < (\text{length } xs) \longrightarrow (xs@ys)!i = (xs!i)$
using nth-append **by** metis

lemma *append-simpl2*: $i \geq (\text{length } xs) \longrightarrow (xs@ys)!i = (ys!(i - (\text{length } xs)))$
using nth-append less-asm leD **by** metis

lemma *append-simpl3*:
assumes $i > (\text{length } y)$

shows $(i < ((\text{length } (z\#zs)) * (\text{length } y)))$
 $\longrightarrow (i - (\text{length } y)) < (\text{length } zs) * (\text{length } y)$
proof –
have $\text{length } (z\#zs) = (\text{length } zs) + 1$
by *auto*
then have $i < ((\text{length } (z\#zs)) * (\text{length } y))$
 $\longrightarrow i < ((\text{length } zs) + 1) * (\text{length } y)$
by *auto*
then have $1: i < ((\text{length } (z\#zs)) * (\text{length } y))$
 $\longrightarrow (i < ((\text{length } zs) * (\text{length } y) + (\text{length } y)))$
by *auto*
have $i < ((\text{length } zs) * (\text{length } y) + (\text{length } y))$
 $= ((i - (\text{length } y)) < ((\text{length } zs) * (\text{length } y)))$
using *assms* **by** *auto*
then have $(i < ((\text{length } (z\#zs)) * (\text{length } y)))$
 $\longrightarrow ((i - (\text{length } y)) < ((\text{length } zs) * (\text{length } y)))$
by *auto*
then show *?thesis* **by** *auto*
qed

lemma *append-simpl4*:
 $(i > (\text{length } y))$
 $\longrightarrow ((i < ((\text{length } (z\#zs)) * (\text{length } y))))$
 $\longrightarrow ((i - (\text{length } y)) < (\text{length } zs) * (\text{length } y))$
using *append-simpl3* **by** *auto*

lemma *vec-vec-Tensor-simpl*:
 $i < (\text{length } y) \longrightarrow (\text{vec-vec-Tensor } (z\#zs) y)!i = (\text{times } z y)!i$
proof –
have $a: \text{vec-vec-Tensor } (z\#zs) y = (\text{times } z y) @ (\text{vec-vec-Tensor } zs y)$
by *auto*
have $b: \text{length } (\text{times } z y) = (\text{length } y)$ **using** *preserving-length* **by** *auto*
have $i < (\text{length } (\text{times } z y))$
 $\longrightarrow ((\text{times } z y) @ (\text{vec-vec-Tensor } zs y))!i = (\text{times } z y)!i$
using *append-simpl* **by** *metis*
with b **have** $i < (\text{length } y)$
 $\longrightarrow ((\text{times } z y) @ (\text{vec-vec-Tensor } zs y))!i = (\text{times } z y)!i$
by *auto*
with a **have** $i < (\text{length } y)$
 $\longrightarrow (\text{vec-vec-Tensor } (z\#zs) y)!i = (\text{times } z y)!i$
by *auto*
then show *?thesis* **by** *auto*
qed

lemma *vec-vec-Tensor-simpl2*:
 $(i \geq (\text{length } y))$
 $\longrightarrow ((\text{vec-vec-Tensor } (z\#zs) y)!i = (\text{vec-vec-Tensor } zs y)!(i - (\text{length } y)))$
using *vec-vec-Tensor.simps(2)* *append-simpl2* *preserving-length*

by *metis*

lemma *division-product*:

assumes $(b::int) > 0$

and $a \geq b$

shows $(a \text{ div } b) = ((a - b) \text{ div } b) + 1$

proof –

fix c

have $a - b \geq 0$

using *assms(2)* **by** *auto*

have $1: a - b = a + (-1)*b$

by *auto*

have $(b \neq 0) \longrightarrow ((a + b * (-1)) \text{ div } b = (-1) + a \text{ div } b)$

using *div-mult-self2* **by** *metis*

with 1 **assms(1)** **have** $((a - b) \text{ div } b) = (-1) + a \text{ div } b$

using *less-int-code(1)* **by** *auto*

then **have** $(a \text{ div } b) = ((a - b) \text{ div } b) + 1$

by *auto*

then **show** *?thesis*

by *auto*

qed

lemma *int-nat-div*:

$(\text{int } a) \text{ div } (\text{int } b) = \text{int } ((a::\text{nat}) \text{ div } b)$

by (*metis zdiv-int*)

lemma *int-nat-eq*:

assumes $\text{int } (a::\text{nat}) = \text{int } b$

shows $a = b$

using *assms of-nat-eq-iff* **by** *auto*

lemma *nat-div*:

assumes $(b::\text{nat}) > 0$

and $a > b$

shows $(a \text{ div } b) = ((a - b) \text{ div } b) + 1$

proof –

fix x

have $1:(\text{int } b) > 0$

using *assms(1) division-product* **by** *auto*

moreover **have** $(\text{int } a) > (\text{int } b)$

using *assms(2)* **by** *auto*

with 1 **have** $2: ((\text{int } a) \text{ div } (\text{int } b))$

$= (((\text{int } a) - (\text{int } b)) \text{ div } (\text{int } b)) + 1$

using *division-product* **by** *auto*

from *int-nat-div* **have** $3: ((\text{int } a) \text{ div } (\text{int } b)) = \text{int } (a \text{ div } b)$

by *auto*

from *int-nat-div* *assms(2)* **have** $4:$

$((\text{int } a) - (\text{int } b)) \text{ div } (\text{int } b) = \text{int } ((a - b) \text{ div } b)$

by (*metis (full-types) less-asm not-less of-nat-diff*)

```

have (int x) + 1 = int (x + 1)
  by auto
with 2 3 4 have int (a div b) = int (((a - b) div b) + 1)
  by auto
with int-nat-eq have (a div b) = ((a - b) div b) + 1
  by auto
then show ?thesis by auto
qed

```

```

lemma mod-eq:
  (m::int) mod n = (m + (-1)*n) mod n
  using mod-mult-self1 by metis

```

```

lemma nat-mod-eq: int m mod int n = int (m mod n)
  by (simp add: of-nat-mod)

```

```

lemma nat-mod:
  assumes (m::nat) > n
  shows (m::nat) mod n = (m - n) mod n
  using assms mod-if not-less-iff-gr-or-eq by auto

```

```

lemma logic:
  assumes A → B
  and ¬A → B
  shows B
  using assms(1) assms(2) by auto

```

```

theorem vec-vec-Tensor-elements:
  assumes (y ≠ [])
  shows
    ∀ i. ((i < ((length x) * (length y)))
      → ((vec-vec-Tensor x y)!i)
        = f (x!(i div (length y))) (y!(i mod (length y))))
  apply(rule allI)
  proof(induct x)
  case Nil
  have (length [] = 0)
    by auto
  also have length (vec-vec-Tensor [] y) = 0
    using vec-vec-Tensor.simps(1) by auto
  then have i < (length (vec-vec-Tensor [] y)) ⇒ False
    by auto
  moreover have (vec-vec-Tensor [] y) = []
    by auto
  moreover have
    (i < (length (vec-vec-Tensor [] y))) →
    ((vec-vec-Tensor x y)!i) = f (x!(i div (length y))) (y!(i mod (length y)))
    by auto
  then show ?case

```

by auto
 next
 case (Cons z zs)
 have 1:vec-vec-Tensor (z#zs) y = (times z y)@(vec-vec-Tensor zs y)
 by auto
 have 2: $i < (\text{length } y) \longrightarrow ((\text{times } z y)!i = f z (y!i))$
 using times-elements by auto
 moreover have 3:
 $i < (\text{length } y)$
 $\longrightarrow (\text{vec-vec-Tensor } (z\#zs) y)!i = (\text{times } z y)!i$
 using vec-vec-Tensor-simpl by auto
 moreover have 35:
 $i < (\text{length } y) \longrightarrow (\text{vec-vec-Tensor } (z\#zs) y)!i = f z (y!i)$
 using calculation(1) calculation(2) by metis
 have 4: $(y \neq []) \longrightarrow (\text{length } y) > 0$
 by auto
 have $(i < (\text{length } y)) \longrightarrow ((i \text{ div } (\text{length } y)) = 0)$
 by auto
 then have 6: $(i < (\text{length } y)) \longrightarrow (z\#zs)!i \text{ div } (\text{length } y) = z$
 using nth-Cons-0 by auto
 then have 7: $(i < (\text{length } y)) \longrightarrow (i \text{ mod } (\text{length } y)) = i$
 by auto
 with 2 6
 have $(i < (\text{length } y))$
 $\longrightarrow (\text{times } z y)!i$
 $= f ((z\#zs)!i \text{ div } (\text{length } y)) (y! (i \text{ mod } (\text{length } y)))$
 by auto
 with 3 have step1:
 $(i < (\text{length } y))$
 $\longrightarrow ((i < ((\text{length } x) * (\text{length } y)))$
 $\longrightarrow ((\text{vec-vec-Tensor } (z\#zs) y)!i$
 $= f$
 $((z\#zs)!i \text{ div } (\text{length } y))$
 $(y! (i \text{ mod } (\text{length } y))))))$
 by auto
 have $(\text{length } y \leq i) \longrightarrow (i - (\text{length } y)) \geq 0$
 by auto
 have step2:
 $(\text{length } y < i)$
 $\longrightarrow ((i < (\text{length } (z\#zs) * (\text{length } y)))$
 $\longrightarrow ((\text{vec-vec-Tensor } (z\#zs) y)!i$
 $= f$
 $((z\#zs)!i \text{ div } (\text{length } y))$
 $(y!(i \text{ mod } (\text{length } y))))$
 proof—
 have $(\text{length } y) > 0$
 using assms by auto
 then have 1:
 $(i > (\text{length } y))$

$\longrightarrow (i \operatorname{div} (\operatorname{length} y)) = ((i - (\operatorname{length} y)) \operatorname{div} (\operatorname{length} y)) + 1$
using *nat-div* **by** *auto*
have $zs!j = (z\#zs)!(j+1)$
by *auto*
then have
 $(zs!((i - (\operatorname{length} y)) \operatorname{div} (\operatorname{length} y)))$
 $= (z\#zs)!(((i - (\operatorname{length} y)) \operatorname{div} (\operatorname{length} y)) + 1)$
by *auto*
with 1 have 2:
 $(i > (\operatorname{length} y))$
 $\longrightarrow (zs!((i - (\operatorname{length} y)) \operatorname{div} (\operatorname{length} y)))$
 $= (z\#zs)!(i \operatorname{div} (\operatorname{length} y))$
by *auto*
have $(i > (\operatorname{length} y))$
 $\longrightarrow ((i \operatorname{mod} (\operatorname{length} y)))$
 $= ((i - (\operatorname{length} y)) \operatorname{mod} (\operatorname{length} y))$
using *nat-mod* **by** *auto*
then have 3:
 $(i > (\operatorname{length} y))$
 $\longrightarrow (y! (i \operatorname{mod} (\operatorname{length} y)))$
 $= (y! ((i - (\operatorname{length} y)) \operatorname{mod} (\operatorname{length} y)))$
by *auto*
have 4: $(i > (\operatorname{length} y))$
 $\longrightarrow (\operatorname{vec-vec-Tensor} (z\#zs) y)!i$
 $= (\operatorname{vec-vec-Tensor} zs y)!(i - (\operatorname{length} y))$
using *vec-vec-Tensor-simpl2* **by** *auto*
have 5: $(i > (\operatorname{length} y))$
 $\longrightarrow ((i < ((\operatorname{length} (z\#zs)) * (\operatorname{length} y))))$
 $= (i - (\operatorname{length} y) < (\operatorname{length} zs) * (\operatorname{length} y))$
by *auto*
then have 6:
 $\forall i. ((i < ((\operatorname{length} zs) * (\operatorname{length} y))))$
 $\longrightarrow ((\operatorname{vec-vec-Tensor} zs y)!i)$
 $= f$
 $(zs!(i \operatorname{div} (\operatorname{length} y)))$
 $(y!(i \operatorname{mod} (\operatorname{length} y)))$
using *Cons.hyps* **by** *auto*
with 5 have $(i > (\operatorname{length} y))$
 $\longrightarrow ((i < ((\operatorname{length} (z\#zs)) * (\operatorname{length} y))))$
 $\longrightarrow ((\operatorname{vec-vec-Tensor} zs y)!(i - (\operatorname{length} y)))$
 $= f$
 $(zs!((i - (\operatorname{length} y)) \operatorname{div} (\operatorname{length} y)))$
 $(y!((i - (\operatorname{length} y)) \operatorname{mod} (\operatorname{length} y)))$
 $= ((i < ((\operatorname{length} zs) * (\operatorname{length} y))))$
 $\longrightarrow ((\operatorname{vec-vec-Tensor} zs y)!i)$
 $= f$
 $(zs!(i \operatorname{div} (\operatorname{length} y)))$
 $(y!(i \operatorname{mod} (\operatorname{length} y)))$
by *auto*

with 6 have
 $(i > (\text{length } y))$
 $\rightarrow ((i < ((\text{length } (z\#zs)) * (\text{length } y))))$
 $\rightarrow ((\text{vec-vec-Tensor } zs \ y)! (i - (\text{length } y)))$
 $= f$
 $(zs! ((i - (\text{length } y)) \text{ div } (\text{length } y)))$
 $(y! ((i - (\text{length } y)) \text{ mod } (\text{length } y)))$
by auto
with 2 3 4 have
 $(i > (\text{length } y))$
 $\rightarrow ((i < ((\text{length } (z\#zs)) * (\text{length } y))))$
 $\rightarrow ((\text{vec-vec-Tensor } (z\#zs) \ y)! i)$
 $= f$
 $((z\#zs)! (i \text{ div } (\text{length } y)))$
 $(y! (i \text{ mod } (\text{length } y)))$
by auto
then show ?thesis by auto
qed
have $((\text{length } y) = i)$
 $\rightarrow ((i < (\text{length } (z\#zs)) * (\text{length } y)))$
 $\rightarrow ((\text{vec-vec-Tensor } (z\#zs) \ y)! i)$
 $= f$
 $((z\#zs)! (i \text{ div } (\text{length } y)))$
 $(y! (i \text{ mod } (\text{length } y)))$
proof-
have 1: $(i = (\text{length } y))$
 $\rightarrow ((\text{vec-vec-Tensor } (z\#zs) \ y)! i)$
 $= (\text{vec-vec-Tensor } zs \ y)! 0$
using *vec-vec-Tensor-simpl2* **by auto**
have 2: $(i = \text{length } y) \rightarrow (i \text{ mod } (\text{length } y)) = 0$
by auto
have 3: $(i = \text{length } y) \rightarrow (i \text{ div } (\text{length } y)) = 1$
using 4 *assms div-self less-numeral-extra(3)*
by auto
have 4: $(i = \text{length } y)$
 $\rightarrow ((i < (\text{length } (z\#zs)) * (\text{length } y)))$
 $= (0 < (\text{length } zs) * (\text{length } y))$
by auto
have $(z\#zs)! 1 = (zs! 0)$
by auto
with 3 have 5: $(i = \text{length } y)$
 $\rightarrow ((z\#zs)! (i \text{ div } (\text{length } y))) = (zs! 0)$
by auto
have $\forall i. ((i < (\text{length } zs) * (\text{length } y))$
 $\rightarrow ((\text{vec-vec-Tensor } (zs) \ y)! i)$
 $= f$
 $((zs)! (i \text{ div } (\text{length } y)))$
 $(y! (i \text{ mod } (\text{length } y)))$
using *Cons.hyps* **by auto**

with 4 have 6: $(i = \text{length } y)$
 $\longrightarrow ((0 < ((\text{length } zs) * (\text{length } y)))$
 $\longrightarrow (((\text{vec-vec-Tensor } (zs) \ y)!0)$
 $= f ((zs)!0) (y!0))$
 $= ((i < ((\text{length } zs) * (\text{length } y)))$
 $\longrightarrow (((\text{vec-vec-Tensor } zs \ y)!i)$
 $= f$
 $((zs)!(i \ \text{div} \ (\text{length } y)))$
 $(y!(i \ \text{mod} \ (\text{length } y))))$

by auto
have 7: $(0 \ \text{div} \ (\text{length } y)) = 0$
by auto
have 8: $(0 \ \text{mod} \ (\text{length } y)) = 0$
by auto
have 9: $(0 < ((\text{length } zs) * (\text{length } y)))$
 $\longrightarrow ((\text{vec-vec-Tensor } zs \ y)!0)$
 $= f (zs!0) (y!0)$

using 7 8 Cons.hyps by auto
with 4 5 8 have $(i = \text{length } y)$
 $\longrightarrow ((i < (\text{length } (z\#zs)) * (\text{length } y))$
 $\longrightarrow (((\text{vec-vec-Tensor } (zs) \ y)!0)$
 $= f ((zs)!0) (y!0))$

by auto
with 1 2 5 have $(i = \text{length } y)$
 $\longrightarrow ((i < (\text{length } (z\#zs)) * (\text{length } y))$
 $\longrightarrow (((\text{vec-vec-Tensor } ((z\#zs) \ y)!i)$
 $= f$
 $((z\#zs)!(i \ \text{div} \ (\text{length } y)))$
 $(y!(i \ \text{mod} \ (\text{length } y))))$

by auto
then show ?thesis by auto
qed

with step2 have step4:
 $(i \geq (\text{length } y))$
 $\longrightarrow ((i < (\text{length } (z\#zs)) * (\text{length } y))$
 $\longrightarrow (((\text{vec-vec-Tensor } ((z\#zs) \ y)!i)$
 $= f$
 $((z\#zs)!(i \ \text{div} \ (\text{length } y)))$
 $(y!(i \ \text{mod} \ (\text{length } y))))$

by auto
have $(i < (\text{length } y)) \vee (i \geq (\text{length } y))$
by auto
with step1 step4 have
 $((i < (\text{length } (z\#zs)) * (\text{length } y))$
 $\longrightarrow (((\text{vec-vec-Tensor } ((z\#zs) \ y)!i)$
 $= f$
 $((z\#zs)!(i \ \text{div} \ (\text{length } y)))$
 $(y!(i \ \text{mod} \ (\text{length } y))))$

using logic by (metis 6 7 35)

then show *?case by auto*
qed

a few more results that will be used later on

lemma *nat-int*: $\text{nat } (\text{int } x + \text{int } y) = x + y$
using *nat-int of-nat-add by auto*

lemma *int-nat-equiv*: $(x > 0) \longrightarrow (\text{nat } ((\text{int } x) + -1) + 1) = x$
proof–

have $1 = \text{nat } (\text{int } 1)$
by *auto*
have $-1 = -\text{int } 1$
by *auto*
then have $1: (\text{nat } ((\text{int } x) + -1) + 1)$
 $= (\text{nat } ((\text{int } x) + -1) + (\text{nat } (\text{int } 1)))$
by *auto*
then have $2: (x > 0)$
 $\longrightarrow \text{nat } ((\text{int } x) + -1) + (\text{nat } (\text{int } 1))$
 $= (\text{nat } (((\text{int } x) + -1) + (\text{int } 1)))$
using *of-nat-add nat-int by auto*
have $(\text{nat } (((\text{int } x) + -1) + (\text{int } 1))) = (\text{nat } ((\text{int } x) + -1 + (\text{int } 1)))$
by *auto*
then have $(\text{nat } (((\text{int } x) + -1) + (\text{int } 1))) = (\text{nat } ((\text{int } x)))$
by *auto*
then have $(\text{nat } (((\text{int } x) + -1) + (\text{int } 1))) = x$
by *auto*
with $1\ 2$ **have** $(x > 0) \longrightarrow \text{nat } ((\text{int } x) + -1) + 1 = x$
by *auto*
then show *?thesis by auto*
qed

lemma *list-int-nat*: $(k > 0) \longrightarrow ((x\#\text{xs})!k = \text{xs}!(\text{nat } ((\text{int } k) + -1)))$
proof–

fix j
have $((x\#\text{xs})!(k+1) = \text{xs}!k)$
by *auto*
have $j = (k+1) \longrightarrow (\text{nat } ((\text{int } j) + -1)) = k$
by *auto*
moreover have $(\text{nat } ((\text{int } j) + -1)) = k$
 $\longrightarrow ((\text{nat } ((\text{int } j) + -1)) + 1) = (k + 1)$
by *auto*
moreover have $(j > 0) \longrightarrow (((\text{nat } ((\text{int } j) + -1)) + 1) = j)$
using *int-nat-equiv by (auto)*
moreover have $(k > 0) \longrightarrow ((x\#\text{xs})!k = \text{xs}!(\text{nat } ((\text{int } k) + -1)))$
using *Suc-eq-plus1 int-nat-equiv nth-Cons-Suc by (metis)*
from this show *?thesis by auto*
qed

```

lemma row-length-eq:
  (mat (row-length (a#b#N)) (length (a#b#N)) (a#b#N))
  →
  (row-length (a#b#N) = (row-length (b#N)))
proof –
  have (mat (row-length (a#b#N)) (length (a#b#N)) (a#b#N))
    → (b ∈ set (a#b#M))
    by auto
  moreover have
    (mat (row-length (a#b#N)) (length (a#b#N)) (a#b#N))
    → (Ball (set (a#b#N)) (vec (row-length (a#b#N))))
    using mat-def by metis
  moreover have (mat (row-length (a#b#N)) (length (a#b#N)) (a#b#N))
    → (b ∈ (set (a#b#N)))
    → (vec (row-length (a#b#N)) b)
    by (metis calculation(2))
  then have (mat (row-length (a#b#N)) (length (a#b#N)) (a#b#N))
    → (length b) = (row-length (a#b#N))
    using vec-def by auto
  then have (mat (row-length (a#b#N)) (length (a#b#N)) (a#b#N))
    → (row-length (b#N))
      = (row-length (a#b#N))
    using row-length-def by auto
  then show ?thesis by auto
qed

```

The following theorem tells us the relationship between entries of `vec_mat_Ten`-`sor v M` and entries of `v` and `M` respectively

```

theorem vec-mat-Tensor-elements:
  ∀ i. ∀ j.
  (((i < ((length v) * (row-length M)))
  ∧ (j < (length M)))
  ∧ (mat (row-length M) (length M) M)
  → ((vec-mat-Tensor v M)!j!i)
    = f (v!(i div (row-length M))) (M!j!(i mod (row-length M))))
apply(rule allI)
apply(rule allI)
proof(induct M)
case Nil
  have row-length [] = 0
    using row-length-def by auto
  from this
  have (length v) * (row-length []) = 0
    by auto
  from this
  have (((i < ((length v) * (row-length []))) ∧ (j < (length []))) → False)
    by auto
  moreover have vec-mat-Tensor v [] = []

```

```

    by auto
  moreover have (((i < ((length v) * (row-length []))) ^ (j < (length [])))
    → ((vec-mat-Tensor v [])!j!i)
      = f (v!(i div (row-length []))) ([]!j!(i mod (row-length [])))

    by auto
  from this
    show ?case by auto
next
case (Cons a M)
have (((i < ((length v) * (row-length (a#M))))
  ^ (j < (length (a#M))))
  ^ (mat (row-length (a#M)) (length (a#M)) (a#M))
  → ((vec-mat-Tensor v (a#M))!j!i)
    = f
      (v!(i div (row-length (a#M))))
      ((a#M)!j!(i mod (row-length (a#M))))
proof(cases a)
case Nil
have row-length ([]#M) = 0
  using row-length-def by auto
then have 1:(length v) * (row-length ([]#M)) = 0
  by auto
then have (((i < ((length v) * (row-length ([]#M))))
  ^ (j < (length ([]#M)))) → False
  by auto
moreover have
  (((i < ((length v) * (row-length ([]#M))))
  ^ (j < (length ([]#M))))
  → ((vec-mat-Tensor v ([]#M))!j!i) =
    f
      (v!(i div (row-length ([]#M))))
      ([]!j!(i mod (row-length ([]#M))))
  using calculation by auto
then show ?thesis using Nil 1 less-nat-zero-code by (metis )
next
case (Cons x xs)
have 1:(a#M)!j+1 = M!j by auto
have (((i < ((length v) * (row-length M)))
  ^ (j < (length M)))
  ^ (mat (row-length M) (length M) M)
  → ((vec-mat-Tensor v M)!j!i) = f
    (v!(i div (row-length M)))
    (M!j!(i mod (row-length M)))
  using Cons.hyps by auto
have 2:(row-length (a#M)) = (length a)
  using row-length-def by auto
then have 3:(i < (row-length (a#M)) * (length v))
  = (i < (length a) * (length v))
  by auto

```

```

have a ≠ []
  using Cons by auto
then have 4:
  ∀ i. ((i < (length a)*(length v))
    → ((vec-vec-Tensor v a)!i) = f
      (v!(i div (length a)))
      (a!(i mod (length a))))
  using vec-vec-Tensor-elements Cons.hyps mult.commute
  by (simp add: mult.commute vec-vec-Tensor-elements)
have (vec-mat-Tensor v (a#M))!0 = (vec-vec-Tensor v a)
  using vec-mat-Tensor.simps(2) by auto
with 2 4 have 5:
  ∀ i. ((i < (row-length (a#M))*(length v))
    → ((vec-mat-Tensor v (a#M))!0!i)
      = f
      (v!(i div (row-length (a#M))))
      ((a#M)!0!(i mod (row-length (a#M)))))

  by auto
have length (a#M) > 0
  by auto
with 5 have 6:
  (j = 0) →
    (((i < (row-length (a#M))*(length v))
      ∧ (j < (length (a#M))))
      ∧ (mat (row-length (a#M)) (length (a#M)) (a#M)))
    → ((vec-mat-Tensor v (a#M))!j!i)
      = f
      (v!(i div (row-length (a#M))))
      ((a#M)!j!(i mod (row-length (a#M)))))

  by auto
have (((i < (row-length (a#M))*(length v))
  ∧ (j < (length (a#M))))
  ∧ (mat (row-length (a#M)) (length (a#M)) (a#M)))
  →
    ((vec-mat-Tensor v (a#M))!j!i) =
      f
      (v!(i div (row-length (a#M))))
      ((a#M)!j!(i mod (row-length (a#M)))))

proof(cases M)
case Nil
  have (length (a#[])) = 1
    by auto
  then have (j < (length (a#[]))) = (j = 0)
    by auto
  then have (((i < (row-length (a#[]))*(length v))
    ∧ (j < (length (a#[]))))
    ∧ (mat (row-length (a#[])) (length (a#[])) (a#[])))
    → ((vec-mat-Tensor v (a#[]))!j!i)
      = f

```

$(v!(i \text{ div } (\text{row-length } (a\#\[])))$
 $((a\#\[])!j!(i \text{ mod } (\text{row-length } (a\#\[]))))$

using 6 Nil **by** auto

then show ?thesis **using** Nil **by** auto

next

case (Cons b N)

have 7:(mat (row-length (a#b#N)) (length (a#b#N)) (a#b#N))
 \rightarrow row-length (a#b#N) = (row-length (b#N))

using row-length-eq **by** metis

have 8: (j>0)
 \rightarrow ((vec-mat-Tensor v (b#N))!(nat ((int j)+-1)))
 $=$ (vec-mat-Tensor v (a#b#N))!j

using vec-mat-Tensor.simps(2) **using** list-int-nat **by** metis

have 9: (j>0)
 \rightarrow (((i < (row-length (b#N))*(length v))
 \wedge ((nat ((int j)+-1)) < (length (b#N))))
 \wedge (mat (row-length (b#N)) (length (b#N)) (b#N))
 \rightarrow
((vec-mat-Tensor v (b#N))!(nat ((int j)+-1))!i)
 $=$ f
(v!(i div (row-length (b#N))))
((b#N)!(nat ((int j)+-1))!(i mod (row-length (b#N))))))

using Cons.hyps Cons mult.commute **by** metis

have (j>0) \rightarrow ((nat ((int j) + -1)) < (length (b#N)))
 \rightarrow ((nat ((int j) + -1) + 1) < length (a#b#N))

by auto

then have
(j>0)
 \rightarrow ((nat ((int j) + -1)) < (length (b#N))) = (j < length (a#b#N))

by auto

then have
(j>0)
 \rightarrow (((i < (row-length (b#N))*(length v)) \wedge (j < length (a#b#N)))
 \wedge (mat (row-length (b#N)) (length (b#N)) (b#N)) \rightarrow
((vec-mat-Tensor v (b#N))!(nat ((int j)+-1))!i)
 $=$ f
(v!(i div (row-length (b#N))))
((b#N)!(nat ((int j)+-1))!(i mod (row-length (b#N))))))

using Cons.hyps Cons mult.commute **by** metis

with 8 **have** (j>0)
 \rightarrow (((i < (row-length (b#N))*(length v))
 \wedge (j < length (a#b#N)))
 \wedge (mat (row-length (b#N)) (length (b#N)) (b#N))
 \rightarrow
((vec-mat-Tensor v (a#b#N))!j!i)
 $=$ f
(v!(i div (row-length (b#N))))
((b#N)!(nat ((int j)+-1))!(i mod (row-length (b#N))))))

by auto

also have $(j > 0) \longrightarrow (b\#N)!(\text{nat } ((\text{int } j) + -1)) = (a\#b\#N)!j$
using *list-int-nat by metis*
moreover have $(j > 0) \longrightarrow$
 $((i < (\text{row-length } (b\#N)) * (\text{length } v))$
 $\wedge (j < \text{length } (a\#b\#N)))$
 $\wedge (\text{mat } (\text{row-length } (b\#N)) (\text{length } (b\#N)) (b\#N))$
 \longrightarrow
 $((\text{vec-mat-Tensor } v (a\#b\#N))!j!i)$
 $= f$
 $(v!(i \text{ div } (\text{row-length } (b\#N))))$
 $((a\#b\#N)!j!(i \text{ mod } (\text{row-length } (b\#N))))$
by (*metis calculation(1) calculation(2)*)
then have
 $(j > 0)$
 $\longrightarrow ((i < (\text{row-length } (b\#N)) * (\text{length } v))$
 $\wedge (j < \text{length } (a\#b\#N)))$
 $\wedge (\text{mat } (\text{row-length } (a\#b\#N)) (\text{length } (a\#b\#N)) (a\#b\#N))$
 \longrightarrow
 $((\text{vec-mat-Tensor } v (a\#b\#N))!j!i)$
 $= f$
 $(v!(i \text{ div } (\text{row-length } (b\#N))))$
 $((a\#b\#N)!j!(i \text{ mod } (\text{row-length } (b\#N))))$
using *reduct-matrix by metis*
moreover have $(\text{mat } (\text{row-length } (a\#b\#N)) (\text{length } (a\#b\#N)) (a\#b\#N))$
 $\longrightarrow (\text{row-length } (b\#N)) = (\text{row-length } (a\#b\#N))$
by (*metis 7 Cons*)
moreover have *10:(j > 0)*
 $\longrightarrow ((i < (\text{row-length } (a\#b\#N)) * (\text{length } v))$
 $\wedge (j < \text{length } (a\#b\#N)))$
 $\wedge (\text{mat } (\text{row-length } (a\#b\#N)) (\text{length } (a\#b\#N)) (a\#b\#N))$
 \longrightarrow
 $((\text{vec-mat-Tensor } v (a\#b\#N))!j!i)$
 $= f (v!(i \text{ div } (\text{row-length } (a\#b\#N))))$
 $((a\#b\#N)!j!(i \text{ mod } (\text{row-length } (a\#b\#N))))$
by (*metis calculation(3) calculation(4)*)
have $(j = 0) \vee (j > 0)$
by *auto*
with *6 10 logic have*
 $((i < (\text{row-length } (a\#b\#N)) * (\text{length } v))$
 $\wedge (j < \text{length } (a\#b\#N)))$
 $\wedge (\text{mat } (\text{row-length } (a\#b\#N)) (\text{length } (a\#b\#N)) (a\#b\#N))$
 \longrightarrow
 $((\text{vec-mat-Tensor } v (a\#b\#N))!j!i)$
 $= f$
 $(v!(i \text{ div } (\text{row-length } (a\#b\#N))))$
 $((a\#b\#N)!j!(i \text{ mod } (\text{row-length } (a\#b\#N))))$
using *Cons by metis*
from this show *?thesis by metis Cons*
qed

from this show ?thesis by (metis mult.commute)
qed
from this show ?case by auto
qed

The following theorem tells us about the relationship between entries of tensor products of two matrices and the entries of matrices

theorem matrix-*Tensor-elements*:

fixes $M1\ M2$

shows

$$\begin{aligned}
& \forall i.\forall j.(((i < ((\text{row-length } M1) * (\text{row-length } M2)))) \\
& \quad \wedge (j < (\text{length } M1) * (\text{length } M2))) \\
& \quad \wedge (\text{mat } (\text{row-length } M1) (\text{length } M1) M1) \\
& \quad \wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2) \\
& \quad \longrightarrow ((M1 \otimes M2)!j!i) = \\
& \quad \quad \quad f \\
& \quad \quad \quad (M1!(j \text{ div } (\text{length } M2))!(i \text{ div } (\text{row-length } M2))) \\
& \quad \quad \quad (M2!(j \text{ mod } \text{length } M2)!(i \text{ mod } (\text{row-length } M2)))
\end{aligned}$$

apply(rule allI)

apply(rule allI)

proof(induct M1)

case Nil

have $(\text{row-length } []) = 0$

using row-length-def **by** auto

then have $(i < ((\text{row-length } []) * (\text{row-length } M2))) \longrightarrow \text{False}$

by auto

from this have $((i < ((\text{row-length } []) * (\text{row-length } M2))))$

$\wedge (j < (\text{length } []) * (\text{length } M2))$

$\wedge (\text{mat } (\text{row-length } []) (\text{length } []) [])$

$\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$

$\longrightarrow \text{False}$

by auto

moreover have $([] \otimes M2) = []$

by auto

moreover have

$((i < ((\text{row-length } []) * (\text{row-length } M2))))$

$\wedge (j < (\text{length } []) * (\text{length } M2))$

$\wedge (\text{mat } (\text{row-length } []) (\text{length } []) [])$

$\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$

$\longrightarrow (([] \otimes M2)!j!i) =$

f

$([]!(j \text{ div } (\text{length } []))!(i \text{ div } (\text{row-length } M2)))$

$(M2!(j \text{ mod } \text{length } [])!(i \text{ mod } (\text{row-length } M2)))$

by auto

then show ?case by auto

next

case (Cons v M)

fix a

have $0:(v\#M) \otimes M2 = (\text{vec-mat-Tensor } v\ M2) @ (\text{Tensor } M\ M2)$

by auto
then have 1:
 $(j < (\text{length } M2)) \longrightarrow ((v\#M) \otimes M2)!j = (\text{vec-mat-Tensor } v \ M2)!j$
using *append-simpl vec-mat-Tensor-length by metis*
have $((i < ((\text{length } a) * (\text{row-length } M2)))$
 $\wedge (j < (\text{length } M2))) \wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$
 $\longrightarrow ((\text{vec-mat-Tensor } a \ M2)!j!i) = f (a!(i \text{ div } (\text{row-length } M2))) (M2!j!(i \text{ mod } (\text{row-length } M2)))$
using *vec-mat-Tensor-elements by auto*
have $(j < (\text{length } M2)) \longrightarrow (j \text{ div } (\text{length } M2)) = 0$
by auto
then have 2: $(j < (\text{length } M2)) \longrightarrow (v\#M)!(j \text{ div } (\text{length } M2)) = v$
by auto
have $(j < (\text{length } M2)) \longrightarrow (j \text{ mod } (\text{length } M2)) = j$
by auto
moreover have $(j < (\text{length } M2)) \longrightarrow (v\#M)!(j \text{ mod } (\text{length } M2)) = (v\#M)!j$

by auto
have step0:
 $(j < (\text{length } M2)) \longrightarrow$
 $((i < ((\text{length } v) * (\text{row-length } M2)))$
 $\wedge (j < (\text{length } M2) * (\text{length } (v\#M))))$
 $\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$
 $\longrightarrow ((\text{Tensor } (v\#M) \ M2)!j!i)$
 $= f$
 $((v\#M)!(j \text{ div } (\text{length } M2))!(i \text{ div } (\text{row-length } M2)))$
 $(M2!(j \text{ mod } (\text{length } M2))!(i \text{ mod } (\text{row-length } M2)))$
using *2 1 calculation(1) vec-mat-Tensor-elements by auto*
have step1:
 $(j < (\text{length } M2))$
 $\longrightarrow (((i < ((\text{row-length } (v\#M)) * (\text{row-length } M2)))$
 $\wedge (j < (\text{length } (v\#M)) * (\text{length } M2)))$
 $\wedge (\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M))$
 $\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$
 $\longrightarrow ((\text{Tensor } (v\#M) \ M2)!j!i) =$
 f
 $((v\#M)!(j \text{ div } (\text{length } M2))!(i \text{ div } (\text{row-length } M2)))$
 $(M2!(j \text{ mod } (\text{length } M2))!(i \text{ mod } (\text{row-length } M2)))$
using *row-length-def step0 by auto*
from 0 have 3:
 $(j \geq (\text{length } M2)) \longrightarrow ((v\#M) \otimes M2)!j = (M \otimes M2)!(j - (\text{length } M2))$
using *vec-mat-Tensor-length add.commute append-simpl2 by metis*
have 4:
 $(j \geq (\text{length } M2)) \longrightarrow$
 $((i < ((\text{row-length } M) * (\text{row-length } M2)))$
 $\wedge ((j - (\text{length } M2)) < (\text{length } M) * (\text{length } M2)))$
 $\wedge (\text{mat } (\text{row-length } M) (\text{length } M) M)$
 $\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$
 $\longrightarrow ((M \otimes M2)!(j - (\text{length } M2))!i)$

$= f$
 $(M!((j - (\text{length } M2)) \text{ div } (\text{length } M2))!(i \text{ div } (\text{row-length } M2)))$
 $(M2!((j - (\text{length } M2)) \bmod \text{length } M2)!(i \bmod (\text{row-length } M2))))$
using *Cons.hyps by auto*
moreover have $(\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M))$
 $\longrightarrow (\text{mat } (\text{row-length } M) (\text{length } M) M)$
using *reduct-matrix by auto*
moreover have 5:
 $(j \geq (\text{length } M2))$
 $\longrightarrow (((i < ((\text{row-length } M) * (\text{row-length } M2))))$
 $\wedge ((j - (\text{length } M2)) < (\text{length } M) * (\text{length } M2)))$
 $\wedge (\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M))$
 $\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$
 $\longrightarrow ((M \otimes M2)!(j - (\text{length } M2))!i)$
 $= f$
 $(M!((j - (\text{length } M2)) \text{ div } (\text{length } M2))!(i \text{ div } (\text{row-length } M2)))$
 $(M2!((j - (\text{length } M2)) \bmod \text{length } M2)!(i \bmod (\text{row-length } M2))))$
using 4 calculation(3) by metis
have $((j - (\text{length } M2)) < (\text{length } M) * (\text{length } M2))$
 $\longrightarrow (j < ((\text{length } M) + 1) * (\text{length } M2))$
by auto
then have 6:
 $((j - (\text{length } M2)) < (\text{length } M) * (\text{length } M2))$
 \longrightarrow
 $(j < ((\text{length } (v\#M)) * (\text{length } M2)))$
by auto
have 7:
 $(j \geq (\text{length } M2))$
 \longrightarrow
 $((j - (\text{length } M2)) \text{ div } (\text{length } M2)) = ((j \text{ div } (\text{length } M2)) - 1)$
using *add-diff-cancel-left' div-add-self1 div-by-0*
le-imp-diff-is-add add.commute zero-diff
by metis
then have 8:
 $(j \geq (\text{length } M2))$
 \longrightarrow
 $M!((j - (\text{length } M2)) \text{ div } (\text{length } M2))$
 $= M!((j \text{ div } (\text{length } M2)) - 1)$
by auto
have step2:
 $(j \geq (\text{length } M2))$
 \longrightarrow
 $((i < ((\text{row-length } (v\#M)) * (\text{row-length } M2))))$
 $\wedge (j < (\text{length } (v\#M)) * (\text{length } M2))$
 $\wedge (\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M))$
 $\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$
 $\longrightarrow ((v\#M) \otimes M2)!j!i =$
 $\overset{f}{((v\#M)! (j \text{ div } (\text{length } M2))! (i \text{ div } (\text{row-length } M2)))}$

```

      (M2!(j mod length M2)!(i mod (row-length M2)))
proof(cases M2)
case Nil
  have (0 = ((row-length (v#M))*(row-length M2)))
    using row-length-def Nil mult-0-right by auto
  then have (i < ((row-length (v#M))*(row-length M2))) → False
    by auto
  then have (j ≥ (length M2))
    →(((i < ((row-length (v#M))*(row-length M2)))
      ∧(j < (length (v#M))*(length M2)))
      ∧(mat (row-length (v#M)) (length (v#M)) (v#M))
      ∧(mat (row-length M2) (length M2) M2))
      → False
    by auto
  then show ?thesis by auto
next
case (Cons w N)
  fix k
  have (k < (length M)) ∧ (k ≥ 1) → M!(k - 1) = (v#M)!k
    using not-one-le-zero nth-Cons' by auto
  have (j ≥ (length (w#N))) → (j div (length (w#N))) ≥ 1
    using div-le-mono div-self length-0-conv neq-Nil-conv by metis
  moreover have (j ≥ (length (w#N))) → (j div (length (w#N))) - 1 ≥ 0
    by auto
  moreover have (j ≥ (length (w#N)))
    → M!((j div (length (w#N))) - 1)
      = (v#M)!(j div (length (w#N)))
    using calculation(1) not-one-le-zero nth-Cons' by auto
  from this 7 have 9: (j ≥ (length (w#N)))
    → M!((j - (length (w#N))) div (length (w#N)))
      = (v#M)!(j div (length (w#N)))
    using Cons by auto
  have 10: (j ≥ (length (w#N)))
    → ((j - (length (w#N))) mod (length (w#N)))
      = (j mod (length (w#N)))
    using mod-if not-less by auto
  with 5 9 have
    (j ≥ (length (w#N))) →
    ((i < ((row-length M)*(row-length (w#N))))
    ∧((j - (length (w#N))) < (length M)*(length (w#N)))
    ∧(mat (row-length (v#M)) (length (v#M)) (v#M))
    ∧(mat (row-length (w#N)) (length (w#N)) (w#N)))
    → (((M ⊗ (w#N))!(j - (length (w#N))))!i)
      = f
      ((v#M)!(j div (length (w#N)))!i div (row-length (w#N)))
      ((w#N)!(j mod length (w#N))!i mod (row-length (w#N))))
    using Cons by auto
  then have
    (j ≥ (length (w#N))) →

```

$$\begin{aligned}
& ((i < ((\text{row-length } M) * (\text{row-length } (w\#N)))) \\
& \wedge (j < ((\text{length } (v\#M)) * (\text{length } (w\#N)))) \\
& \wedge (\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M)) \\
& \wedge (\text{mat } (\text{row-length } (w\#N)) (\text{length } (w\#N)) (w\#N)) \\
& \longrightarrow (((M \otimes (w\#N))! (j - (\text{length } (w\#N))))! i) \\
& = f \\
& ((v\#M)! (j \text{ div } (\text{length } (w\#N))))! (i \text{ div } (\text{row-length } (w\#N))) \\
& ((w\#N)! (j \text{ mod } \text{length } (w\#N)))! (i \text{ mod } (\text{row-length } (w\#N)))
\end{aligned}$$

using 6 by auto

then have 11:

$$\begin{aligned}
& (j \geq (\text{length } (w\#N))) \longrightarrow \\
& ((i < ((\text{row-length } M) * (\text{row-length } (w\#N)))) \\
& \wedge (j < ((\text{length } (v\#M)) * (\text{length } (w\#N)))) \\
& \wedge (\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M)) \\
& \wedge (\text{mat } (\text{row-length } (w\#N)) (\text{length } (w\#N)) (w\#N)) \\
& \longrightarrow (((v\#M) \otimes (w\#N))! j! i) = \\
& f \\
& ((v\#M)! (j \text{ div } (\text{length } (w\#N))))! (i \text{ div } (\text{row-length } (w\#N))) \\
& ((w\#N)! (j \text{ mod } \text{length } (w\#N)))! (i \text{ mod } (\text{row-length } (w\#N)))
\end{aligned}$$

using 3 Cons by auto

have

$$\begin{aligned}
& (j \geq (\text{length } (w\#N))) \longrightarrow \\
& ((i < ((\text{row-length } (v\#M)) * (\text{row-length } (w\#N)))) \\
& \wedge (j < ((\text{length } (v\#M)) * (\text{length } (w\#N)))) \\
& \wedge (\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M)) \\
& \wedge (\text{mat } (\text{row-length } (w\#N)) (\text{length } (w\#N)) (w\#N)) \\
& \longrightarrow (((v\#M) \otimes (w\#N))! j! i) \\
& = f \\
& ((v\#M)! (j \text{ div } (\text{length } (w\#N))))! (i \text{ div } (\text{row-length } (w\#N))) \\
& ((w\#N)! (j \text{ mod } \text{length } (w\#N)))! (i \text{ mod } (\text{row-length } (w\#N)))
\end{aligned}$$

proof (cases M)

case Nil

have Nil0: $(\text{length } (v\#\[])) = 1$

by auto

then have Nil1:

$$(j < ((\text{length } (v\#\[])) * (\text{length } (w\#N)))) = (j < (\text{length } (w\#N)))$$

by (metis Nil nat-mult-1)

have

$$\text{row-length } (v\#\[]) = (\text{length } v)$$

using row-length-def by auto

then have Nil2:

$$\begin{aligned}
& (i < ((\text{row-length } (v\#M)) * (\text{row-length } (w\#N)))) \\
& = (i < ((\text{length } v) * (\text{row-length } (w\#N))))
\end{aligned}$$

using Nil by auto

then have $(j < (\text{length } (w\#N))) \longrightarrow (j \text{ div } (\text{length } (w\#N))) = 0$

by auto

from this have Nil3:

$$(j < (\text{length } (w\#N))) \longrightarrow (v\#M)! (j \text{ div } (\text{length } (w\#N))) = v$$

using Nil by auto

then have Nil4:
 $(j < (\text{length } (w\#N))) \longrightarrow (j \bmod (\text{length } (w\#N))) = j$
by auto

then have Nil5: $(v\#M) \otimes (w\#N) = \text{vec-mat-Tensor } v (w\#N)$
using Nil Tensor.simps(2) Tensor.simps(1)
by auto

from vec-mat-Tensor-elements have

$$\begin{aligned} &(((i < ((\text{length } v) * (\text{row-length } (w\#N)))) \\ &\wedge (j < (\text{length } (w\#N)))) \\ &\wedge (\text{mat } (\text{row-length } (w\#N)) (\text{length } (w\#N)) (w\#N)) \\ &\longrightarrow ((\text{vec-mat-Tensor } v (w\#N))!j!i) \\ &= f \\ &\quad (v!(i \text{ div } (\text{row-length } (w\#N)))) \\ &\quad ((w\#N)!j!(i \bmod (\text{row-length } (w\#N)))) \end{aligned}$$

by metis

then have

$$\begin{aligned} &((i < ((\text{row-length } (v\#M)) * (\text{row-length } (w\#N)))) \\ &\wedge (j < ((\text{length } (v\#M)) * (\text{length } (w\#N)))) \\ &\wedge (\text{mat } (\text{row-length } (w\#N)) (\text{length } (w\#N)) (w\#N)) \\ \longrightarrow &((\text{vec-mat-Tensor } v (w\#N))!j!i) \\ &= f (v!(i \text{ div } (\text{row-length } (w\#N)))) \\ &\quad ((w\#N)!j!(i \bmod (\text{row-length } (w\#N)))) \end{aligned}$$

using Nil1 Nil2 Nil by auto

then have

$$\begin{aligned} &((i < ((\text{row-length } (v\#M)) * (\text{row-length } (w\#N)))) \\ &\wedge (j < ((\text{length } (v\#M)) * (\text{length } (w\#N)))) \\ &\wedge (\text{mat } (\text{row-length } (w\#N)) (\text{length } (w\#N)) (w\#N)) \\ \longrightarrow &(((v\#M) \otimes (w\#N))!j!i) \\ &= f \\ &\quad ((v\#M)!(j \text{ div } (\text{length } (w\#N)))(i \text{ div } (\text{row-length } (w\#N)))) \\ &\quad ((w\#N)!(j \bmod (\text{length } (w\#N)))(i \bmod (\text{row-length } (w\#N)))) \end{aligned}$$

using Nil3 Nil4 Nil5 Nil by auto

then have

$$\begin{aligned} &((i < ((\text{row-length } (v\#M)) * (\text{row-length } (w\#N)))) \\ &\wedge (j < ((\text{length } (v\#M)) * (\text{length } (w\#N)))) \\ &\wedge (\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M)) \\ &\wedge (\text{mat } (\text{row-length } (w\#N)) (\text{length } (w\#N)) (w\#N)) \\ \longrightarrow &(((v\#M) \otimes (w\#N))!j!i) \\ &= f \\ &\quad ((v\#M)!(j \text{ div } (\text{length } (w\#N)))(i \text{ div } (\text{row-length } (w\#N)))) \\ &\quad ((w\#N)!(j \bmod (\text{length } (w\#N)))(i \bmod (\text{row-length } (w\#N)))) \end{aligned}$$

by auto

from this show ?thesis by auto
next
case (Cons u P)
have $(\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M)) \longrightarrow (\text{row-length } (v\#M)) = (\text{row-length } M)$
using Cons row-length-eq by metis
from this 11 show ?thesis by auto

qed
from this show *?thesis using Cons by auto*
qed
have $(j < (\text{length } M2)) \vee (j \geq (\text{length } M2))$ **by auto**
from this step1 step2 logic have
 $((i < ((\text{row-length } (v\#M)) * (\text{row-length } M2))))$
 $\wedge (j < (\text{length } M2) * (\text{length } (v\#M)))$
 $\wedge (\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M))$
 $\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$
 $\longrightarrow (((v\#M) \otimes M2)!j!i$
 $\quad = f$
 $\quad \quad ((v\#M)!(j \text{ div } (\text{length } M2))!(i \text{ div } (\text{row-length } M2)))$
 $\quad \quad (M2!(j \text{ mod } (\text{length } M2))!(i \text{ mod } (\text{row-length } M2))))$
using *mult.commute by metis*
from this show *?case by (metis mult.commute)*
qed

we restate the theorem in two different forms for convenience of reuse

theorem *effective-matrix-tensor-elements:*
 $((i < ((\text{row-length } M1) * (\text{row-length } M2))))$
 $\wedge (j < (\text{length } M1) * (\text{length } M2))$
 $\wedge (\text{mat } (\text{row-length } M1) (\text{length } M1) M1)$
 $\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$
 $\implies ((M1 \otimes M2)!j!i$
 $\quad = f (M1!(j \text{ div } (\text{length } M2))!(i \text{ div } (\text{row-length } M2)))$
 $\quad \quad (M2!(j \text{ mod } \text{length } M2)!(i \text{ mod } (\text{row-length } M2))))$
using *matrix-Tensor-elements by auto*

theorem *effective-matrix-tensor-elements2:*
assumes $i < (\text{row-length } M1) * (\text{row-length } M2)$
and $j < (\text{length } M1) * (\text{length } M2)$
and $\text{mat } (\text{row-length } M1) (\text{length } M1) M1$
and $\text{mat } (\text{row-length } M2) (\text{length } M2) M2$
shows $(M1 \otimes M2)!j!i =$
 $(M1!(j \text{ div } (\text{length } M2))!(i \text{ div } (\text{row-length } M2)))$
 $\quad * (M2!(j \text{ mod } \text{length } M2)!(i \text{ mod } (\text{row-length } M2)))$
using *assms matrix-Tensor-elements by auto*

the following lemmas are useful in proving associativity of tensor products

lemma *div-left-ineq:*
assumes $(x::\text{nat}) < y*z$
shows $(x \text{ div } z) < y$
proof(*rule ccontr*)
assume $0: \neg((x \text{ div } z) < y)$
then have $1: x \text{ div } z \geq y$
by auto
then have $2:(x \text{ div } z)*z \geq y*z$
by auto
then have $3:(x \text{ div } z)*z + (x \text{ mod } z) = z$

```

    using div-mult-mod-eq
    add-leD1 assms minus-mod-eq-div-mult [symmetric] le-diff-conv2 mod-less-eq-dividend
not-less
    by metis
then have 4:(x div z)*z ≤ z
    by auto
then have 5:z ≥ y*z
    using 2 by auto
then have 6:z div z ≥ (y*z) div z
    by auto
then have (y*z) div z ≤ 1
    by auto
with 6 have 1 ≥ y
    using 1 3 assms div-self less-nat-zero-code mult-zero-left
    mult.commute mod-div-mult-eq
    by auto
then have 7:(y = 0) ∨ (y = 1)
    by auto
have (y = 0) ⇒ x < 0
    using assms by auto
moreover have x ≥ 0
    by auto
then have 8:(y = 0) ⇒ False
    using calculation less-nat-zero-code by auto
moreover have (y = 1) ⇒ (x < z)
    using assms by auto
then have (y = 1) ⇒ (x div z) = 0
    by (metis div-less)
then have (y = 1) ⇒ (x div z) < y
    by auto
then have (y = 1) ⇒ False
    using 0 by auto
then show False using 7 8 by auto
qed

```

```

lemma div-right-ineq:
  assumes (x::nat) < y*z
  shows (x div y) < z
  using assms div-left-ineq mult.commute by (metis)

```

In the following theorem, we obtain columns of `vec_mat_Tensor` of a vector `v` and a matrix `M` in terms of the vector `v` and columns of the matrix `M`

```

lemma col-vec-mat-Tensor-prelim:
  ∀ j.(j < (length M)
  →
  col (vec-mat-Tensor v M) j = vec-vec-Tensor v (col M j))
  unfolding col-def
  apply(rule allI)
  proof(induct M)

```

```

case Nil
  show ?case using Nil by auto
next
case (Cons w N)
  have Cons-1:vec-mat-Tensor v (w#N)
    = (vec-vec-Tensor v w)#(vec-mat-Tensor v N)
    using vec-mat-Tensor.simps Cons by auto
  then show ?case
  proof(cases j)
  case 0
    have vec-mat-Tensor v (w#N)!0 = (vec-vec-Tensor v w)
      by auto
    then show ?thesis using 0 by auto
  next
  case (Suc k)
    have vec-mat-Tensor v (w#N)!j = (vec-mat-Tensor v N)!(k)
      using Cons-1 Suc by auto
    moreover have j < length (w#N)  $\implies$  k < length N
      using Suc by (metis length-Suc-conv not-less-eq)
    moreover then have k < length (N)
       $\implies$  (vec-mat-Tensor v N)!k = vec-vec-Tensor v (N!k)
      using Cons.hyps by auto
    ultimately show ?thesis using Suc by auto
  qed
qed

```

```

lemma col-vec-mat-Tensor:fixes j M v
  assumes j < (length M)
  shows col (vec-mat-Tensor v M) j = vec-vec-Tensor v (col M j)
  using col-vec-mat-Tensor-prelim assms by auto

```

```

lemma col-formula:
  fixes M1 and M2
  shows  $\forall j. ((j < (\text{length } M1) * (\text{length } M2))$ 
     $\wedge$  (mat (row-length M1) (length M1) M1)
     $\wedge$  (mat (row-length M2) (length M2) M2)
     $\longrightarrow$  col (M1  $\otimes$  M2) j
      = vec-vec-Tensor
        (col M1 (j div length M2))
        (col M2 (j mod length M2)))

```

```

apply (rule allI)
proof(induct M1)
case Nil
  show ?case using Nil by auto
next
case (Cons v M)
  have j < (length (v#M))*length M2
     $\wedge$  mat (row-length (v # M)) (length (v # M)) (v # M)
     $\wedge$  mat (row-length M2) (length M2) M2  $\implies$ 

```

```

      (col (v # M ⊗ M2) j
       = vec-vec-Tensor
         (col (v # M) (j div length M2))
         (col M2 (j mod length M2)))
proof-
fix k
assume 0:j < (length (v#M))*(length M2)
      ∧ mat (row-length (v # M)) (length (v # M)) (v # M)
      ∧ mat (row-length M2) (length M2) M2
then have 1:mat (row-length M) (length M) M
      by (metis reduct-matrix)
have j < (1 + length M)*(length M2)
      using 0 by auto
then have j < (length M2)+ (length M)*(length M2)
      by auto
then have 2:j ≥ (length M2)
      ⇒ j - (length M2) < (length M)*(length M2)
      using add-0-iff add-diff-inverse diff-is-0-eq
      less-diff-conv less-imp-le linorder-cases add.commute
      neq0-conv
      by (metis (opaque-lifting, no-types))
have 3:(v#M)⊗M2 = (vec-mat-Tensor v M2)@(M ⊗ M2)
      using Tensor.simps by auto
have (col ((v#M)⊗M2) j) = (col ((vec-mat-Tensor v M2)@(M ⊗ M2)) j)
      using col-def by auto
then have j < length (vec-mat-Tensor v M2)
      ⇒ (col ((v#M)⊗M2) j) = (col (vec-mat-Tensor v M2) j)
      unfolding col-def using append-simpl by auto
then have 4:j < length M2 ⇒
      (col ((v#M)⊗M2) j) = (col (vec-mat-Tensor v M2) j)
      using vec-mat-Tensor-length by simp
then have j < length M2 ⇒
      (col (vec-mat-Tensor v M2) j)
      = vec-vec-Tensor v (col M2 j)
      using col-vec-mat-Tensor by auto
then have
      j < length M2 ⇒
      (col (vec-mat-Tensor v M2) j)
      = vec-vec-Tensor
        ((v#M)!(j div length M2))
        (col M2 (j mod (length M2)))
      by auto
then have step-1:j < length M2 ⇒
      (col ((v#M)⊗M2) j)
      = vec-vec-Tensor
        ((v#M)!(j div length M2))
        (col M2 (j mod (length M2)))
      using 4 by auto
have 4:j ≥ length M2

```

$\implies (\text{col } ((v\#M)\otimes M2) j) = (M \otimes M2)!(j - (\text{length } M2))$
unfolding col-def using 3 append-simpl2 vec-mat-Tensor-length
by metis

then have 5:
 $j \geq \text{length } M2 \implies$
 $\text{col } (M \otimes M2) (j - \text{length } M2)$
 $= \text{vec-vec-Tensor}$
 $(\text{col } M ((j - \text{length } M2) \text{ div } \text{length } M2))$
 $(\text{col } M2 ((j - \text{length } M2) \text{ mod } \text{length } M2))$
using 1 0 2 Cons by auto

then have 6:
 $j \geq \text{length } M2 \implies$
 $(j - \text{length } M2) \text{ div } (\text{length } M2) + 1 = j \text{ div } (\text{length } M2)$
using 2 div-0 div-self
le-neq-implies-less less-nat-zero-code
monoid-add-class.add.right-neutral mult-0 mult-cancel2
add.commute nat-div neq0-conv div-add-self1 le-add-diff-inverse
by metis

then have
 $j \geq \text{length } M2 \implies$
 $((j - \text{length } M2) \text{ mod } \text{length } M2) = j \text{ mod } (\text{length } M2)$
using le-mod-geq by metis

with 6 have 7:
 $j \geq \text{length } M2 \implies$
 $\text{col } (M \otimes M2) (j - \text{length } M2)$
 $= \text{vec-vec-Tensor } (\text{col } M ((j - \text{length } M2) \text{ div } \text{length } M2))$
 $(\text{col } M2 (j \text{ mod } \text{length } M2))$
using 5 by auto

moreover have $k < (\text{length } M) \implies (\text{col } M k) = (\text{col } (v\#M) (k+1))$
unfolding col-def by auto

ultimately have $j \geq \text{length } M2 \implies$
 $\text{col } (M \otimes M2) (j - \text{length } M2)$
 $= \text{vec-vec-Tensor } (\text{col } (v\#M) (j \text{ div } \text{length } M2))$
 $(\text{col } M2 (j \text{ mod } \text{length } M2))$

proof-
assume temp: $j \geq \text{length } M2$
have $j - (\text{length } M2) < (\text{length } M) * (\text{length } M2)$
using 2 temp by auto
then have $(j - (\text{length } M2)) \text{ div } (\text{length } M2) < (\text{length } M)$
using div-right-ineq mult.commute by metis
moreover have
 $((j - (\text{length } M2)) \text{ div } (\text{length } M2) < (\text{length } M))$
 $\longrightarrow (\text{col } M ((j - (\text{length } M2)) \text{ div } (\text{length } M2)))$
 $= (\text{col } (v\#M) ((j - (\text{length } M2)) \text{ div } (\text{length } M2) + 1))$
unfolding col-def by auto
ultimately have temp1:
 $(\text{col } (v\#M) (((j - \text{length } M2) \text{ div } \text{length } M2) + 1))$
 $= (\text{col } M (((j - \text{length } M2) \text{ div } \text{length } M2)))$
by auto

```

then have (col (v#M) (((j-length M2) div length M2)+1))
            = (col (v#M) (j div length M2))
using 6 temp by auto
then show ?thesis using temp1 7 by (metis temp)
qed
then have  $j \geq \text{length } M2 \implies$ 
            col ((v#M)  $\otimes$  M2) j
            = vec-vec-Tensor (col (v#M) (j div length M2))
                          (col M2 (j mod length M2))
using col-def 4 by metis
then show ?thesis
using step-1 col-def le-refl nat-less-le nat-neq-iff
by (metis)
qed
then show ?case by auto
qed

lemma row-Cons:row (v#M) i = (v!i)#(row M i)
unfolding row-def map-def by auto

lemma row-append:row (A@B)i = (row A i)@(row B i)
unfolding row-def map-append by auto

lemma row-empty:row [] i = []
unfolding row-def by auto

lemma vec-vec-Tensor-right-empty:vec-vec-Tensor x [] = []
using vec-vec-Tensor.simps times.simps length-0-conv mult-0-right vec-vec-Tensor-length

by (metis)

lemma vec-mat-Tensor v ([]#[]) = [[]]
using vec-mat-Tensor.simps by (metis vec-vec-Tensor-right-empty)

lemma  $i < 0 \implies [[]!i] = []$ 
by auto

lemma row-vec-mat-Tensor-prelim:
 $\forall i.$ 
  (( $i < (\text{length } v) * (\text{row-length } M)$ )  $\wedge$  (mat nr (length M) M)
    $\implies$  row (vec-mat-Tensor v M) i
   = times (v!(i div row-length M)) (row M (i mod row-length M)))
apply(rule allI)
proof(induct M)
case Nil
  show ?case using Nil by (metis less-nat-zero-code mult-0-right row-length-Nil)
next
case (Cons w N)
  have row (vec-mat-Tensor v (w#N)) i

```

```

      = row ((vec-vec-Tensor v w)#(vec-mat-Tensor v N)) i
    using vec-mat-Tensor.simps by auto
  then have 1:... = ((vec-vec-Tensor v w)!i)#(row (vec-mat-Tensor v N) i)
    using row-Cons by auto
  have 2:row-length (w#N) = length w
    using row-length-def by auto
  then have 3:(mat nr (length (w#N)) (w#N)) ==> nr = length w
    using hd-in-set list.distinct(1) mat-uniqueness matrix-row-length by metis
  then have ((i < (length v)*(row-length (w#N)))
    ^ (mat nr (length (w#N)) (w#N))
    ==> row (vec-mat-Tensor v (w#N)) i
    = times
      (v!(i div row-length (w#N)))
      (row (w#N) (i mod row-length (w#N))))
  proof-
  assume assms: i < (length v)*(row-length (w#N))
    ^ (mat nr (length (w#N)) (w#N))
  show ?thesis
  proof(cases N)
  case Nil
  have row (vec-mat-Tensor v (w#N)) i = [(vec-vec-Tensor v w)!i]
    using 1 vec-mat-Tensor.simps Nil row-empty by auto
  then show ?thesis
  proof(cases w)
  case Nil
  have (vec-vec-Tensor v w) = []
    using Nil vec-vec-Tensor-right-empty by auto
  moreover have (length v)*(row-length (w#N)) = 0
    using Nil row-length-def by auto
  then have [(vec-vec-Tensor v [])!i] = []
    using assms less-nat-zero-code by metis
  ultimately show ?thesis
    using vec-vec-Tensor.simps row-empty Nil assms list.distinct(1) by
(metis)
  next
  case (Cons a w1)
  have 1:w ≠ []
    using Cons by auto
  then have i < (length v)*(length w)
    using assms row-length-def by auto
  then have (vec-vec-Tensor v w)!i
    = f
      (v!(i div (length w)))
      (w!(i mod (length w)))
    using vec-vec-Tensor-elements 1 allI by auto
  then have (row (vec-mat-Tensor v (w#N)) i)
    = times
      (v!(i div row-length (w#N)))
      (row (w#N) (i mod (length w)))

```

```

using Cons vec-mat-Tensor.simps row-def row-length-def 2 Nil row-Cons
      row-empty times.simps(1) times.simps(2) by metis
then show ?thesis using row-def 2 by metis
qed
next
case (Cons w1 N1)
  have Cons-0:row-length N = length w1
    using Cons row-length-def by auto
  have mat nr (length (w#w1#N1)) (w#w1#N1)
    using assms Cons by auto
  then have Cons-1:
    mat (row-length (w#w1#N1)) (length (w#w1#N1)) (w#w1#N1)

    by (metis matrix-row-length)
  then have Cons-2:
    mat (row-length (w1#N1)) (length (w1#N1)) (w1#N1)
    by (metis reduct-matrix)
  then have Cons-3:(length w1 = length w)
    using Cons-1
    unfolding mat-def row-length-def Ball-def vec-def
    by (metis 2 Cons-0 Cons-1 local.Cons row-length-eq)
  then have Cons-4:mat nr (length (w1#N1)) (w1#N1)
    using 3 Cons-2 assms hd-conv-nth list.distinct(1) nth-Cons-0 row-length-def
    by metis
  moreover have i < (length v)*(row-length (w1#N1))
    using assms Cons-3 row-length-def by auto
  ultimately have Cons-5:row (vec-mat-Tensor v N) i
    = times
      (v ! (i div row-length N))
      (row N (i mod row-length N))

    using Cons Cons.hyps by auto
  then show ?thesis
proof(cases w)
  case Nil
    have (vec-vec-Tensor v w) = []
      using Nil vec-vec-Tensor-right-empty by auto
    moreover have (length v)*(row-length (w#N)) = 0
      using Nil row-length-def by auto
    then have [(vec-vec-Tensor v [])!i] = []
      using assms by (metis less-nat-zero-code)
    ultimately show ?thesis
      using vec-vec-Tensor.simps row-empty Nil assms
      by (metis list.distinct(1))
  next
  case (Cons a w2)
    have 1:w ≠ []
      using Cons by auto
    then have i < (length v)*(length w)

```

```

using assms row-length-def by auto
then have ConsCons-2:
  (vec-vec-Tensor v w)!i = f
    (v!(i div (length w)))
    (w!(i mod (length w)))
using vec-vec-Tensor-elements 1 allI by auto
moreover have
  times
    (v!(i div row-length (w#N)))
    (row (w#N) (i mod row-length (w#N)))
  = (f
    (v!(i div (length w)))
    (w!(i mod (length w))))
    #(times (v ! (i div row-length N))
      (row N (i mod row-length N)))

proof-
have temp:row-length (w#N) = (row-length N)
using row-length-def 2 Cons-3 Cons-0 by auto
have (row (w#N) (i mod row-length (w#N)))
  = (w!(i mod (row-length (w#N))))
    #(row N (i mod row-length (w#N)))
unfolding row-def by auto
then have ...
  = (w!(i mod (length w)))
    #(row N (i mod row-length N))
using Cons-3 3 assms 2 neq-Nil-conv row-Cons row-empty
  row-length-eq by (metis (opaque-lifting, no-types))
then have times
  (v!(i div row-length (w#N)))
  ((w!(i mod (length w))))
  #(row N (i mod row-length N))
  = (f
    (v!(i div row-length (w#N)))
    (w!(i mod (length w))))
    #(times (v!(i div row-length (w#N))
      (row N (i mod row-length N))))

by auto
then have ... = (f
  (v!(i div length w))
  (w!(i mod (length w))))
  #(times (v!(i div row-length N)
    (row N (i mod row-length N))))
using 3 Cons-3 assms temp row-length-def by auto
then show ?thesis using times.simps 2 row-Cons temp by metis
qed
then show ?thesis using Cons-5 ConsCons-2 1
  row-Cons vec-mat-Tensor.simps(2) by (metis)
qed
qed

```

qed
then show ?case by auto
qed

The following lemma gives us a formula for the row of a tensor of two matrices

lemma *row-formula:*

fixes $M1$ and $M2$

shows $\forall i. ((i < (\text{row-length } M1) * (\text{row-length } M2))$
 $\wedge (\text{mat } (\text{row-length } M1) (\text{length } M1) M1)$
 $\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$
 $\longrightarrow \text{row } (M1 \otimes M2) i$
 $= \text{vec-vec-Tensor}$
 $(\text{row } M1 (i \text{ div } \text{row-length } M2))$
 $(\text{row } M2 (i \text{ mod } \text{row-length } M2)))$

apply(*rule allI*)

proof(*induct M1*)

case *Nil*

show ?case **using** *Nil* **by** (*metis less-nat-zero-code mult-0 row-length-Nil*)

next

case (*Cons v M*)

have

$((i < (\text{row-length } (v\#M)) * (\text{row-length } M2))$
 $\wedge (\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M))$
 $\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$
 $\implies \text{row } ((v\#M) \otimes M2) i = \text{vec-vec-Tensor}$
 $(\text{row } (v\#M) (i \text{ div } \text{row-length } M2))$
 $(\text{row } M2 (i \text{ mod } \text{row-length } M2)))$

proof–

assume *assms:*

$(i < (\text{row-length } (v\#M)) * (\text{row-length } M2))$
 $\wedge (\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M))$
 $\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$

have $0 : i < (\text{length } v) * (\text{row-length } M2)$

using *assms row-length-def* **by** *auto*

have $1 : \text{mat } (\text{row-length } M) (\text{length } M) M$

using *assms reduct-matrix* **by** (*metis*)

have $\text{row } ((v\#M) \otimes M2) i = \text{row } ((\text{vec-mat-Tensor } v M2) @ (M \otimes M2)) i$
by *auto*

then have $2 : \dots = (\text{row } (\text{vec-mat-Tensor } v M2) i) @ (\text{row } (M \otimes M2) i)$
using *row-append* **by** *auto*

then show ?thesis

proof(*cases M*)

case *Nil*

have $\text{row } ((v\#M) \otimes M2) i = (\text{row } (\text{vec-mat-Tensor } v M2) i)$
using *Nil 2* **by** *auto*

moreover have $\text{row } (\text{vec-mat-Tensor } v M2) i = \text{times}$

$(v!(i \text{ div } \text{row-length } M2))$
 $(\text{row } M2 (i \text{ mod } \text{row-length } M2))$

```

      using row-vec-mat-Tensor-prelim assms 0 by auto
    ultimately show ?thesis using vec-vec-Tensor-def
      Nil append-Nil2 vec-vec-Tensor.simps(1)
      vec-vec-Tensor.simps(2) row-Cons row-empty by (metis)
  next
  case (Cons w N)
  have Cons-Cons-1:mat (row-length M) (length M) M
    using assms reduct-matrix by auto
  then have row-length (w#N) = row-length (v#M)
    using assms Cons unfolding mat-def Ball-def vec-def
    using append-Cons hd-in-set list.distinct(1)
    rotate1.simps(2) set-rotate1
    by auto
  then have Cons-Cons-2:i < (row-length M)*(row-length M2)
    using assms Cons by auto
  then have Cons-Cons-3:(row (M ⊗ M2) i) = vec-vec-Tensor
    (row M (i div row-length M2))
    (row M2 (i mod row-length M2))
    using Cons.hyps Cons-Cons-1 assms by auto
  moreover have row (vec-mat-Tensor v M2) i
    = times
      (v!(i div row-length M2))
      (row M2 (i mod row-length M2))
    using row-vec-mat-Tensor-prelim assms 0 by auto
  then have row ((v#M)⊗M2) i =
    (times
      (v!(i div row-length M2))
      (row M2 (i mod row-length M2)))
      @ (vec-vec-Tensor
        (row M (i div row-length M2))
        (row M2 (i mod row-length M2)))
    using 2 Cons-Cons-3 by auto
  moreover have ... = (vec-vec-Tensor
    ((v!(i div row-length M2))
      #(row M (i div row-length M2)))
    (row M2 (i mod row-length M2)))
    using vec-vec-Tensor.simps(2) by auto
  moreover have ... = (vec-vec-Tensor (row (v#M) (i div row-length M2))
    (row M2 (i mod row-length M2)))
    using row-Cons by metis
  ultimately show ?thesis by metis
qed
qed
then show ?case by auto
qed

```

lemma *effective-row-formula*:
 fixes $M1$ and $M2$
 assumes $i < (\text{row-length } M1) * (\text{row-length } M2)$

and (*mat* (*row-length* *M1*) (*length* *M1*) *M1*)
and (*mat* (*row-length* *M2*) (*length* *M2*) *M2*)
shows *row* (*M1* \otimes *M2*) *i*
 = *vec-vec-Tensor*
 (*row* *M1* (*i div row-length* *M2*))
 (*row* *M2* (*i mod row-length* *M2*))
using *assms row-formula* **by** *auto*

lemma *alt-effective-matrix-tensor-elements*:
 (((*i* < ((*row-length* *M2*)*(*row-length* *M3*)))
 \wedge (*j* < (*length* *M2*)*(*length* *M3*)))
 \wedge (*mat* (*row-length* *M2*) (*length* *M2*) *M2*)
 \wedge (*mat* (*row-length* *M3*) (*length* *M3*) *M3*)
 \implies ((*M2* \otimes *M3*)!*j*!*i*) = *f* (*M2*!*j div (length* *M3*)!*i div (row-length* *M3*))
 (*M3*!*j mod length* *M3*)!*i mod (row-length* *M3*)))
using *matrix-Tensor-elements* **by** *auto*

lemma *trans-impl*: ($\forall i j. (P i j \longrightarrow Q i j) \wedge (\forall i j. (Q i j \longrightarrow R i j))$)
 \implies ($\forall i j. (P i j \longrightarrow R i j)$)
by *auto*

lemma ((*x::nat*) *div* *y*) *div* *z* = (*x div (y*z)*)
using *div-mult2-eq* **by** *auto*

lemma ($\neg((a::nat) < b)$) \implies (*a* \geq *b*)
by *auto*

lemma *not-null*: *xs* \neq [] \implies $\exists y ys. xs = y\#ys$
by (*metis neq-Nil-conv*)

lemma (*y::nat*) \neq 0 \implies (*x mod y*) < *y*
using *mod-less-divisor* **by** *auto*

lemma *mod-prop1*: ((*a::nat*) *mod* (*b*c*)) *mod* *c* = (*a mod c*)
proof (*cases c = 0*)
case *True*
 have *b*c = 0*
 by (*metis True mult-0-right*)
 then have (*a::nat*) *mod* (*b*c*) = *a*
 by *auto*
 then have ((*a::nat*) *mod* (*b*c*)) *mod* *c* = *a mod c*
 by *auto*
 then show *?thesis* **by** *auto*
next
case *False*
 let *?x* = (*a::nat*) *mod* (*b*c*)
 let *?z* = *?x mod c*

```

have  $\exists m. a = m*(b*c) + ?x$ 
  by (metis div-mult-mod-eq)
then obtain  $m1$  where  $a = m1*(b*c) + ?x$ 
  by auto
then have  $?x = (a - m1*(b*c))$ 
  by auto
then have  $\exists m. (?x = m*c + ?z)$ 
  using mod-div-decomp by blast
then obtain  $m$  where  $(?x = m*c + ?z)$ 
  by auto
then have  $(a - m1*(b*c)) = m*c + ?z$ 
  using  $\langle a \bmod (b * c) = a - m1 * (b * c) \rangle$  by (metis)
then have  $a = m1*b*c + m*c + ?z$ 
  using  $\langle a = m1 * (b * c) + a \bmod (b * c) \rangle \langle a \bmod (b * c) = m * c + a \bmod (b * c) \bmod c \rangle$ 
  by (metis ab-semigroup-add-class.add-ac(1) ab-semigroup-mult-class.mult-ac(1))
then have  $1:a = (m1*b + m)*c + ?z$ 
  by (metis add-mult-distrib2 mult.commute)
let  $?y = (a \bmod c)$ 
have  $\exists n. a = n*(c) + ?y$ 
  by (metis 1  $\langle a \bmod (b * c) = m * c + a \bmod (b * c) \bmod c \rangle$  mod-mult-self3)

then obtain  $n$  where  $a = n*(c) + ?y$ 
  by auto
with 1 have  $(m1*b + m)*c + ?z = n*c + ?y$ 
  by auto
then have  $(m1*b + m)*c - (n*c) = ?y - ?z$ 
  by auto
then have  $(m1*b + m - n)*c = (?y - ?z)$ 
  by (metis diff-mult-distrib2 mult.commute)
then have  $c \text{ dvd } (?y - ?z)$ 
  by (metis dvd-triv-right)
moreover have  $?y < c$ 
  using mod-less-divisor False by auto
moreover have  $?z < c$ 
  using mod-less-divisor False by auto
moreover have  $?y - ?z < c$ 
  using calculation(2) less-imp-diff-less by blast
ultimately have  $?y - ?z = 0$ 
  by (metis dvd-imp-mod-0 mod-less)
then show  $?thesis$  using False
  by (metis 1 mod-add-right-eq mod-mult-self2 add.commute mult.commute)
qed

lemma mod-div-relation:  $((a::nat) \bmod (b*c)) \text{ div } c = (a \text{ div } c) \bmod b$ 
proof (cases  $b*c = 0$ )
case True
  have  $T-1:(b = 0) \vee (c = 0)$ 

```

```

    using True by auto
show ?thesis
proof(cases (b = 0))
case True
  have a mod (b*c) = a
    using True by auto
  then show ?thesis using True by auto
next
case False
  have c = 0
    using T-1 False by auto
  then show ?thesis by auto
qed
next
case False
  have F-1:(b > 0) ∧ (c > 0)
    using False by auto
  have ∃ x. a = x*(b*c) + (a mod (b*c))
    using mod-div-decomp by blast
  then obtain x where a = x*(b*c) + (a mod (b*c))
    by auto
  then have a div c = ((x*(b*c)) div c) + ((a mod (b*c)) div c)
    using div-add1-eq mod-add-self1 mod-add-self2
    mod-by-0 mod-div-trivial mod-prop1 mod-self
    by (metis)
  then have a div c = (((x*b)*c) div c) + ((a mod (b*c)) div c)
    by auto
  then have F-2:a div c = (x*b) + ((a mod (b*c)) div c)
    by (metis F-1 nonzero-mult-div-cancel-left mult.commute neq0-conv)
  have ∃ y. a div c = (y*b) + ((a div c) mod b)
    by (metis add.commute mod-div-mult-eq)
  then obtain y where a div c = (y*b) + ((a div c) mod b)
    by auto
  with F-2 have F-3: (x*b) + ((a mod (b*c)) div c) = (y*b) + ((a div c) mod b)
    by auto
  then have (x*b) - (y * b) = ((a div c) mod b) - ((a mod (b*c)) div c)
    by auto
  then have (x - y) * b = ((a div c) mod b) - ((a mod (b*c)) div c)
    by (metis diff-mult-distrib2 mult.commute)
  then have F-4:b dvd (((a div c) mod b) - ((a mod (b*c)) div c))
    by (metis dvd-eq-mod-eq-0 mod-mult-self1-is-0 mult.commute)
  have F-5:b > ((a div c) mod b)
    by (metis F-1 mod-less-divisor)
  have b*c > (a mod (b*c))
    by (metis False mod-less-divisor neq0-conv)
  moreover then have (b*c) div c > (a mod (b*c)) div c
    by (metis F-1 div-left-ineq nonzero-mult-div-cancel-right neq0-conv)
  then have b > (a mod (b*c)) div c
    by (metis calculation div-right-ineq mult.commute)

```

```

with F-4 F-5
have F-6:((a div c) mod b) - ((a mod (b*c)) div c) = 0
  using less-imp-diff-less nat-dvd-not-less by blast
from F-3 have (y * b) - (x*b)
              = ((a mod (b*c)) div c) - ((a div c) mod b)
  by auto
then have (y - x) * b = ((a mod (b*c)) div c) - ((a div c) mod b)
  by (metis diff-mult-distrib2 mult.commute)
then have F-7:b dvd (((a mod (b*c)) div c) - ((a div c) mod b))
  by (metis dvd-eq-mod-eq-0 mod-mult-self1-is-0 mult.commute)
have F-8:b > ((a div c) mod b)
  by (metis F-1 mod-less-divisor)
have b*c > (a mod (b*c))
  by (metis False mod-less-divisor neq0-conv)
moreover then have (b*c) div c > (a mod (b*c)) div c
  by (metis F-1 div-left-ineq nonzero-mult-div-cancel-right neq0-conv)
then have b > (a mod (b*c)) div c
  by (metis calculation div-right-ineq mult.commute)
with F-7 F-8
have ((a mod (b*c)) div c) - ((a div c) mod b) = 0
  by (metis F-2 cancel-comm-monoid-add-class.diff-cancel mod-if-mod-mult-self3)
with F-6 have ((a mod (b*c)) div c) = ((a div c) mod b)
  by auto
then show ?thesis using False by auto
qed

```

The following lemma proves that the tensor product of matrices is associative

lemma *associativity*:

fixes $M1\ M2\ M3$

shows

$$\begin{aligned}
& (\text{mat } (\text{row-length } M1) (\text{length } M1) M1) \\
& \wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2) \\
& \wedge (\text{mat } (\text{row-length } M3) (\text{length } M3) M3) \\
& \implies \\
& M1 \otimes (M2 \otimes M3) = (M1 \otimes M2) \otimes M3 \text{ (is } ?x \implies ?l = ?r)
\end{aligned}$$

proof–

fix j

assume 0 : $(\text{mat } (\text{row-length } M1) (\text{length } M1) M1)$

$\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$

$\wedge (\text{mat } (\text{row-length } M3) (\text{length } M3) M3)$

have 1 : $\text{length } ((M1 \otimes M2) \otimes M3)$

$$= (\text{length } M1) * (\text{length } M2) * (\text{length } M3)$$

proof–

have $\text{length } (M2 \otimes M3) = (\text{length } M2) * (\text{length } M3)$

by $(\text{metis length-Tensor})$

then have $\text{length } (M1 \otimes (M2 \otimes M3))$

$$= (\text{length } M1) * (\text{length } M2) * (\text{length } M3)$$

using $\text{mult.assoc length-Tensor}$ **by** auto

moreover have $\text{length } (M1 \otimes M2) = (\text{length } M1) * (\text{length } M2)$

by (*metis length-Tensor*)
 ultimately show *?thesis using mult.assoc length-Tensor by auto*
 qed
 have 2: $\text{row-length } ((M1 \otimes M2) \otimes M3)$
 $= (\text{row-length } M1) * (\text{row-length } M2) * (\text{row-length } M3)$
 proof –
 have $\text{row-length } (M2 \otimes M3) = (\text{row-length } M2) * (\text{row-length } M3)$
 using *row-length-mat assoc by auto*
 then have $\text{row-length } (M1 \otimes (M2 \otimes M3))$
 $= (\text{row-length } M1) * (\text{row-length } M2) * (\text{row-length } M3)$
 using *row-length-mat assoc by auto*
 moreover have $\text{row-length } (M1 \otimes M2)$
 $= (\text{row-length } M1) * (\text{row-length } M2)$
 using *row-length-mat by auto*
 ultimately show *?thesis using row-length-mat assoc by auto*
 qed
 have 3:
 $\forall i. \forall j. (((i < ((\text{row-length } M1) * (\text{row-length } M2) * (\text{row-length } M3)))$
 $\wedge (j < (\text{length } M1) * (\text{length } M2) * (\text{length } M3)))$
 \longrightarrow
 $((M1 \otimes M2) \otimes M3)!j!i$
 $= f$
 $((M1 \otimes M2)!j \text{ div } (\text{length } M3)!i \text{ div } (\text{row-length } M3))$
 $(M3)!j \text{ mod } \text{length } M3!i \text{ mod } (\text{row-length } M3))$
 using *0 matrix-Tensor-elements 1 2 effective-well-defined-Tensor*
 length-Tensor row-length-mat
 by *auto*
 moreover have
 $\forall j. (j < (\text{length } M1) * (\text{length } M2) * (\text{length } M3))$
 $\longrightarrow (j \text{ div } (\text{length } M3)) < (\text{length } M1) * (\text{length } M2)$
 apply (*rule allI*)
 apply (*simp add: div-left-ineq*)
 done
 moreover have $\forall i. (i < (\text{row-length } M1) * (\text{row-length } M2) * (\text{row-length } M3))$
 $\longrightarrow (i \text{ div } (\text{row-length } M3))$
 $< (\text{row-length } M1) * (\text{row-length } M2)$
 apply (*rule allI*)
 apply (*simp add: div-left-ineq*)
 done
 ultimately have 4: $\forall i. \forall j. (((i < ((\text{row-length } M1) * (\text{row-length } M2) * (\text{row-length } M3))$
 $M3)))$
 $\wedge (j < (\text{length } M1) * (\text{length } M2) * (\text{length } M3))$
 \longrightarrow
 $((i \text{ div } (\text{row-length } M3)) < (\text{row-length } M1) * (\text{row-length } M2))$
 $\wedge ((j \text{ div } (\text{length } M3)) < (\text{length } M1) * (\text{length } M2))$
 using *allI 0 by auto*
 have $(\text{mat } (\text{row-length } M1) (\text{length } M1) M1)$
 $\wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)$
 using *0 by auto*

then have $\forall i.\forall j.(((i \text{ div } (\text{row-length } M3)) < (\text{row-length } M1)*(\text{row-length } M2))$
 $\wedge ((j \text{ div } (\text{length } M3)) < (\text{length } M1)*(\text{length } M2))$
 \longrightarrow
 $((M1 \otimes M2)!(j \text{ div } (\text{length } M3))!(i \text{ div } \text{row-length } M3))$
 $= f$
 $((M1)!(j \text{ div } (\text{length } M3)) \text{div } (\text{length } M2))$
 $!((i \text{ div } (\text{row-length } M3)) \text{div } (\text{row-length } M2)))$
 $(M2!(j \text{ div } (\text{length } M3)) \text{mod } (\text{length } M2))$
 $!((i \text{ div } (\text{row-length } M3)) \text{mod } (\text{row-length } M2))))$
using *effective-matrix-tensor-elements* **by** *auto*
with 4 have 5: $\forall i.\forall j.(((i < ((\text{row-length } M1)*(\text{row-length } M2)*(\text{row-length } M3)))$
 $\wedge (j < (\text{length } M1)*(\text{length } M2)*(\text{length } M3)))$
 $\longrightarrow ((M1 \otimes M2)!(j \text{ div } (\text{length } M3))!(i \text{ div } \text{row-length } M3))$
 $= f$
 $((M1)!(j \text{ div } (\text{length } M3)) \text{div } (\text{length } M2))$
 $!((i \text{ div } (\text{row-length } M3)) \text{div } (\text{row-length } M2)))$
 $(M2!(j \text{ div } (\text{length } M3)) \text{mod } (\text{length } M2))$
 $!((i \text{ div } (\text{row-length } M3)) \text{mod } (\text{row-length } M2))))$
by *auto*
with 3 have 6:
 $\forall i.\forall j.(((i < ((\text{row-length } M1)*(\text{row-length } M2)*(\text{row-length } M3)))$
 $\wedge (j < (\text{length } M1)*(\text{length } M2)*(\text{length } M3)))$
 \longrightarrow
 $((M1 \otimes M2) \otimes M3)!j!i$
 $= f$
 $(f$
 $((M1)!(j \text{ div } (\text{length } M3)) \text{div } (\text{length } M2))$
 $!((i \text{ div } (\text{row-length } M3)) \text{div } (\text{row-length } M2)))$
 $(M2!(j \text{ div } (\text{length } M3)) \text{mod } (\text{length } M2))$
 $!((i \text{ div } (\text{row-length } M3)) \text{mod } (\text{row-length } M2))))$
 $(M3!(j \text{ mod } \text{length } M3)!(i \text{ mod } (\text{row-length } M3))))$
by *auto*
have $(j \text{ div } (\text{length } M3)) \text{div } (\text{length } M2) = (j \text{ div } ((\text{length } M3)*(\text{length } M2)))$
using *div-mult2-eq* **by** *auto*
moreover have $((i \text{ div } (\text{row-length } M3)) \text{div } (\text{row-length } M2)) = (i \text{ div } ((\text{row-length } M3)*(\text{row-length } M2)))$
using *div-mult2-eq* **by** *auto*
ultimately have *step1*: $\forall i.\forall j.(((i < ((\text{row-length } M1)*(\text{row-length } M2)*(\text{row-length } M3)))$
 $\wedge (j < (\text{length } M1)*(\text{length } M2)*(\text{length } M3)))$
 \longrightarrow
 $((M1 \otimes M2) \otimes M3)!j!i$
 $= f$
 $(f$
 $((M1)!(j \text{ div } ((\text{length } M3)*(\text{length } M2)))! (i \text{ div } ((\text{row-length } M3)*(\text{row-length } M2))))$
 $(M2!(j \text{ div } (\text{length } M3)) \text{mod } (\text{length } M2))!((i \text{ div } (\text{row-length } M3)) \text{mod } (\text{row-length } M2))))$
 $(M3!(j \text{ mod } \text{length } M3)!(i \text{ mod } (\text{row-length } M3))))$

using 6 by (*metis 3 5 div-mult2-eq*)
then have *step1*: $\forall i j.(((i < ((\text{row-length } M1) * (\text{row-length } M2) * (\text{row-length } M3)))$
 $\wedge (j < (\text{length } M1) * (\text{length } M2) * (\text{length } M3)))$
 \longrightarrow
 $((M1 \otimes M2) \otimes M3)!j!i$
 $= f$
 $(f$
 $((M1)! (j \text{ div } ((\text{length } M2) * (\text{length } M3)))! (i \text{ div } ((\text{row-length } M2) * (\text{row-length } M3))))$
 $(M2! ((j \text{ div } (\text{length } M3)) \bmod (\text{length } M2))! ((i \text{ div } (\text{row-length } M3)) \bmod$
 $(\text{row-length } M2))))$
 $(M3! (j \bmod \text{length } M3)! (i \bmod (\text{row-length } M3))))$
by (*metis mult.commute*)
have 7:
 $\forall i. \forall j. (((i < ((\text{row-length } M1) * (\text{row-length } M2) * (\text{row-length } M3)))$
 $\wedge (j < (\text{length } M1) * (\text{length } M2) * (\text{length } M3)))$
 \longrightarrow
 $((M1 \otimes (M2 \otimes M3))!j!i$
 $= f$
 $((M1)! (j \text{ div } (\text{length } (M2 \otimes M3)))! (i \text{ div } (\text{row-length } (M2 \otimes M3))))$
 $((M2 \otimes M3)! (j \bmod \text{length } (M2 \otimes M3))! (i \bmod (\text{row-length } (M2 \otimes$
 $M3))))$
using 0 matrix-Tensor-elements 1 2 effective-well-defined-Tensor
length-Tensor row-length-mat
by auto
then have
 $\forall i. \forall j. (((i < ((\text{row-length } M1) * (\text{row-length } M2) * (\text{row-length } M3)))$
 $\wedge (j < (\text{length } M1) * (\text{length } M2) * (\text{length } M3)))$
 \longrightarrow
 $((M1 \otimes (M2 \otimes M3))!j!i$
 $= f$
 $((M1)! (j \text{ div } ((\text{length } M2) * (\text{length } M3)))! (i \text{ div } ((\text{row-length } M2) * (\text{row-length } M3))))$
 $((M2 \otimes M3)! (j \bmod \text{length } (M2 \otimes M3))! (i \bmod (\text{row-length } (M2 \otimes$
 $M3))))$
using length-Tensor row-length-mat by auto
then have
 $\forall i. \forall j. (((i < ((\text{row-length } M1) * (\text{row-length } M2) * (\text{row-length } M3)))$
 $\wedge (j < (\text{length } M1) * (\text{length } M2) * (\text{length } M3)))$
 \longrightarrow
 $((M1 \otimes (M2 \otimes M3))!j!i$
 $= f$
 $((M1)! (j \text{ div } ((\text{length } M3) * (\text{length } M2)))$
 $! (i \text{ div } ((\text{row-length } M3) * (\text{row-length } M2))))$
 $((M2 \otimes M3)! (j \bmod \text{length } (M2 \otimes M3))$
 $! (i \bmod (\text{row-length } (M2 \otimes M3))))$
using mult.commute by (*metis*)
have 8:

```

       $\forall j.((j < (\text{length } M1)*(\text{length } M2)*(\text{length } M3)))$ 
       $\longrightarrow (j \bmod (\text{length } (M2 \otimes M3))) < (\text{length } (M2 \otimes M3))$ 
proof(cases length (M2  $\otimes$  M3) = 0)
  case True
    have (length M2)*(length M3) = 0
      using length-Tensor True by auto
    then have (length M1)*(length M2)*(length M3) = 0
      by auto
    then show ?thesis by (metis less-nat-zero-code)
  next
  case False
    have length (M2  $\otimes$  M3) > 0
      using False by auto
    then show ?thesis using mod-less-divisor by auto
qed
then have 9:
   $\forall i.((i < (\text{row-length } M1)*(\text{row-length } M2)*(\text{row-length } M3)))$ 
   $\longrightarrow (i \bmod (\text{row-length } (M2 \otimes M3))) < (\text{row-length } (M2 \otimes M3))$ 
proof(cases row-length (M2  $\otimes$  M3) = 0)
  case True
    have (row-length M2)*(row-length M3) = 0
      using True by (metis row-length-mat)
    then have (row-length M1)*(row-length M2)*(row-length M3) = 0
      by auto
    then show ?thesis by (metis less-nat-zero-code)
  next
  case False
    have row-length (M2  $\otimes$  M3) > 0
      using False by auto
    then show ?thesis using mod-less-divisor by auto
qed
with 8 have 10: $\forall i.\forall j.(((i < ((\text{row-length } M1)*(\text{row-length } M2)*(\text{row-length } M3))))$ 
 $\wedge (j < (\text{length } M1)*(\text{length } M2)*(\text{length } M3)))$ 
   $\longrightarrow$ 
   $(i \bmod (\text{row-length } (M2 \otimes M3))) < (\text{row-length } (M2 \otimes M3))$ 
   $\wedge (j \bmod (\text{length } (M2 \otimes M3))) < (\text{length } (M2 \otimes M3))$ 
  by auto
then have 11: $\forall i j.(((i < ((\text{row-length } M1)*(\text{row-length } M2)*(\text{row-length } M3))))$ 
 $\wedge (j < (\text{length } M1)*(\text{length } M2)*(\text{length } M3)))$ 
   $\longrightarrow$ 
   $(i \bmod (\text{row-length } (M2 \otimes M3)))$ 
   $< (\text{row-length } M2)*(\text{row-length } M3)$ 
   $\wedge (j \bmod (\text{length } (M2 \otimes M3))) < (\text{length } M2)*(\text{length } M3))$ 
  using length-Tensor row-length-mat by auto
have (mat (row-length M2) (length M2) M2)
   $\wedge$  (mat (row-length M3) (length M3) M3)
  using 0 by auto
then have  $\forall i j.(((i \bmod (\text{row-length } (M2 \otimes M3)))$ 
 $< (\text{row-length } M2)*(\text{row-length } M3))$ 

```

$$\begin{aligned} & \wedge((j \bmod (\text{length } (M2 \otimes M3))) < (\text{length } M2) * (\text{length } M3)) \\ \longrightarrow & \\ & (((M2 \otimes M3))!(j \bmod (\text{length } (M2 \otimes M3)))(i \bmod \text{row-length } (M2 \otimes M3))) \\ & = f \\ & ((M2)!((j \bmod (\text{length } (M2 \otimes M3))) \text{div } (\text{length } M3)) \\ & \quad !((i \bmod (\text{row-length } (M2 \otimes M3))) \text{div } (\text{row-length } M3))) \\ & (M3)!((j \bmod (\text{length } (M2 \otimes M3))) \bmod (\text{length } M3)) \\ & \quad !((i \bmod (\text{row-length } (M2 \otimes M3))) \bmod (\text{row-length } M3))) \\ & \text{using matrix-Tensor-elements by auto} \\ \text{then have } \forall i j. & \\ & ((i < (\text{row-length } M1) * (\text{row-length } M2) * (\text{row-length } M3)) \\ & \wedge (j < (\text{length } M1) * (\text{length } M2) * (\text{length } M3))) \\ \longrightarrow & \\ & (((M2 \otimes M3))!(j \bmod (\text{length } (M2 \otimes M3))) \\ & \quad ! (i \bmod \text{row-length } (M2 \otimes M3))) \\ & = \\ & f \\ & ((M2)!((j \bmod (\text{length } (M2 \otimes M3))) \text{div } (\text{length } M3)) \\ & \quad !((i \bmod (\text{row-length } (M2 \otimes M3))) \text{div } (\text{row-length } M3))) \\ & (M3)!((j \bmod (\text{length } (M2 \otimes M3))) \bmod (\text{length } M3)) \\ & \quad !((i \bmod (\text{row-length } (M2 \otimes M3))) \bmod (\text{row-length } M3))) \\ & \text{using 11 by auto} \\ \text{moreover then have } \forall j. & (j \bmod (\text{length } (M2 \otimes M3))) \bmod (\text{length } M3) \\ & = j \bmod (\text{length } M3) \\ \text{proof} & \\ \text{have } \forall j. & ((j \bmod (\text{length } (M2 \otimes M3))) \\ & = (j \bmod ((\text{length } M2) * (\text{length } M3)))) \\ & \text{using length-Tensor by auto} \\ \text{moreover have} & \\ \forall j. & ((j \bmod ((\text{length } M2) * (\text{length } M3))) \bmod (\text{length } M3)) \\ & = (j \bmod (\text{length } M3)) \\ & \text{using mod-prop1 by auto} \\ \text{ultimately show ?thesis} & \text{ by auto} \\ \text{qed} & \\ \text{moreover then have } \forall i. & (i \bmod (\text{row-length } (M2 \otimes M3))) \bmod (\text{row-length } M3) \\ & = i \bmod (\text{row-length } M3) \\ \text{proof} & \\ \text{have } \forall i. & ((i \bmod (\text{row-length } (M2 \otimes M3))) \\ & = (i \bmod ((\text{row-length } M2) * (\text{row-length } M3)))) \\ & \text{using row-length-mat by auto} \\ \text{moreover have } \forall i. & ((i \bmod ((\text{row-length } M2) * (\text{row-length } M3))) \\ & \quad \bmod (\text{row-length } M3)) \\ & = (i \bmod (\text{row-length } M3)) \\ & \text{using mod-prop1 by auto} \\ \text{ultimately show ?thesis} & \text{ by auto} \\ \text{qed} & \\ \text{ultimately have 12:} & \forall i j. ((i < (\text{row-length } M1) \\ & \quad * (\text{row-length } M2)) \end{aligned}$$

$$\begin{aligned}
& *(row-length\ M3)) \\
& \wedge(j < (length\ M1)*(length\ M2)*(length\ M3)) \\
\longrightarrow & \\
& (((M2 \otimes M3)!(j\ mod\ (length\ (M2 \otimes M3))) \\
& \quad !(i\ mod\ row-length\ (M2 \otimes M3))) \\
= f & \\
& ((M2)!(j\ mod\ (length\ (M2 \otimes M3))\ div\ (length\ M3)) \\
& \quad !((i\ mod\ (row-length\ (M2 \otimes M3))\ div\ (row-length\ M3))) \\
& (M3!(j\ mod\ (length\ M3))!(i\ mod\ (row-length\ M3)))) \\
& \text{by auto} \\
\text{moreover have } \forall j.(j\ mod\ (length\ (M2 \otimes M3))\ div\ (length\ M3)) & \\
& = (j\ div\ (length\ M3))\ mod\ (length\ M2) \\
\text{proof-} & \\
\text{have } \forall j.((j\ mod\ (length\ (M2 \otimes M3))) & \\
& = (j\ mod\ ((length\ M2)*(length\ M3)))) \\
& \text{using length-Tensor by auto} \\
\text{then show ?thesis using mod-div-relation by auto} & \\
\text{qed} & \\
\text{moreover have } \forall i.(i\ mod\ (row-length\ (M2 \otimes M3))\ div\ (row-length\ M3)) & \\
& = (i\ div\ (row-length\ M3))\ mod\ (row-length\ M2) \\
\text{proof-} & \\
\text{have } \forall i.((i\ mod\ (row-length\ (M2 \otimes M3))) & \\
& = (i\ mod\ ((row-length\ M2)*(row-length\ M3)))) \\
& \text{using row-length-mat by auto} \\
\text{then show ?thesis using mod-div-relation by auto} & \\
\text{qed} & \\
\text{ultimately have } \forall i\ j. & \\
& ((i < (row-length\ M1)*(row-length\ M2)*(row-length\ M3)) \\
& \wedge(j < (length\ M1)*(length\ M2)*(length\ M3))) \\
\longrightarrow & \\
& (((M2 \otimes M3)!(j\ mod\ (length\ (M2 \otimes M3))) \\
& \quad !(i\ mod\ row-length\ (M2 \otimes M3))) \\
= f & \\
& ((M2)!(j\ div\ (length\ M3))\ mod\ (length\ M2)) \\
& \quad !((i\ div\ (row-length\ M3))\ mod\ (row-length\ M2))) \\
& (M3!(j\ mod\ (length\ M3))!(i\ mod\ (row-length\ M3)))) \\
& \text{by auto} \\
\text{with 7 have } 13:\forall i\ j.(((i < ((row-length\ M1)*(row-length\ M2)*(row-length\ M3))) & \\
& \wedge(j < (length\ M1)*(length\ M2)*(length\ M3))) \\
\longrightarrow & \\
& ((M1 \otimes (M2 \otimes M3))!j!i) \\
= f & \\
& ((M1)!(j\ div\ ((length\ M2)*(length\ M3))) \\
& \quad !((i\ div\ ((row-length\ M2)*(row-length\ M3)))) \\
(f & \\
& ((M2)!(j\ div\ (length\ M3))\ mod\ (length\ M2))!((i\ div\ (row-length\ M3))\ mod \\
(row-length\ M2))) & \\
& (M3!(j\ mod\ (length\ M3)) \\
& \quad !((i\ mod\ (row-length\ M3))))))
\end{aligned}$$

using *length-Tensor row-length-mat* **by** *auto*
moreover have $\forall i j. (f$
 $((M1)!(j \text{ div } ((\text{length } M2)*(\text{length } M3)))$
 $!(i \text{ div } ((\text{row-length } M2)*(\text{row-length } M3))))$
 $(f$
 $((M2)!(j \text{ div } (\text{length } M3)) \text{ mod } (\text{length } M2))!((i \text{ div } (\text{row-length}$
 $M3)) \text{ mod } (\text{row-length } M2)))$
 $(M3!(j \text{ mod } (\text{length } M3))$
 $!(i \text{ mod } (\text{row-length } M3))))$
 $= f (f$
 $((M1)!(j \text{ div } ((\text{length } M2)*(\text{length } M3)))$
 $!(i \text{ div } ((\text{row-length } M2)*(\text{row-length } M3))))$
 $((M2)!(j \text{ div } (\text{length } M3)) \text{ mod } (\text{length } M2))$
 $!((i \text{ div } (\text{row-length } M3)) \text{ mod } (\text{row-length } M2))))$
 $(M3!(j \text{ mod } (\text{length } M3))$
 $!(i \text{ mod } (\text{row-length } M3)))$
using *assoc* **by** *auto*
with *13* **have** $\forall i j. (((i < ((\text{row-length } M1)*(\text{row-length } M2)*(\text{row-length } M3)))$
 $\wedge (j < (\text{length } M1)*(\text{length } M2)*(\text{length } M3)))$
 \longrightarrow
 $((M1 \otimes (M2 \otimes M3))!j!i)$
 $= f (f$
 $((M1)!(j \text{ div } ((\text{length } M2)*(\text{length } M3)))$
 $!(i \text{ div } ((\text{row-length } M2)*(\text{row-length } M3))))$
 $((M2)!(j \text{ div } (\text{length } M3)) \text{ mod } (\text{length } M2))$
 $!((i \text{ div } (\text{row-length } M3)) \text{ mod } (\text{row-length } M2))))$
 $(M3!(j \text{ mod } (\text{length } M3))$
 $!(i \text{ mod } (\text{row-length } M3)))$
by *auto*
with *step1* **have** *step2*:
 $\forall i j. (((i < ((\text{row-length } M1)*(\text{row-length } M2)*(\text{row-length } M3)))$
 $\wedge (j < (\text{length } M1)*(\text{length } M2)*(\text{length } M3)))$
 \longrightarrow
 $((M1 \otimes (M2 \otimes M3))!j!i) = (((M1 \otimes M2) \otimes M3)!j!i)$
by *auto*
moreover have *mat* $((\text{row-length } M1)*(\text{row-length } M2)*(\text{row-length } M3))$
 $((\text{length } M1)*(\text{length } M2)*(\text{length } M3))$
 $(M1 \otimes (M2 \otimes M3))$
proof—
have *mat* $((\text{row-length } M2)*(\text{row-length } M3)) ((\text{length } M2)*(\text{length } M3)) (M2$
 $\otimes M3)$
using *0 effective-well-defined-Tensor row-length-mat length-Tensor*
by *auto*
moreover have *mat* $((\text{row-length } M1)*((\text{row-length } (M2 \otimes M3))))$
 $((\text{length } M1)*((\text{length } (M2 \otimes M3))))$
 $(M1 \otimes (M2 \otimes M3))$
using *0 effective-well-defined-Tensor row-length-mat length-Tensor*
by *metis*
ultimately show *?thesis* **using** *row-length-mat length-Tensor mult.assoc*

```

    by (simp add: length-Tensor row-length-mat semigroup-mult-class.mult.assoc)
qed
moreover have mat ((row-length M1)*(row-length M2)*(row-length M3))
  ((length M1)*(length M2)*(length M3))
  ((M1 ⊗ M2) ⊗ M3)
proof-
  have mat ((row-length M1)*(row-length M2)) ((length M1)*(length M2)) (M1
⊗ M2)
    using 0 effective-well-defined-Tensor row-length-mat length-Tensor by
auto
  moreover have mat ((row-length (M1 ⊗ M2))*(row-length M3))
    ((length (M1 ⊗ M2))*(length M3))
    ((M1 ⊗ M2) ⊗ M3)
    using 0 effective-well-defined-Tensor row-length-mat length-Tensor by
metis
  ultimately show ?thesis using row-length-mat length-Tensor by (metis mult.assoc)
qed
  ultimately show ?thesis
    by (auto intro: mat-eqI)
qed
end

```

```

lemma  $\wedge(a::nat) b.(times a b) = (times b a)$ 
  by auto

```

1.2 Associativity and Distributive properties

```

locale plus-mult =
  mult +
  fixes zer::'a
  fixes g:: 'a ⇒ 'a ⇒ 'a (infixl <+> 60)
  fixes inver::'a ⇒ 'a
  assumes plus-comm: g a b = g b a
  assumes plus-assoc: (g (g a b) c) = (g a (g b c))
  assumes plus-left-id: g zer x = x
  assumes plus-right-id: g x zer = x
  assumes plus-left-distributivity: f a (g b c) = g (f a b) (f a c)
  assumes plus-right-distributivity: f (g a b) c = g (f a c) (f b c)
  assumes plus-left-inverse: (g x (inver x)) = zer
  assumes plus-right-inverse: (g (inver x) x) = zer

```

```

context plus-mult
begin

```

```

lemma fixes M1 M2 M3
  shows (mat (row-length M1) (length M1) M1)
     $\wedge$ (mat (row-length M2) (length M2) M2)

```

$\wedge(\text{mat } (\text{row-length } M3) (\text{length } M3) M3)$
 $\implies (M1 \otimes (M2 \otimes M3)) = ((M1 \otimes M2) \otimes M3)$
using *associativity by auto*

matrix__mult refers to multiplication of matrices in the locale plus__mult

abbreviation *matrix-mult*::'a mat \Rightarrow 'a mat \Rightarrow 'a mat (**infixl** $\langle \circ \rangle$ 65)

where

matrix-mult M1 M2 \equiv (*mat-multI* zer g f (row-length M1) M1 M2)

definition *scalar-product* :: 'a vec \Rightarrow 'a vec \Rightarrow 'a **where**

scalar-product v w = *scalar-prodI* zer g f v w

lemma *ma* :

assumes *wf1*: mat nr n m1

and *wf2*: mat n nc m2

and *i*: i < nr

and *j*: j < nc

shows *mat-multI* zer g f nr m1 m2 ! j ! i

= *scalar-prodI* zer g f (row m1 i) (col m2 j)

using *mat-mult-index* i j *wf1* *wf2* **by** *metis*

lemma *matrix-index*:

assumes *wf1*: mat (row-length m1) n m1

and *wf2*: mat n nc m2

and *i*: i < (row-length m1)

and *j*: j < nc

shows *matrix-mult* m1 m2 ! j ! i

= *scalar-product* (row m1 i) (col m2 j)

using *wf1* *wf2* i j *ma* *scalar-product-def* **by** *auto*

lemma *unique-row-col*:

assumes mat nr1 nc1 M **and** mat nr2 nc2 M **and** M \neq []

shows nr1 = nr2 **and** nc1 = nc2

proof(*cases* M)

case *Nil*

show nr1 = nr2 **using** *assms*(3) *Nil* **by** *auto*

next

case (*Cons* v M)

have 1:v \in set (v#M)

using *Cons* **by** *auto*

then **have** length v = nr1

using *assms*(1) *mat-def* *Ball-def* *vec-def* *Cons* **by** *metis*

moreover **then** **have** length v = nr2

using 1 *assms*(2) *mat-def* *Ball-def* *vec-def* *Cons* **by** *metis*

ultimately **show** nr1 = nr2

by *auto*

next

have length M = nc1

```

    using mat-def assms(1) by auto
  moreover have length M = nc2
    using mat-def assms(2) by auto
  ultimately show nc1 = nc2
    by auto
qed

```

```

lemma matrix-mult-index:
  assumes m1 ≠ []
  and wf1: mat nr n m1
  and wf2: mat n nc m2
  and i: i < nr
  and j: j < nc
  shows matrix-mult m1 m2 ! j ! i = scalar-product (row m1 i) (col m2 j)
  using matrix-index unique-row-col assms by (metis matrix-row-length)

```

the following definition checks if the given four matrices are such that the compositions in the mixed-product property which will be proved, hold true. It further checks that the matrices are non empty and valid

```

definition matrix-match::'a mat ⇒ 'a mat ⇒ 'a mat ⇒ 'a mat ⇒ bool
where
matrix-match A1 A2 B1 B2 ≡
  (mat (row-length A1) (length A1) A1)
  ∧ (mat (row-length A2) (length A2) A2)
  ∧ (mat (row-length B1) (length B1) B1)
  ∧ (mat (row-length B2) (length B2) B2)
  ∧ (length A1 = row-length A2)
  ∧ (length B1 = row-length B2)
  ∧ (A1 ≠ []) ∧ (A2 ≠ []) ∧ (B1 ≠ []) ∧ (B2 ≠ [])

```

```

lemma non-empty-mat-mult:
  assumes wf1:mat nr n A
  and wf2:mat n nc B
  and A ≠ [] and B ≠ []
  shows A ∘ B ≠ []
proof-
  have mat nr nc (A ∘ B)
    using assms(1) assms(2) mat-mult assms(3) matrix-row-length unique-row-col(1)
  by (metis)
  then have length (A ∘ B) = nc
    using mat-def by auto
  moreover have nc > 0
proof-
  have length B = nc
    using assms(2) mat-def by auto
  then show ?thesis using assms(4) by auto
qed
moreover then have length (A ∘ B) > 0

```

by (*metis calculation*(1))
 then show *?thesis* by auto
 qed

lemma *tensor-compose-distribution1*:

assumes *wf1*:*mat* (row-length *A1*) (length *A1*) *A1*
 and *wf2*:*mat* (row-length *A2*) (length *A2*) *A2*
 and *wf3*:*mat* (row-length *B1*) (length *B1*) *B1*
 and *wf4*:*mat* (row-length *B2*) (length *B2*) *B2*
 and *matchAA*:length *A1* = row-length *A2*
 and *matchBB*:length *B1* = row-length *B2*
 and *non-Nil*:(*A1* ≠ []) ∧ (*A2* ≠ []) ∧ (*B1* ≠ []) ∧ (*B2* ≠ [])
shows *mat* ((row-length *A1*)*(row-length *B1*))
 ((length *A2*)*(length *B2*))
 ((*A1* ◦ *A2*) ⊗ (*B1* ◦ *B2*))

proof –

have *0*:*mat* (row-length *A1*) (length *A2*) (*matrix-mult* *A1* *A2*)
 using *wf1 wf2 mat-mult matchAA* by auto
then have *1*:*mat* (row-length (*A1* ◦ *A2*)) (length (*A1* ◦ *A2*)) (*matrix-mult* *A1* *A2*)
 by (*metis matrix-row-length*)
then have *2*: (row-length (*A1* ◦ *A2*)) = (row-length *A1*) and length (*A1* ◦ *A2*)
 = length *A2*
 using *non-empty-mat-mult unique-row-col 0*
 apply (*metis length-0-conv mat-empty-column-length non-Nil*)
 by (*metis 0 1 mat-empty-column-length unique-row-col*(2))
moreover have *3*:*mat* (row-length *B1*) (length *B2*) (*matrix-mult* *B1* *B2*)
 using *wf3 wf4 matchBB mat-mult* by auto
then have *4*:*mat* (row-length (*B1* ◦ *B2*)) (length (*B1* ◦ *B2*)) (*matrix-mult* *B1* *B2*)
 by (*metis matrix-row-length*)
then have *5*: (row-length (*B1* ◦ *B2*)) = (row-length *B1*) and length (*B1* ◦ *B2*)
 = length *B2*
 using *non-empty-mat-mult unique-row-col 3*
 apply (*metis length-0-conv mat-empty-column-length non-Nil*)
 by (*metis 3 4 mat-empty-column-length unique-row-col*(2))
then show *?thesis* using *1 4 5 well-defined-Tensor*
 by (*metis 2 calculation*(2))

qed

lemma *effective-tensor-compose-distribution1*:

matrix-match *A1* *A2* *B1* *B2* ⇒ *mat* ((row-length *A1*)*(row-length *B1*))
 ((length *A2*)*(length *B2*))
 ((*A1* ◦ *A2*) ⊗ (*B1* ◦ *B2*))
 using *tensor-compose-distribution1 unfolding matrix-match-def* by auto

lemma *tensor-compose-distribution2*:

assumes *wf1*:*mat* (row-length *A1*) (length *A1*) *A1*

and $wf2:mat$ (row-length $A2$) (length $A2$) $A2$
and $wf3:mat$ (row-length $B1$) (length $B1$) $B1$
and $wf4:mat$ (row-length $B2$) (length $B2$) $B2$
and $matchAA: length A1 = row-length A2$
and $matchBB: length B1 = row-length B2$
and $non-Nil: (A1 \neq []) \wedge (A2 \neq []) \wedge (B1 \neq []) \wedge (B2 \neq [])$
shows mat ((row-length $A1$)*(row-length $B1$))
((length $A2$)*(length $B2$))
(($A1 \otimes B1$) \circ ($A2 \otimes B2$))

proof–

have mat

((row-length $A1$)*(row-length $B1$))
((length $A1$)*(length $B1$))
($A1 \otimes B1$)

using $wf1 wf3$ well-defined-Tensor **by** *auto*

moreover have mat

((row-length $A2$)*(row-length $B2$))
((length $A2$)*(length $B2$))
($A2 \otimes B2$)

using $wf2 wf4$ well-defined-Tensor **by** *auto*

moreover have ((length $A1$)*(length $B1$))
= ((row-length $A2$)*(row-length $B2$))

using $matchAA matchBB$ **by** *auto*

ultimately show *?thesis using mat-mult row-length-mat by simp*
qed

theorem *tensor-non-empty*: **assumes** $A \neq []$ **and** $B \neq []$

shows $A \otimes B \neq []$

using $assms(1) assms(2)$ length-0-conv length-Tensor mult-is-0 **by** *metis*

theorem *non-empty-distribution*:

assumes mat $nr1$ $n1$ $A1$

and mat $n1$ $nc1$ $A2$

and mat $nr2$ $n2$ $B1$

and mat $n2$ $nc2$ $B2$

and $A1 \neq []$ **and** $B1 \neq []$ **and** $A2 \neq []$ **and** $B2 \neq []$

shows $((A1 \circ A2) \otimes (B1 \circ B2)) \neq []$

proof–

have $A1 \circ A2 \neq []$

using $assms$ non-empty-mat-mult **by** *auto*

moreover have $B1 \circ B2 \neq []$

using $assms$ non-empty-mat-mult **by** *auto*

ultimately show *?thesis using tensor-non-empty by auto*

qed

lemma *effective-tensor-compose-distribution2:matrix-match* $A1 A2 B1 B2 \implies$

mat ((row-length $A1$)*(row-length $B1$))
((length $A2$)*(length $B2$))
(($A1 \otimes B1$) \circ ($A2 \otimes B2$))

shows

$$\begin{aligned} & ((\text{matrix-mult } A1 \ A2) \otimes (\text{matrix-mult } B1 \ B2))!j!i \\ &= f(\text{scalar-product } (\text{row } A1 \ (i \ \text{div } (\text{row-length } B1))) \\ &\quad (\text{col } A2 \ (j \ \text{div } (\text{length } B2)))) \\ &\quad (\text{scalar-product } (\text{row } B1 \ (i \ \text{mod } (\text{row-length } B1))) \\ &\quad (\text{col } B2 \ (j \ \text{mod } (\text{length } B2)))) \end{aligned}$$

proof–

have $0:((\text{matrix-mult } A1 \ A2) \otimes (\text{matrix-mult } B1 \ B2)) \neq []$
using *non-empty-distribution* **assms** **by** *auto*
then have $1:\text{mat } ((\text{row-length } A1) * (\text{row-length } B1))$
 $((\text{length } A2) * (\text{length } B2))$
 $((A1 \circ A2) \otimes (B1 \circ B2))$
using *tensor-compose-distribution1* **assms** **by** *auto*
then have $2:\text{mat } (\text{row-length } ((A1 \circ A2) \otimes (B1 \circ B2)))$
 $(\text{length } ((A1 \circ A2) \otimes (B1 \circ B2)))$
 $((A1 \circ A2) \otimes (B1 \circ B2))$
by *metis matrix-row-length*
then have $3:((\text{row-length } A1) * (\text{row-length } B1))$
 $= (\text{row-length } ((A1 \circ A2) \otimes (B1 \circ B2)))$
and $((\text{length } A2) * (\text{length } B2)) = (\text{length } ((A1 \circ A2) \otimes (B1 \circ B2)))$
using *0 1 unique-row-col*
apply *metis*
using *0 1 2 unique-row-col* **by** *metis*
then have $i:(i < ((\text{row-length } A1) * (\text{row-length } B1)))$
 $= (i < (\text{row-length } ((A1 \circ A2) \otimes (B1 \circ B2))))$
by *auto*
moreover have $j:(j < ((\text{length } A2) * (\text{length } B2)))$
 $= (j < (\text{length } ((A1 \circ A2) \otimes (B1 \circ B2))))$
using $\langle \text{length } A2 * \text{length } B2 = \text{length } (A1 \circ A2 \otimes B1 \circ B2) \rangle$
by *metis*
have $4:\text{mat } (\text{row-length } A1) (\text{length } A2) (A1 \circ A2)$
using *assms mat-mult* **by** *auto*
then have $5:\text{mat } (\text{row-length } (A1 \circ A2)) (\text{length } (A1 \circ A2)) (A1 \circ A2)$
using *matrix-row-length* **by** *metis*
with 4 **have** $6:\text{row-length } A1 = \text{row-length } (A1 \circ A2)$
by *metis 0 Tensor.simps(1) unique-row-col(1)*
with $4 \ 5$ **have** $7:\text{length } A2 = \text{length } (A1 \circ A2)$
by *metis mat-empty-column-length unique-row-col(2)*
then have $8:\text{mat } (\text{row-length } B1) (\text{length } B2) (B1 \circ B2)$
using *assms mat-mult* **by** *auto*
then have $9:\text{mat } (\text{row-length } (B1 \circ B2)) (\text{length } (B1 \circ B2)) (B1 \circ B2)$
using *matrix-row-length* **by** *metis*
with $7 \ 8$ **have** $10:\text{row-length } B1 = \text{row-length } (B1 \circ B2)$
by *metis 3 6 assms(8) less-nat-zero-code mult-cancel2 mult-is-0 mult commute*
row-length-mat
with $7 \ 8 \ 9$ **have** $11:\text{length } B2 = \text{length } (B1 \circ B2)$
by *metis mat-empty-column-length unique-row-col(2)*
from $6 \ 10$ **have** $12:$
 $(i < ((\text{row-length } A1) * (\text{row-length } B1)))$

$= (i < (\text{row-length } (A1 \circ A2)) * (\text{row-length } (B1 \circ B2)))$

by auto

then have 13: $(i < (\text{row-length } (A1 \circ A2)) * (\text{row-length } (B1 \circ B2)))$

using assms by auto

from 7 11 have 14:

$(j < ((\text{length } A2) * (\text{length } B2)))$
 $= (j < (\text{length } (A1 \circ A2)) * (\text{length } (B1 \circ B2)))$

by auto

then have 15: $(j < (\text{length } (A1 \circ A2)) * (\text{length } (B1 \circ B2)))$

using assms by auto

then have step-1: $((A1 \circ A2) \otimes (B1 \circ B2))!j!i$
 $= f ((A1 \circ A2)! (j \text{ div } (\text{length } (B1 \circ B2)))$
 $\quad ! (i \text{ div } (\text{row-length } (B1 \circ B2))))$
 $\quad ((B1 \circ B2)! (j \text{ mod } \text{length } (B1 \circ B2)))$
 $\quad ! (i \text{ mod } (\text{row-length } (B1 \circ B2))))$

using 5 9 13 15 effective-matrix-Tensor-elements by auto

then have $((A1 \circ A2) \otimes (B1 \circ B2))!j!i$
 $= f ((A1 \circ A2)! (j \text{ div } (\text{length } B2))! (i \text{ div } (\text{row-length } B1)))$
 $\quad ((B1 \circ B2)! (j \text{ mod } \text{length } B2)! (i \text{ mod } (\text{row-length } B1)))$

using 10 11 by auto

moreover have $((A1 \circ A2)! (j \text{ div } (\text{length } B2))! (i \text{ div } (\text{row-length } B1)))$
 $= (\text{scalar-product } (\text{row } A1 \ (i \text{ div } (\text{row-length } B1))) \ (\text{col } A2 \ (j \text{ div } (\text{length } B2))))$

proof-

have $j \text{ div } (\text{length } B2) < (\text{length } A2)$

using div-left-ineq assms by auto

moreover have $i \text{ div } (\text{row-length } B1) < (\text{row-length } A1)$

using assms div-left-ineq by auto

moreover have $\text{mat } (\text{length } A1) \ (\text{length } A2) \ A2$

using wf2 matchAA by auto

ultimately show ?thesis using wf1 non-Nil matrix-mult-index by blast

qed

moreover have $((B1 \circ B2)! (j \text{ mod } (\text{length } B2))! (i \text{ mod } (\text{row-length } B1)))$
 $= (\text{scalar-product}$
 $\quad (\text{row } B1 \ (i \text{ mod } (\text{row-length } B1)))$
 $\quad (\text{col } B2 \ (j \text{ mod } (\text{length } B2))))$

proof-

have $j < (\text{length } A2) * (\text{length } B2)$

using assms by auto

then have $j \text{ mod } (\text{length } B2) < (\text{length } B2)$

by (metis calculation less-nat-zero-code mod-less-divisor mult-is-0 neq0-conv)

moreover have $i \text{ mod } (\text{row-length } B1) < (\text{row-length } B1)$

by (metis assms(8) less-nat-zero-code mod-less-divisor mult-is-0 neq0-conv)

moreover have $\text{mat } (\text{length } B1) \ (\text{length } B2) \ B2$

using wf4 matchBB by auto

ultimately show ?thesis

using wf3 non-Nil matrix-mult-index by blast

qed
ultimately show *?thesis by auto*
qed

lemma *effective-elements-matrix-distribution1*:
matrix-compose-cond A1 A2 B1 B2 i j \implies
 $((\text{matrix-mult } A1 \ A2) \otimes (\text{matrix-mult } B1 \ B2))!j!i$
 $= f$ (*scalar-product* (*row A1* (*i div (row-length B1)*))) (*col A2* (*j div (length B2)*)))
 $(\text{scalar-product} (\text{row } B1 \ (i \text{ mod } (\text{row-length } B1))) (\text{col } B2 \ (j \text{ mod } (\text{length } B2))))$
using *elements-matrix-distribution-1 matrix-compose-cond-def* **by auto**

lemma *matrix-match-condn-1*:
matrix-match A1 A2 B1 B2
 $\wedge(i < (\text{row-length } A1) * (\text{row-length } B1))$
 $\wedge(j < (\text{length } A2) * (\text{length } B2))$
 $\implies ((\text{matrix-mult } A1 \ A2) \otimes (\text{matrix-mult } B1 \ B2))!j!i$
 $= f$
 $(\text{scalar-product}$
 $(\text{row } A1 \ (i \text{ div } (\text{row-length } B1)))$
 $(\text{col } A2 \ (j \text{ div } (\text{length } B2))))$
 $(\text{scalar-product}$
 $(\text{row } B1 \ (i \text{ mod } (\text{row-length } B1)))$
 $(\text{col } B2 \ (j \text{ mod } (\text{length } B2))))$
using *elements-matrix-distribution-1 unfolding matrix-match-def* **by auto**

lemma *effective-matrix-match-condn-1*:
assumes (*matrix-match A1 A2 B1 B2*)
shows $\forall i \ j. ((i < (\text{row-length } A1) * (\text{row-length } B1))$
 $\wedge(j < (\text{length } A2) * (\text{length } B2))$
 $\longrightarrow ((A1 \circ A2) \otimes (B1 \circ B2))!j!i$
 $= f$
 $(\text{scalar-product}$
 $(\text{row } A1 \ (i \text{ div } (\text{row-length } B1)))$
 $(\text{col } A2 \ (j \text{ div } (\text{length } B2))))$
 $(\text{scalar-product}$
 $(\text{row } B1 \ (i \text{ mod } (\text{row-length } B1)))$
 $(\text{col } B2 \ (j \text{ mod } (\text{length } B2))))$
using *assms matrix-match-condn-1 unfolding matrix-match-def*
by auto

theorem *elements-matrix-distribution2*:
fixes *A1 A2 B1 B2 i j*
assumes *wf1:mat (row-length A1) (length A1) A1*
and *wf2:mat (row-length A2) (length A2) A2*
and *wf3:mat (row-length B1) (length B1) B1*
and *wf4:mat (row-length B2) (length B2) B2*
and *matchAA:length A1 = row-length A2*

and *matchBB*: $\text{length } B1 = \text{row-length } B2$
and *non-Nil*: $(A1 \neq []) \wedge (A2 \neq []) \wedge (B1 \neq []) \wedge (B2 \neq [])$
and $i < (\text{row-length } A1) * (\text{row-length } B1)$ **and** $j < (\text{length } A2) * (\text{length } B2)$
shows
 $((A1 \otimes B1) \circ (A2 \otimes B2))!j!i$
 $= \text{scalar-product}$
 $(\text{vec-vec-Tensor}$
 $(\text{row } A1 \ (i \ \text{div} \ \text{row-length } B1))$
 $(\text{row } B1 \ (i \ \text{mod} \ \text{row-length } B1)))$
 $(\text{vec-vec-Tensor}$
 $(\text{col } A2 \ (j \ \text{div} \ \text{length } B2))$
 $(\text{col } B2 \ (j \ \text{mod} \ \text{length } B2)))$

proof –
have $1:\text{mat}$
 $((\text{row-length } A1) * (\text{row-length } B1))$
 $((\text{length } A1) * (\text{length } B1))$
 $(A1 \otimes B1)$
using *wf1 wf3 well-defined-Tensor by auto*
moreover have $2:\text{mat}$
 $((\text{row-length } A2) * (\text{row-length } B2))$
 $((\text{length } A2) * (\text{length } B2))$
 $(A2 \otimes B2)$
using *wf2 wf4 well-defined-Tensor by auto*
moreover have $3:((\text{length } A1) * (\text{length } B1))$
 $= ((\text{row-length } A2) * (\text{row-length } B2))$
using *matchAA matchBB by auto*
ultimately have $4:((A1 \otimes B1) \circ (A2 \otimes B2))!j!i$
 $= \text{scalar-product } (\text{row } (A1 \otimes B1) \ i) \ (\text{col } (A2 \otimes B2) \ j)$
using *i j matrix-mult-index non-Nil mat-mult-index*
 $\text{row-length-mat scalar-product-def}$
by auto
moreover have $(\text{row } (A1 \otimes B1) \ i)$
 $= \text{vec-vec-Tensor}$
 $(\text{row } A1 \ (i \ \text{div} \ \text{row-length } B1))$
 $(\text{row } B1 \ (i \ \text{mod} \ \text{row-length } B1))$
using *wf1 wf3 i effective-row-formula by auto*
moreover have $\text{col } (A2 \otimes B2) \ j = \text{vec-vec-Tensor } (\text{col } A2 \ (j \ \text{div} \ \text{length } B2))$
 $(\text{col } B2 \ (j \ \text{mod} \ \text{length } B2))$
using *wf2 wf4 j col-formula by auto*
ultimately show *?thesis by auto*
qed

lemma *matrix-match-condn-2*:
matrix-match $A1 \ A2 \ B1 \ B2$
 $\wedge ((i < (\text{row-length } A1) * (\text{row-length } B1))$
 $\wedge (j < (\text{length } A2) * (\text{length } B2)))$
 $\implies ((A1 \otimes B1) \circ (A2 \otimes B2))!j!i$
 $= \text{scalar-product}$

```

      (vec-vec-Tensor
        (row A1 (i div row-length B1))
        (row B1 (i mod row-length B1)))
      (vec-vec-Tensor
        (col A2 (j div length B2))
        (col B2 (j mod length B2)))

```

using *elements-matrix-distribution2* **unfolding** *matrix-match-def* **by** *auto*

lemma *effective-matrix-match-condn-2*:
assumes (*matrix-match* A1 A2 B1 B2)
shows $\forall i j. ((i < (\text{row-length } A1) * (\text{row-length } B1))$
 $\wedge (j < (\text{length } A2) * (\text{length } B2))$
 $\longrightarrow ((A1 \otimes B1) \circ (A2 \otimes B2))!j!i$
 $= \text{scalar-product}$
 (*vec-vec-Tensor*
 (*row* A1 (i div row-length B1))
 (*row* B1 (i mod row-length B1)))
 (*vec-vec-Tensor*
 (*col* A2 (j div length B2))
 (*col* B2 (j mod length B2)))

using *assms matrix-match-condn-2* **unfolding** *matrix-match-def* **by** *auto*

lemma *zip-Nil*: *zip* [] [] = []
using *zip-def* **by** *auto*

lemma *zer-left-mult*: *f* zer *x* = zer

proof –

have *g* zer zer = zer

using *plus-left-id* **by** *auto*

then have *f* zer *x* = *f* (*g* zer zer) *x*

by *auto*

then have *f* zer *x* = (*f* zer *x*) + (*f* zer *x*)

using *plus-right-distributivity* **by** *auto*

then have (*f* zer *x*) + (*inver* (*f* zer *x*)) = (*f* zer *x*) + (*f* zer *x*) + (*inver* (*f* zer *x*))

by *auto*

then have zer = (*f* zer *x*) + zer

using *plus-left-inverse plus-assoc* **by** (*metis*)

then show *?thesis*

using *plus-right-id* **by** *simp*

qed

lemma *zip-Cons*: (*length* *v* = *length* *w*) \implies *zip* (*a*#*v*) (*b*#*w*) = (*a*,*b*)#(*zip* *v* *w*)
unfolding *zip-def* **by** *auto*

lemma *scalar-product-times*:

$\forall w1 w2. (\text{length } w1 = \text{length } w2) \wedge (\text{length } w1 = n) \longrightarrow$
 $(f (x*y) (\text{scalar-product } w1 w2))$

$$= (\text{scalar-product} \\ (\text{times } x \ w1) \\ (\text{times } y \ w2))$$

```

apply(rule allI)
apply (rule allI)
proof(induct n)
case 0
  have (length w1 = length w2)  $\wedge$ (length w1 = 0)  $\implies$  ?case
  proof-
    assume assms:(length w1 = length w2)  $\wedge$ (length w1 = 0)
    have 1: w1 = []
      using assms by auto
    moreover have 2:(length w1 = length w2)  $\wedge$ (length w1 = 0)  $\longrightarrow$  w2 = []
      by auto
    ultimately have (length w1 = length w2)  $\wedge$ (length w1 = 0)
       $\longrightarrow$  scalar-product w1 w2 = zer
      unfolding scalar-product-def scalar-prodI-def by auto
    then have 3:(length w1 = length w2)  $\wedge$ (length w1 = 0)
       $\longrightarrow$  (f (x*y) (scalar-product w1 w2)) = zer
      using comm zer-left-mult by metis
    then have times x w1 = []
      using 1 by auto
    moreover have times y w2 = []
      using 2 assms by auto
    ultimately have (scalar-product (times x w1) (times y w2)) = zer
      unfolding scalar-product-def scalar-prodI-def by auto
    with 3 show ?thesis by auto
  qed
then show ?case by auto
next
case (Suc k)
  have (length w1 = length w2)  $\wedge$ (length w1 = (Suc k))  $\implies$  ?case
  proof-
    assume assms:(length w1 = length w2)  $\wedge$ (length w1 = (Suc k))
    have  $\exists$  a1 u1.(w1 = a1#u1) $\wedge$ (length u1 = k)
      using assms by (metis length-Suc-conv)
    then obtain a1 u1 where (w1 = a1#u1) $\wedge$ (length u1 = k)
      by auto
    then have Cons-1:(w1 = a1#u1) $\wedge$ (length u1 = k)
      by auto
    have length w2 = (Suc k)
      using assms by auto
    then have  $\exists$  a2 u2.(w2 = a2#u2) $\wedge$ (length u2 = k)
      using assms by (metis length-Suc-conv)
    then obtain a2 u2 where (w2 = a2#u2) $\wedge$ (length u2 = k)
      by auto
    then have Cons-2:(w2 = a2#u2) $\wedge$ (length u2 = k)
      by auto
    then have (length u1 = length u2) $\wedge$ (length u1 = k)

```

```

    using Cons-1 by auto
  then have Cons-3:  $x * y * \text{scalar-product } u1 \ u2$ 
    =  $\text{scalar-product } (\text{times } x \ u1) \ (\text{times } y \ u2)$ 
    using Suc assms by auto
  have  $\text{scalar-product } (a1 \# u1) \ (a2 \# u2) = (a1 * a2) + (\text{scalar-product } u1 \ u2)$ 
    unfolding scalar-product-def scalar-prodI-def zip-def by auto
  then have  $\text{scalar-product } w1 \ w2 = (a1 * a2) + (\text{scalar-product } u1 \ u2)$ 
    using Cons-1 Cons-2 by auto
  then have  $(x * y) * (\text{scalar-product } w1 \ w2)$ 
    =  $((x * y) * (a1 * a2)) + ((x * y) * (\text{scalar-product } u1 \ u2))$ 
    using plus-right-distributivity by (metis plus-left-distributivity)
  then have Cons-4:  $(x * y) * (\text{scalar-product } w1 \ w2)$ 
    =  $(x * a1 * y * a2) + ((x * y) * (\text{scalar-product } u1 \ u2))$ 
    using comm assoc by metis
  have  $(\text{times } x \ w1) = (x * a1) \# (\text{times } x \ u1)$ 
    using times.simps Cons-1 by auto
  moreover have  $(\text{times } y \ w2) = (y * a2) \# (\text{times } y \ u2)$ 
    using times.simps Cons-2 by auto
  ultimately have Cons-5:  $\text{scalar-product } (\text{times } x \ w1) \ (\text{times } y \ w2)$ 
    =  $\text{scalar-product}$ 
       $((x * a1) \# (\text{times } x \ u1))$ 
       $((y * a2) \# (\text{times } y \ u2))$ 
    by auto
  then have ... =  $((x * a1) * (y * a2))$ 
    +  $\text{scalar-product } (\text{times } x \ u1) \ (\text{times } y \ u2)$ 
    unfolding scalar-product-def scalar-prodI-def zip-def by auto
  with Cons-3 Cons-4 Cons-5 show ?thesis using assoc by auto
qed
then show ?case by auto
qed

```

lemma *effective-scalar-product-times*:
assumes $(\text{length } w1 = \text{length } w2)$
shows $(f \ (x * y) \ (\text{scalar-product } w1 \ w2))$
 $= (\text{scalar-product } (\text{times } x \ w1) \ (\text{times } y \ w2))$
using *scalar-product-times assms* **by** auto

lemma *zip-append*: $(\text{length } zs = \text{length } ws) \wedge (\text{length } xs = \text{length } ys)$
 $\implies (\text{zip } (xs @ zs) \ (ys @ ws)) = (\text{zip } xs \ ys) @ (\text{zip } zs \ ws)$
using *zip-append1 zip-append2* **by** auto

lemma *scalar-product-append*:
 $\forall xs \ ys \ zs \ ws. (\text{length } zs = \text{length } ws)$
 $\wedge (\text{length } xs = \text{length } ys)$
 $\wedge (\text{length } xs = n) \implies$
 $(\text{scalar-product } (xs @ zs) \ (ys @ ws))$

$$= (\text{scalar-product } xs \ ys) \\ + (\text{scalar-product } zs \ ws)$$

apply(rule allI)
apply(rule allI)
apply(rule allI)
apply(rule allI)
proof(induct n)
case 0
have (length zs = length ws) \wedge (length xs = length ys) \wedge (length xs = 0)
 \implies
(scalar-product (xs@zs) (ys@ws))
= (scalar-product xs ys)
+ (scalar-product zs ws)

proof–
assume *assms*:(length zs = length ws) \wedge (length xs = length ys)
 \wedge (length xs = 0)
have 1:xs = []
using *assms* **by** auto
moreover **have** 2:ys = []
using *assms* **by** auto
ultimately **have** scalar-product xs ys = zer
unfolding scalar-product-def scalar-prodI-def zip-def **by** auto
then **have** (scalar-product xs ys) + (scalar-product zs ws)
= (scalar-product zs ws)
using plus-left-id **by** auto
moreover **have** (scalar-product (xs@zs) (ys@ws)) = (scalar-product zs ws)
using 1 2 **by** auto
ultimately **show** ?thesis **by** auto
qed
then **show** ?case **by** auto
next
case (Suc k)
have (length zs = length ws) \wedge (length xs = length ys) \wedge (length xs = (Suc k)) \implies
(scalar-product (xs@zs) (ys@ws))
= (scalar-product xs ys)
+ (scalar-product zs ws)

proof–
assume *assms*:(length zs = length ws)
 \wedge (length xs = length ys)
 \wedge (length xs = (Suc k))
have $\exists x \ xss. (xs = x\#xss) \wedge (length \ xss = k)$
using *assms* **by** (metis Suc-length-conv)
then **obtain** x xss **where** (xs = x#xss) \wedge (length xss = k)
by auto
then **have** 1:(xs = x#xss) \wedge (length xss = k)
by auto
have $\exists y \ yss. (ys = y\#yss) \wedge (length \ yss = k)$
using *assms* **by** (metis Suc-length-conv)
then **obtain** y yss **where** (ys = y#yss) \wedge (length yss = k)

by auto
then have 2: $(ys = y\#yss) \wedge (\text{length } yss = k)$
by auto
with 1 **have** $\text{length } xss = \text{length } yss \wedge \text{length } xss = k$
by auto
then have 3: $(\text{scalar-product } (xss@zs) (yss@ws))$
 $= (\text{scalar-product } xss \ yss)$
 $+ (\text{scalar-product } zs \ ws)$
using 1 2 *assms Suc* **by auto**
then have 4: $(\text{scalar-product } ((x\#xss)@zs) ((y\#yss)@ws)) =$
 $(\text{scalar-product } (x\#(xss@zs)) (y\#(yss@ws)))$
by auto
then have ... = $(x*y) + (\text{scalar-product } (xss@zs) (yss@ws))$
unfolding *scalar-product-def scalar-prodI-def*
using *zip-Cons scalar-prodI-def scalar-prod-cons*
by (*metis*)
with 4 **have** 5: $(\text{scalar-product } (xs@zs) ((ys)@ws))$
 $= (x*y) + (\text{scalar-product } (xss@zs) (yss@ws))$
using 1 2 **by auto**
moreover have $(\text{scalar-product } xs \ ys) = (x*y) + (\text{scalar-product } xss \ yss)$
unfolding *scalar-product-def scalar-prodI-def*
using *zip-Cons*
by (*metis* 1 2 *scalar-prodI-def scalar-prod-cons*)
moreover then have $(\text{scalar-product } xs \ ys) + (\text{scalar-product } zs \ ws)$
 $= (x*y)$
 $+ (\text{scalar-product } xss \ yss)$
 $+ (\text{scalar-product } zs \ ws)$
by auto
ultimately show ?thesis **using** 3 *plus-assoc* **by auto**
qed
then show ?case **by auto**
qed

lemma *effective-scalar-product-append*:

assumes $\text{length } zs = \text{length } ws$ **and** $(\text{length } xs = \text{length } ys)$

shows $(\text{scalar-product } (xs@zs) (ys@ws)) = (\text{scalar-product } xs \ ys) + (\text{scalar-product } zs \ ws)$

using *scalar-product-append assms* **by auto**

lemma *scalar-product-distributivity*:

$\forall v1 \ v2 \ w1 \ w2. ((\text{length } v1 = \text{length } v2) \wedge (\text{length } v1 = n) \wedge (\text{length } w1 = \text{length } w2))$

$\longrightarrow (\text{scalar-product } v1 \ v2) * (\text{scalar-product } w1 \ w2)$

$= \text{scalar-product } (\text{vec-vec-Tensor } v1 \ w1) (\text{vec-vec-Tensor } v2 \ w2)$

apply (*rule allI*)

apply (*rule allI*)

apply (*rule allI*)

apply (*rule allI*)

proof(*induct n*)

case 0

```

have ((length v1 = length v2)^(length v1 = 0)^(length w1 = length w2))
  →length v1 = 0
  using 0 by auto
then have 1:((length v1 = length v2)
  ^ (length v1 = 0)
  ^ (length w1 = length w2))
  →v1 = []

  by auto
moreover have ((length v1 = length v2)
  ^ (length v1 = 0)
  ^ (length w1 = length w2))
  →length v2 = 0
  using 0 by auto
moreover then have 2:((length v1 = length v2)
  ^ (length v1 = 0)
  ^ (length w1 = length w2))
  →v2 = []

  by auto
ultimately have 3:
  ((length v1 = length v2)^(length v1 = 0)^(length w1 = length w2))
  →scalar-product v1 v2 = zer
  unfolding scalar-product-def scalar-prodI-def using zip-Nil by auto
then have 4:f zer (scalar-product w1 w2) = zer
  using zer-left-mult by auto
have ((length v1 = length v2)^(length v1 = 0)^(length w1 = length w2))
  →vec-vec-Tensor v1 w1 = []
  using 1 by auto
moreover have ((length v1 = length v2)
  ^ (length v1 = 0)
  ^ (length w1 = length w2))
  →vec-vec-Tensor v2 w2 = []

  using 2 by auto
ultimately have ((length v1 = length v2)
  ^ (length v1 = 0)
  ^ (length w1 = length w2))
  → scalar-product
    (vec-vec-Tensor v1 w1)
    (vec-vec-Tensor v2 w2) = zer
  unfolding scalar-product-def scalar-prodI-def using zip-Nil by auto
with 3 4 show ?case by auto
next
case (Suc k)
have ((length v1 = length v2)^(length v1 = Suc k)
  ^ (length w1 = length w2))
  ⇒ f (scalar-product v1 v2) (scalar-product w1 w2)
  = scalar-product (vec-vec-Tensor v1 w1) (vec-vec-Tensor v2 w2)
proof-
  assume assms:((length v1 = length v2)^(length v1 = Suc k)
  ^ (length w1 = length w2))

```

```

have length v1 = Suc k
  using Suc assms by auto
then have ( $\exists a1 u1.(v1 = a1\#u1)\wedge(\text{length } u1 = k)$ )
  using assms Suc-length-conv by metis
then obtain a1 u1 where ( $v1 = a1\#u1$ ) $\wedge(\text{length } u1 = k)$ 
  using assms by auto
then have Cons-1:( $v1 = a1\#u1$ ) $\wedge(\text{length } u1 = k)$ 
  by auto
moreover have length v2 = Suc k
  using assms Suc by auto
then have ( $\exists a2 u2.(v2 = a2\#u2)\wedge(\text{length } u2 = k)$ )
  using Suc-length-conv by metis
then obtain a2 u2 where ( $v2 = a2\#u2$ ) $\wedge(\text{length } u2 = k)$ 
  by auto
then have Cons-2: ( $v2 = a2\#u2$ ) $\wedge(\text{length } u2 = k)$ 
  by simp
then have length u1 = length u2
  using Cons-1 by auto
then have Cons-3:(scalar-product u1 u2) * scalar-product w1 w2 =
  scalar-product (vec-vec-Tensor u1 w1) (vec-vec-Tensor u2 w2)
  using Suc Cons-1 Cons-2 assms by auto
then have zip v1 v2 = (a1,a2)#(zip u1 u2)
  using zip-Cons Cons-1 Cons-2 by auto
then have Cons-4:scalar-product v1 v2 = (a1*a2)+ (scalar-product u1 u2)
  unfolding scalar-product-def scalar-prodI-def by auto
then have f (scalar-product v1 v2) (scalar-product w1 w2)
  = ((a1*a2)+ (scalar-product u1 u2))*(scalar-product w1 w2)
  by auto
then have ... = ((a1*a2)*(scalar-product w1 w2))
  + ((scalar-product u1 u2)*(scalar-product w1 w2))
  using plus-right-distributivity by auto
then have Cons-5:... = ((a1*a2)*(scalar-product w1 w2))
  + scalar-product (vec-vec-Tensor u1 w1) (vec-vec-Tensor u2 w2)
  using Cons-3 by auto
then have Cons-6:... = (scalar-product (times a1 w1) (times a2 w2))
  + scalar-product (vec-vec-Tensor u1 w1) (vec-vec-Tensor u2 w2)
  using assms effective-scalar-product-times by auto
then have scalar-product (vec-vec-Tensor v1 w1) (vec-vec-Tensor v2 w2)
  = scalar-product (vec-vec-Tensor (a1#u1) w1) (vec-vec-Tensor
(a2#u2) w2)
  using Cons-1 Cons-2 by auto
moreover have (vec-vec-Tensor (a1#u1) w1) = (times a1 w1)@(vec-vec-Tensor
u1 w1)
  using vec-vec-Tensor.simps by auto
moreover have (vec-vec-Tensor (a2#u2) w2) = (times a2 w2)@(vec-vec-Tensor
u2 w2)
  using vec-vec-Tensor.simps by auto
ultimately have Cons-7:scalar-product (vec-vec-Tensor v1 w1) (vec-vec-Tensor
v2 w2)

```

$$= \text{scalar-product } ((\text{times } a1 \ w1) @ (\text{vec-vec-Tensor } u1 \ w1)) \\ ((\text{times } a2 \ w2) @ (\text{vec-vec-Tensor } u2 \ w2))$$

by auto
moreover have $\text{length } (\text{vec-vec-Tensor } u2 \ w2) = \text{length } (\text{vec-vec-Tensor } u1 \ w1)$
using *assms* **by** (*metis Cons-1 Cons-2 vec-vec-Tensor-length*)
moreover have $\text{length } (\text{times } a1 \ w1) = (\text{length } (\text{times } a2 \ w2))$
using *assms* **by** (*metis preserving-length*)
ultimately have $\text{scalar-product } ((\text{times } a1 \ w1) @ (\text{vec-vec-Tensor } u1 \ w1)) \\ ((\text{times } a2 \ w2) @ (\text{vec-vec-Tensor } u2 \ w2)) =$
 $(\text{scalar-product } (\text{times } a1 \ w1) (\text{times } a2 \ w2))$
 $+ \text{scalar-product } (\text{vec-vec-Tensor } u1 \ w1) (\text{vec-vec-Tensor } u2 \ w2)$
using *effective-scalar-product-append* **by auto**
then show *?thesis*
using *Cons-6 Cons-7* $\langle a1 * a2 + \text{scalar-product } u1 \ u2 * \text{scalar-product}$
 $w1 \ w2$
 $= a1 * a2 * \text{scalar-product } w1 \ w2$
 $+ (\text{scalar-product } u1 \ u2 * \text{scalar-product } w1 \ w2) \rangle$
by (*metis Cons-3 Cons-4*)
qed
then show *?case* **by auto**
qed

lemma *effective-scalar-product-distributivity*:
assumes $\text{length } v1 = \text{length } v2$ **and** $\text{length } w1 = \text{length } w2$
shows $(\text{scalar-product } v1 \ v2) * (\text{scalar-product } w1 \ w2)$
 $= \text{scalar-product } (\text{vec-vec-Tensor } v1 \ w1) (\text{vec-vec-Tensor } v2 \ w2)$
using *assms* *scalar-product-distributivity* **by auto**

lemma *row-length-constant*: **assumes** *mat nr nc A* **and** $j < \text{length } A$
shows $\text{length } (A!j) = (\text{row-length } A)$
proof (*cases A*)
case *Nil*
have $\text{length } (A!j) = 0$
using *assms(2) Nil* **by auto**
then show *?thesis* **using** *assms(2) Nil row-length-Nil* **by** (*metis*)
next
case (*Cons v B*)
have $1: \forall x. ((x \in \text{set } A) \longrightarrow \text{length } x = nr)$
using *assms unfolding mat-def Ball-def vec-def* **by auto**
moreover have $(A!j) \in \text{set } A$
using *assms(2)* **by auto**
ultimately have $2: \text{length } (A!j) = nr$
by auto
have $hd \ A \in \text{set } A$
using *hd-def Cons* **by auto**
then have $\text{row-length } A = nr$
using *row-length-def 1* **by auto**

then show *?thesis using 2 by auto*
 qed

theorem *row-col-match:*

fixes *A1 A2 B1 B2 i j*
assumes *wf1:mat (row-length A1) (length A1) A1*
and *wf2:mat (row-length A2) (length A2) A2*
and *wf3:mat (row-length B1) (length B1) B1*
and *wf4:mat (row-length B2) (length B2) B2*
and *matchAA:length A1 = row-length A2*
and *matchBB:length B1 = row-length B2*
and *non-Nil:(A1 ≠ [] ∧ A2 ≠ [] ∧ B1 ≠ [] ∧ B2 ≠ [])*
and *i:i < (row-length A1)*(row-length B1) and j:j < (length A2)*(length B2)*
shows *length (row A1 (i div (row-length B1)))*
 = length (col A2 (j div (length B2)))
and *length (row B1 (i mod (row-length B1)))*
 = length (col B2 (j mod (length B2)))

proof–

have *i div (row-length B1) < row-length A1*
using *i by (metis div-left-ineq)*
then have *1:length (row A1 (i div (row-length B1))) = length A1*
unfolding *row-def by auto*
have *j div (length B2) < length A2*
using *j by (metis div-left-ineq)*
then have *2:length (col A2 (j div (length B2))) = row-length A2*
using *row-length-constant wf2 unfolding col-def by auto*
with *1 matchAA show length (row A1 (i div (row-length B1))) = length (col A2 (j div (length B2)))*
by auto
have *i mod (row-length B1) < row-length B1*
using *i by (metis less-nat-zero-code mod-less-divisor mult-is-0 neq0-conv)*
then have *2:length (row B1 (i mod (row-length B1))) = length B1*
unfolding *row-def by auto*
have *j mod (length B2) < length B2*
using *j by (metis less-nat-zero-code mod-less-divisor mult-is-0 neq0-conv)*
then have *length (col B2 (j mod (length B2))) = row-length B2*
using *row-length-constant wf4 unfolding col-def by auto*
with *2 matchBB show length (row B1 (i mod (row-length B1))) = length (col B2 (j mod (length B2)))*
by auto
 qed

lemma *effective-row-col-match: assumes matrix-match A1 A2 B1 B2*

shows $\forall i j. ((i < (row-length A1)*(row-length B1)) \wedge (j < (length A2)*(length B2)))$

\longrightarrow *length (row A1 (i div (row-length B1))) = length (col A2 (j div (length*

$B2)))$
 $\forall i j. ((i < (\text{row-length } A1) * (\text{row-length } B1)) \wedge (j < (\text{length } A2) * (\text{length } B2)))$
 $\longrightarrow \text{length } (\text{row } B1 \text{ (} i \text{ mod } (\text{row-length } B1))) = \text{length } (\text{col } B2 \text{ (} j \text{ mod } (\text{length } B2)))$
using *assms row-col-match unfolding matrix-match-def by auto*

theorem *prelim-element-match:*

$\text{matrix-match } A1 \ A2 \ B1 \ B2 \Longrightarrow (\forall i j. ((i < (\text{row-length } A1) * (\text{row-length } B1))$
 $\wedge (j < (\text{length } A2) * (\text{length } B2)))$
 \longrightarrow
 $((A1 \circ A2) \otimes (B1 \circ B2))!j!i$
 $= ((A1 \otimes B1) \circ (A2 \otimes B2))!j!i)$

proof –

assume *assms:matrix-match A1 A2 B1 B2*

have *1:matrix-match A1 A2 B1 B2*

using *assms matrix-compose-cond-def by auto*

then have *2:*

$\forall i j. ((i < (\text{row-length } A1) * (\text{row-length } B1)) \wedge (j < (\text{length } A2) * (\text{length } B2)))$
 \longrightarrow
 $((A1 \circ A2) \otimes (B1 \circ B2))!j!i$
 $= (\text{scalar-product}$
 $\text{ (row } A1 \text{ (} i \text{ div } (\text{row-length } B1))) \text{ (col } A2 \text{ (} j \text{ div } (\text{length } B2))))$
 $* (\text{scalar-product}$
 $\text{ (row } B1 \text{ (} i \text{ mod } (\text{row-length } B1))) \text{ (col } B2 \text{ (} j \text{ mod } (\text{length } B2))))$)

using *effective-matrix-match-condn-1 assms by metis*

moreover from 1 have *3:* $\forall i j. ((i < (\text{row-length } A1) * (\text{row-length } B1)) \wedge (j < (\text{length } A2) * (\text{length } B2))) \longrightarrow$

$((A1 \otimes B1) \circ (A2 \otimes B2))!j!i =$
 scalar-product
 $(\text{vec-vec-Tensor } (\text{row } A1 \text{ (} i \text{ div } \text{row-length } B1)) \text{ (row } B1 \text{ (} i \text{ mod } \text{row-length } B1)))$
 $(\text{vec-vec-Tensor } (\text{col } A2 \text{ (} j \text{ div } \text{length } B2)) \text{ (col } B2 \text{ (} j \text{ mod } \text{length } B2)))$

using *effective-matrix-match-condn-2 by auto*

have $\forall i j. ((i < (\text{row-length } A1) * (\text{row-length } B1)) \wedge (j < (\text{length } A2) * (\text{length } B2)))$

$\longrightarrow \text{length } (\text{row } A1 \text{ (} i \text{ div } (\text{row-length } B1)))$
 $= \text{length } (\text{col } A2 \text{ (} j \text{ div } (\text{length } B2)))$

and $\forall i j. ((i < (\text{row-length } A1) * (\text{row-length } B1)) \wedge (j < (\text{length } A2) * (\text{length } B2)))$

$\longrightarrow \text{length } (\text{row } B1 \text{ (} i \text{ mod } (\text{row-length } B1)))$
 $= \text{length } (\text{col } B2 \text{ (} j \text{ mod } (\text{length } B2)))$

using *assms effective-row-col-match by auto*

then have $\forall i j. ((i < (\text{row-length } A1) * (\text{row-length } B1)) \wedge (j < (\text{length } A2) * (\text{length } B2)))$

\longrightarrow
 $(\text{scalar-product } (\text{row } A1 \text{ (} i \text{ div } (\text{row-length } B1))) \text{ (col } A2 \text{ (} j \text{ div } (\text{length } B2))))$
 $* (\text{scalar-product } (\text{row } B1 \text{ (} i \text{ mod } (\text{row-length } B1))) \text{ (col } B2 \text{ (} j \text{ mod } (\text{length } B2))))$

$$= \text{scalar-product}$$

$$(\text{vec-vec-Tensor } (\text{row } A1 \text{ (} i \text{ div row-length } B1)) \text{ (row } B1 \text{ (} i \text{ mod row-length } B1)))$$

$$(\text{vec-vec-Tensor } (\text{col } A2 \text{ (} j \text{ div length } B2)) \text{ (col } B2 \text{ (} j \text{ mod length } B2)))$$
using *effective-scalar-product-distributivity* **by** *auto*
then show *?thesis* **using** 2 3 **by** *auto*
qed

theorem *element-match*:

$$\text{matrix-match } A1 \ A2 \ B1 \ B2 \implies (\forall i < ((\text{row-length } A1) * (\text{row-length } B1)).$$

$$\forall j < ((\text{length } A2) * (\text{length } B2)).$$

$$(((A1 \circ A2) \otimes (B1 \circ B2))!j!i$$

$$= ((A1 \otimes B1) \circ (A2 \otimes B2))!j!i)$$
using *prelim-element-match* **by** *auto*

lemma *application: fixes* $m1 \ m2$
shows $\forall m1 \ m2. (\text{mat } nr \ nc \ m1)$

$$\wedge (\text{mat } nr \ nc \ m2)$$

$$\wedge (\forall j < nc. \forall i < nr. m1 ! j ! i = m2 ! j ! i)$$

$$\longrightarrow (m1 = m2)$$
using *mat-eqI* **by** *blast*

theorem *tensor-compose-condn*:
assumes $wf1: \text{mat } nr \ nc \ ((A1 \circ A2) \otimes (B1 \circ B2))$
and $wf2: \text{mat } nr \ nc \ ((A1 \otimes B1) \circ (A2 \otimes B2))$
and $wf3: \forall j < nc. \forall i < nr. (((A1 \circ A2) \otimes (B1 \circ B2))!j!i$

$$= ((A1 \otimes B1) \circ (A2 \otimes B2))!j!i)$$

shows $((A1 \circ A2) \otimes (B1 \circ B2))$

$$= ((A1 \otimes B1) \circ (A2 \otimes B2))$$

using *application wf1 wf2 wf3* **by** *blast*

The following theorem gives us the distributivity relation of tensor product with matrix multiplication

theorem *distributivity*:
assumes *matrix-match* $A1 \ A2 \ B1 \ B2$
shows $((A1 \circ A2) \otimes (B1 \circ B2)) = ((A1 \otimes B1) \circ (A2 \otimes B2))$
proof –
let $?nr = ((\text{row-length } A1) * (\text{row-length } B1))$
let $?nc = ((\text{length } A2) * (\text{length } B2))$
have $\text{mat } ?nr \ ?nc \ ((A1 \circ A2) \otimes (B1 \circ B2))$
by (*metis assms effective-tensor-compose-distribution1*)
moreover have $\text{mat } ?nr \ ?nc \ ((A1 \otimes B1) \circ (A2 \otimes B2))$
using *assms* **by** (*metis effective-tensor-compose-distribution2*)
moreover have $\forall j < ?nc. \forall i < ?nr.$

$$(((A1 \circ A2) \otimes (B1 \circ B2))!j!i$$

$$= ((A1 \otimes B1) \circ (A2 \otimes B2))!j!i)$$
using *element-match assms* **by** *auto*
ultimately show *?thesis*

```
    using application by blast  
qed  
end  
end
```