Tensor Product of Matrices

T.V.H. Prathamesh

August 16, 2018

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
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

times a v , gives us the product of the vector v with multiplied pointwise with a

primrec times:: 'a ⇒ 'a vec ⇒ 'a vec
where
  times n [] = []
  times n (y#ys) = (f n y)#(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

\[ vec_vec_Tensor(v_1, v_2, \ldots, v_n, w_1, w_2, \ldots, w_m) = (v_1 \cdot w_1, \ldots, v_1 \cdot w_m, \ldots, v_n \cdot w_1, \ldots, v_n \cdot w_m) \]

primrec vec_vec-Tensor:: 'a vec ⇒ 'a vec ⇒ 'a vec
where
  vec_vec-Tensor [] ys = []
  vec_vec-Tensor (x#xs) ys = (times x ys)#(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 :
  (length(vec_vec-Tensor x y)) = (length x)*((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
vec.mat_Tensor \( (v_1, v_2, ... v_n) (C_1, C_2, ... C_m) = (v_1 \cdot C_1, ..., v_n \cdot C_1, ..., v_1 \cdot C_m, ..., v_n \cdot C_m) \)

primrec vec.mat-Tensor :: \\
'a vec ⇒ \\
'a mat ⇒ \\
'a mat
where
vec.mat-Tensor xs [] = []
vec.mat-Tensor xs (ys # yss) = (vec.vec-Tensor xs ys) #(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 M
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) = (vec.vec-Tensor v y) #(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 mat-def
by (metis in-set-member member-rec(1) 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 ∈ set M
shows ∃ ys.∃ zs.(ys @ x # zs = M)
using assms set-def in-set-conv-decomp by metis

primrec reduct :: 'a mat ⇒ 'a mat
where
reduct [] = []
lemma length-reduct:
  assumes m ≠ []
  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 ≠ []
  shows (∀ x ∈ (set M). (nr1 = nr2))
proof –
  have A: z ∈ 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 ∈ (set M)) → (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 ∈ (set M)) → (vec nr2 x) using Ball-def assms by auto
  from step1 and step2
  have step3: ∀ x.((x ∈ (set M)) → ((vec nr1 x) ∧ (vec nr2 x)))
  by (metis Ball (set M) (vec nr1) Ball (set M) (vec nr2))
  have ((vec nr1 z) ∧ (vec nr2 z)) → (nr1 = nr2) using vec-uniqueness by auto
  with step3
  have (∀ x.((x ∈ (set M)) → ((nr1 = nr2))))) by (metis vec-uniqueness)
  then
  have (∀ x ∈ (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 \( \text{mat } 0 \text{ nc M} \)
proof
  have \( \text{set } M = \{\} \) using \text{mat-def} \text{ assms} \text{ empty-set by auto} 
  then have \( \text{Ball } (\text{set } M) \) (\( \text{vec } 0 \)) using \text{Ball-def} \text{ by auto} 
  then have \( \text{mat } 0 \text{ nc M} \) using \text{mat-def} \text{ assms(1) assms(2) gen-length-code(1)} 
  length-code 
  by \( \text{metis (full-types)} \) 
  then have \( \text{Ball } (\text{set } M) \) (\( \text{vec } 0 \)) using \text{Ball-def} \text{ by auto} 
  then have \( \text{mat } 0 \text{ nc M} \) using \text{mat-def} \text{ assms(1) assms(2) gen-length-code(1)} 
  length-code 
  by \( \text{metis (full-types)} \) 
  then show \?thesis by auto
qed

abbreviation \text{null-matrix}::'a list list 
where 
\text{null-matrix} \equiv [\text{Nil}]

lemma \text{null-mat}:\text{null-matrix} = [[]] 
by auto

lemma \text{zero-matrix}: \text{mat } 0 \text{ 0 } [[]] using \text{mat-def} \text{ in-set-insert insert-Nil list.size(3)} 
\text{not-Cons-self2} 
by \( \text{metis (full-types)} \)

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

\text{definition} \text{row-length}:: 'a mat \Rightarrow \text{nat} 
where 
\text{row-length } xs \equiv \text{if } (xs = [[]]) \text{ then } 0 \text{ else } (\text{length } (\text{hd } xs))

lemma \text{row-length-Nil}: 
\text{row-length } [[]] = 0 
using \text{row-length-def by (metis )}

lemma \text{row-length-Null}: 
\text{row-length } [[]] = 0 
using \text{row-length-def by auto}

lemma \text{row-length-vec-mat}: 
\text{row-length } (\text{vec-mat-Tensor v m}) = (\text{length } v) * (\text{row-length m}) 
proof \( \text{induct m} \) 
  case Nil 
  have \( \text{row-length } [[]] = 0 \) 
  using \text{row-length-Nil by simp} 
  moreover have \( \text{vec-mat-Tensor v } [[]] = [[]] \) 
  using \text{vec-mat-Tensor.simps(1) by auto} 
  ultimately have 
  \( \text{row-length } (\text{vec-mat-Tensor v } [[]]) = (\text{length } v) * (\text{row-length } [[]]) \) 
  using \text{mult-0-right by (metis )} 
  then show \?case by metis
next
  fix a m
\textbf{assume} \( \text{A:row-length (vec-mat-Tensor v m)} = \text{length v} \ast \text{row-length m} \)

\text{let \ ?case =}

\text{row-length (vec-mat-Tensor v (a\#m)) = (length v)\ast\text{row-length (a\#m)}}

\text{have \ A:row-length (a \# m) = length a}

\text{using \ row-length-def \ list.distinct(1) \ by \ auto}

\text{have (vec-mat-Tensor v (a\#m)) = (vec-vec-Tensor v a)\#(vec-mat-Tensor v m)}

\text{using \ vec-mat-Tensor-def \ vec-mat-Tensor.simps(2) \ by \ auto}

\text{from \ this \ have}

\text{row-length (vec-mat-Tensor v (a\#m)) = length (vec-vec-Tensor v a)}

\text{using \ row-length-def \ list.distinct(1) \ vec-mat-Tensor.simps(2) \ by \ auto}

\text{from \ this \ and \ vec-vec-Tensor-length \ have}

\text{row-length (vec-mat-Tensor v (a\#m)) = (length v)\ast\text{(length a)}}

\text{by \ auto}

\text{from \ this \ and \ A \ have}

\text{row-length (vec-mat-Tensor v (a\#m)) = (length v)\ast\text{row-length (a\#m)}}

\text{by \ auto}

\text{from \ this \ show \ ?case \ by \ auto}

\text{qed}

Tensor is the tensor product of matrices

\textbf{primrec Tensor:: \ 'a mat \Rightarrow \ 'a mat \Rightarrow \ 'a mat \ (infixl \ \otimes \ 63)}

\textbf{where}

\text{Tensor [] xs = []}

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

\textbf{lemma Tensor-null: \ xs \otimes [] = []}

\text{by (induction xs)(auto)}

Tensor commutes with left and right identity

\textbf{lemma Tensor-left-id: \ [[id]] \otimes xs = xs}

\text{by (induction xs)(auto simp add: times-scalar-id)}

\textbf{lemma Tensor-right-id: \ xs \otimes [[id]] = xs}

\text{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

\textbf{lemma row-length-mat:}

\( \text{(row-length (m1}\otimes m2)) = \text{(row-length m1)\ast(row-length m2)} \)

\textbf{proof (induct m1)}

\textbf{case Nil}

\textbf{have row-length ([]}\otimes m2) = \emptyset

\text{using \ Tensor.simps(1) \ row-length-def}

\text{by \ metis}

\textbf{from this}
have row-length ([] ⊗ 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 ⊗ m2) = row-length m1 * row-length m2
let ?case = row-length ((a # m1) ⊗ 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) ⊗ 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 ⊗ m2) = row-length (a # m1) * row-length m2
let ?case = row-length (a # m1 ⊗ aa # m2)
  = row-length (a # m1) * row-length (aa # m2)
have aa#m2 ≠ []
  by auto
from this have non-zero: (vec-mat-Tensor a (aa#m2)) ≠ []
  using vec-mat-Tensor-def by auto
from this have
  hd ((vec-mat-Tensor a (aa#m2))@(m1 ⊗ m2))
  = hd (vec-mat-Tensor a (aa#m2))
by auto
from this have
  hd ((a#m1)⊗(aa#m2)) = hd (vec-mat-Tensor a (aa#m2))
using Tensor.simps(2) by auto
from this have s1: row-length ((a#m1)⊗(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-vec-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
7
qed
from this show ?case by auto
qed

lemma hd-set\): assumes \( x \in \text{set} (a\#M) \) shows \( (x = a) \lor (x \in (\text{set} M)) \)
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 \((x \in \text{set}(a \#N)) \longrightarrow (x = a) \lor (x \in (\text{set} N))\)
  using hd-set by auto
from this and 1 have 2: vec nr a
  using mat-def by (metis Ball-set-list-all list-all-simps(1))
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 in-set-member mat-def member-rec(1) vec-def
    by metis
  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 in-set-member mat-def member-rec(1)
    vec-uniqueness
    by metis
  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:
(mat (row-length M) (length M) M) —> (mat
\((\text{row-length } M) \cdot (\text{length } v)\)

\((\text{length } M)\)

\((\text{vec-mat-Tensor } v M)\)

**proof** (**induct** \( M \))

**case** **Nil**

**have** \((\text{vec-mat-Tensor } v \ []\)) = \[

**using** \(\text{vec-mat-Tensor}.\text{simps}(1)\) Nil

**by** simp

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

**using** \(\text{row-length-def } \text{Nil}\)

**by** metis

**moreover have** \((\text{length } \ []\) = 0\)

**using** Nil by simp

ultimately have

\(\text{mat} ((\text{row-length } \ []) \cdot (\text{length } v)) (\text{length } \ []) (\text{vec-mat-Tensor } v \ [])\)

**using** \(\text{zero-matrix}\) by (metis \text{mult-zero-left})

then show \(?\text{case}\) by simp

next

fix \( a \) \( M \)

assume \( \text{hyp} : \)

\((\text{mat} \ (\text{row-length } M) \ (\text{length } M) \ M) \implies \text{mat} \ ((\text{row-length } M) \cdot (\text{length } v)) (\text{length } M) (\text{vec-mat-Tensor } v M))\)

mat \((\text{row-length } (a\#M)) \ (\text{length } (a\#M)) \ (a\#M)\)

let \(?\text{case } = \)

\(\text{mat} ((\text{row-length } (a\#M)) \cdot (\text{length } v)) (\text{length } (a\#M)) (\text{vec-mat-Tensor } v (a\#M))\)

**have** \(\text{step1} : \text{mat} \ (\text{row-length } M) \ (\text{length } M) M\)

**using** \(\text{hyp}(2)\) \text{reduct-matrix} by auto

then have \(\text{step2} : \)

\(\text{mat} \ (\text{row-length } M) \cdot (\text{length } v) \ (\text{length } M) (\text{vec-mat-Tensor } v M)\)

**using** \(\text{hyp}(1)\) by auto

have

\(\text{mat} \ (\text{row-length } (a\#M) \cdot (\text{length } v)) \ (\text{length } (a\#M)) \ (\text{vec-mat-Tensor } v (a\#M))\)

**proof** (**cases** \( M \))

**case** **Nil**

fix \( x \)

**have** \(1: (\text{vec-mat-Tensor } v (a\#M)) = [\text{vec-vec-Tensor } v a]\)

**using** \(\text{vec-mat-Tensor}.\text{simps}\) Nil by auto

**have** \((x \in \text{(set } [\text{vec-vec-Tensor } v a]) \implies x = (\text{vec-vec-Tensor } v a)\)

**using** \(\text{set-def}\) by auto

then have \(2: \)

\((x \in \text{(set } [\text{vec-vec-Tensor } v a])) \implies (\text{vec} \ (\text{length } (\text{vec-vec-Tensor } v a)) x)\)

**using** \(\text{vec-def}\) by metis

**have** \(3: \text{length } (\text{vec-vec-Tensor } v a) = (\text{length } v) \cdot (\text{length } a)\)

**using** \(\text{vec-vec-Tensor-length}\) by auto

then have \(4:\)
\[
\text{length (vec-vec-Tensor } v \ a) = (\text{length } v) \ast (\text{row-length } (a\#M))
\]

using \text{row-length-def list.distinct(1)}

by auto

have \textbf{6}: \text{length (vec-mat-Tensor } v \ (a\#M)) = (\text{length } (a\#M))

using \text{vec-mat-Tensor-length by auto}

hence \text{mat (length (vec-vec-Tensor } v \ a)) \ast (\text{length } (a \# M)) \ [\text{vec-vec-Tensor } v \ a] \]

by (simp add: Nil mat-def vec-def)

hence

\[
\text{mat (row-length } (a\#M) \ast \text{length } v)
\]

(\text{length (vec-mat-Tensor } v \ (a\#M)))

(\text{vec-mat-Tensor } v \ (a\#M))

using \textbf{1 \& \textbf{6}} by (simp add: mult.commute)

then show \textit{thesis} using \textbf{6} by auto

next
case (\textbf{Cons } b \ L)

fix \textbf{x}

have \textbf{1}: \textbf{x} \in (\text{set } (a\#M)) \longrightarrow ((x=a) \vee (x \in (\text{set } M)))

using \text{hd-set by auto}

have \text{mat (row-length } (a\#M) \ast \text{length } v)

(\text{length (vec-mat-Tensor } v \ (a\#M)))

(\text{vec-mat-Tensor } v \ (a\#M))

using \text{hyp by auto}

then have \textbf{x} \in (\text{set } (a\#M)) \longrightarrow (\text{vec (row-length } (a\#M)) \ x)

using \text{mat-def Ball-def by metis}

then have \textbf{x} \in (\text{set } (a\#M)) \longrightarrow (\text{vec (length } a) \ x)

using \text{row-length-def list.distinct(1)}

by auto

with \textbf{1} have \textbf{x} \in (\text{set } M) \longrightarrow (\text{vec (length } a) \ x)

by auto

moreover have \textbf{b} \in (\text{set } M)

using \text{Cons by auto}

ultimately have \text{vec (length } a) \ b

using \text{hyp(2) in-set-member mat-def member-rec(1) vec-def by (metis)}

then have \textbf{(length } b) = (\text{length } a)

using \text{vec-def vec-uniqueness by auto}

then have \textbf{2}: \text{row-length } M = (\text{length } a)

using \text{row-length-def Cons list.distinct(1) by auto}

have \text{mat (row-length } M \ast \text{length } v) \ast (\text{length } M) \ (\text{vec-mat-Tensor } v \ M)

using \text{step2 by auto}

then have \textbf{3}:

\[
\text{Ball (set (vec-mat-Tensor } v \ M)) \ (\text{vec ((row-length } M) \ast (\text{length } v)))
\]

using \text{mat-def by auto}

then have \textbf{(x} \in (\text{set (vec-mat-Tensor } v \ M))

\longrightarrow (\text{vec ((row-length } M) \ast (\text{length } v)) \ x)

using \text{mat-def Ball-def by auto}

then have \textbf{4}: (x \in (\text{set (vec-mat-Tensor } v \ M))

\longrightarrow (\text{vec ((length } a) \ast (\text{length } v)) \ x)

using \text{2 by auto}

have \textbf{5}: \text{length (vec-vec-Tensor } v \ a) = (\text{length } a) \ast (\text{length } v)

using \text{vec-vec-Tensor-length by auto}
then have 6: vec ((length a)\ast(lengt h 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 \(x \in \text{set} \ (\text{vec-mat-Tensor} \ v \ (a\#M))\)
  \(\rightarrow ((x = (\text{vec-vec-Tensor} \ v \ a))\)
  \(\lor (x \in (\text{set} \ (\text{vec-mat-Tensor} \ v \ M))))\)
  using hd-set by auto
with 4 6 have \(x \in \text{set} \ (\text{vec-mat-Tensor} \ v \ (a\#M))\)
  \(\rightarrow \text{vec} \ ((\text{length} a)\ast(\text{length} v)) \ x\)
  by auto
with 7 have \(x \in \text{set} \ (\text{vec-mat-Tensor} \ v \ (a\#M))\)
  \(\rightarrow \text{vec} \ ((\text{row-length} (a\#M))\ast(\text{length} v)) \ x\)
  by auto
then have \(\forall x.((x \in \text{set} \ (\text{vec-mat-Tensor} \ v \ (a\#M)))\)
  \(\rightarrow \text{vec} \ ((\text{row-length} (a\#M))\ast(\text{length} v)) \ x)\)
  using 2 3 6 7 hd-set vec-mat-Tensor.simps(2) by auto
then have 7: Ball \((\text{set} \ (\text{vec-mat-Tensor} \ v \ (a\#M)))\)
  \((\text{vec} \ ((\text{row-length} (a\#M))\ast(\text{length} v))))\)
  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
  \((\text{row-length} (a\#M))\ast(\text{length} v))\)
  \(\text{length} (a\#M)\)
  \((\text{vec-mat-Tensor} \ v \ (a\#M))\)
  using mat-def 5 length-code
  by (metis (hide-lams, no-types))
then show \(\text{thesis} by auto\)
qed
with hyp show \(\text{case} by auto\)
qed

The following theorem gives length of tensor product of two matrices

**Lemma** length-Tensor: \((\text{length} \ (M1 \otimes M2)) = (\text{length} M1)\ast(\text{length} M2)\)
**Proof** (induct M1)
  case Nil
  show \(\text{case} by auto\)
next
case (Cons a M1)
  have ((a \# M1) \otimes M2) = (vec-mat-Tensor a M2)\otimes(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:length \((\text{vec-mat-Tensor a } M2) @ (M1 \otimes M2)\)
  \(=\) length \((\text{vec-mat-Tensor a } M2) +\) length \((M1 \otimes M2)\)
  using append-def
  by auto

have 3:(\text{length } (\text{vec-mat-Tensor a } M2)) = \text{length } M2
  using vec-mat-Tensor-length by (auto)

have 4:length \((M1 \otimes M2) = (\text{length } M1) \ast (\text{length } M2)\)
  using Cons.hyps by auto

with 2 3 have \(\text{length } ((\text{vec-mat-Tensor a } M2) @ (M1 \otimes M2)) = (\text{length } M2) + (\text{length } M1) \ast (\text{length } M2)\)
  by auto

then have 5:
  \(\text{length } ((\text{vec-mat-Tensor a } M2) @ (M1 \otimes M2)) = (1 + (\text{length } M1)) \ast (\text{length } M2)\)
  by auto

with 1 have \(\text{length } ((a \# M1) \otimes M2) = ((\text{length } (a \# M1)) \ast (\text{length } M2))\)
  by auto

then show ?case by auto
qed

lemma append-reduct-matrix:
\((\text{mat } (\text{row-length } (M1 @ M2)) (\text{length } (M1 @ M2)) (M1 @ M2)) \implies (\text{mat } (\text{row-length } M2) (\text{length } M2) M2)\)
proof(induct M1)
  case Nil
  show ?thesis using Nil append.simps(1) by auto
next
case (Cons a M1)
  have \(\text{mat } (\text{row-length } (M1 @ M2)) (\text{length } (M1 @ M2)) (M1 @ M2)\)
    using reduct-matrix Cons.prems append.Cons by metis
  from this have \(\text{mat } (\text{row-length } M2) (\text{length } M2) M2\)
    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:
\((\text{mat } (\text{row-length } M1) (\text{length } M1) M1) \wedge (\text{mat } (\text{row-length } M2) (\text{length } M2) M2) \implies (\text{mat } ((\text{row-length } M1) \ast (\text{row-length } M2)) ((\text{length } M1) \ast (\text{length } M2)) (M1 \otimes M2))\)
proof(induct M1)
  case Nil
  have \((\text{row-length } []) \ast (\text{row-length } M2) = 0\)
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) ([] ⊗ M2)
using zero-matrix bymetis
then show ?case by simp
next
case (Cons a M1)
have step1: mat (row-length (a # M1)) (length (a # M1)) (a # M1)
using Cons.prems 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.prems by auto
ultimately have step2:
mat (row-length M1 * row-length M2) (length M1 * length M2) (M1 ⊗ 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 ⊗ M2)
proof (cases M1)
case Nil
have (mat ((row-length M2)* (length a)) (length M2) (vec-mat-Tensor a M2))
using Cons.prems well-defined-vec-mat-Tensor by auto
moreover have (length (a # M1)) * (length M2) = length M2
using Nil by auto
moreover have (a#M1) ⊗ 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) ⊗ M2))
using 0 by auto
then show ?thesis by (simp add: mult.commute)
next
case (Cons b N1)
fix x
have 1:x ∈ (set (a#M1)) —→ ((x=a) ∨ (x ∈ (set M1)))
using hd-set by auto
have mat (row-length (a#M1)) (length (a#M1)) (a#M1)
using Cons.prems by auto
then have x∈ (set (a#M1)) —→ (vec (row-length (a#M1)) x)
Using `mat-def` `Ball-def` by `metis`
then have \( x \in (\text{set} \ (a \# M1)) \rightarrow (\text{vec} \ (\text{length} \ a) \ x) \)
using `row-length-def` `list.distinct(1)` by `auto`
with \( I \) have \( x \in (\text{set} \ M1) \rightarrow (\text{vec} \ (\text{length} \ a) \ x) \)
by `auto`
moreover have \( b \in (\text{set} \ M1) \)
using `Cons` by `auto`
ultimately have \( \text{vec} \ (\text{length} \ a) \ b \)
using `Cons`.

prems in `set-member` `mat-def` `member-rec` `(1)` `vec-def`
by `metis`
then have \( \text{(length} \ b) = (\text{length} \ a) \)
using `vec-def` `vec-uniqueness` by `auto`
then have `2: row-length M1 = (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` `(\text{set} \ (M1 \otimes M2)) \ (\text{vec} \ ((\text{length} \ a) * (\text{row-length} \ M2)))`
using `mat-def` by `auto`
from this have `3`:
\( \forall x. \ x \in (\text{set} \ (M1 \otimes M2)) \rightarrow (\text{vec} \ ((\text{length} \ a) * (\text{row-length} \ M2)) \ x) \)
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.prems`
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`:
\( \forall x. \ x \in (\text{set} \ (\text{vec-mat-Tensor} \ a \ M2)) \)
\( \rightarrow (\text{vec} \ ((\text{length} \ a) * (\text{row-length} \ M2)) \ x) \)
using `mult.commute` by `metis`
with `3` have `5`:
\( \forall x. \ (x \in (\text{set} \ (\text{vec-mat-Tensor} \ a \ M2))) \)
\( \forall x \in (\text{set} \ (M1 \otimes M2)) \rightarrow (\text{vec} \ ((\text{length} \ a) * (\text{row-length} \ M2)) \ x) \)
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))`
\( \rightarrow (x \in (\text{set} \ (\text{vec-mat-Tensor} \ a \ M2))) \forall x \in (\text{set} \ (M1 \otimes M2)) \)`
using set-def append-def by auto

with 5 have 7:\forall x. (x \in (set (a \# M1 \otimes M2)))
           \rightarrow (vec ((length a)\ast (row-length M2)) x)
           by auto
then have 8:
Ball (set (a \# M1 \otimes M2)) (vec ((row-length (a#M1))\ast (row-length M2)))
using Ball-def 0 by auto
have (length ((a#M1)\otimes M2)) = (length (a#M1))\ast (length M2)
using length-Tensor by metis

with 7 8
have mat
(row-length (a \# M1) \ast row-length M2)
(length (a \# M1) \ast length M2)
(a \# M1 \otimes 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)\ast (row-length M2))
((length M1)\ast (length M2))
(M1 \otimes M2)
using well-defined-Tensor assms by auto

definition natmod::nat \Rightarrow nat \Rightarrow nat (infixl nmod 50)
where
natmod x y = nat ((int x) mod (int y))

theorem times-elements:
\forall i.((i<\langle length v\rangle)) \rightarrow (times a v)!i = f a (v!i)
apply(rule allI)
proof(induct v)
case Nil
have (length [] = 0)
by auto
then have i <\langle length []\rangle \Rightarrow False
by auto
moreover have (times a []) = []
using times.simps(1) by auto
ultimately have (i<\langle length []\rangle) \rightarrow (times a [])!i = f a ([]!i)
by auto
then have \forall i. ((i<\langle length []\rangle)) \rightarrow (times a [])!i = f a ([]!i)
by auto
then show \( \exists \text{case} \) by auto

next

case (\( \text{Cons} \ x \ xs \))

have \( \forall i.((x\#xs)! (i+1) = (xs)!i) \)

by auto

have 0:((\( i < \text{length} \ (x\#xs) \)) \rightarrow ((\( i < \text{length} \ xs \)) \lor (i = (\text{length} \ xs))))

by auto

have 1:((\( i < \text{length} \ xs \)) \rightarrow ((\text{times} \ a \ xs)!i = f a (xs)!i))

by (metis \text{Cons.hyps})

have \( \forall i.((x\#xs)!(i+1) = (xs)!i) \) by auto

have ((\( i < \text{length} \ (x\#xs) \)) \rightarrow ((\text{times} \ (x\#xs))!i = f a ((x\#xs)!i))

proof (cases \( i \))

case 0

have ((\text{times} \ a \ (x\#xs))!i) = f a x

using 0 times.simps(2) by auto

then have ((\text{times} \ a \ (x\#xs))!i) = f a ((x\#xs)!i)

using 0 by auto

then show \( \exists \text{thesis} \) by auto

next

case (\( \text{Suc} \ j \))

have 1:(\( \text{times} \ a \ (x\#xs))!i = ((f a x)\#(\text{times} \ a \ xs))!i)

using times.simps(2) by auto

have 2:(f a x)\#(\text{times} \ a \ xs)!i = (\text{times} \ a \ xs)!j

using \text{Suc by auto}

have 3:((\( i < \text{length} \ (x\#xs) \)) \rightarrow (j < \text{length} \ xs))

using \text{One-nat-def Suc-eq-plus1 list.size(4) not-less-eq}

by metis

have 4:((\text{times} \ a \ xs)!j = (f a ((xs)!j)))

using 1 by (metis \text{Cons.hyps})

have 5:((x\#xs)!i) = (xs)!j

using Suc by (metis nth-Cons-Suc)

with 1 2 4 have ((\( j < \text{length} \ xs \)) \rightarrow ((\text{times} \ a \ (x\#xs))!i = (f a ((x\#xs)!i))))

by auto

with 3 have ((\( i < \text{length} \ (x\#xs) \)) \rightarrow ((\text{times} \ a \ (x\#xs))!i = (f a ((x\#xs)!i))))

by auto

then show \( \exists \text{thesis} \) by auto

qed

then show \( \exists \text{case} \) by auto

qed

lemma simpl-times-elements:

assumes \( (i < \text{length} \ xs) \)

shows ((\( i < \text{length} \ v \)) \rightarrow ((\text{times} \ a \ v)!i = f a ((v)!i))

using times-elements by auto

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

\textbf{lemma} \textit{append-simpl2}:
\[ i \geq \left( \text{length } xs \right) \longrightarrow (xs@ys)!i = (ys!(i - \left( \text{length } xs \right))) \]
using \textit{nth-append less-asym leD} by \textit{metis}

\textbf{lemma} \textit{append-simpl3}:
\begin{itemize}
  \item \textbf{assumes} \[ i > \left( \text{length } y \right) \]
  \item \textbf{shows} \[ (i < ((\text{length } (z#zs)) \ast (\text{length } y))) \longrightarrow (i - (\text{length } y)) < (\text{length } zs) \ast (\text{length } y)) \]
\end{itemize}
\textbf{proof–}
\begin{itemize}
  \item have \[ \text{length } (z#zs) = (\text{length } zs) + 1 \]
    by \textit{auto}
  \item then have \[ i < ((\text{length } (z#zs)) \ast (\text{length } y)) \rightarrow i < ((\text{length } zs) + 1) \ast (\text{length } y) \]
    by \textit{auto}
  \item then have \[ 1: \ i < ((\text{length } (z#zs)) \ast (\text{length } y)) \rightarrow (i < ((\text{length } zs) \ast (\text{length } y)) + (\text{length } y)) \]
    by \textit{auto}
  \item have \[ i < ((\text{length } zs) \ast (\text{length } y)) + (\text{length } y)) \]
    \[ = ((i - (\text{length } y)) < ((\text{length } zs) \ast (\text{length } y))) \]
    using \textit{assms} by \textit{auto}
  \item then have \[ (i < ((\text{length } (z#zs)) \ast (\text{length } y)) \rightarrow (i - (\text{length } y)) < ((\text{length } zs) \ast (\text{length } y)) \]
    by \textit{auto}
  \item then show \textbf{?thesis} by \textit{auto}
\end{itemize}
qed

\textbf{lemma} \textit{append-simpl4}:
\[ (i > (\text{length } y)) \rightarrow (i < ((\text{length } (z#zs)) \ast (\text{length } y)) \rightarrow (i - (\text{length } y)) < (\text{length } zs) \ast (\text{length } y)) \]
using \textit{append-simpl3} by \textit{auto}

\textbf{lemma} \textit{vec-vec-Tensor-simpl}:
\[ i < (\text{length } y) \rightarrow (\text{vec-vec-Tensor } (z#zs) \ y)!i = (\times z \ y)!i \]
\textbf{proof–}
\begin{itemize}
  \item have \textbf{a}: \[ \text{vec-vec-Tensor } (z#zs) \ y = (\times z \ y)@(\text{vec-vec-Tensor } zs \ y) \]
    by \textit{auto}
  \item have \textbf{b}:
    \begin{itemize}
      \item \textbf{length} \ (\times z \ y) = (\text{length } y) \textbf{using preserving-length} by \textit{auto}
      \item \textbf{have} \[ i < (\times z y) \ast (\times z y)!i = (\times z y)!i \]
        using \textit{append-simpl} by \textit{metis}
      \item with \textbf{b} have \[ i < (\times z y)!i = (\times z y)!i \]
        by \textit{auto}
      \item with \textbf{a} have \[ i < (\times z y)!i = (\times z y)!i \]
        by \textit{auto}
    \end{itemize}
  \item then show \textbf{?thesis} by \textit{auto}
\end{itemize}
qed

**lemma** \texttt{vec-vec-Tensor-simpl2}: \begin{itemize} 
\item \(i \geq (\text{length } y)\) \rightarrow ((\text{vec-vec-Tensor } (z \# z s) y)!i = (\text{vec-vec-Tensor } z s y)! (i - (\text{length } y)))\) 
\end{itemize}
\texttt{using vec-vec-Tensor.simps(2) append-simpl2 preserving-length} by \texttt{metis}

**lemma** \texttt{division-product}: 
\begin{itemize} 
\item \texttt{assumes (b::int)>0 and a \geq b}\) \texttt{shows (a \div b) = ((a - b) \div b) + 1}\) 
\end{itemize}
\texttt{proof}--
\begin{itemize} 
\item \texttt{fix c}\) 
\item \texttt{have a - b \geq 0 by auto}\) 
\item \texttt{using assms(2) by auto}\) 
\item \texttt{have 1: a - b = a + (-1)*b by auto}\) 
\item \texttt{have (b \neq 0) \rightarrow ((a + b * (-1)) \div b = (-1) + a \div b) by auto}\) 
\item \texttt{using div-mult-self2 by auto}\) 
\item \texttt{with 1 assms(1) have ((a - b) \div b) = (-1) + a \div b by auto}\) 
\item \texttt{then have (a \div b) = ((a - b) \div b) + 1 by auto}\) 
\end{itemize}
\texttt{then show ?thesis by auto}\) 
qed

**lemma** \texttt{int-nat-div}: 
\begin{itemize} 
\item \texttt{(int a) \div (int b) = int ((a::nat) \div b)} \) 
\end{itemize}
\texttt{by (metis zdiv-int)}

**lemma** \texttt{int-nat-eq}: 
\begin{itemize} 
\item \texttt{assumes int (a::nat) = int b}\) \texttt{shows a = b by auto}\) 
\end{itemize}

**lemma** \texttt{nat-div}: 
\begin{itemize} 
\item \texttt{assumes (b::nat) > 0 and a > b}\) \texttt{shows (a \div b) = ((a - b) \div b) + 1}\) 
\end{itemize}
\texttt{proof}--
\begin{itemize} 
\item \texttt{fix x}\) 
\item \texttt{have 1:(int b)>0 by auto}\) 
\item \texttt{using assms(1) division-product by auto}\) 
\item \texttt{moreover have (int a)>(int b) by auto}\) 
\end{itemize}
\texttt{with 1 have 2: ((int a) \div (int b))
int a - (int b) div (int b) + 1

using division-product by auto
from int-nat-div have 3: (int a) div (int b)) = int (a div b)
  by auto
from int-nat-div assms(2) have 4:
  (((int a) - (int b)) div (int b)) = int ((a - b) div b)
  by (metis (full-types) less-asym 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
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 \( \texttt{vec-vec-Tensor } \emptyset y = \emptyset \)
  by auto
moreover have
\( (i < \langle \text{length } \texttt{vec-vec-Tensor } \emptyset y \rangle) \longrightarrow \)
\( \langle \text{vec-vec-Tensor } x y \rangle!i = f \langle x! (i \text{ div } \langle \text{length } y \rangle) \rangle \langle y! (i \text{ mod } \langle \text{length } y \rangle) \rangle \)
by auto
then show case
by auto
next
\begin{itemize}
  \item \textbf{case} (\texttt{Cons } z \texttt{ zs})
  \item have \texttt{1:vec-vec-Tensor } (z\#zs) \texttt{ y} = \langle \texttt{times } z \texttt{ y} \rangle @ \texttt{vec-vec-Tensor zs y} \by auto
  \item have \texttt{2:i < \langle \text{length } y \rangle \longrightarrow \langle \texttt{times } z \texttt{ y} \rangle!i = f z \langle y!i \rangle \rangle}
    \begin{itemize}
      \item using \texttt{times-elements} by auto
    \end{itemize}
moreover have \texttt{3:}
\( i < \langle \text{length } y \rangle \)
\( \longrightarrow \langle \text{vec-vec-Tensor } (z\#zs) \rangle \!i = \langle \texttt{times } z \texttt{ y} \rangle!i \)
using \texttt{vec-vec-Tensor-simpl} by auto
moreover have \texttt{35:}
\( i < \langle \text{length } y \rangle \longrightarrow \langle \text{vec-vec-Tensor } (z\#zs) \rangle \!i = f z \langle y!i \rangle \)
using \texttt{calculation(1) calculation(2) by metis}
have \texttt{4:}\( y \neq \emptyset \imeq \langle \text{length } y \rangle > 0 \)
by auto
have \texttt{6:i < \langle \text{length } y \rangle \longrightarrow \langle \langle i < \langle \text{length } x \rangle \rangle \ast \langle \text{length } y \rangle \rangle \longrightarrow \langle \text{vec-vec-Tensor } (z\#zs) \rangle \!i = f \langle (z\#zs)!((i \text{ div } \langle \text{length } y \rangle)) \rangle \langle y! (i \text{ mod } \langle \text{length } y \rangle) \rangle \rangle \}
by auto
\end{itemize}
then have \texttt{7:}\( i < \langle \text{length } y \rangle \longrightarrow \langle i \text{ mod } \langle \text{length } y \rangle \rangle = i \)
by auto
with \texttt{2 6}
\begin{itemize}
  \item have \texttt{3 have step1:}
    \( \langle i < \langle \text{length } y \rangle \rangle \)
    \( \longrightarrow \langle \langle i < \langle \langle \text{length } x \rangle \rangle \ast \langle \text{length } y \rangle \rangle \rangle \longrightarrow \langle \text{vec-vec-Tensor } (z\#zs) \rangle \!i \)
    \begin{itemize}
      \item \texttt{f \langle (z\#zs)!((i \text{ div } \langle \text{length } y \rangle)) \rangle \langle y! (i \text{ mod } \langle \text{length } y \rangle) \rangle \rangle \}}
    \end{itemize}
by auto
  \item have \texttt{3 have step2:}
    \( \langle \langle \text{length } y \rangle \leq i \rangle \longrightarrow \langle i - \langle \text{length } y \rangle \rangle \geq 0 \)
by auto
have \texttt{step2:}
\( \langle \langle \text{length } y \rangle \leq i \rangle \)
\( \longrightarrow \langle \langle i < \langle \langle \text{length } (z\#zs) \rangle \ast \langle \text{length } y \rangle \rangle \rangle \rangle \longrightarrow \langle \text{vec-vec-Tensor } (z\#zs) \rangle \!i \)
\( \texttt{= f} \)
proof–

have \((\text{length } y) > 0\)
using \(\text{assms}\) by auto
then have 1:
\((i > (\text{length } y))\)
\(\rightarrow (i \div (\text{length } y)) = ((i - (\text{length } y)) \div (\text{length } y)) + 1\)
using \(\text{nat-div}\) by auto
have \((\text{zs}!j = (z\#\text{zs})!(j + 1))\)
by auto
then have
\([(\text{zs}!((i - (\text{length } y)) \div (\text{length } y))) = (z\#\text{zs})!(((i - (\text{length } y)) \div (\text{length } y)) + 1))\)
by auto

with 1 have 2:
\((i > (\text{length } y))\)
\(\rightarrow (\text{zs}!((i - (\text{length } y)) \div (\text{length } y))) = (z\#\text{zs})!((i \div (\text{length } y)))\)
by auto
have \((i > (\text{length } y))\)
\(\rightarrow ((y! (i \mod (\text{length } y))) = (y! ((i - (\text{length } y)) \mod (\text{length } y))))\)
using \(\text{nat-mod}\) by auto
then have 3:
\((i > (\text{length } y))\)
\(\rightarrow (((y! (i \mod (\text{length } y))) = (y! ((i - (\text{length } y)) \mod (\text{length } y))))\)
by auto
have 4:\((i > (\text{length } y))\)
\(\rightarrow (\text{vec-vec-Tensor } (z\#\text{zs}) y)!i = (\text{vec-vec-Tensor } z\text{zs } y)!((i - (\text{length } y)))\)
using \(\text{vec-vec-Tensor-simpl2}\) by auto
have 5:\((i > (\text{length } y))\)
\(\rightarrow ((i < ((\text{length } (z\#\text{zs})) \ast (\text{length } y)))) = (i - (\text{length } y) < (\text{length } z\#\text{zs}) \ast (\text{length } y))\)
by auto
then have 6:
\(\forall i.((i < ((\text{length } z\#\text{zs}) \ast (\text{length } y))))\)
\(\rightarrow ((\text{vec-vec-Tensor } z\#\text{zs } y)!i) = f (\text{zs}!((i \div (\text{length } y)))
(y!((i \mod (\text{length } y))))\)
using \(\text{Cons.hyps}\) by auto
with 5 have \((i > (\text{length } y))\)
\(\rightarrow ((i < ((\text{length } (z\#\text{zs})) \ast (\text{length } y))))\)
\(\rightarrow ((\text{vec-vec-Tensor } z\#\text{zs } y)!((i - (\text{length } y)))) = f (\text{zs}!((i - (\text{length } y)) \div (\text{length } y)))\)
\[(y!((i - (\text{length } y)) \mod (\text{length } y))))\]
\[= ((i < ((\text{length } z) \ast (\text{length } y))))\]
\[\rightarrow ((\text{vec-vec-Tensor } z \# y)!i)\]
\[= f\]
\[\rightarrow ((z!((i - (\text{length } y)) \div (\text{length } y))))\]
\[\rightarrow ((y!((i - (\text{length } y)) \mod (\text{length } y))))\]

by auto

with 6 have
\[\rightarrow ((i < ((\text{length } z) \ast (\text{length } y))))\]
\[\rightarrow ((\text{vec-vec-Tensor } z !i)\]
\[= f\]
\[(z!((i - (\text{length } y)) \div (\text{length } y))))\]
\[\rightarrow ((y!((i - (\text{length } y)) \mod (\text{length } y))))\]

by auto

with 2 3 4 have
\[\rightarrow ((i < ((\text{length } z) \ast (\text{length } y))))\]
\[\rightarrow ((\text{vec-vec-Tensor } z !i))\]
\[= f\]
\[(\rightarrow ((z!((i - (\text{length } y)) \div (\text{length } y))))\]
\[\rightarrow ((y!((i - (\text{length } y)) \mod (\text{length } y))))\]

by auto

then show \(?\text{thesis}\) by auto

qed

have \(((\text{length } y) = i)\)
\[\rightarrow ((i < ((\text{length } z) \ast (\text{length } y))))\]
\[\rightarrow ((\text{vec-vec-Tensor } (z \# y)!i)\]
\[= f\]
\[\rightarrow ((z!((i - (\text{length } y)) \div (\text{length } y))))\]
\[\rightarrow ((y!((i - (\text{length } y)) \mod (\text{length } y))))\]

proof-

have 1: \((i = (\text{length } y))\)
\[\rightarrow ((\text{vec-vec-Tensor } (z \# y)!i)\]
\[= (\text{vec-vec-Tensor } z \# y)!0\]

using vec-vec-Tensor-simpl2 by auto

have 2: \((i = \text{length } y)\) \rightarrow \((i \mod (\text{length } y)) = 0\)
by auto

have 3: \((i = \text{length } y)\) \rightarrow \((i \div (\text{length } y)) = 1\)
using 4 assms div-self less-numeral-extra(3)
by auto

have 4: \((i = \text{length } y)\)
\[\rightarrow ((i < ((\text{length } z) \ast (\text{length } y)))\]
\[= (0 < ((\text{length } z) \ast (\text{length } y)))))\]

by auto

have \((z \# y)!1 = (z!0)\)
by auto

with 3 have 5: \((i = \text{length } y)\)
\[\rightarrow ((z!((i - (\text{length } y))))) = (z!0)\]
\begin{verbatim}
by auto
have \( \forall i . (i < (\text{length } zs) \ast (\text{length } y)) \rightarrow ((\text{vec-vec-Tensor } \langle zs, y \rangle)!i) = f ((zs)!((i \div (\text{length } y))) (y!(i \mod (\text{length } y)))) \)

using Cons.hyps by auto
with \( \alpha \) have 6: \((i = \text{length } y) \rightarrow ((0 < ((\text{length } zs) \ast (\text{length } y))) \rightarrow (((\text{vec-vec-Tensor } \langle zs, y \rangle)!0) = f ((zs)!0) (y!0)) \)

by auto
have 7: \((0 \div (\text{length } y)) = 0 \)
by auto
have 8: \((0 \mod (\text{length } y)) = 0 \)
by auto
have 9: \((0 < ((\text{length } zs) \ast (\text{length } y))) \rightarrow (((\text{vec-vec-Tensor } \langle zs, y \rangle)!0) = f (zs!0) (y!0) \)

using 7 8 Cons.hyps by auto
with \( \alpha \beta \gamma \) have \((i = \text{length } y) \rightarrow ((i < ((\text{length } (z \# zs)) \ast (\text{length } y))) \rightarrow (((\text{vec-vec-Tensor } \langle z \# zs, y \rangle)!i) = f ((z \# zs)!((i \div (\text{length } y))) (y!(i \mod (\text{length } y)))) \)

by auto
with \( \alpha \beta \) have \((i = \text{length } y) \rightarrow ((i < ((\text{length } (z \# zs)) \ast (\text{length } y))) \rightarrow (((\text{vec-vec-Tensor } \langle z \# zs, y \rangle)!i) = f ((z \# zs)!((i \div (\text{length } y))) (y!(i \mod (\text{length } y)))) \)

by auto
then show ?thesis by auto
qed
with step2 have step4:
\((i \geq (\text{length } y)) \rightarrow ((i < ((\text{length } (z \# zs)) \ast (\text{length } y))) \rightarrow (((\text{vec-vec-Tensor } \langle z \# zs, y \rangle)!i) = f ((z \# zs)!((i \div (\text{length } y))) (y!(i \mod (\text{length } y)))) \)

by auto
have \((i < (\text{length } y)) \lor (i \geq (\text{length } y)) \)
by auto
\end{verbatim}
with step1 step4 have
((i < (length (z#zs))*(length y))
→ (((vec-vec-Tensor ((z#zs)) y)!i)
= f
((z#zs)!(i div (length y))))
(y!(i mod (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: nat (int x + int y) = x + y
using nat-int of-nat-add by auto

lemma int-nat-equiv: (x > 0) → (nat ((int x) + -1)+1) = x
proof
have 1 = nat (int 1)
  by auto
have -1 = -int 1
  by auto
then have 1:(nat ((int x) + -1)+1)
   = (nat ((int x) + -1) + (nat (int 1)))
  by auto
then have 2:(x > 0)
   → nat ((int x) + -1 ) + (nat (int 1))
   = (nat (((int x) + -1) + (int 1)))
using of-nat-add nat-int by auto
have (nat (((int x) + -1) + (int 1))) = (nat ((int x) + -1 + (int 1)))
by auto
then have (nat (((int x) + -1) + (int 1))) = (nat ((int x)))
by auto
then have (nat (((int x) + -1) + (int 1))) = x
by auto
with 1 2 have (x > 0) → nat ((int x) + -1 ) + 1 = x
by auto
then show ?thesis by auto
qed

lemma list-int-nat: (k>0) → ((x#xs)!k = xs!(nat ((int k)+-1)))
proof-
  fix j
have ((x#xs)!(k+1) = xs!k)
  by auto
have j = (k+1) → (nat ((int j)+-1)) = k
  by auto
moreover have (nat ((int j)+-1)) = k
  → ((nat ((int j)+-1)) + 1) = (k +1)
  by auto
moreover have (j>0)→(((nat ((int j)+-1)) + 1) = j)
using int-nat-equiv by (auto)
moreover have \((k>0) \rightarrow ((x#xs)!k = xs!(nat ((int k)+-1)))\)
using Suc-eq-plus1 int-nat-equiv nth-Cons-Suc by (metis)

from this show ?thesis by auto
qed

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

The following theorem tells us the relationship between entries of vec_mat_Tensor v M and entries of v and M respectively

theorem vec-mat-Tensor-elements:
\(\forall i.\forall j.\)
\(((i<((\text{length} \ v)*\text{row-length} \ M))\)
\&(j < (\text{length} \ M)))
\&(\text{mat} \ (\text{row-length} \ M) \ (\text{length} \ M) \ M)\)
\(\rightarrow\) \((\text{vec-mat-Tensor} \ v \ M)!j!i\)
\(=\) \(f \ ((v!((i \ div \ (\text{row-length} \ M)))) \ (M!y!((i \ mod \ (\text{row-length} \ M))))))\)
apply(rule allI)
apply(rule allI)
proof(induct M)
case Nil
have \(\text{row-length} \ [] = 0\)
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
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))))
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!(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
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 (row-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)))
    ((a # M) ![0] ![i] (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)))
    ((a # M) ![j] ![i] (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)))
    ((a # M) ![j] ![i] (i mod (row-length (a # M))))
  proof(cases M)
case Nil
  have (length (a # [])) = 1

proof(cases M)
case Nil
  have (length (a # [])) = 1

by auto
then have \((j < (\text{length } (a[]))) = (j = 0)\)
by auto
then have \(((i < (\text{row-length } (a[]))*(\text{length } v))\)
\&\((j < (\text{length } (a[]))))\)
\& (\text{mat} (\text{row-length } (a[])) (\text{length } (a[])) (a[]))
\rightarrow ((\text{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:((\text{mat} (\text{row-length } (a\#b\#N)) (\text{length } (a\#b\#N)) (a\#b\#N))
\rightarrow \text{row-length } (a\#b\#N) = (\text{row-length } (b\#N))
using \text{row-length-eq} by metis
have 8: \((j > 0)\)
\rightarrow ((\text{vec-mat-Tensor } v (b\#N))(\text{nat} ((\text{int } j)+-1)))
    = (\text{vec-mat-Tensor } v (a\#b\#N))|j
using \text{vec-mat-Tensor.simps(2)} using list-int-nat by metis
have 9: \((j > 0)\)
\rightarrow (((i < (\text{row-length } (b\#N))*(\text{length } v))\)
\&((\text{nat} ((\text{int } j)+-1)) < (\text{length } (b\#N))))
\&(\text{mat} (\text{row-length } (b\#N)) (\text{length } (b\#N)) (b\#N))
\rightarrow ((\text{vec-mat-Tensor } v (b\#N))(\text{nat} ((\text{int } j)+-1))|i)
    = f
        (v!\((i \text{ div } (\text{row-length } (b\#N)))\))
        ((b\#N)|(\text{nat} ((\text{int } j)+-1))|j!\((i \text{ mod } (\text{row-length } (b\#N)))\)))
using Cons.hyps Cons mult.commute by metis
have \((j > 0)\) \rightarrow ((\text{nat} ((\text{int } j) + -1)) < (\text{length } (b\#N)))
\rightarrow ((\text{nat} ((\text{int } j) + -1) + 1) < \text{length } (a\#b\#N))
by auto
then have \((j > 0)\)
\rightarrow ((\text{nat} ((\text{int } j) + -1)) < \text{length } (a\#b\#N)) = (j < \text{length } (a\#b\#N))
by auto
then have \((j > 0)\)
\rightarrow ((\text{nat} ((\text{int } j) + -1)) < \text{length } (a\#b\#N))
\& (j < \text{length } (a\#b\#N))\)
\&(\text{mat} (\text{row-length } (b\#N)) (\text{length } (b\#N)) (b\#N))
\rightarrow ((\text{vec-mat-Tensor } v (b\#N))(\text{nat} ((\text{int } j)+-1))|i)
    = f
        (v!\((i \text{ div } (\text{row-length } (b\#N)))\))
        ((b\#N)|(\text{nat} ((\text{int } j)+-1))|j!\((i \text{ mod } (\text{row-length } (b\#N)))\)))
using Cons.hyps Cons mult.commute by metis
with 8 have \((j > 0)\)
\rightarrow (((i < (\text{row-length } (b\#N))*(\text{length } v))\)
moreover have

\[ (\forall i < j \quad (i < (row-length (b\#N)) \land j < length (b\#N))) \]

\[ \land (mat (row-length (b\#N)) (length (b\#N)) (a\#b\#N)) \]

\[ \quad \rightarrow \]

\[ ((\text{vec-mat-Tensor} v (a\#b\#N))!j!i) \]

\[ = f \]

\[ (v!(i div (row-length (b\#N)))) \]

\[ ((a\#b\#N)!j!(i mod (row-length (b\#N)))) \]

by auto

also have \((j > 0) \quad \rightarrow \quad (b\#N)!((nat ((int j) - 1))!i mod (row-length (b\#N)))) \]

using list-int-nat by metis

moreover have \((j > 0) \quad \rightarrow \quad (((i < (row-length (b\#N)) \land (j < length (a\#b\#N))) \land (mat (row-length (a\#b\#N)) (length (a\#b\#N)) (a\#b\#N)) \]

\[ \quad \rightarrow \]

\[ ((\text{vec-mat-Tensor} v (a\#b\#N))!j!i) \]

\[ = f \]

\[ (v!(i div (row-length (b\#N)))) \]

\[ ((a\#b\#N)!j!(i mod (row-length (b\#N)))) \]

by (metis calculation(1) calculation(2))

then have

\((j > 0) \quad \rightarrow \quad (((i < (row-length (b\#N)) \land (j < length (a\#b\#N))) \land (mat (row-length (a\#b\#N)) (length (a\#b\#N)) (a\#b\#N)) \]

\[ \quad \rightarrow \]

\[ ((\text{vec-mat-Tensor} v (a\#b\#N))!j!i) \]

\[ = f \]

\[ (v!(i div (row-length (b\#N)))) \]

\[ ((a\#b\#N)!j!(i mod (row-length (b\#N)))) \]

using reduct-matrix by (metis)

moreover have \((mat (row-length (a\#b\#N)) (length (a\#b\#N)) (a\#b\#N)) \rightarrow (row-length (b\#N)) = (row-length (a\#b\#N)) \]

by (metis 7 Cons)

moreover have \(10: (j > 0) \quad \rightarrow \quad (((i < (row-length (a\#b\#N)) \land (j < length (a\#b\#N))) \land (mat (row-length (a\#b\#N)) (length (a\#b\#N)) (a\#b\#N)) \]

\[ \quad \rightarrow \]

\[ ((\text{vec-mat-Tensor} v (a\#b\#N))!j!i) \]

\[ = f \quad (v!(i div (row-length (a\#b\#N)))) \]

\[ ((a\#b\#N)!j!(i mod (row-length (a\#b\#N)))) \]

by (metis calculation(3) calculation(4))

have \((j = 0) \quad \lor \quad (j > 0) \]

by auto

with \(6 \quad 10\) logic have

\[((i < (row-length (a\#b\#N)) \land (j < length (a\#b\#N))) \land (mat (row-length (a\#b\#N)) (length (a\#b\#N)) (a\#b\#N)) \]

\[ \quad \rightarrow \]

\[ ((\text{vec-mat-Tensor} v (a\#b\#N))!j!i) \]

\[ = f \quad (v!(i div (row-length (a\#b\#N)))) \]

\[ ((a\#b\#N)!j!(i mod (row-length (a\#b\#N)))) \]

by (auto)
\[ \begin{align*} \rightarrow \\
& ((\text{vec-mat-Tensor } v \ (a\#b\#N)!j!i) \\
& = f \\
& (v!(i \text{ div } (\text{row-length (a\#b\#N)}))) \\
& ((a\#b\#N)!j(i \text{ mod } (\text{row-length (a\#b\#N)}))) \\
\text{using Cons by metis} \end{align*} \]

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

\textbf{theorem matrix-Tensor-elements:}

\textbf{fixes} \ M1 M2

\textbf{shows}

\[ \forall i.\forall j.((i<(\text{row-length } M1)\ast(\text{row-length } M2)) \\
\land(j < (\text{length } M1)\ast(\text{length } M2))) \\
\land(\text{mat (row-length } M1) \ (\text{length } M1) \ M1) \\
\land(\text{mat (row-length } M2) \ (\text{length } M2) \ M2) \\
\rightarrow ((M1 \otimes M2)!j!i) = f \\
\text{M1}(j \text{ div } (\text{length } M2))!(i \text{ div } (\text{row-length } M2))) \\
\text{M2}(j \text{ mod length } M2)!(i \text{ mod } (\text{row-length } M2))) \]

apply(rule allI)
apply(rule allI)

\textbf{proof (induct } M1)

\textbf{case } Nil

have (row-length []) = 0

\textbf{using row-length-def by auto}

then have (i<((row-length [])\ast(\text{row-length } M2))) \rightarrow False

by auto

from this have ((i<(\text{row-length []})\ast(\text{row-length } M2)) \\
\land(j < (\text{length []})\ast(\text{length } M2))) \\
\land(\text{mat (row-length [])} \ (\text{length []} [])) \\
\land(\text{mat (row-length } M2) \ (\text{length } M2) \ M2) \\
\rightarrow False \\

\textbf{by auto}

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

\textbf{by auto}

moreover have ((i<(row-length [])\ast(row-length M2))) \\
\land(j < (\text{length []})\ast(\text{length } M2))) \\
\land(\text{mat (row-length [])} \ (\text{length []} [])) \\
\land(\text{mat (row-length } M2) \ (\text{length } M2) \ M2) \\
\rightarrow (([] \otimes M2)!j!i) = f \]
by auto
then show \( \text{case by auto} \)
next
case (Cons \( v \ M \))
fix \( a \)
have \( 0:\( v \# M \) \otimes M2 = (\text{vec-mat-Tensor } v \ M2) \otimes (\text{Tensor } M \ M2) \)
by auto
then have \( 1:\)
\( (j < (\text{length } M2)) \rightarrow ((v \# M) \otimes M2)!j = (\text{vec-mat-Tensor } v \ M2)!j \)
using append-simpl vec-mat-Tensor-length by metis
have \( (j < (\text{length } M2)) \land (\text{mat } \text{row-length } M2 \ (\text{length } M2) \ M2) \rightarrow ((\text{vec-mat-Tensor } a \ M2)!j!i) = f \ (a!(i \div \text{row-length } M2)) \ \text{(M2)!j!(i \mod (\text{row-length } M2)))} \)
using vec-mat-Tensor-elements by auto
have \( (j < (\text{length } M2)) \rightarrow (j \div \text{length } M2) = 0 \)
by auto
then have \( 2:\)
\( (j < (\text{length } M2)) \rightarrow (v \# M)!j \div ((\text{length } M2)) = v \)
by auto
have \( (j < (\text{length } M2)) \rightarrow (j \mod \text{length } M2) = j \)
by auto
moreover have \( (j < (\text{length } M2)) \rightarrow (v \# M)!j \mod (\text{length } M2) = (v \# M)!j \)
by auto
have \( \) step0:
\( (j < (\text{length } M2)) \rightarrow ((i < (\text{length } v)*\text{row-length } M2)) \land (j < (\text{length } M2) \land (\text{length } (v \# M))) \land (\text{mat } \text{row-length } M2 \ (\text{length } M2) \ M2) \rightarrow ((\text{Tensor } (v \# M) \ M2)!j!i) = f \)
\((v \# M)!j \div ((\text{row-length } M2)!j \mod (\text{row-length } M2))) \)
using 2 1 calculation(1) vec-mat-Tensor-elements by auto
have \( \) step1:
\( (j < (\text{length } M2)) \rightarrow ((i < (\text{row-length } (v \# M)) \land (\text{length } M2))) \land (j < (\text{length } (v \# M)) \land (\text{length } M2))) \land (\text{mat } (\text{row-length } (v \# M)) \ (\text{length }v \# M)) \land (\text{mat } \text{row-length } M2 \ (\text{length } M2) \ M2) \rightarrow ((\text{Tensor } (v \# M) \ M2)!j!i) = f \)
\((v \# M)!j \div ((\text{row-length } M2)!j \mod (\text{row-length } M2))) \)
using row-length-def step0 by auto
from 0 have \( 3:\)
\( (j \geq (\text{length } M2)) \rightarrow ((v \# M) \otimes M2)!j = (M \otimes M2)!j \rightarrow (\text{length } M2)) \)
32
using vec-mat-Tensor-length add.commute append-simpl2 by metis
have 4:
\((j \geq (\text{length } M2)) \rightarrow ((i < (\text{row-length } M) \ast (\text{row-length } M2)) \land (j - (\text{length } M2)) < (\text{length } M) \ast (\text{length } M2)) \land (\text{mat } (\text{row-length } M2) (\text{length } M) M) \land (\text{mat } (\text{row-length } M2) (\text{length } M2) M2) \rightarrow ((M \otimes M2)!((j - (\text{length } M2))!i)) = f ((M!((j - (\text{length } M2)) \div (\text{length } M2))!((i \div (\text{row-length } M2)))) \land (M2!((j - (\text{length } M2)) \mod \text{length } M2)!((i \mod (\text{row-length } M2))))))
using Cons.hyps by auto
moreover have (mat (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M)) \rightarrow (mat (\text{row-length } M) (\text{length } M) M)
using reduct-matrix by auto
moreover have 5:
\((j \geq (\text{length } M2)) \rightarrow ((i < (\text{row-length } M) \ast (\text{row-length } M2)) \land (j - (\text{length } M2)) < (\text{length } M) \ast (\text{length } M2)) \land (\text{mat } (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M)) \land (\text{mat } (\text{row-length } M2) (\text{length } M2) M2) \rightarrow ((M \otimes M2)!((j - (\text{length } M2))!i)) = f ((M!((j - (\text{length } M2)) \div (\text{length } M2))!((i \div (\text{row-length } M2)))) \land (M2!((j - (\text{length } M2)) \mod \text{length } M2)!((i \mod (\text{row-length } M2))))))
using 4 calculation(3) by metis
have (((j - (\text{length } M2)) < (\text{length } M) \ast (\text{length } M2))) \rightarrow ((j < (((\text{length } M) + 1) \ast (\text{length } M2)))))
by auto
then have 6:
\(((j - (\text{length } M2)) < (\text{length } M) \ast (\text{length } M2))) \rightarrow (j < (((\text{length } (v\#M)) \ast (\text{length } M2)))))
by auto
have 7:
\((j \geq (\text{length } M2)) \rightarrow ((j - (\text{length } M2)) \div (\text{length } M2)) = (j \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)) \rightarrow (M!((j - (\text{length } M2)) \div (\text{length } M2)) = M!((j \div (\text{length } M2)) - 1))
by auto
have step2:
\((j \geq (\text{length } M2))

33
→

\((i < ((\text{row-length } (v\#M)) \times (\text{row-length } M2)))\)
\(\land (j < (\text{length } (v\#M)) \times (\text{length } M2)))\)
\(\land (\text{mat } (\text{row-length } (v\#M)) \text{ (length } (v\#M)) \text{ (v\#M)})\)
\(\land (\text{mat } (\text{row-length } M2) \text{ (length } M2) \text{ M2}))\)

\(\rightarrow ((v\#M) \land M2)!j!i = \)

\(f\)
\((v\#M)!j \text{ div } (\text{length } M2)!)(i \text{ div } (\text{row-length } M2))\)
\((M2!)(j \text{ mod } \text{length } M2)!)(i \text{ mod } (\text{row-length } M2))\)

**proof** *(cases M2)*

**case** \(\text{Nil}\)

**have** \((0 = ((\text{row-length } (v\#M)) \times (\text{row-length } M2)))\)

**using** \(\text{row-length-def Nil mult-0-right by auto}\)

**then have** \((i < ((\text{row-length } (v\#M)) \times (\text{row-length } M2))) \rightarrow \text{False}\)

**by auto**

**then have** \((j \geq (\text{length } M2))\)

\(\rightarrow (((i < ((\text{row-length } (v\#M)) \times (\text{row-length } M2)))\)
\(\land (j < (\text{length } (v\#M)) \times (\text{length } M2)))\)
\(\land (\text{mat } (\text{row-length } (v\#M)) \text{ (length } (v\#M)) \text{ (v\#M)})\)
\(\land (\text{mat } (\text{row-length } M2) \text{ (length } M2) \text{ M2}))\)

\(\rightarrow \text{False}\)

**by auto**

**then show** \(?\text{thesis by auto}\)

**next**

**case** \((\text{Cons } w \text{ N})\)

**fix** \(k\)

**have** \((k \not< (\text{length } M)) \land (k \geq 1) \rightarrow \text{M}(k - 1) = (v\#M)!k\)

**using** \(\text{not-one-le-zero nth-Cons’ by auto}\)

**have** \((j \geq (\text{length } (w\#N))) \rightarrow (j \text{ div } (\text{length } (w\#N))) \geq 1\)

**using** \(\text{div-le-mono div-self length-0-conv neq-Nil-conv by metis}\)

**moreover have** \((j \geq (\text{length } (w\#N))) \rightarrow (j \text{ div } (\text{length } (w\#N))) - 1 \geq 0\)

**by auto**

**moreover have** \((j \geq (\text{length } (w\#N)))\)

\(\rightarrow \text{M}((j - (\text{length } (w\#N))) \text{ div } (\text{length } (w\#N)))\)

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

**using** \(\text{calculation(1) not-one-le-zero nth-Cons’ by auto}\)

**from this 7 have** 9: \((j \geq (\text{length } (w\#N)))\)

\(\rightarrow \text{M}((j - (\text{length } (w\#N))) \text{ div } (\text{length } (w\#N)))\)

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

**using** \(\text{Cons by auto}\)

**have** 10: \((j \geq (\text{length } (w\#N)))\)

\(\rightarrow ((j - (\text{length } (w\#N))) \text{ mod } (\text{length } (w\#N)))\)

\(= (j \text{ mod } (\text{length } (w\#N)))\)

**using** \(\text{mod-if not-less by auto}\)

**with** 5 9 have

\((j \geq (\text{length } (w\#N))) \rightarrow\)

\((i < ((\text{row-length } M) \times (\text{row-length } (w\#N))))\)

\((j - (\text{length } (w\#N))) < (\text{length } M) \times (\text{length } (w\#N)))\)

\(\land (\text{mat } (\text{row-length } (v\#M)) \text{ (length } (v\#M)) \text{ (v\#M)})\)

34
\(11\)

then have

\[ \begin{align*}
&\text{using } \text{Cons by auto} \\
&\text{then have } (j \geq \text{(length (w#N)))} \\
&\quad \text{using } 6 \text{ by auto} \\
&\text{then have } 11:\]

\[\begin{align*}
&\text{have } (j \geq \text{(length (w#N)))} \\
&\quad \text{using } 3 \text{ Cons by auto} \\
&\text{case Nil} \\
&\text{have } \text{Nil0: (length (v[#0]))} = 1 \\
&\quad \text{by auto} \\
&\text{then have } \text{Nil1:} \\
&\quad (j < \text{(length (v[#0]))* (length (w#N)))} = (j < \text{(length (w#N))}) \\
&\quad \text{by (metis Nil nat-mult-1)} \\
&\text{have } \text{row-length (v[#0])} = \text{(length v)} \\
&\quad \text{using } \text{row-length-def by auto} \\
&\text{then have } \text{Nil2:}
\end{align*}\]
\[(i < ((\text{row-length} (v \# M)) \ast (\text{row-length} (w \# N))))\]
\[(\rightarrow (i < ((\text{length} v) \ast (\text{row-length} (w \# N))))\]
\[\text{using Nil by auto}\]
then have \((j < (\text{length} (w \# N))) \rightarrow (j \text{ div} (\text{length} (w \# N))) = 0\]
by auto
from this have Nil3:
\[(j < (\text{length} (w \# N))) \rightarrow (v \# M)!(j \text{ div} (\text{length} (w \# N))) = v\]
\[\text{using Nil by auto}\]
then have Nil4:
\[(j < (\text{length} (w \# N))) \rightarrow (j \text{ mod} (\text{length} (w \# N))) = j\]
by auto
then have Nil5: \((v \# M) \otimes (w \# N) = \text{vec-mat-Tensor} v (w \# N)\]
\[\text{using Nil Tensor.simps(2) Tensor.simps(1)}\]
by auto
from vec-mat-Tensor-elements have
\[((i < ((\text{length} v) \ast (\text{row-length} (w \# N))))\]
\[\land (j < (\text{length} (w \# N))))\]
\[\land (\text{mat} (\text{row-length} (w \# N)) (\text{length} (w \# N)) (w \# N))\]
\[\rightarrow ((\text{vec-mat-Tensor} v (w \# N))!j!i)\]
\[= f\]
\[(v!(i \text{ div} (\text{row-length} (w \# N))))\]
\[((w \# N)!j!(i \text{ mod} (\text{row-length} (w \# N))))\]
\[\text{by metis}\]
then have
\[((i < ((\text{row-length} (v \# M)) \ast (\text{row-length} (w \# N))))\]
\[\land (j < (\text{length} (v \# M)) \ast (\text{length} (w \# N))))\]
\[\land (\text{mat} (\text{row-length} (w \# N)) (\text{length} (w \# N)) (w \# N))\]
\[\rightarrow (((v \# M) \otimes (w \# N))!j!i)\]
\[= f\]
\[((v \# M)!j!(i \text{ div} (\text{length} (w \# N)))!j!(i \text{ mod} (\text{row-length} (w \# N))))\]
\[(w \# N)!j!(i \text{ mod} (\text{row-length} (w \# N))))\]
\[\text{using Nil1 Nil2 Nil by auto}\]
then have
\[((i < ((\text{row-length} (v \# M)) \ast (\text{row-length} (w \# N))))\]
\[\land (j < (\text{length} (v \# M)) \ast (\text{length} (w \# N))))\]
\[\land (\text{mat} (\text{row-length} (w \# N)) (\text{length} (v \# M)) (v \# M))\]
\[\land (\text{mat} (\text{row-length} (w \# N)) (\text{length} (w \# N)) (w \# N))\]
\[\rightarrow (((v \# M) \otimes (w \# N))!j!i)\]
\[= f\]
\[((v \# M)!j!(i \text{ div} (\text{length} (w \# N)))!j!(i \text{ div} (\text{row-length} (w \# N))))\]
\[(w \# N)!j!(i \text{ mod} (\text{length} (w \# N)))!j!(i \text{ mod} (\text{row-length} (w \# N))))\]
\[\text{using Nil3 Nil4 Nil5 Nil by auto}\]
then have
\[((i < ((\text{row-length} (v \# M)) \ast (\text{row-length} (w \# N))))\]
\[\land (j < (\text{length} (v \# M)) \ast (\text{length} (w \# N))))\]
\[\land (\text{mat} (\text{row-length} (v \# M)) (\text{length} (v \# M)) (v \# M))\]
\[\land (\text{mat} (\text{row-length} (w \# N)) (\text{length} (w \# N)) (w \# N))\]
\[\rightarrow (((v \# M) \otimes (w \# N))!j!i)\]
\[= f\]
\[((v \# M)!j!(i \text{ div} (\text{length} (w \# N)))!j!(i \text{ div} (\text{row-length} (w \# N))))\]
\[(w \# N)!j!(i \text{ mod} (\text{length} (w \# N)))!j!(i \text{ mod} (\text{row-length} (w \# N))))\]
\[\text{using Nil3 Nil4 Nil5 Nil by auto}\]
by auto

from this show ?thesis by auto

next

case (Cons u P)
  have (mat (row-length (v#M)) (length (v#M)) (v#M)) → (row-length (v#M)) = (row-length M)
    using Cons row-length-eq by metis
  from this show ?thesis by auto

qed

from this show ?thesis using Cons by auto

qed

have (j<(length M2)) ∨ (j ≥ (length M2)) by auto

from this step1 step2 logic have
  (((i<(row-length (v#M))*(row-length M2)))
    ∧ (j < (length M2) * (length (v#M))))
    ∧ (mat (row-length (v#M)) (length (v#M)) (v#M))
    ∧ (mat (row-length M2) (length M2) M2)
    → ((v#M)⊗ M2)!j!i)
  = f
    (M2!(j mod (length M2))!(i mod (row-length M2)))
    (v#M)!(i div (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<(row-length M1)* (row-length M2)))
    ∧ (j < (length M1) * (length M2)))
    ∧ (mat (row-length M1) (length M1) M1)
    ∧ (mat (row-length M2) (length M2) M2)
    ⇒ ((M1 ⊗ M2)!j!i)
  = f (M1!(j div (length M2))!(i div (row-length M2)))
    (M2!(j mod length M2))!(i mod (row-length M2)))

using matrix-Tensor-elements by auto

theorem effective-matrix-tensor-elements2:
  assumes i<(row-length M1)* (row-length M2)
    and j < (length M1) * (length M2)
    and mat (row-length M1) (length M1) M1
    and mat (row-length M2) (length M2) M2
  shows (M1 ⊗ M2)!j!i =
    (M1!(j div (length M2))!(i div (row-length M2)))
    * (M2!(j mod length M2)!((i mod (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::nat) < y*z
shows \((x \div z) < y\)
proof (rule ccontr)
assume 0: \(!((x \div z) < y)\)
then have 1: \(x \div z \geq y\)
  by auto
then have 2: \((x \div z) \ast z \geq y \ast z\)
  by auto
then have 3: \((x \div z) \ast z + (x \mod z) = z\)
  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: \((x \div z) \ast z \leq z\)
  by auto
then have 5: \(z \geq y \ast z\)
  using 2 by auto
then have 6: \(z \div z \geq (y \ast z) \div z\)
  by auto
then have \((y \ast z) \div z \leq 1\)
  by auto
with 6 have 1: \(y \geq y\)
  using 1 3 assms div-self less-nat-zero-code mult-zero-left
  mult.commute mod-div-mult-eq
  by auto
then have 7: \((y = 0) \lor (y = 1)\)
  by auto
have \((y = 0) \implies x < 0\)
  using assms by auto
moreover have \(x \geq 0\)
  by auto
then have 8: \((y = 0) \implies \text{False}\)
  using calculation less-nat-zero-code by auto
moreover have \((y = 1) \implies (x < z)\)
  using assms by auto
then have \((y = 1) \implies (x \div z) = 0\)
  by (metis div-less)
then have \((y = 1) \implies (x \div z) < y\)
  by auto
then have \((y = 1) \implies \text{False}\)
  using 0 by auto
then show \(\text{False}\) using 7 8 by auto
qed

lemma div-right-ineq:
assumes \((x :: \text{n}at) < y \cdot z\)
shows \((x \div y) < z\)
using assms div-left-ineq mult.commute by (metis)

In the following theorem, we obtain columns of vec_matTensor 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:**
\[
\forall j. (j < (\text{length } M)) \\
\rightarrow \quad \text{col} (\text{vec-mat-Tensor } v M) j = \text{vec-vec-Tensor } v (\text{col } M j)
\]

**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)!(j div length M2)
  using Cons-1 Suc by auto
  moreover have j < length (w#N) \rightarrow k < length N
  using Suc by (metis length-Suc-conv not-less-eq)
  moreover then have k < length (N)
  \rightarrow (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 < (length M1)*(length M2)) \\
\wedge (\text{mat} (\text{row-length } M1) (\text{length } M1) M1) \\
\wedge (\text{mat} (\text{row-length } M2) (\text{length } M2) M2) \\
\rightarrow \text{col} (M1 \otimes M2) j = \text{vec-vec-Tensor} \\
\quad (\text{col } M1 (j \text{ div length } M2)) \\
\quad (\text{col } M2 (j \text{ 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)
    \land mat (row-length (v # M)) (length (v # M)) (v # M)
    \land mat (row-length M2) (length M2) M2 \rightarrow
    (col (v # M) \otimes 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)
    \land mat (row-length (v # M)) (length (v # M)) (v # M)
    \land 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 \geq (length M2)
    \rightarrow 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 (hide-lams, no-types))
  have 3:(v # M) \otimes M2 = (vec-mat-Tensor v M2) @ (M \otimes M2)
  using Tensor.simps by auto
  have (col ((v # M) \otimes M2) j) = (col ((vec-mat-Tensor v M2) @ (M \otimes M2)) j)
  using col-def by auto
then have j < length (vec-mat-Tensor v M2)
  \rightarrow (col ((v # M) \otimes M2) j) = (col (vec-mat-Tensor v M2) j)
  unfolding col-def using append-simpl by auto
then have 4:j < length M2 \rightarrow
  (col ((v # M) \otimes M2) j) = (col (vec-mat-Tensor v M2) j)
  using vec-mat-Tensor-length by simp
then have j < length M2 \rightarrow
  (col (vec-mat-Tensor v M2) j)
  = vec-vec-Tensor (col M2 j)
  using col-vec-mat-Tensor by auto
then have
  j < length M2 \rightarrow
  (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: \( j < \text{length } M_2 \implies (\text{col } ((v\#M) \otimes M_2) j) = \text{vec-vec-Tensor} ((v\#M)! (j \text{ div length } M_2) \text{ mod length } M_2)) \)
using 4 by auto
have 4: \( j \geq \text{length } M_2 \implies (\text{col } ((v\#M) \otimes M_2) j) = (M \otimes M_2)! (j - (\text{length } M_2)) \)
unfolding col-def using 3 append-simpl2 vec-mat-Tensor-length by metis
then have 5:
\( j \geq \text{length } M_2 \implies \text{col } (M \otimes M_2) (j - \text{length } M_2) = \text{vec-vec-Tensor} (\text{col } M ((j - \text{length } M_2) \text{ div length } M_2)) \text{ mod length } M_2) \)
using 1 0 2 Cons by auto
then have 6:
\( j \geq \text{length } M_2 \implies (j - \text{length } M_2) \text{ div length } M_2 + 1 = j \text{ div length } M_2 \)
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-cone div-add-self1 le-add-diff-inverse
by metis
then have
\( j \geq \text{length } M_2 \implies ((j - \text{length } M_2) \text{ mod length } M_2) = j \text{ mod length } M_2 \)
using le-mod-geq by metis
with 6 have 7:
\( j \geq \text{length } M_2 \implies \text{col } (M \otimes M_2) (j - \text{length } M_2) = \text{vec-vec-Tensor} (\text{col } M ((j - \text{length } M_2) \text{ div length } M_2)) \text{ mod length } M_2) \)
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 \( j \geq \text{length } M_2 \implies \text{col } (M \otimes M_2) (j - \text{length } M_2) = \text{vec-vec-Tensor} (\text{col } (v\#M) (j \text{ div length } M_2)) \text{ mod length } M_2) \)
proof
assume temp:j \geq \text{length } M_2
have \( j - (\text{length } M_2) < (\text{length } M) \times (\text{length } M_2) \)
using 2 temp by auto
then have \( (j - (\text{length } M_2)) \text{ div length } M_2 < (\text{length } M) \)
using div-right-ineq mult.commute by metis
moreover have
\[(j - (\text{length } M2)) \div (\text{length } M2)) \leq (\text{length } M)\]

\[= (\text{col } M ((j - (\text{length } M2)) \div (\text{length } M2)))\]

unfolding col-def by auto
ultimately have temp1:
\[(\text{col } (v \# M) (((j - \text{length } M2) \div \text{length } M2))) + 1)\]

by auto
then have \((\text{col } (v \# M) (((j - \text{length } M2) \div \text{length } M2)))\]

using 6 temp by auto
then show \(?thesis\) using temp1 7 by (metis temp)
qed
then have \(j \geq \text{length } M2 \Rightarrow \)
\[(\text{col } ((v \# M) \otimes M2)) j\]

= vec-vec-Tensor (\(\text{col } (v \# M) (j \div \text{length } M2)\))
\[(\text{col } M2 (j \mod \text{length } M2))\]

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:** \(\text{row } (v \# M) i = (v!i)\#(\text{row } M i)\)
unfolding row-def map-def by auto

**Lemma row-append:** \(\text{row } (A@B)i = (\text{row } A i)@(\text{row } B i)\)
unfolding row-def map-append by auto

**Lemma row-empty:** \(\text{row } [] i = []\)
unfolding row-def by auto

**Lemma vec-vec-Tensor-right-empty:** vec-vec-Tensor \(x [] = []\)
using vec-vec-Tensor.simps times.simps length-0-cons 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 \(\Rightarrow [[i]] = []\)
by auto

**Lemma row-vec-mat-Tensor-prelim:**
\[\forall i. \]
\[(i < (\text{length } v))\ast(\text{row-length } M)) \land (\text{mat nr } (\text{length } M) M) \Rightarrow \text{row } (\text{vec-mat-Tensor } v M) i\]
\[ = \text{times} (v(i \text{ div row-length } M)) (\text{row } M (i \text{ mod row-length } M)) \]

\textbf{apply (rule allI)}

\textbf{proof (induct } M)\]

\textbf{case Nil}

\textbf{show ?case using Nil by (metis less-nat-zero-code mult-0-right row-length-Nil)}

\textbf{next}

\textbf{case (Cons } w \ N)\]

\textbf{have \text{row} (vec-mat-Tensor v (w\#N)) i} = \text{row} ((vec-vec-Tensor v w)\#(vec-mat-Tensor v N)) i

\textbf{using} \text{vec-mat-Tensor..simps by auto}

\textbf{then have \text{1:}... = ((vec-vec-Tensor v w)!i)\#(row (vec-mat-Tensor v N) i)}

\textbf{using row-Cons by auto}

\textbf{have 2:row-length (w\#N) = length w}

\textbf{using row-length-def by auto}

\textbf{then have 3:(mat nr (length (w\#N)) (w\#N)) \implies nr = length w}

\textbf{using hd-in-set list.distinct(1) mat-uniqueness matrix-row-length by metis}

\textbf{then have \text{((i < (length v)\star(row-length (w\#N)))}}

\textbf{\wedge (mat nr (length (w\#N)) (w\#N))}

\textbf{\implies \text{row} (vec-mat-Tensor v (w\#N)) i}

\textbf{= \text{times}}

\textbf{((v(i \text{ div row-length } (w\#N)))}

\textbf{(row (w\#N) (i \text{ mod row-length } (w\#N))))}

\textbf{proof –}

\textbf{assume \text{assms:} i < (length v)\star(row-length (w\#N)))}

\textbf{\wedge (mat nr (length (w\#N)) (w\#N))}

\textbf{show \text{thesis}}

\textbf{proof (cases N)}

\textbf{case Nil}

\textbf{have \text{row} (vec-mat-Tensor v (w\#N)) i} = [(vec-vec-Tensor v w)!i]

\textbf{using 1 vec-mat-Tensor.simps Nil row-empty by auto}

\textbf{then show \text{thesis}}

\textbf{proof (cases w)}

\textbf{case Nil}

\textbf{have (vec-vec-Tensor v w) = []}

\textbf{using Nil vec-vec-Tensor-right-empty by auto}

\textbf{moreover have (length v)\star(row-length (w\#N)) = 0}

\textbf{using Nil row-length-def by auto}

\textbf{then have [(vec-vec-Tensor v [])!i] = []}

\textbf{using assms less-nat-zero-code by metis}

\textbf{ultimately show \text{thesis}}

\textbf{using vec-vec-Tensor.simps row-empty Nil assms list.distinct(1) by (metis)}

\textbf{next}

\textbf{case (Cons a w1)}

\textbf{have \text{1:w} \neq []}

\textbf{using Cons by auto}

\textbf{then have i < (length v)\star(length w)}

\textbf{using assms row-length-def by auto}

\textbf{then have (vec-vec-Tensor v w)!i}
= f
  (v ![i die (length w)])
  (w ![i mod (length w)])

using vec-vec-Tensor-elements 1 all/ by auto
then have (row (vec-mat-Tensor v (w # N)) i)
  = times
  (v ![i die 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 die 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 (hide-lams, 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))
The following lemma gives us a formula for the row of a tensor of two matrices:

**Lemma (row-formula):**

**Fixes** $M_1$ and $M_2$

**Shows** $\forall i. ((i < (\text{row-length } M_1) \times (\text{row-length } M_2))$

\[ \land (\text{mat } (\text{row-length } M_1) (\text{length } M_1) M_1) \land (\text{mat } (\text{row-length } M_2) (\text{length } M_2) M_2) \]

\[ \rightarrow \text{row } (M_1 \otimes M_2) i = \text{vec-vec-Tensor} \]

\[ \begin{align*}
\quad &\text{row } (M_1 (i \div \text{row-length } M_2)) \\
\quad &\text{row } (M_2 (i \mod \text{row-length } M_2))
\end{align*} \]

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**

\[ ((i < (\text{row-length } (v \# M)) \times (\text{row-length } M_2)) \land (\text{mat } (\text{row-length } (v \# M)) (\text{length } (v \# M)) (v \# M)) \land (\text{mat } (\text{row-length } M_2) (\text{length } M_2) M_2) \]

\[ \rightarrow \text{row } ((v \# M) \otimes M_2) i = \text{vec-vec-Tensor} \]

\[ \begin{align*}
\quad &\text{row } (v \# M) (i \div \text{row-length } M_2) \\
\quad &\text{row } (M_2 (i \mod \text{row-length } M_2))
\end{align*} \]

**Proof—**

**Assume** assms:

\[(i < (\text{row-length } (v \# M)) \times (\text{row-length } M_2)) \land (\text{mat } (\text{row-length } (v \# M)) (\text{length } (v \# M)) (v \# M)) \land (\text{mat } (\text{row-length } M_2) (\text{length } M_2) M_2)\]

**Have 0:** $i < (\text{length } v) \times (\text{row-length } M_2)$

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

**Have 1:** mat (row-length M) (length M) M

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

**Have** row $((v \# M) \otimes M_2) i = \text{row } ((\text{vec-mat-Tensor } v M_2) \otimes (M \otimes M_2)) i$

**By** auto

**Then have 2:** $= \text{row } ((\text{vec-mat-Tensor } v M_2) i) \otimes (\text{row } (M \otimes M_2) i)$

**Using** row-append by auto

46
then show \(?thesis\)

proof (cases \(M\))

\begin{enumerate}
\item case Nil

\begin{itemize}
\item have \(\text{row } ((v \# M) \odot M^2) \ i = \ (\text{row } (\text{vec-mat-Tensor } v \ M^2) \ i)\)
\end{itemize}

\begin{itemize}
\item using Nil 2 by auto
\end{itemize}

\begin{itemize}
\item moreover have \(\text{row } (\text{vec-mat-Tensor } v \ M^2) \ i = \ \text{times}\)
\end{itemize}

\begin{itemize}
\item \(\text{v!}(i \div \text{row-length } M^2)\)
\item \(\text{row } M^2 \ (i \mod \text{row-length } M^2)\)
\end{itemize}

\begin{itemize}
\item using row-vec-mat-Tensor-prelim assms 0 by auto
\end{itemize}

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)

\item next

\begin{itemize}
\item case \((\text{Cons } w \ N)\)
\end{itemize}

\begin{itemize}
\item have Cons-Cons-1:mat \((\text{row-length } M) \ (\text{length } M) \ M)\)
\end{itemize}

\begin{itemize}
\item using assms reduct-matrix by auto
\end{itemize}

then have \(\text{row-length } (w \# N) = \text{row-length } (v \# M)\)

\begin{itemize}
\item using assms Cons unfolding mat-def Ball-def vec-def
\end{itemize}

\begin{itemize}
\item using append-Cons hd-in-set list.distinct(1)
\item rotate1.simps(2) set-rotate1
\end{itemize}

\begin{itemize}
\item by auto
\end{itemize}

then have Cons-Cons-2:i < \((\text{row-length } M) \ast (\text{row-length } M^2)\)

\begin{itemize}
\item using assms Cons by auto
\end{itemize}

then have Cons-Cons-3:\((\text{row } (M \odot M^2) \ i) = \text{vec-vec-Tensor}\)

\begin{itemize}
\item \((\text{row } M \ (i \div \text{row-length } M^2))\)
\item \((\text{row } M^2 \ (i \mod \text{row-length } M^2))\)
\end{itemize}

\begin{itemize}
\item using Cons.hyps Cons-Cons-1 assms by auto
\end{itemize}

moreover have \(\text{row } (\text{vec-mat-Tensor } v \ M^2) \ i = \ \text{times}\)

\begin{itemize}
\item \(\text{v!}(i \div \text{row-length } M^2)\)
\item \(\text{row } M^2 \ (i \mod \text{row-length } M^2)\)
\end{itemize}

\begin{itemize}
\item using row-vec-mat-Tensor-prelim assms 0 by auto
\end{itemize}

then have \((v \# M) \odot M^2) \ i = \ (\text{times}\)

\begin{itemize}
\item \(\text{v!}(i \div \text{row-length } M^2)\)
\item \(\text{row } M^2 \ (i \mod \text{row-length } M^2)\)
\end{itemize}

\begin{itemize}
\item using 2 Cons-Cons-3 by auto
\end{itemize}

moreover have ... = \((\text{vec-vec-Tensor}\)

\begin{itemize}
\item \((\text{v!}(i \div \text{row-length } M^2)\)
\item \(\text{#(row } M \ (i \div \text{row-length } M^2))\)
\item \(\text{row } M^2 \ (i \mod \text{row-length } M^2)\))
\end{itemize}

\begin{itemize}
\item using vec-vec-Tensor.simps(2) by auto
\end{itemize}

moreover have ... = \((\text{vec-vec-Tensor } (\text{row } (v \# M) \ (i \div \text{row-length } M^2))\)

\begin{itemize}
\item \(\text{row } M^2 \ (i \mod \text{row-length } M^2)\))
\end{itemize}

\begin{itemize}
\item using row-Cons by metis
\end{itemize}

ultimately show \(?thesis\) by metis
\end{enumerate}
qed

then show ?case by auto

qed

lemma effective-row-formula:
fixes M1 and M2
assumes \( i < (\text{row-length} M1) \times (\text{row-length} M2) \)
and \( (\text{mat} (\text{row-length} M1) (\text{length} M1) M1) \)
and \( (\text{mat} (\text{row-length} M2) (\text{length} M2) M2) \)
shows \( \text{row} (M1 \otimes M2) i = \text{vec-vec-Tensor} \)
\[ \begin{aligned}
& (\text{row} M1 (i \text{ div} \text{row-length} M2)) \\
& (\text{row} M2 (i \text{ mod} \text{row-length} M2))
\end{aligned} \]
using assms row-formula by auto

lemma alt-effective-matrix-tensor-elements:
\( ((i < ((\text{row-length} M2) \times (\text{row-length} M3)))) \)
\( (j < (\text{length} M2) \times (\text{length} M3))) \)
\( (\text{mat} (\text{row-length} M2) (\text{length} M2) M2) \)
\( (\text{mat} (\text{row-length} M3) (\text{length} M3) M3) \)
\( \Rightarrow ((M2 \otimes M3) ![ji]) = f (M2 ![j \text{ div} (\text{length} M3)])(i \text{ div} (\text{row-length} M3)) \)
\( (M3 ![j \text{ mod} \text{length} M3])(i \text{ mod} (\text{row-length} M3))) \)
using matrix-Tensor-elements by auto

lemma trans-impl:
\( (\forall i j. (P i j \rightarrow Q i j)) \land (\forall i j. (Q i j \rightarrow R i j)) \)
\( \Rightarrow (\forall i j. (P i j \rightarrow R i j)) \)
by auto

lemma \( ((x::\text{nat}) \text{ div} y) \text{ div} z = (x \text{ div} (y \ast z)) \)
using div-mult2-eq by auto

lemma \( (\neg((a::\text{nat}) < b)) \Rightarrow (a \geq b) \)
by auto

lemma not-null: \( xs \neq [] \Rightarrow \exists y \text{ ys}. \ xs = y \# \text{ys} \)
by (metis neq-Nil-conv)

lemma \( (y::\text{nat}) \neq 0 \Rightarrow (x \text{ mod} y) < y \)
using mod-less-divisor by auto

lemma mod-prop1:\( ((a::\text{nat}) \text{ mod} (b \ast c)) \text{ mod} c = (a \text{ mod} c) \)
proof (cases \( c = 0 \))
case True
have \( b \ast c = 0 \)
  by (metis True mult-0-right)
then have \( (a::\text{nat}) \text{ mod} (b \ast c) = a \)
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 \(?m. a = m*(b*c) + ?x\) 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 \(?z = m*c + ?z\) 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 \((a mod (b * c)) = a - m1 * (b * c)\) by (metis)
then have \(a = m1*b*c + m*c + ?z\)
using \((a = m1*(b * c) + a \mod (b * c); (a mod (b * c)) = m*c + a \mod (b * c) mod c)\) 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 \(?y = (a \mod c)\)
have \(?y = (a mod c)\)
by (metis 1 ⟨a mod (b * c) = m * c + a mod (b * c) mod c⟩ mod-mult-self3)
then obtain n where \(a = n*(c) + ?y\)
by auto
with I 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 \(c dvd (?y - ?z)\)
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 \textbf{thesis} using \textit{False}
  by (metis 1 mod-add-right-eq mod-mult-self2 add.commute mult.commute)
qed

\textbf{lemma \textit{mod-div-relation}}: \((a::\text{nat}) \mod (b\cdot c)) \div c = (a \div c) \mod b
\textbf{proof} (cases \(b\cdot c = 0\))
  case True
  have T-1: \((b = 0) \vee (c = 0)\)
   using True by auto
  show \textbf{thesis}
   proof (cases \((b = 0)\))
    case True
    have \(a \mod (b\cdot c) = a\)
     using True by auto
    then show \textbf{thesis} using True by auto
   next
    case False
    have \(c = 0\)
     using T-1 False by auto
    then show \textbf{thesis} by auto
   qed

  next
  case False
  have F-1: \((b > 0) \land (c > 0)\)
   using False by auto
  have \(\exists x. \ a = x \cdot (b\cdot c) + (a \mod (b\cdot c))\)
   using mod-div-decomp by blast
  then obtain \(x\) where \(a = x \cdot (b\cdot c) + (a \mod (b\cdot c))\)
   by auto
  then have \(a \div c = ((x \cdot (b\cdot c)) \div c) + ((a \mod (b\cdot c)) \div c)\)
   using div-add1-eq mod-add-self1 mod-add-self2
     mod-by-0 mod-div-trivial mod-prop1 mod-self
   by (metis)
  then have \(a \div c = (((x \cdot b) \cdot c) \div c) + ((a \mod (b\cdot c)) \div c)\)
   by auto
  then have F-2: \(a \div c = (x \cdot b) + ((a \mod (b\cdot c)) \div c)\)
   by (metis F-1 nonzero-mult-div-cancel-left mult.commute neq0-conv)
  have \(\exists y. \ a \div c = (y \cdot b) + ((a \div c) \mod b)\)
   by (metis add.commute mod-div-mult-eq)
  then obtain \(y\) where \(a \div c = (y \cdot b) + ((a \div c) \mod b)\)
   by auto
  with F-2 have F-3: \((x \cdot b) + ((a \mod (b\cdot c)) \div c) = (y \cdot b) + ((a \div c) \mod b)\)
   by auto
  then have \((x \cdot b) - (y \cdot b) = ((a \div c) \mod b) - ((a \mod (b\cdot c)) \div c)\)
   by auto
  then have \((x - y) \cdot b = ((a \div c) \mod b) - ((a \mod (b\cdot c)) \div c)\)
   by (metis diff-mult-distrib2 mult.commute)
  then have F-4: \(b \dvd (((a \div c) \mod b) - ((a \mod (b\cdot c)) \div c))\)
by (metis dvd-eq-mod-eq-0 mod-mult-self1-is-0 mult.commute)
have F-5: \( b > ((a \mod b) \mod c) \)
  by (metis F-1 mod-less-divisor)
have \( b \cdot c > (a \mod (b \cdot c)) \)
  by (metis False mod-less-divisor neq0-conv)
moreover then have \( (b \cdot c) \div c > (a \mod (b \cdot c)) \div c \)
  by (metis F-1 div-left-inq nonzero-mult-cancel-right neq0-conv)
then have \( b > (a \mod (b \cdot c)) \div c \)
  by (metis calculation div-right-inq mult.commute)
with F-4 F-5
have F-6: ((a \mod b) - ((a \mod (b \cdot c)) \div c)) = 0
  using less-imp-diff-less nat-dvd-not-less by blast
from F-3 have \( (y \cdot b) - (z \cdot b) = ((a \mod (b \cdot c)) \div c) - ((a \div c) \mod b) \)
  by auto
then have \( (y - z) \cdot b = ((a \mod (b \cdot c)) \div c) - ((a \div c) \mod b) \)
  by (metis diff-mult-distrib2 mult.commute)
then have F-7: \( b \cdot ((a \mod (b \cdot c)) \div c) - ((a \div c) \mod b) = 0 \)
  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)
moreover then have \( (b \cdot c) \div c > (a \mod (b \cdot c)) \div c \)
  by (metis calculation div-right-inq mult.commute)
with F-7 F-8
have \( ((a \mod (b \cdot c)) \div c) - ((a \div c) \mod b) = 0 \)
  by (metis F-2 cancel-comm-monoid-add-class.diff-cancel mod-if mod-mult-self3)
with F-6 have \( (a \mod (b \cdot c)) \div c = ((a \div c) \mod b) \)
  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**
- \( (\text{mat} \ (\text{row-length} \ M1) \ (\text{length} \ M1) \ M1) \)
- \( (\text{mat} \ (\text{row-length} \ M2) \ (\text{length} \ M2) \ M2) \)
- \( (\text{mat} \ (\text{row-length} \ M3) \ (\text{length} \ M3) \ M3) \)

\[ \implies M1 \otimes (M2 \otimes M3) = (M1 \otimes M2) \otimes M3 \]

**proof**
- fix \( j \)
**assume** \( 0: \ (\text{mat} \ (\text{row-length} \ M1) \ (\text{length} \ M1) \ M1) \)
- \( (\text{mat} \ (\text{row-length} \ M2) \ (\text{length} \ M2) \ M2) \)
- \( (\text{mat} \ (\text{row-length} \ M3) \ (\text{length} \ M3) \ M3) \)

51
have 1:\\ length \ ((M_1 \otimes M_2) \otimes M_3) \\\\
\quad = (\\ length M_1) \ast (\\ length M_2) \ast (\\ length M_3) \\
proof - \\
have \ length \ M_2 \otimes M_3 = (\\ length M_2) \ast (\\ length M_3) \\
\quad by (metis \ length-Tensor) \\
then have \ length \ (M_1 \otimes (M_2 \otimes M_3)) \\
\quad = (\\ length M_1) \ast (\\ length M_2) \ast (\\ length M_3) \\
\quad using \ mult.assoc \ length-Tensor \ by \ auto \\
moreover have \ length \ (M_1 \otimes M_2) = (\\ length M_1) \ast (\\ length M_2) \\
\quad by (metis \ length-Tensor) \\
ultimately show \ ?thesis \ using \ mult.assoc \ length-Tensor \ by \ auto \\
qed \\
have 2:row-length \ ((M_1 \otimes M_2) \otimes M_3) \\
\quad = (row-length M_1) \ast (row-length M_2) \ast (row-length M_3) \\
proof - \\
have \ row-length \ M_2 \otimes M_3 = (row-length M_2) \ast (row-length M_3) \\
\quad using \ row-length-mat \ assoc \ by \ auto \\
then have \ row-length \ (M_1 \otimes (M_2 \otimes M_3)) \\
\quad = (row-length M_1) \ast (row-length M_2) \ast (row-length M_3) \\
\quad using \ row-length-mat \ assoc \ by \ auto \\
moreover have \ row-length \ (M_1 \otimes M_2) \\
\quad = (row-length M_1) \ast (row-length M_2) \\
\quad using \ row-length-mat \ by \ auto \\
ultimately show \ ?thesis \ using \ row-length-mat \ assoc \ by \ auto \\
qed \\
have 3: \\
\forall i. \forall j. (((i < (row-length M_1) \ast (row-length M_2) \ast (row-length M_3))) \\
\quad \land (j < (\\ length M_1) \ast (\\ length M_2) \ast (\\ length M_3))) \\
\quad \rightarrow \\
\quad ( ((M_1 \otimes M_2) \otimes M_3) ![j][i]) \\
\quad = f \\
\quad ((M_1 \otimes M_2) ![j \ div \ (\\ length M_3)] ![i \ div \ (row-length M_3)]) \\
\quad (M_3 ![j \ mod \ length M_3] ![i \ mod \ (row-length M_3)])) \\
\quad using \ 0 \ matrix-Tensor-elements \ 1 \ 2 \ effective-well-defined-Tensor \ length-Tensor \ row-length-mat \\
\quad by \ auto \\
moreover have \\
\forall j. (j < (\\ length M_1) \ast (\\ length M_2) \ast (\\ length M_3)) \\
\quad \rightarrow \ (j \ div \ (\\ length M_3)) < (\\ length M_1) \ast (\\ length M_2) \\
apply \ (rule \ allI) \\
apply \ (simp \ add: div-left-ineq) \\
done \\
moreover have \ \forall i. (i < (row-length M_1) \ast (row-length M_2) \ast (row-length M_3)) \\
\quad \rightarrow \ (i \ div \ (row-length M_3)) \\
\quad < (row-length M_1) \ast (row-length M_2) \\
apply \ (rule \ allI) \\
apply \ (simp \ add: div-left-ineq) \\
done \\
ultimately have 4: \ \forall i. \forall j. ( ((i < (row-length M_1) \ast (row-length M_2) \ast (row-length M_3))) \\
\quad \land (j < (\\ length M_1) \ast (\\ length M_2) \ast (\\ length M_3))) \\
\quad \rightarrow \\
\quad ( ((M_1 \otimes M_2) \otimes M_3) ![j][i]) \\
\quad = f \\
\quad ((M_1 \otimes M_2) ![i \ div \ (row-length M_3)] ![j \ div \ (\\ length M_3)]) \\
\quad (M_3 ![i \ mod \ (row-length M_3)] ![j \ mod \ length M_3])) \\
\quad using \ 0 \ matrix-Tensor-elements \ 1 \ 2 \ effective-well-defined-Tensor \ length-Tensor \ row-length-mat
ultimately have

\[ \forall \ i. \forall \ j. ((i \div \text{row-length } M3) < (\text{row-length } M1) \ast (\text{row-length } M2)) \land (j \div \text{length } M3) < (\text{length } M1) \ast (\text{length } M2)) \]

using \[ \text{all } 0 \ by \ auto \]

have \[ (\text{mat } (\text{row-length } M1) \ (\text{length } M1) \ M1) \land (\text{mat } (\text{row-length } M2) \ (\text{length } M2) \ M2) \]

using \[ 0 \ by \ auto \]

then have \[ \forall i. \forall j. ((i \div \text{row-length } M3) < (\text{row-length } M1) \ast (\text{row-length } M2)) \land (j \div \text{length } M3) < (\text{length } M1) \ast (\text{length } M2)) \]

\[ \rightarrow \]

\[ (((M1 \otimes M2) \ast (j \div \text{length } M3))!\ (i \div \text{row-length } M3)) = f \]

\[ ((M1)!(j \div \text{length } M3) \div (\text{length } M2)) \land ((i \div \text{row-length } M3) \div (\text{row-length } M2))) \land ((j \div \text{length } M3) \mod (\text{length } M2)) \land ((i \div \text{row-length } M3) \mod (\text{row-length } M2))) \]

using \[ \text{effective-matrix-tensor-elements by auto} \]

with \[ \& \ j \ have \ 5: \forall i. \forall j. ((i < ((\text{row-length } M1) \ast (\text{row-length } M2) \ast (\text{row-length } M3))) \land (j < (\text{length } M1) \ast (\text{length } M2) \ast (\text{length } M3))) \]

\[ \rightarrow \]

\[ (((M1 \otimes M2) \otimes M3)!(i \div j \div i)) = f \]

\[ \{(M1)!(j \div \text{length } M3) \div (\text{length } M2)) \land ((i \div \text{row-length } M3) \div (\text{row-length } M2))) \land ((j \div \text{length } M3) \mod (\text{length } M2)) \land ((i \div \text{row-length } M3) \mod (\text{row-length } M2))) \]

by \[ \text{auto} \]

have \[ (j \div \text{length } M3) \div (\text{length } M2) = (j \div ((\text{length } M3) \ast (\text{length } M2))) \]

using \[ \text{div-mult2-eq by auto} \]

moreover have \[ ((i \div \text{row-length } M3) \div (\text{row-length } M2)) = (i \div ((\text{row-length } M3) \ast (\text{row-length } M2))) \]

using \[ \text{div-mult2-eq by auto} \]

ultimately have \[ \forall i. \forall j. ((i < ((\text{row-length } M1) \ast (\text{row-length } M2) \ast (\text{row-length } M3))) \land (j < (\text{length } M1) \ast (\text{length } M2) \ast (\text{length } M3))) \]

\[ = 53 \]
\[
(\langle (M_1 \otimes M_2) \otimes M_3 \rangle !_j)_i
= f
\]
\[
= f
\]
\[
(\langle (M_1) \otimes ((\text{length } M_2) \times (\text{length } M_3)) \rangle !_j)_i
\]
\[
= f
\]
\[
= (\langle (M_1) \otimes ((\text{length } M_2) \times (\text{length } M_3)) \rangle !_j)_i
\]
\[
= f
\]
\[
\text{using 6 by (metis 3 5 \text{div-mult2-eq})}
\]
\[
\text{then have step1: } \forall \ i, j. ((i < ((\text{row-length } M_1) \times (\text{row-length } M_2) \times (\text{row-length } M_3)))
\]
\[
\wedge (j < (\text{length } M_1) \times (\text{length } M_2) \times (\text{length } M_3)))
\]
\[
\rightarrow 
(\langle (M_1 \otimes (M_2 \otimes M_3)) \rangle !_j)_i
= f
\]
\[
(\langle (M_1) \otimes ((\text{length } M_2) \times (\text{length } M_3)) \rangle !_j)_i
(\langle (M_2 \otimes M_3) \rangle !_j)_{\text{mod length } (M_2 \otimes M_3)}
(\langle (M_3) \rangle !_j)_{\text{mod length } (M_2 \otimes M_3)}
\]
\[
\text{using 0 matrix-Tensor-elements 1 2 effective-well-defined-Tensor}
\]
\[
\text{length-Tensor row-length-mat}
\]
\[
\text{by auto}
\]
\[
\text{then have}
\]
\[
\forall \ i, j. ((i < ((\text{row-length } M_1) \times (\text{row-length } M_2) \times (\text{row-length } M_3)))
\]
\[
\wedge (j < (\text{length } M_1) \times (\text{length } M_2) \times (\text{length } M_3)))
\]
\[
\rightarrow 
(\langle (M_1 \otimes (M_2 \otimes M_3)) \rangle !_j)_i
= f
\]
\[
(\langle (M_1) \otimes ((\text{length } M_2) \times (\text{length } M_3)) \rangle !_j)_i
(\langle (M_2 \otimes M_3) \rangle !_j)_{\text{mod length } (M_2 \otimes M_3)}
(\langle (M_3) \rangle !_j)_{\text{mod length } (M_2 \otimes M_3)}
\]
\[
\text{using length-Tensor row-length-mat by auto}
\]
\[
\text{then have}
\]
\[
\forall \ i, j. ((i < ((\text{row-length } M_1) \times (\text{row-length } M_2) \times (\text{row-length } M_3)))
\]
\[
\wedge (j < (\text{length } M_1) \times (\text{length } M_2) \times (\text{length } M_3)))
\]
\[
\rightarrow ((M1 \otimes (M2 \otimes M3))!j!i) = f \((M1)!((j \text{ div } ((\text{length } M3) \ast (\text{length } M2)))

!((i \text{ div } ((\text{row-length } M3) \ast (\text{row-length } M2))))

((M2 \otimes M3)!((j \mod \text{length } (M2 \otimes M3))\n
!((i \mod (\text{row-length } (M2 \otimes M3))))))
\]

using mult.commute by (metis)

have 8: \(\forall j.((j < (\text{length } M1) \ast (\text{length } M2) \ast (\text{length } M3)))
\rightarrow ((j \mod \text{length } M2 \otimes M3))) < (\text{length } M2 \otimes M3))

proof (cases length (M2 \otimes M3) = 0)

  case True

  have \((\text{length } M2) \ast (\text{length } M3) = 0\)

  using length-Tensor True by auto

  then have \((\text{length } M1) \ast (\text{length } M2) \ast (\text{length } M3) = 0\)

  by auto

  then show ?thesis by (metis less-nat-zero-code)

next

  case False

  have \((\text{length } M2 \otimes M3) > 0\)

  using False by auto

  then show ?thesis using mod-less-divisor by auto

qed

then have 9: \(\forall i.((i < ((\text{row-length } M1) \ast (\text{row-length } M2) \ast (\text{row-length } M3)))
\rightarrow ((i \mod (\text{row-length } M2 \otimes M3))) < (\text{row-length } (M2 \otimes M3)))

proof (cases row-length (M2 \otimes M3) = 0)

  case True

  have \((\text{row-length } M2) \ast (\text{row-length } M3) = 0\)

  using True by (metis row-length-mat)

  then have \((\text{row-length } M1) \ast (\text{row-length } M2) \ast (\text{row-length } M3) = 0\)

  by auto

  then show ?thesis by (metis less-nat-zero-code)

next

  case False

  have \((\text{row-length } M2 \otimes M3) > 0\)

  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) \ast (\text{row-length } M2) \ast (\text{row-length } M3)))
\wedge (j < (\text{length } M1) \ast (\text{length } M2) \ast (\text{length } M3)))
\rightarrow ((i \mod (\text{row-length } (M2 \otimes M3))) < (\text{row-length } (M2 \otimes M3))
\wedge (j \mod (\text{length } (M2 \otimes M3))) < (\text{length } (M2 \otimes M3)))

by auto

then have 11: \(\forall i.\forall j.((i < ((\text{row-length } M1) \ast (\text{row-length } M2) \ast (\text{row-length } M3)))
\wedge (j < (\text{length } M1) \ast (\text{length } M2) \ast (\text{length } M3)))
\rightarrow \)

55
(i mod (row-length (M2 ⊗ M3)))
< (row-length M2)* (row-length M3)
∧ (j mod (length (M2 ⊗ M3))) < (length M2)* (length M3)

using length-Tensor row-length-mat by auto

have (mat (row-length M2) (length M2) M2)
∧ (mat (row-length M3) (length M3) M3)
using 0 by auto

then have ∀ i j. (((i mod (row-length (M2 ⊗ M3)))
< (row-length M2)* (row-length M3))
∧ (j mod (length (M2 ⊗ M3))) < (length M2)* (length M3))

→ (((M2 ⊗ M3)!(j mod (length (M2 ⊗ M3)))!(i mod row-length (M2 ⊗ M3)))

= f
((M2)!(j mod (length (M2 ⊗ M3))) div (length M3))
!(i mod (row-length (M2 ⊗ M3))) div (row-length M3))
(M3!(j mod (length (M2 ⊗ M3))) mod (length M3))
!(i mod (row-length (M2 ⊗ M3))) mod (row-length M3)))

using matrix-Tensor-elements by auto

then have ∀ i j.
((i < (row-length M1)* (row-length M2)* (row-length M3))
∧ (j < (length M1)* (length M2)* (length M3))

→ (((M2 ⊗ M3)!(j mod (length (M2 ⊗ M3)))!(i mod row-length (M2 ⊗ M3)))

= f
((M2)!(j mod (length (M2 ⊗ M3))) div (length M3))
!(i mod (row-length (M2 ⊗ M3))) div (row-length M3))
(M3!(j mod (length (M2 ⊗ M3))) mod (length M3))
!(i mod (row-length (M2 ⊗ M3))) mod (row-length M3)))

using 11 by auto

moreover then have ∀ j. (j mod (length (M2 ⊗ M3))) mod (length M3)
= j mod (length M3)

proof
have ∀ j. (j mod (length (M2 ⊗ M3)))
= (j mod ((length M2)* (length M3)))

using length-Tensor by auto

moreover have
∀ j. (j mod ((length M2)* (length M3))) mod (length M3)
= (j mod (length M3))

using mod-prop1 by auto

ultimately show ?thesis by auto

qed

moreover then have ∀ i. (i mod (row-length (M2 ⊗ M3))) mod (row-length M3)
= i mod (row-length M3)

proof
have ∀ i. (i mod (row-length (M2 ⊗ M3)))
= (i mod ((row-length M2)* (row-length M3))))
using row-length-mat by auto
moreover have \( \forall i.((i \mod ((\text{row-length } M2)\ast(\text{row-length } M3))) \mod (\text{row-length } M3)) = (i \mod (\text{row-length } M3)) \)
using mod-prop1 by auto
ultimately show ?thesis by auto
qed
ultimately have 12: \( \forall i j.((i < (\text{row-length } M1) \ast(\text{row-length } M2) \ast(\text{row-length } M3))) \)
\( \wedge(j < (\text{length } M1)\ast(\text{length } M2)\ast(\text{length } M3)) \)
\( \rightarrow (((M2 \otimes M3))!(j \mod (\text{length } (M2 \otimes M3)))\notag
\quad !(i \mod \text{row-length } (M2 \otimes M3)))\notag
\quad = f \notag
\quad ((M2)!(j \mod (\text{length } (M2 \otimes M3))) \div (\text{length } M3))\notag
\quad !(i \mod (\text{row-length } (M2 \otimes M3))))\notag
\) by auto
moreover have \( \forall j.((j \mod (\text{length } (M2 \otimes M3))) \div (\text{length } M3))\notag
\quad = (j \mod (((\text{length } M2)\ast(\text{length } M3))))\notag
\) using length-Tensor by auto
then show ?thesis using mod-div-relation by auto
qed
moreover have \( \forall i.((i \mod (\text{row-length } (M2 \otimes M3))) \div (\text{row-length } M3))\notag
\quad = (i \mod ((\text{row-length } M2)\ast(\text{row-length } M3))))\notag
\) using row-length-mat by auto
then show ?thesis using mod-div-relation by auto
qed
ultimately have \( \forall i j.((i < (\text{row-length } M1)\ast(\text{row-length } M2)\ast(\text{row-length } M3)))\notag
\quad \wedge(j < (\text{length } M1)\ast(\text{length } M2)\ast(\text{length } M3)) \)
\( \rightarrow (((M2 \otimes M3))!(j \mod (\text{length } (M2 \otimes M3)))\notag
\quad !(i \mod \text{row-length } (M2 \otimes M3)))\notag
\quad = f \notag
\quad ((M2)!(j \div (\text{length } M3)) \mod (\text{length } M2))\notag
\quad !(i \div (\text{row-length } M3)))\notag
\quad !(i \mod (\text{row-length } (M2 \otimes M3))))\notag
\) by auto
with 7 have 13: \( \forall i j.(((i < ((\text{row-length } M1)\ast(\text{row-length } M2)\ast(\text{row-length } M3)))\notag
\quad \wedge(j < (\text{length } M1)\ast(\text{length } M2)\ast(\text{length } M3))) \)
\( \rightarrow \notag\)
\[
((M_1 \otimes (M_2 \otimes M_3))!j!i) = f
((M_1)!((j \div ((\text{length } M_2) \times (\text{length } M_3))))
\!!(i \div ((\text{row-length } M_2) \times (\text{row-length } M_3))))
\]

\[
(f
((M_2)!((j \div (\text{length } M_3)) \mod (\text{length } M_2))!((i \div (\text{row-length } M_3)) \mod (\text{row-length } M_2)))
(M_3)!((j \mod (\text{length } M_3))
\!!(i \mod (\text{row-length } M_3))))
\]

using \text{length-Tensor row-length-mat} by auto

moreover have \forall i j. (f
((M_1)!((j \div ((\text{length } M_2) \times (\text{length } M_3))))
\!!(i \div ((\text{row-length } M_2) \times (\text{row-length } M_3))))
\]

\[
(f
(((M_2)!((j \div (\text{length } M_3)) \mod (\text{length } M_2))!((i \div (\text{row-length } M_3)) \mod (\text{row-length } M_2)))
(M_3)!((j \mod (\text{length } M_3))
\!!(i \mod (\text{row-length } M_3))))
\]

by auto

\begin{proof}

\text{with 13 have} \forall i.j.((i < ((\text{row-length } M_1) \times (\text{row-length } M_2) \times (\text{row-length } M_3)))
\land (j < (\text{length } M_1) \times (\text{length } M_2) \times (\text{length } M_3))
\rightarrow
((M_1 \otimes (M_2 \otimes M_3))!j!i)

f (f
((M_1)!((j \div ((\text{length } M_2) \times (\text{length } M_3))))
\!!(i \div ((\text{row-length } M_2) \times (\text{row-length } M_3))))
\]

\[
((M_2)!((j \div (\text{length } M_3)) \mod (\text{length } M_2))!((i \div (\text{row-length } M_3)) \mod (\text{row-length } M_2)))
(M_3)!((j \mod (\text{length } M_3))
\!!(i \mod (\text{row-length } M_3))))
\]

by auto

\text{with step1 have step2:}
\forall i.j.((i < ((\text{row-length } M_1) \times (\text{row-length } M_2) \times (\text{row-length } M_3)))
\land (j < (\text{length } M_1) \times (\text{length } M_2) \times (\text{length } M_3))
\rightarrow
((M_1 \otimes (M_2 \otimes M_3))!j!i) = (((M_1 \otimes M_2) \otimes M_3)!j!i))

by auto

\text{moreover have} \text{mat} (((\text{row-length } M_1) \times (\text{row-length } M_2) \times (\text{row-length } M_3))
((\text{length } M_1) \times (\text{length } M_2) \times (\text{length } M_3))
(M_1 \otimes (M_2 \otimes M_3))

\end{proof}
\(M3\)

using 0 effective-well-defined-Tensor row-length-mat length-Tensor
by auto

moreover have mat ((row-length M1)*((row-length (M2 \(\otimes\) M3))))
((length M1)*((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 \(\otimes\) M2)

using 0 effective-well-defined-Tensor row-length-mat length-Tensor by auto

moreover have mat ((row-length (M1 \(\otimes\) M2))*(row-length M3))
((length (M1 \(\otimes\) M2))*(length M3))
(M1 \(\otimes\) M2)

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 using mat-eqI by blast

qed

end

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

1.2 Associativity and Distributive properties

locale plus-mult =
    mult +
    fixes zer::'a
    fixes g:: 'a \Rightarrow 'a \Rightarrow 'a (infixl + 60)
    fixes inver::'a \Rightarrow 'a
    assumes plus-comm: g a b = g b a
    assumes plus-associ: (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

59
context plus-mult

begin

lemma fixes M1 M2 M3
  shows \((\text{mat} \ (\text{row-length} \ M1) \ (\text{length} \ M1) \ M1) \\\n\land (\text{mat} \ (\text{row-length} \ M2) \ (\text{length} \ M2) \ M2) \\\n\land (\text{mat} \ (\text{row-length} \ M3) \ (\text{length} \ M3) \ M3) \\\n\implies (M1 \otimes (M2 \otimes M3)) = ((M1 \otimes M2) \otimes M3)\)
  using associativity by auto

matrix_mult refers to multiplication of matrices in the locale plus_mult

abbreviation matrix-mult:: 'a mat ⇒ 'a mat ⇒ 'a mat
  where
  matrix-mult M1 M2 ≡ (mat-multI zer g f (row-length M1) M1 M2)

definition scalar-product :: 'a vec ⇒ 'a vec ⇒ '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 ≠ []
  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 ∈ set (v#M)
      using Cons by auto

60
then have \( \text{length } v = nr1 \)
using \text{assms(1)} mat-def Ball-def vec-def \( \text{Cons by metis} \)
moreover then have \( \text{length } v = nr2 \)
using 1 \text{assms(2)} mat-def Ball-def vec-def \( \text{Cons by metis} \)
ultimately show \( nr1 = nr2 \)
by auto

next
have \( \text{length } M = nc1 \)
using mat-def \text{assms(1)} by auto
moreover have \( \text{length } M = nc2 \)
using mat-def \text{assms(2)} by auto
ultimately show \( nc1 = nc2 \)
by auto

qed

\textbf{lemma} matrix-mult-index:
\begin{itemize}
\item \textbf{assumes} \( m1 \neq [] \)
\item and \( \text{wf1}: \text{mat } nr \ n \ m1 \)
\item and \( \text{wf2}: \text{mat } n \ nc \ m2 \)
\item and \( i: i < nr \)
\item and \( j: j < nc \)
\item \text{shows} matrix-mult \( m1 \ m2 \ ! \ j \ ! \ i = \text{scalar-product \ (row } m1 \ i \) \ (\text{col } m2 \ j) \)
\item using matrix-index unique-row-col \text{assms by (metis matrix-row-length)}
\end{itemize}

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

\textbf{definition} matrix-match::\( \forall a \text{ mat } \Rightarrow \forall a \text{ mat } \Rightarrow \forall a \text{ mat } \Rightarrow \forall a \text{ mat } \Rightarrow \text{bool} \)
where
\begin{align*}
\text{matrix-match } A1 \ A2 \ B1 \ B2 & \equiv \\
(\text{mat (row-length } A1) \ (\text{length } A1) \ A1) \\
\land (\text{mat (row-length } A2) \ (\text{length } A2) \ A2) \\
\land (\text{mat (row-length } B1) \ (\text{length } B1) \ B1) \\
\land (\text{mat (row-length } B2) \ (\text{length } B2) \ B2) \\
\land (\text{length } A1 = \text{row-length } A2) \\
\land (\text{length } B1 = \text{row-length } B2) \\
\land (A1 \neq []) \land (A2 \neq []) \land (B1 \neq []) \land (B2 \neq [])
\end{align*}

\textbf{lemma} non-empty-mat-mult:
\begin{itemize}
\item \textbf{assumes} \( \text{wf1}: \text{mat } nr \ n \ A \)
\item and \( \text{wf2}: \text{mat } n \ nc \ B \)
\item and \( A \neq [] \) and \( B \neq [] \)
\item \text{shows} \( A \circ B \neq [] \)
\item \textbf{proof}–
\item have \( \text{mat } nr \ nc \ (A \circ B) \)
\item using \text{assms(1)} \text{assms(2)} mat-mult \text{assms(3)} matrix-row-length unique-row-col(1)
\item by (metis)
\item then have \( \text{length } (A \circ B) = nc \)
\end{itemize}
using \texttt{mat-def} by \texttt{auto}
moreover have \( nc > 0 \)
proof –
  have \( \text{length} \, B = nc \)
    using \texttt{assms}(2) \texttt{mat-def} by \texttt{auto}
  then show \( \text{?thesis} \) using \texttt{assms}(4) by \texttt{auto}
qed
moreover then have \( \text{length} \, (A \circ B) > 0 \)
  by (metis calculation(1))
then show \( \text{?thesis} \) by \texttt{auto}
qed

lemma \texttt{tensor-compose-distribution1}:
assumes \( \text{wf1:mat} \, (\text{row-length} \, A1) \, (\text{length} \, A1) \, A1 \)
  and \( \text{wf2:mat} \, (\text{row-length} \, A2) \, (\text{length} \, A2) \, A2 \)
  and \( \text{wf3:mat} \, (\text{row-length} \, B1) \, (\text{length} \, B1) \, B1 \)
  and \( \text{wf4:mat} \, (\text{row-length} \, B2) \, (\text{length} \, B2) \, B2 \)
  and \( \text{matchAA:}\, \text{length} \, A1 = \text{row-length} \, A2 \)
  and \( \text{matchBB:}\, \text{length} \, B1 = \text{row-length} \, B2 \)
  and \( \text{non-Nil}: (A1 \neq []) \land (A2 \neq []) \land (B1 \neq []) \land (B2 \neq []) \)
shows \( \text{mat} \, ((\text{row-length} \, A1) \ast (\text{row-length} \, B1)) \)
  \( (\text{length} \, A2) \ast (\text{length} \, B2)) \)
  \( (A1 \circ A2) \otimes (B1 \circ B2)) \)
proof –
  have \( 0: \text{mat} \, (\text{row-length} \, A1) \, (\text{length} \, A2) \, (\text{matrix-mult} \, A1 \, A2) \)
    using \texttt{wf1} \texttt{wf2} \texttt{mat-mult} \texttt{matchAA} by \texttt{auto}
  then have \( 1: \text{mat} \, (\text{row-length} \, (A1 \circ A2)) \, (\text{length} \, (A1 \circ A2)) \, (\text{matrix-mult} \, A1 \, A2) \)
    by (metis \text{matrix-row-length})
  then have \( 2: \, (\text{row-length} \, (A1 \circ A2)) = (\text{row-length} \, A1) \, \text{and} \, \text{length} \, (A1 \circ A2) = \text{length} \, A2 \)
    using \texttt{non-empty-mat-mult unique-row-col 0}
    apply (metis \text{length-0-conv} \text{mat-empty-column-length} \text{non-Nil})
    by (metis 0 1 \text{mat-empty-column-length} unique-row-col(2))
moreover have \( 3: \text{mat} \, (\text{row-length} \, B1) \, (\text{length} \, B2) \, (\text{matrix-mult} \, B1 \, B2) \)
  using \texttt{wf3} \texttt{wf4} \texttt{matchBB} \texttt{mat-mult} by \texttt{auto}
then have \( 4: \text{mat} \, (\text{row-length} \, (B1 \circ B2)) \, (\text{length} \, (B1 \circ B2)) \, (\text{matrix-mult} \, B1 \, B2) \)
  by (metis \text{matrix-row-length})
then have \( 5: \, (\text{row-length} \, (B1 \circ B2)) = (\text{row-length} \, B1) \, \text{and} \, \text{length} \, (B1 \circ B2) = \text{length} \, B2 \)
  using \texttt{non-empty-mat-mult unique-row-col 3}
  apply (metis \text{length-0-conv} \text{mat-empty-column-length} \text{non-Nil})
  by (metis 3 4 \text{mat-empty-column-length} unique-row-col(2))
then show \( \text{?thesis} \) using \( 1 \, 4 \, 5 \) \texttt{well-defined-Tensor}
  by (metis 2 calculation(2))
qed

lemma \texttt{effective-tensor-compose-distribution1}:
matrix-match \( A_1 \ A_2 \ B_1 \ B_2 \rightarrow \text{mat} \ ((\text{row-length } A_1)\ast(\text{row-length } B_1)) \)
\( ((\text{length } A_2)\ast(\text{length } B_2)) \)
\( ((A_1 \circ A_2)\otimes(B_1 \circ B_2)) \)
using \( \text{tensor-compose-distribution1} \) unfolding \( \text{matrix-match-def} \) by auto

**Lemma tensor-compose-distribution2:**

**Assumes**
- \( wf_1 : \text{mat} \ (\text{row-length } A_1) \ (\text{length } A_1) \ A_1 \)
- \( wf_2 : \text{mat} \ (\text{row-length } A_2) \ (\text{length } A_2) \ A_2 \)
- \( wf_3 : \text{mat} \ (\text{row-length } B_1) \ (\text{length } B_1) \ B_1 \)
- \( wf_4 : \text{mat} \ (\text{row-length } B_2) \ (\text{length } B_2) \ B_2 \)
- \( \text{matchAA} : \text{length } A_1 = \text{row-length } A_2 \)
- \( \text{matchBB} : \text{length } B_1 = \text{row-length } B_2 \)
- \( \text{non-Nil} : (A_1 \neq []) \land (A_2 \neq []) \land (B_1 \neq []) \land (B_2 \neq []) \)

**Shows**
- \( \text{mat} \ ((\text{row-length } A_1)\ast(\text{row-length } B_1)) \)
- \( ((\text{length } A_2)\ast(\text{length } B_2)) \)
- \( ((A_1 \otimes B_1) \circ (A_2 \otimes B_2)) \)

**Proof**
- have \( \text{mat} \)
  - \( ((\text{row-length } A_1)\ast(\text{row-length } B_1)) \)
  - \( ((\text{length } A_1)\ast(\text{length } B_1)) \)
  - \( (A_1 \otimes B_1) \)
  - using \( \text{wf}_1 \ \text{wf}_3 \) well-defined-Tensor by auto
- moreover have \( \text{mat} \)
  - \( ((\text{row-length } A_2)\ast(\text{row-length } B_2)) \)
  - \( ((\text{length } A_2)\ast(\text{length } B_2)) \)
  - \( (A_2 \otimes B_2) \)
  - using \( \text{wf}_2 \ \text{wf}_4 \) well-defined-Tensor by auto
- moreover have \( ((\text{length } A_1)\ast(\text{length } B_1)) \)
  - \( = ((\text{row-length } A_2)\ast(\text{row-length } B_2)) \)
  - using \( \text{matchAA} \ \text{matchBB} \) by auto
- ultimately show \( \text{thesis} \) using \( \text{mat-mult} \ \text{row-length-mat} \) by simp
- qed

**Theorem tensor-non-empty:**

**Assumes**
- \( A \neq [] \) and \( B \neq [] \)

**Shows**
- \( A \otimes B \neq [] \)

**Using**
- \( \text{assms}(1) \) \( \text{assms}(2) \) length-0-cone length-Tensor mult-is-0 by metis

**Theorem non-empty-distribution:**

**Assumes**
- \( \text{mat} \ n_1 \ n_1 A_1 \)
- \( \text{mat} \ n_2 \ n_1 A_2 \)
- \( \text{mat} \ n_2 \ n_2 B_1 \)
- \( \text{mat} \ n_2 \ n_2 B_2 \)
- \( A_1 \neq [] \) and \( B_1 \neq [] \) and \( A_2 \neq [] \) and \( B_2 \neq [] \)

**Shows**
- \( ((A_1 \circ A_2)\otimes(B_1 \circ B_2)) \neq [] \)

**Proof**
- have \( A_1 \circ A_2 \neq [] \)
  - using \( \text{assms} \) non-empty-mat-mult by auto
- moreover have \( B_1 \circ B_2 \neq [] \)
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 \(A1 A2 B1 B2\) \(\Rightarrow\) 
\(\mat((\text{row-length } A1)\ast(\text{row-length } B1))\)
\((\text{length } A2)\ast(\text{length } B2))\)
\((A1 \otimes B1) \circ (A2 \otimes B2))\)
using tensor-compose-distribution2 unfolding matrix-match-def by auto

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

theorem effective-matrix-Tensor-elements2:
fixes \(M1 M2\)
assumes \(\mat(\text{row-length } M1) (\text{length } M1) M1\)
and \(\mat(\text{row-length } M2) (\text{length } M2) M2\)
shows
\((\forall i < ((\text{row-length } M1)\ast(\text{row-length } M2)).\)
\((\forall j < ((\text{length } M1)\ast(\text{length } M2)).\)
\((M1 \otimes M2)!i!j) = f (M1!(j \text{ div } (\text{length } M2))!(i \text{ div } (\text{row-length } M2)))\)
\((M2!(j \text{ mod } (\text{length } M2))!(i \text{ mod } (\text{row-length } M2)))\))
using matrix-Tensor-elements assms by auto

definition matrix-compose-cond::\text{"a mat} \Rightarrow \text{"a mat} \Rightarrow \text{"a mat} \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{bool}
where
matrix-compose-cond \(A1 A2 B1 B2 i j\) \(\equiv\)
\((\mat(\text{row-length } A1) (\text{length } A1) A1\)
\&(\mat(\text{row-length } A2) (\text{length } A2) A2\)
\&(\mat(\text{row-length } B1) (\text{length } B1) B1\)
\&(\mat(\text{row-length } B2) (\text{length } B2) B2\)
\&(\text{length } A1 = \text{row-length } A2\)
\&(\text{length } B1 = \text{row-length } B2\)
\&(A1 \neq \text{[]})\&(A2 \neq \text{[]})\&(B1 \neq \text{[]})\&(B2 \neq \text{[]})\)
\&(i < (\text{row-length } A1)\ast(\text{row-length } B1))\&(j < (\text{length } A2)\ast(\text{length } B2))\)

theorem elements-matrix-distribution-1:
assumes \(wf1: \text{mat} \ (\text{row-length } A1) \ (\text{length } A1) \ A1\)
and \(wf2: \text{mat} \ (\text{row-length } A2) \ (\text{length } A2) \ A2\)
and \(wf3: \text{mat} \ (\text{row-length } B1) \ (\text{length } B1) \ B1\)
and \(wf4: \text{mat} \ (\text{row-length } B2) \ (\text{length } B2) \ B2\)
and \(\text{matchAA} : \text{length } A1 = \text{row-length } A2\)
and \(\text{matchBB} : \text{length } B1 = \text{row-length } B2\)
and \(\text{non-Nil}: (A1 \neq [] \land (A2 \neq []) \land (B1 \neq []) \land (B2 \neq []))\)
and \(i < (\text{row-length } A1) \ast (\text{row-length } B1)\) and \(j < (\text{length } A2) \ast (\text{length } B2)\)

shows

\[(\text{matrix-mul} \ A1 \ A2) \times (\text{matrix-mul} \ B1 \ B2)) = \ ! \ i \ j \ i\]

\(\text{proof –}\)

have \(0: (\text{matrix-mul} \ A1 \ A2) \times (\text{matrix-mul} \ B1 \ B2)) \neq []\)
using \(\text{non-empty-distribution assms by auto}\)

then have \(1: \text{mat} \ ((\text{row-length } A1) \ast (\text{row-length } B1)) \)
\(\ (\text{length } A2) \ast (\text{length } B2)\)
\(\ (A1 \circ A2) \times (B1 \circ B2))\)
using \(\text{tensor-compose-distribution1 assms by auto}\)

then have \(2: \text{mat} \ (\text{row-length} \ ((A1 \circ A2) \times (B1 \circ B2)))\)
\(\ (\text{length} \ ((A1 \circ A2) \times (B1 \circ B2)))\)
\(\ (A1 \circ A2) \times (B1 \circ B2))\)
by \(\text{metis matrix-row-length}\)

then have \(3: ((\text{row-length } A1) \ast (\text{row-length } B1)) \)
\(\ = (\text{row-length} \ ((A1 \circ A2) \times (B1 \circ B2)))\)
and \((\text{length } A2) \ast (\text{length } B2) = (\text{length} \ ((A1 \circ A2) \times (B1 \circ B2)))\)
using \(0 \ 1 \ 2 \ unique-row-col\) by \(\text{metis}\)

apply \(\text{metis}\)
using \(0 \ 1 \ 2 \ unique-row-col\) by \(\text{metis}\)

then have \(i: (i < ((\text{row-length } A1) \ast (\text{row-length } B1)))\)
\(\ = (i < (\text{row-length} \ ((A1 \circ A2) \times (B1 \circ B2))))\)
by \(\text{auto}\)

moreover have \(j: (j < ((\text{length } A2) \ast (\text{length } B2)))\)
\(\ = (j < (\text{length} \ ((A1 \circ A2) \times (B1 \circ B2))))\)
using \(3 \ (\text{length } A2 \ast \text{length } B2 = \text{length} \ ((A1 \circ A2) \times (B1 \circ B2)))\)
by \(\text{metis}\)

have \(4: \text{mat} \ (\text{row-length } A1) \ (\text{length } A2) \ (A1 \circ A2)\)
using \(\text{assms mat-mul} \text{ by auto}\)

then have \(5: \text{mat} \ (\text{row-length} \ ((A1 \circ A2)) \ (\text{length } (A1 \circ A2)) \ (A1 \circ A2)\)
using \(\text{matrix-row-length} \ text{ by } \text{metis}\)

with \(4\) have \(6: \text{row-length } A1 = \text{row-length } (A1 \circ A2)\)
by \(\text{metis } 0 \ Tensor.simps(1) \ unique-row-col(1))\)

with \(4 \ 5\) have \(7: \text{length } A2 = \text{length } (A1 \circ A2)\)
by \(\text{metis } \text{mat-empty-column-length unique-row-col}(2)\)
then have \(8: \text{mat} \ (\text{row-length } B1) \ (\text{length } A2) \ (B1 \circ B2)\)
using \(\text{assms mat-mul} \text{ by auto}\)

then have \(9: \text{mat} \ (\text{row-length} \ ((B1 \circ B2)) \ (\text{length } (B1 \circ B2)) \ (B1 \circ B2)\)

65
using \texttt{matrix-row-length by (metis)}

with 7 8 have 10: 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: 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:\( ((A1 \circ A2) \circ (B1 \circ B2)) \!\! j \!\! i \)
  \( = f (((A1 \circ A2)) ! (j \text{ div } (\text{length } (B1 \circ B2))) ! (i \text{ div } (\text{row-length } (B1 \circ B2)))) ((B1 \circ B2)) !(j \text{ mod } \text{length } (B1 \circ B2)) (! (i \text{ mod } (\text{row-length } (B1 \circ B2)))) \)
  using 5 9 13 15 effective-matrix-Tensor-elements by auto

then have \( ((A1 \circ A2) \circ (B1 \circ B2)) \!\! j \!\! i \)
  \( = f (((A1 \circ A2)) ! (j \text{ div } (\text{length } B2)) ! (i \text{ div } (\text{row-length } B1)) ((B1 \circ B2)) !(j \text{ mod } \text{length } B2) !(i \text{ mod } (\text{row-length } B1))) \)
  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 \texttt{thesis using wf1 non-Nil matrix-mult-index by blast}

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

proof
  have \( j \text{ mod } (\text{length } B2) \)
    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)

qed
moreover have \( i \mod (\text{row-length } B1) < (\text{row-length } B1) \)
by \( \text{metis assms(8) less-nat-zero-code mod-less-divisor mult-is-0} \)
moreover have \( \text{mat (length } B1) (\text{length } B2) B2 \)
using \( \text{wf4 matchBB} \) by auto
ultimately show \( \text{thesis} \)
using \( \text{wf3 non-Nil matrix-mult-index} \) by blast
qed
ultimately show \( \text{thesis} \) by auto
qed

\begin{align*}
\text{lemma } \text{effective-elements-matrix-distribution1:} \\
\text{matrix-compose-cond } A1 A2 B1 B2 i j \Rightarrow \\
(((\text{matrix-mult } A1 A2) \circ (\text{matrix-mult } B1 B2))!/j!i \\
= f \left( \text{scalar-product} \left( \text{row } A1 \left( i \div (\text{row-length } B1) \right) \right) \left( \text{col } A2 \left( j \div (\text{length } B2) \right) \right) \right) \\
\left( \text{scalar-product} \left( \text{row } B1 \left( i \mod (\text{row-length } B1) \right) \right) \left( \text{col } B2 \left( j \mod (\text{length } B2) \right) \right) \right) \\
\text{using } \text{elements-matrix-distribution-1 matrix-compose-cond-def by auto} \\
\end{align*}

\begin{align*}
\text{lemma } \text{matrix-match-condn-1:} \\
\text{matrix-match } A1 A2 B1 B2 \\
\land ((i < (\text{row-length } A1) \ast (\text{row-length } B1)) \\
\land (j < (\text{length } A2) \ast (\text{length } B2))) \\
\Rightarrow ((\text{matrix-mult } A1 A2) \circ (\text{matrix-mult } B1 B2))!/j!i \\
= f \left( \text{scalar-product} \left( \text{row } A1 \left( i \div (\text{row-length } B1) \right) \right) \left( \text{col } A2 \left( j \div (\text{length } B2) \right) \right) \right) \\
\left( \text{scalar-product} \left( \text{row } B1 \left( i \mod (\text{row-length } B1) \right) \right) \left( \text{col } B2 \left( j \mod (\text{length } B2) \right) \right) \right) \\
\text{using } \text{elements-matrix-distribution-1 unfolding matrix-match-def by auto} \\
\end{align*}

\begin{align*}
\text{lemma } \text{effective-matrix-match-condn-1:} \\
\text{assumes } (\text{matrix-match } A1 A2 B1 B2) \\
\text{shows } \forall i j. ((i < (\text{row-length } A1) \ast (\text{row-length } B1)) \\
\land (j < (\text{length } A2) \ast (\text{length } B2))) \\
\Rightarrow ((A1 \circ A2) \circ (B1 \circ B2))/j!i \\
= f \left( \text{scalar-product} \left( \text{row } A1 \left( i \div (\text{row-length } B1) \right) \right) \left( \text{col } A2 \left( j \div (\text{length } B2) \right) \right) \right) \\
\left( \text{scalar-product} \left( \text{row } B1 \left( i \mod (\text{row-length } B1) \right) \right) \left( \text{col } B2 \left( j \mod (\text{length } B2) \right) \right) \right) \\
\text{using } \text{assms matrix-match-condn-1 unfolding matrix-match-def by auto} \\
\end{align*}
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:length B1 = row-length B2
and non-nil:(A1 ≠ [] ∧ (A2 ≠ [] ∧ (B1 ≠ [] ∧ (B2 ≠ [])))
and i:<(row-length A1) * (row-length B1) and j:<(length A2) * (length B2)
shows ((A1 ⊗ B1) ⊕ (A2 ⊗ B2))!j!i
= scalar-products
(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)))
proof -
have 1:mat
  ((row-length A1) * (row-length B1))
  ((length A1) * (length B1))
  (A1 ⊗ B1)
  using wf1 wf3 well-defined-Tensor by auto
moreover have 2:mat
  ((row-length A2) * (row-length B2))
  ((length A2) * (length B2))
  (A2 ⊗ B2)
  using wf2 wf4 well-defined-Tensor by auto
moreover have 3:((length A1) * (length B1))
  = ((row-length A2) * (row-length B2))
  using matchAA matchBB by auto
ultimately have 4:((A1 ⊗ B1) ⊕ (A2 ⊗ B2))!j!i
  = scalar-products (row (A1 ⊗ B1) i) (col (A2 ⊗ B2) j)
  using i j matrix-mult-index non-nil mat-mult-index
  row-length-mat scalar-product-def
  by auto
moreover have (row (A1 ⊗ B1) i)
  = vec-vec-Tensor
  (row A1 (i div row-length B1))
  (row B1 (i mod row-length B1))
  using wf1 wf3 i effective-row-formula by auto
moreover have col (A2 ⊗ B2) j = vec-vec-Tensor (col A2 (j div length B2))
  (col B2 (j mod 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
  \(\land (i < (\text{row-length } A1) \times (\text{row-length } B1))\)
  \(\land (j < (\text{length } A2) \times (\text{length } B2))\)
  \(\Rightarrow ((A1 \otimes B1) \circ (A2 \otimes B2))!j!i\)
  = scalar-product
  \[
  \begin{align*}
  &\text{(vec-vec-Tensor)} \\
  &\text{(row } A1 (i \text{ div row-length } B1)) \\
  &\text{(row } B1 (i \text{ mod row-length } B1)) \\
  &\text{(vec-vec-Tensor)} \\
  &\text{(col } A2 (j \text{ div length } B2)) \\
  &\text{(col } B2 (j \text{ mod length } B2))
  \end{align*}
  \]
  using elements-matrix-distribution2 unfolding matrix-match-def by auto

lemma effective-matrix-match-condn-2:
  assumes (matrix-match A1 A2 B1 B2)
  shows \(\forall i j. (i < (\text{row-length } A1) \times (\text{row-length } B1))\)
  \(\land (j < (\text{length } A2) \times (\text{length } B2))\)
  \(\Rightarrow ((A1 \otimes B1) \circ (A2 \otimes B2))!j!i\)
  = scalar-product
  \[
  \begin{align*}
  &\text{(vec-vec-Tensor)} \\
  &\text{(row } A1 (i \text{ div row-length } B1)) \\
  &\text{(row } B1 (i \text{ mod row-length } B1)) \\
  &\text{(vec-vec-Tensor)} \\
  &\text{(col } A2 (j \text{ div length } B2)) \\
  &\text{(col } B2 (j \text{ mod length } B2))
  \end{align*}
  \]
  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) ⇒ zip (a#v) (b#w) = (a,b)#(zip v w)

unfolding zip-def by auto

lemma scalar-product-times:
∀ w1 w2.(length w1 = length w2) ∧(length w1 = n) ⇒
(f (x*y)) (scalar-product w1 w2))
= (scalar-product
(times x w1)
(times y w2))

apply(rule allI)
apply (rule allI)
proof (induct n)
case 0
have (length w1 = length w2) ∧(length w1 = 0) ⇒ ?case
proof−
assume assms:(length w1 = length w2) ∧(length w1 = 0)
have 1: w1 = []
using assms by auto
moreover have 2:(length w1 = length w2) ∧(length w1 = 0) ⇒ w2 = []
by auto
ultimately have (length w1 = length w2) ∧(length w1 = 0)
⇒ scalar-product w1 w2 = zer
unfolding scalar-product-def scalar-prodI-def by auto
then have 3:(length w1 = length w2) ∧(length w1 = 0)
⇒ (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) ∧(length w1 = (Suc k)) ⇒ ?case
proof−
assume assms:(length w1 = length w2) ∧(length w1 = (Suc k))
have ∃aI u1.(w1 = a1#u1)∧(length u1 = k)
using assms by (metis length-Suc-conv)
then obtain a1 u1 where (w1 = a1#u1)∧(length u1 = k)
by auto
then have Cons-1:(w1 = a1#u1)∧(length u1 = k)
by auto
have length w2 = (Suc k)
using assms by auto
then have \( \exists a_2 \ u_2 \cdot (w_2 = a_2 \# u_2) \land (\text{length } u_2 = k) \)
using assms by (metis length-Suc-conv)
then obtain \( a_2 \ u_2 \)
where \( (w_2 = a_2 \# u_2) \land (\text{length } u_2 = k) \)
by auto
then have \( \text{Cons-2:} (w_2 = a_2 \# u_2) \land (\text{length } u_2 = k) \)
by auto
then have \( (\text{length } u_1 = \text{length } u_2) \land (\text{length } u_1 = k) \)
using CONS-1 by auto
then have \( \text{Cons-3:} x \ast y \ast \text{scalar-product } u_1 \ u_2 = \text{scalar-product} (\times x \ u_1) \ (\times y \ u_2) \)
using Suc assms by auto
have \( \times a_1 \ast \times a_2 \ = (\times a_1 \ast a_2) + (\text{scalar-product } u_1 \ u_2) \)
unfolding scalar-product-def scalar-prodI-def zip-def by auto
then have \( \text{Cons-4:} (x \ast y) \ast (\text{scalar-product } w_1 \ w_2) = ((x \ast y) \ast (a_1 \ast a_2)) + ((x \ast y) \ast (\text{scalar-product } u_1 \ u_2)) \)
using \( \text{comm } \text{assoc} \) by metis
have \( (\times x \ w_1) = (x \ast a_1) \# (\times x \ u_1) \)
using \( \times \ast \text{simps } \text{Cons-1} \) by auto
moreover have \( (\times y \ w_2) = (y \ast a_2) \# (\times y \ u_2) \)
using \( \times \ast \text{simps } \text{Cons-2} \) by auto
ultimately have \( \text{Cons-5:} \text{scalar-product} (\times x \ w_1) \ (\times y \ w_2) = \text{scalar-product} ((x \ast a_1) \# (\times x \ u_1)) \ (y \ast a_2) \# (\times y \ u_2)) \)
by auto
then have \( ... = ((x \ast a_1) \ast (y \ast a_2)) + \text{scalar-product} (\times x \ u_1) \ (\times y \ u_2) \)
unfolding scalar-product-def scalar-prodI-def zip-def by auto
with \( \text{Cons-3 } \text{Cons-4 } \text{Cons-5} \) show \(?\text{thesis}\) using \( \text{assoc} \) by auto
qed
then show \(?\text{case}\) by auto
qed

lemma \textit{effective-scalar-product-times}:\nassumes \( (\text{length } w_1 = \text{length } w_2) \)
shows \( (f \ (x \ast y) \ (\text{scalar-product } w_1 \ w_2)) = (\text{scalar-product} \ (\times x \ w_1) \ (\times y \ w_2)) \)
using scalar-product-times assms by auto

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

lemma scalar-product-append:
\[ \forall \, xs \, ys \, zs \, ws. (\text{length } zs = \text{length } ws) \land (\text{length } xs = \text{length } ys) \land (\text{length } xs = n) \implies \]
\[ (\text{scalar-product } (xs@zs) (ys@ws)) = (\text{scalar-product } xs \, ys) + (\text{scalar-product } zs \, ws) \]

apply(rule allI)
apply(rule allI)
apply(rule allI)
apply(rule allI)
proof(induct n)
case 0
have (length zs = length ws) \land (length xs = length ys) \land (length xs = 0)
implies
\[ (\text{scalar-product } (xs@zs) (ys@ws)) = (\text{scalar-product } xs \, ys) + (\text{scalar-product } zs \, ws) \]
proof-
assume assms:(length zs = length ws)\land(length xs = length ys)
\land(length xs = 0)
have 1:xs = []
using assms by auto
moreover have 2:ys = []
using assms by auto
ultimately have scalar-product xs ys = zero
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)\land(length xs = length ys)\land(length xs = (Suc k)) \implies \]
\[ (\text{scalar-product } (xs@zs) (ys@ws)) = (\text{scalar-product } xs \, ys) + (\text{scalar-product } zs \, ws) \]
proof-
assume assms:(length zs = length ws)
\land(length xs = length ys)
\land(length xs = (Suc k))
have \exists x \, xss. (xs = x#xss)\land(length xss = k)
using assms by (metis Suc-length-conv)
then obtain \( x \) \( xss \) where \( (xs = x#xss) \land (\text{length } xss = k) \)
by auto
then have 1: \( (xs = x#xss) \land (\text{length } xss = k) \)
by auto
have \( \exists \ y \ yss. (ys = y#yss) \land (\text{length } yss = k) \)
using assms by (metis Suc-length-conv)
then obtain \( y \) \( yss \) where \( (ys = y#yss) \land (\text{length } yss = k) \)
by auto
then have 2: \( (ys = y#yss) \land (\text{length } yss = k) \)
by auto
with 1 have \( \text{length } xss = \text{length } yss \land \text{length } xss = k \)
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 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)
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 \( ?\text{thesis} \) using 3 plus-assoc by auto
qed
then show \( ?\text{case} \) by auto
qed

lemma effective-scalar-product-append:
assumes \( \text{length } zs = \text{length } ws \) and \( (\text{length } xs = \text{length } ys) \)
sows \( (\text{scalar-product} (xs@zs) (ys@ws)) = (\text{scalar-product} xs ys) + (\text{scalar-product} zs ws) \)
using scalar-product-append assms by auto

lemma scalar-product-distributivity:
\( \forall v1 v2 w1 w2 . \ ((\text{length } v1 = \text{length } v2) \land (\text{length } v1 = n) \land (\text{length } w1 = \text{length } w2) \)
→ (scalar-product v1 v2)∗(scalar-product w1 w2)
= scalar-product (vec-vec-Tensor v1 w1) (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

74
case (Suc \( k \))
have \((\text{length } v_1 = \text{length } v_2) \land (\text{length } v_1 = \text{Suc } k)\)
\[ \implies f (\text{scalar-product } v_1 v_2) (\text{scalar-product } w_1 w_2) = \text{scalar-product} (\text{vec-vec-Tensor } v_1 w_1) (\text{vec-vec-Tensor } v_2 w_2) \]
proof
assume assms: \((\text{length } v_1 = \text{length } v_2) \land (\text{length } v_1 = \text{Suc } k)\)
have \(\text{length } v_1 = \text{Suc } k\)
using \(\text{Suc assms}\) by auto
then have \((\exists a_1 u_1. (v_1 = a_1 \# u_1) \land (\text{length } u_1 = k))\)
using \(\text{Suc-length-conv}\) by metis
then obtain \(a_1 u_1\) where \((v_1 = a_1 \# u_1) \land (\text{length } u_1 = k)\)
by auto
moreover have \(\text{length } v_2 = \text{Suc } k\)
using \(\text{assms \ Suc by auto}\)
then have \((\exists a_2 u_2. (v_2 = a_2 \# u_2) \land (\text{length } u_2 = k))\)
using \(\text{Suc-length-conv}\) by metis
then obtain \(a_2 u_2\) where \((v_2 = a_2 \# u_2) \land (\text{length } u_2 = k)\)
by auto
then have \(\text{Cons-1: } (v_1 = a_1 \# u_1) \land (\text{length } u_1 = k)\)
by simp
then have \(\text{length } u_1 = \text{length } u_2\)
using \(\text{Cons-1 by auto}\)
then have \(\text{Cons-3: } (\text{scalar-product } u_1 u_2) * (\text{scalar-product } w_1 w_2) = \text{scalar-product} (\text{vec-vec-Tensor } u_1 w_1) (\text{vec-vec-Tensor } u_2 w_2)\)
using \(\text{Suc Cons-1 Cons-2 assms by auto}\)
then have \(\text{zip } v_1 v_2 = (a_1, a_2) \# (\text{zip } u_1 u_2)\)
using \(\text{zip-Cons Cons-1 Cons-2 by auto}\)
then have \(\text{Cons-4: } (\text{scalar-product } v_1 v_2) (\text{scalar-product } w_1 w_2) = ((a_1 * a_2) + (\text{scalar-product } u_1 u_2)) * (\text{scalar-product } w_1 w_2)\)
by auto
then have \(\ldots = ((a_1 * a_2) * (\text{scalar-product } u_1 u_2)) + (\text{scalar-product } u_1 u_2) * (\text{scalar-product } w_1 w_2)\)
using \(\text{plus-right-distributivity by auto}\)
then have \(\text{Cons-5: } \ldots = ((a_1 * a_2) * (\text{scalar-product } u_1 u_2)) + \text{scalar-product} (\text{vec-vec-Tensor } u_1 w_1) (\text{vec-vec-Tensor } u_2 w_2)\)
using \(\text{Cons-3 by auto}\)
then have \(\text{Cons-6: } \ldots = (\text{scalar-product } (\text{times } a_1 w_1) (\text{times } a_2 w_2)) + \text{scalar-product} (\text{vec-vec-Tensor } u_1 w_1) (\text{vec-vec-Tensor } u_2 w_2)\)
using \(\text{assms effective-scalar-product-times by auto}\)
then have \(\text{scalar-product} (\text{vec-vec-Tensor } v_1 w_1) (\text{vec-vec-Tensor } v_2 w_2) = \text{scalar-product} (\text{vec-vec-Tensor } (a_1 \# u_1) w_1) (\text{vec-vec-Tensor } (a_2 \# u_2) w_2)\)
using \(\text{Cons-1 Cons-2 by auto}\)
moreover have \((\text{vec-vec-Tensor } (a1 \# u1) w1) = (\text{times } a1 w1) \otimes (\text{vec-vec-Tensor } u1 w1)\)
using \text{vec-vec-Tensor.simps by auto}
moreover have \((\text{vec-vec-Tensor } (a2 \# u2) w2) = (\text{times } a2 w2) \otimes (\text{vec-vec-Tensor } u2 w2)\)
using \text{vec-vec-Tensor.simps by auto}
ultimately have Cons-7: scalar-product \((\text{vec-vec-Tensor } v1 w1) (\text{vec-vec-Tensor } v2 w2)\)
\[= \text{scalar-product } (\text{times } a1 w1) \otimes (\text{vec-vec-Tensor } u1 w1))
(\text{times } a2 w2) \otimes (\text{vec-vec-Tensor } u2 w2)\)
by \text{auto}
moreover have length \((\text{vec-vec-Tensor } u2 w2) = \text{length } (\text{vec-vec-Tensor } u1 w1)\)
using \text{assms by (metis Cons-1 Cons-2 vec-vec-Tensor-length)}
moreover have length \((\text{times } a1 w1) = \text{length } (\text{times } a2 w2)\)
using \text{assms by (metis preserving-length)}
ultimately have scalar-product \((\text{times } a1 w1) \otimes (\text{vec-vec-Tensor } u1 w1))
(\text{times } a2 w2) \otimes (\text{vec-vec-Tensor } u2 w2)\)
using \text{effective-scalar-product-append by auto}
then show \(\?thesis\)
using Cons-6 Cons-7 \((a1 * a2 + \text{scalar-product } u1 u2 * \text{scalar-product } w1 w2)\)
\[= a1 * a2 + \text{scalar-product } u1 w2\]
\[+ (\text{scalar-product } u1 u2 * \text{scalar-product } w1 w2)\]
by \text{by (metis Cons-3 Cons-4 )}
qed
then show \(\?case\) by \text{auto}
qed

lemma effective-scalar-product-distributivity:
assumes length \(v1 = \text{length } v2\) and 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 \text{assms scalar-product-distributivity by auto}

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

\text{theorem row-col-match:}
fixes \(A1 A2 B1 B2 i j\)
assumes \(\text{wf1 : mat (row-length } A1) (\text{length } A1) A1\)
and \(\text{wf2 : mat (row-length } A2) (\text{length } A2) A2\)
and \(\text{wf3 : mat (row-length } B1) (\text{length } B1) B1\)
and \(\text{wf4 : mat (row-length } B2) (\text{length } B2) B2\)
and \(\text{matchAA : length } A1 = \text{row-length } A2\)
and \(\text{matchBB : length } B1 = \text{row-length } B2\)
and \(\text{non-nil : } (\text{A1} \neq []) \land (\text{A2} \neq []) \land (\text{B1} \neq []) \land (\text{B2} \neq [])\)
and \(i : i < (\text{row-length } A1) \ast (\text{row-length } B1)\) and \(j : j < (\text{length } A2) \ast (\text{length } B2)\)
shows \(\text{length } (\text{row } A1 (i \text{ div } (\text{row-length } B1))) = \text{length } (\text{col } A2 (j \text{ div } (\text{length } B2)))\)
and \(\text{length } (\text{row } B1 (i \text{ mod } (\text{row-length } B1))) = \text{length } (\text{col } B2 (j \text{ mod } (\text{length } B2)))\)

\text{proof –}

\begin{itemize}
  \item have \(i \text{ div } (\text{row-length } B1) < \text{row-length } A1\)
      using \(i\) by (metis \text{div-left-ineq})
  \item then have \(\text{1 : length } (\text{row } A1 (i \text{ div } (\text{row-length } B1))) = \text{length } A1\)
      unfolding \text{row-def by auto}
  \item have \(j \text{ div } (\text{length } B2) < \text{length } A2\)
      using \(j\) by (metis \text{div-left-ineq})
  \item then have \(\text{2 : length } (\text{col } A2 (j \text{ div } (\text{length } B2))) = \text{row-length } A2\)
      using \text{row-length-constant \(\text{wf2\) unfolding \text{col-def by auto}}\)
  \item with \(1\) match\(\text{AA\) show \(\text{length } (\text{row } A1 (i \text{ div } (\text{row-length } B1))) = \text{length } (\text{col } A2 (j \text{ div } (\text{length } B2)))\)\)
      by auto
  \item have \(i \text{ mod } (\text{row-length } B1) < \text{row-length } B1\)
      using \(i\) by (metis \text{less-nat-zero-code mod-less-divisor mult-is-0 neq0-conv})
  \item then have \(\text{2 : length } (\text{row } B1 (i \text{ mod } (\text{row-length } B1))) = \text{length } B1\)
      unfolding \text{row-def by auto}
  \item have \(j \text{ mod } (\text{length } B2) < \text{length } B2\)
      using \(j\) by (metis \text{less-nat-zero-code mod-less-divisor mult-is-0 neq0-conv})
  \item then have \(\text{length } (\text{col } B2 (j \text{ mod } (\text{length } B2))) = \text{row-length } B2\)
      using \text{row-length-constant \(\text{wf4\) unfolding \text{col-def by auto}}\)
  \item with \(2\) match\(\text{BB\) show \(\text{length } (\text{row } B1 (i \text{ mod } (\text{row-length } B1))) = \text{length } (\text{col } B2 (j \text{ mod } (\text{length } B2)))\)\)
\end{itemize}

77
by auto

lemma effective-row-col-match: assumes matrix-match A1 A2 B1 B2
shows\(\forall i\ j. ((i<(\text{row-length } A1)\wedge (j<(\text{length } A2)\wedge (\text{length } B2)))\)
\[\rightarrow \text{length (row } A1 \ (i \ \text{div} \ \text{row-length } B1)\) = \text{length (col } A2 \ (j \ \text{div} \ \text{length } B2))\]
\(\forall i\ j. ((i<(\text{row-length } A1)\wedge (j<(\text{length } A2)\wedge (\text{length } B2)))\)
\[\rightarrow \text{length (row } B1 \ (i \ \text{mod} \ \text{row-length } B1)\) = \text{length (col } B2 \ (j \ \text{mod} \ \text{length } B2))\]
using assms row-col-match unfolding matrix-match-def by auto

theorem prelim-element-match:
matrix-match A1 A2 B1 B2 \implies (\forall i\ j. ((i<(\text{row-length } A1)\wedge (j<(\text{length } A2)\wedge (\text{length } B2)))\)
\[\rightarrow (((A1 \circ A2)\otimes(B1 \circ B2))\lbrack i\rbrack = ((A1 \otimes B1)\circ(A2 \odot B2))\rbrack i)\]
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:
\(\forall i\ j. ((i<(\text{row-length } A1)\wedge (j<(\text{length } A2)\wedge (\text{length } B2)))\)
\[\rightarrow (((A1 \circ A2)\otimes(B1 \circ B2))\lbrack i\rbrack = ((A1 \otimes B1)\circ(A2 \odot B2))\rbrack i)\]
using effective-matrix-match-condn-1 assms by metis
moreover from 1 have 3:\(\forall i\ j. ((i<(\text{row-length } A1)\wedge (j<(\text{length } A2)\wedge (\text{length } B2)))\)
\[\rightarrow (((vec-vec-Tensor \ (row A1 \ (i \ \text{div} \ \text{row-length } B1)\) \ (row B1 \ (i \ \text{mod} \ \text{row-length } B1))) \ (vec-vec-Tensor \ (col A2 \ (j \ \text{div} \ \text{length } B2)\) \ (col B2 \ (j \ \text{mod} \ \text{length } B2)))\)
using effective-matrix-match-condn-2 by auto
have \(\forall i\ j. ((i<(\text{row-length } A1)\wedge (j<(\text{length } A2)\wedge (\text{length } B2)))\)
\[\rightarrow \text{length (row } A1 \ (i \ \text{div} \ \text{row-length } B1)\) = \text{length (col } A2 \ (j \ \text{div} \ \text{length } B2))\]
and \(\forall i\ j. ((i<(\text{row-length } A1)\wedge (j<(\text{length } A2)\wedge (\text{length } B2)))\)
\[\rightarrow \text{length (row } B1 \ (i \ \text{mod} \ \text{row-length } B1)\) = \text{length (col } B2 \ (j \ \text{mod} \ \text{length } B2))\]
using assms effective-row-col-match by auto
then have \( \forall \ i \ j. \ ((i < (\text{row-length } A1) \ast (\text{row-length } B1)) \land (j < (\text{length } A2) \ast (\text{length } B2))) \)
\[ \rightarrow \]
\[(\text{scalar-product } (\text{row } A1 \ (i \text{ div } (\text{row-length } B1))) \ (\text{col } A2 \ (j \text{ div } (\text{length } B2)))) \]
\[\ast (\text{scalar-product } (\text{row } B1 \ (i \text{ mod } (\text{row-length } B1))) \ (\text{col } B2 \ (j \text{ mod } (\text{length } B2)))) \]
\[= \text{ scalar-product } \]
\[(\text{vec-vec-Tensor } (\text{row } A1 \ (i \text{ div } \text{row-length } B1))) \ (\text{col } B2 \ (j \text{ mod } \text{length } B2))) \]
using effective-scalar-product-distributivity by auto
then show \(?thesis using 2 3 \) by auto
qed

theorem element-match:
matrix-match A1 A2 B1 B2 \[\implies (\forall \ i < ((\text{row-length } A1) \ast (\text{row-length } B1)). \]
\[\forall j < ((\text{length } A2) \ast (\text{length } B2)). \]
\[((A1 \circ A2) \circ (B1 \circ B2))! j! i \]
\[= ((A1 \otimes B1) \circ (A2 \otimes B2))! j! i \]
using prelim-element-match by auto

lemma application: fixes \( m1 \ m2 \)
sows \( \forall \ m1 \ m2. (\mat nr nc m1) \)
\[\land (\mat nr nc m2) \]
\[\land (\forall j < nc. \forall i < nr. m1 \ ! j \ ! i = m2 \ ! j \ ! i) \]
\[\rightarrow (m1 = m2) \]
using mat-eqI by blast

theorem tensor-compose-condn:
assumes \( wf1:\mat nr nc ((A1 \circ A2) \circ (B1 \circ B2)) \)
and \( wf2:\mat nr nc ((A1 \otimes B1) \circ (A2 \otimes B2)) \)
and \( wf3:\forall j < nc. \forall i < nr. ((A1 \circ A2) \circ (B1 \circ B2))! j! i \)
\[= ((A1 \otimes B1) \circ (A2 \otimes B2))! j! i \]
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 \( \mat-match A1 A2 B1 B2 \)
sows \( ((A1 \circ A2) \otimes (B1 \circ B2)) = ((A1 \otimes B1) \circ (A2 \otimes B2)) \)
proof –
let \( ?nr = ((\text{row-length } A1) \ast (\text{row-length } B1)) \)
let \( ?nc = ((\text{length } A2) \ast (\text{length } B2)) \)
have \( \mat ?nr ?nc ((A1 \circ A2) \otimes (B1 \circ B2)) \)

79
by (metis assms effective-tensor-compose-distribution1)
moreover have mat ?nr ?nc ((A1 ⊗ B1)⊙(A2 ⊗ B2))
  using assms by (metis effective-tensor-compose-distribution2)
moreover have ∀ j<?nc.∀ i<?nr.
  (((A1 ⊗ A2)⊙(B1 ⊗ B2))!j!i
  = ((A1 ⊗ B1)⊙(A2 ⊗ B2))!j!i)
  using element-match assms by auto
ultimately show ?thesis
  using application by blast
qed
end