Tensor Product of Matrices

T.V.H. Prathamesh

April 20, 2020

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, ...v_n)(w_1, w_2, ...w_m) = (v_1\cdot w_1, ..., v_1\cdot w_m, ..., v_n\cdot w_1, ..., 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
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
even 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 [] = []
\texttt{|reduct (x\#xs) = xs}

\textbf{lemma} length-reduct:
\texttt{assumes m \neq []
show 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))}

\textbf{lemma} mat-empty-column-length: \texttt{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 \texttt{?thesis} by simp
qed}

\textbf{lemma} vec-uniqueness:
\texttt{assumes vec m v and vec n v
shows m = n
using vec-def assms by metis}

\textbf{lemma} mat-uniqueness:
\texttt{assumes mat nr1 nc M and mat nr2 nc M and z = hd M and M \neq []
shows (\forall x \in (set M).(nr1 = nr2))
proof–
have A:z \in set M using assms(1) assms(3) assms(4) set-def mat-def
by (metis hd-in-set)
have Ball (set M) (vec nr1) using mat-def assms(1) by auto
then have step1: ((x \in (set M)) \rightarrow (vec nr1 x)) using Ball-def assms by auto
have Ball (set M) (vec nr2) using mat-def assms(2) by auto
then have step2: ((x \in (set M)) \rightarrow (vec nr2 x)) using Ball-def assms by auto
from step1 and step2
\begin{itemize}
\item have step3:\forall x.(x \in (set M)) \rightarrow (\forall (vec nr1 x)\land (vec nr2 x))
by (metis Ball (set M) (vec nr1) Ball (set M) (vec nr2))
\item have ((vec nr1 x)\land (vec nr2 x)) \rightarrow (nr1 = nr2) using vec-uniqueness by auto
\end{itemize}
with step3
\begin{itemize}
\item have (\forall x.(x \in (set M)) \rightarrow (\forall (nr1 = nr2))) by (metis vec-uniqueness)
\end{itemize}
then
\begin{itemize}
\item have (\forall x \in (set M).(nr1 = nr2)) by auto
\end{itemize}
then
show ?thesis by auto
qed}

\textbf{lemma} mat-empty-row-length: \texttt{assumes mat nr nc M and M = []}
shows \( \text{mat 0 nc } M \)

**proof**
- have \( \text{set } M = \{ \} \) using mat-def assms empty-set by auto
- then have \( \text{Ball (set } M) \text{ (vec 0) } \) using Ball-def by auto
- then have \( \text{mat 0 nc } M \) using mat-def assms(1) assms(2) gen-length-code(1)

length-code
  by (metis (full-types))
- then show \(?thesis by auto
qed

**abbreviation** null-matrix::'a list list

where
null-matrix \( \equiv [\text{Nil}] \)

**lemma** null-mat: null-matrix = [[]] by auto

**lemma** zero-matrix: mat 0 0 [[]] using mat-def in-set-insert insert-nil list.size(3) not-Cons-self2
  by (metis (full-types))

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

**definition** row-length:: 'a mat \( \Rightarrow \) nat

where
row-length xs \( \equiv \text{ if } (xs = []) \text{ then 0 else length (hd xs)} \)

**lemma** row-length-Nil:
  row-length [[]] =0
  using row-length-def by (metis )

**lemma** row-length-Null:
  row-length [ [[]] ] =0
  using row-length-def by auto

**lemma** row-length-vect-mat:
  row-length (vec-mat-Tensor v m) = length v*(row-length m)

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

let ?case = row-length (vec-mat-Tensor v (a#m)) = (length v) * (row-length (a#m))

have A: row-length (a # m) = length a using row-length-def list.distinct(1) by auto
have (vec-mat-Tensor v (a#m)) = (vec-vec-Tensor v a)#(vec-mat-Tensor v m)
  using vec-mat-Tensor-def vec-mat-Tensor.simps(2) by auto

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

done

Tensor is the tensor product of matrices

primrec Tensor:: 'a mat ⇒ 'a mat ⇒ 'a mat (infixl ⊗ 63)
where
  Tensor [] xs = []
  Tensor (x#xs) ys = (vec-mat-Tensor x ys)@ (Tensor xs ys)

lemma Tensor-null: xs ⊗ [] = []
  by (induction xs) (auto)

Tensor commutes with left and right identity

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

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

row-length of tensor product of matrices is the product of their respective row lengths

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

  ...
have row-length ([\bigotimes] m2) = (row-length []) \ast (row-length m2)
using row-length-Nil
by auto
then show ?case by metis

next
fix a m1
assume row-length (m1 \bigotimes m2) = row-length m1 \ast row-length m2
let ?case =
row-length ((a \# m1) \bigotimes m2) = row-length (a \# m1) \ast 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) \bigotimes m2) = row-length (a \# m1) \ast 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 \bigotimes m2) = row-length (a \# m1) \ast row-length m2
let ?case =
row-length (a \# m1 \bigotimes aa \# m2)
= row-length (a \# m1) \ast row-length (aa \# m2)
by auto
from this have non-zero: (vec-mat-Tensor a (aa\#m2)) \neq []
  using vec-mat-Tensor-def by auto
from this have
  hd ((vec-mat-Tensor a (aa\#m2)) \bigotimes (m1 \bigotimes m2))
  = hd (vec-mat-Tensor a (aa\#m2))
by auto
from this have
  hd ((a\#m1) \bigotimes (aa\#m2)) = hd (vec-mat-Tensor a (aa\#m2))
using Tensor.simps(2) by auto
from this have s1: row-length ((a\#m1) \bigotimes (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) \ast 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) \ast row-length(aa\#m2)
by auto
from this and B
  have row-length (vec-mat-Tensor a (aa\#m2))
    = (row-length (a\#m1)) \ast row-length(aa\#m2)
by auto
from this and s1 show ?case by auto
from this show ?case by auto

lemma hd-set: assumes x ∈ set (a#M) shows (x = a) ∨ (x∈(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 ∈ set (a #N)) → (x = a) ∨ (x ∈ (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
\[(\text{row-length } M) \ast (\text{length } v)\]
\[(\text{length } M)\]
\[(\text{vec-mat-Tensor } v \ M)\]

\textbf{proof}\(\text{(induct } M)\)
\begin{description}
\item[case Nil] have \((\text{vec-mat-Tensor } v \ []\) = []
\hspace{1em}\text{using vec-mat-Tensor.simps(1) Nil}
\hspace{1em}\text{by simp}
moreover have \((\text{row-length } [] = 0)\)
\hspace{1em}\text{using row-length-def Nil}
\hspace{1em}\text{bymetis}
moreover have \((\text{length } []) = 0\)
\hspace{1em}\text{using Nil by simp}
ultimately have
\hspace{1em}\text{mat } ((\text{row-length } []) \ast (\text{length } v)) (\text{length } []) (\text{vec-mat-Tensor } v \ [])
\hspace{1em}\text{using zero-matrix by (metis mult-zero-left)}
\hspace{1em}then show \?case by simp
\end{description}

next
fix \(a \ M\)
assume \text{hyp}:
\begin{align*}
(\text{mat } (\text{row-length } M) (\text{length } M) \ M) \\
\implies (\text{mat } (\text{row-length } M \ast \text{length } v) (\text{length } M) (\text{vec-mat-Tensor } v \ M)) \\
(\text{mat } (\text{row-length } (a\text{#}M)) (\text{length } (a\text{#}M)) (a\text{#}M))
\end{align*}
let \?case =
\begin{align*}
\text{mat } (\text{row-length } (a\text{#}M) \ast \text{length } v) (\text{length } (a\text{#}M)) (\text{vec-mat-Tensor } v \ (a\text{#}M))
\end{align*}
have \text{step1}: \(\text{mat } (\text{row-length } M) (\text{length } M) \ M\)
\hspace{1em}\text{using hyp(2) reduct-matrix by auto}
then have \text{step2}:
\begin{align*}
\text{mat } (\text{row-length } M \ast \text{length } v) (\text{length } M) (\text{vec-mat-Tensor } v \ M)
\end{align*}
\hspace{1em}\text{using hyp(1) by auto}
have
\begin{align*}
\text{mat } \\
(\text{row-length } (a\text{#}M) \ast \text{length } v) \\
(\text{length } (a\text{#}M)) \\
(\text{vec-mat-Tensor } v \ (a\text{#}M))
\end{align*}
\textbf{proof}\(\text{(cases } M)\)
\begin{description}
\item[case Nil] fix \(x\)
have \(1\):\((\text{vec-mat-Tensor } v \ (a\text{#}M)) = \text{[vec-mat-Tensor } v \ a]\)
\hspace{1em}\text{using vec-mat-Tensor.simps Nil by auto}
have \((x \in (\text{set } \text{[vec-mat-Tensor } v \ a])) \implies x = \text{[vec-mat-Tensor } v \ a]\)
\hspace{1em}\text{using set-def by auto}
then have \(2\):
\begin{align*}
(x \in (\text{set } \text{[vec-mat-Tensor } v \ a])) \\
\implies (\text{vec } \text{(length } (\text{vec-mat-Tensor } v \ a)) \ x)
\end{align*}
\hspace{1em}\text{using vec-def by metis}
have \(3\):\((\text{length } (\text{vec-mat-Tensor } v \ a)) = (\text{length } v) \ast (\text{length } a)\)
\hspace{1em}\text{using vec-Tensor-length by auto}
then have \(4\):
\[ \text{length (vec-vec-Tensor } v \ a) = (\text{length } v) \ast (\text{row-length (a#M)}) \]

\[ \text{using row-length-def list.different(1)} \]

\[ \text{by auto} \]

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

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

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

\[ \text{by (simp add: Nil mat-def vec-def)} \]

\[ \text{hence} \]

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

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

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

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

\textbf{then show ?thesis using 6 by auto} \]

\textbf{next}

\textbf{case (Cons b L)}

\textbf{fix x}

\[ \text{have 1:x \in (set (a#M)) \rightarrow (((x=a) \lor (x \in (set M)))} \]

\[ \text{using hd-set by auto} \]

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

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

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

\[ \text{using hyp by auto} \]

\[ \text{then have x\in (set (a#M)) \rightarrow (vec (row-length (a#M)) x)} \]

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

\[ \text{then have x\in (set (a#M))\rightarrow (vec (length a) x)} \]

\[ \text{using row-length-def list.different(1)} \]

\[ \text{by auto} \]

\[ \text{with } \text{1 have x\in (set M)\rightarrow (vec (length a) x)} \]

\[ \text{by auto} \]

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

\[ \text{using Cons by auto} \]

\textbf{ultimately have vec (length a) b} \]

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

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

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

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

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

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

\[ \text{using step2 by auto} \]

\[ \text{then have 3:} \]

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

\[ \text{using mat-def by auto} \]

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

\[ \rightarrow (vec ((row-length M)*length v)) x \]

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

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

\[ \rightarrow (vec (length a)*length v) x \]

\[ \text{using 2 by auto} \]

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

\[ \text{using vec-vec-Tensor-length by auto} \]
then have \(6\): \(\text{vec} ((\text{length } a) \ast (\text{length } v)) (\text{vec-vec-Tensor } v a)\)
using \(\text{vec-vec-Tensor-length vec-def by (metis (full-types))}\)

have \(7\):(\(\text{length } a) = (\text{row-length } (a \# M))\)
using \(\text{row-length-def list.distinct(1) by auto}\)

have \(\text{vec-mat-Tensor } v (a \# M)\)
\[= (\text{vec-vec-Tensor } v a)\#(\text{vec-mat-Tensor } v M)\]
using \(\text{vec-mat-Tensor.simps(2) by auto}\)
then have \(x \in \text{set (vec-mat-Tensor } v (a \# M))\)
\[\rightarrow ((x = (\text{vec-vec-Tensor } v a))\]
\[\vee (x \in (\text{set (vec-mat-Tensor } v M))))\]
using \(\text{hd-set by auto}\)
with \(4 \, 6\) have \(x \in \text{set (vec-mat-Tensor } v (a \# M))\)
\[\rightarrow \text{vec } ((\text{length } a) \ast (\text{length } v)) x\]
by \(\text{auto}\)
with \(7\) have \(x \in \text{set (vec-mat-Tensor } v (a \# M))\)
\[\rightarrow \text{vec } ((\text{row-length } (a \# M)) \ast (\text{length } v)) x\]
by \(\text{auto}\)
then have \(\forall x.((x \in \text{set (vec-mat-Tensor } v (a \# M)))\)
\[\rightarrow \text{vec } ((\text{row-length } (a \# M)) \ast (\text{length } v)) x\]
using \(2 \, 3 \, 6 \, 7 \, \text{hd-set vec-mat-Tensor.simps(2) by auto}\)
then have \(7\): \(\text{Ball}\)
\[(\text{set (vec-mat-Tensor } v (a \# M)))\]
\[= (\text{vec } ((\text{row-length } (a \# M)) \ast (\text{length } v)))\]
using \(\text{Ball-def by auto}\)

have \(8\): \(\text{length } (\text{vec-mat-Tensor } v (a \# M)) = \text{length } (a \# M)\)
using \(\text{vec-mat-Tensor-length by auto}\)
with \(6 \, 7\) have mat
\[(\text{row-length } (a \# M)) \ast (\text{length } v)\]
\[= (\text{vec-mat-Tensor } v (a \# M))\]
using \(\text{mat-def 5 length-code by (metis (hide-lams, no-types))}\)
then show \(?\text{thesis by auto}\)
qed
with \(\text{hyp show ?case by auto}\)
qed

The following theorem gives length of tensor product of two matrices

\textbf{lemma length-Tensor}: \(\text{(length } (M1 \otimes M2)) = (\text{length } M1) \ast (\text{length } M2)\)
\textbf{proof}(\text{induct } M1) \text{ case } \text{Nil} \text{ show } ?\text{case by auto} \text{ next}
\text{case } (\text{Cons } a M1) \text{ have } ((a \# M1) \otimes M2) = (\text{vec-mat-Tensor } a M2) @ (M1 \otimes M2)
using \(\text{Tensor.simps(2) by auto}\)
then have \(1\):
\[ \text{length } ((a \# M1) \otimes M2) = \text{length } ((\text{vec-mat-Tensor } a M2) @ (M1 \otimes M2)) \]

by auto
have 2: \[ \text{length } ((\text{vec-mat-Tensor } a M2) @ (M1 \otimes M2)) = \text{length } (\text{vec-mat-Tensor } a M2) + \text{length } (M1 \otimes M2) \]
using append-def
by auto
have 3: \[ (\text{length } (\text{vec-mat-Tensor } a M2) @ (M1 \otimes M2)) = (\text{length } M2) + (\text{length } M1) \times (\text{length } M2) \]
using Cons.hyps by auto
with 2 3 have \[ (\text{length } ((\text{vec-mat-Tensor } a M2) @ (M1 \otimes M2))) = (\text{length } M2) + (\text{length } M1) \times (\text{length } M2) \]
by auto
then have 5: \[ (\text{length } ((\text{vec-mat-Tensor } a M2) @ (M1 \otimes M2))) = (1 + (\text{length } M1)) \times (\text{length } M2) \]
by auto
with 1 have \[ (\text{length } ((a \# M1) \otimes M2)) = ((\text{length } (a \# M1)) \times (\text{length } M2)) \]
by auto
then show \?case by auto
qed

Lemma append-reduct-matrix:
[mat (row-length (M1 @ M2)) (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 [mat (row-length (M1 @ M2)) (length (M1 @ M2)) (M1 @ M2)]
using reduct-matrix Cons.prems append-Cons by metis
from this have [mat (row-length M2) (length M2) M2]
using Cons.hyps by auto
from this show \?thesis by simp
qed

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

Theorem well-defined-Tensor:
[mat (row-length M1) (length M1) M1]
\[ \land [\text{mat } (\text{row-length } M2) (\text{length } M2) M2] \]
\[ \implies [\text{mat } ((\text{row-length } M1) \times (\text{row-length } M2)) ((\text{length } M1) \times (\text{length } M2)) (M1 @ M2)] \]
proof (induct M1)
case Nil
have [\text{row-length } []] \times (\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 by metis
then show ?case by simp
next
case (Cons a M1)
  have step1: mat (row-length (a#M1)) (length (a#M1))(a#M1)
    using Cons.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)

14
using mat-def Ball-def by metis
then have $x \in (\text{set}(\text{a}\#\text{M1}) \rightarrow (\text{vec}(\text{length}\ a)\ x))$
using row-length-def list.distinct(1)
by auto
with I have $x \in (\text{set}\ \text{M1}) \rightarrow (\text{vec}(\text{length}\ a)\ x)$
by auto
moreover have $b \in (\text{set}\ \text{M1})$
using Cons by auto
ultimately have $\text{vec}(\text{length}(\text{a})\ x)$
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 $2:\text{row-length}\ \text{M1} = (\text{length}\ a)$
using row-length-def Cons by auto
then have $\text{mat} \ ((\text{length}\ a)\ *\ \text{row-length}\ \text{M2})$
$(\text{length}\ \text{M1} \ *\ \text{length}\ \text{M2})$
$(M1 \odot M2)$
using step2 by auto
then have Ball $(\text{set}\ (\text{M1} \odot \text{M2}))\ (\text{vec}\ ((\text{length}\ a)\ *\ (\text{row-length}\ \text{M2})))$
using mat-def by auto
from this have 3:
$\forall x.\ x \in (\text{set}\ (\text{M1} \odot \text{M2})) \rightarrow (\text{vec}\ ((\text{length}\ a)\ *\ (\text{row-length}\ \text{M2}))\ x)$
using Ball-def by auto
have $\text{mat} \ ((\text{row-length}\ \text{M2})\ * (\text{length}\ a))$
$(\text{length}\ \text{M2})$
$(\text{vec-mat-Tensor}\ \text{a}\ \text{M2})$
using well-defined-vec-mat-Tensor Cons.prems
by auto
then have Ball $(\text{set}\ (\text{vec-mat-Tensor}\ \text{a}\ \text{M2}))$
$(\text{vec}\ ((\text{row-length}\ \text{M2})\ * (\text{length}\ a)))$
using mat-def by auto
then have 4:
$\forall x.\ x \in (\text{set}\ (\text{vec-mat-Tensor}\ \text{a}\ \text{M2}))$
$\rightarrow (\text{vec}\ ((\text{length}\ a)\ * (\text{row-length}\ \text{M2}))\ x)$
using mult.commute by metis
with 3 have 5: $\forall x.\ (x \in (\text{set}\ (\text{vec-mat-Tensor}\ \text{a}\ \text{M2})))$
$\forall(x \in (\text{set}\ (\text{M1} \odot \text{M2})))$
$\rightarrow (\text{vec}\ ((\text{length}\ a)\ * (\text{row-length}\ \text{M2}))\ x)$
by auto
have 6: $(a \# M1 \odot M2) = (\text{vec-mat-Tensor}\ \text{a}\ \text{M2})\odot (M1 \odot M2)$
using Tensor.simps(2) by auto
then have $x \in (\text{set}\ (a \# M1 \odot M2))$
$\rightarrow (x \in (\text{set}\ (\text{vec-mat-Tensor}\ \text{a}\ \text{M2})))\ {\forall(x \in (\text{set}\ (\text{M1} \odot \text{M2})))}$
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 \textit{thesis} by auto
qed
then show \textit{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<(length v)) \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 <(length []) \Rightarrow False
    by auto
  moreover have (times a []) = []
    using times.simps(1) by auto
  ultimately have (i<(length [])) \rightarrow (times a [])!i = f a ([]!i)
    by auto
  then have \forall i. ((i<(length [])) \rightarrow (times a [])!i = f a ([]!i))
    by auto
then show \(?\)case by auto
next
case (Cons x xs)
  have \(\forall i.((x\#xs)!(i+1) = (xs)!i)\)
    by auto
  have 0: \(\langle i < \text{length } (x\#xs) \rangle \rightarrow \langle (i < \text{length } xs) \lor (i = \text{length } xs) \rangle \)
    by auto
  have 1: \(\langle i < \text{length } xs \rangle \rightarrow \langle (\text{times } a xs)!i = f a ((xs)!i) \rangle \)
    by (metis Cons.hyps)
  have \(\forall i.((x\#xs)!(i+1) = (xs)!i)\) by auto
  have \(\langle i < \text{length } (x\#xs) \rangle \rightarrow \langle \text{times } (x\#xs)!i = f a ((x\#xs)!i) \rangle \)
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 ?thesis by auto
next
case (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 Suc by auto
  have 3: \(\langle i < \text{length } (x\#xs) \rangle \rightarrow \langle j < \text{length } xs \rangle \)
    using One-nat-def Suc-Suc-ev plus1 list.size(4) not-less-eq
    by metis
  have 4: \(\langle j < \text{length } xs \rangle \rightarrow \langle (\text{times } a xs)!j = (f a (xs)!j) \rangle \)
    using 1 by (metis Cons.hyps)
  have 5: \((x\#xs)!i = (xs)!j\)
    using Suc by (metis nth-Cons-Suc)
with 1 2 4 have \(\langle j < \text{length } xs \rangle \rightarrow \langle (\text{times } a (x\#xs))!i = (f a ((x\#xs)!i)) \rangle \)
  by auto
with 3 have \(\langle i < \text{length } (x\#xs) \rangle \rightarrow \langle (\text{times } a (x\#xs))!i = (f a ((x\#xs)!i)) \rangle \)
  by auto
then show ?thesis by auto
qed
then show ?case by auto
qed

lemma simpl-times-elements:
  assumes \(\langle i < \text{length } xs \rangle\)
  shows \(\langle i < \text{length } v \rangle \rightarrow \langle (\text{times } a) v)!i = f a (v)!i \rangle \)
  using times-elements by auto

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

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

lemma append-simpl3:
assumes \( i > (\text{length } y) \)
shows \( (i < ((\text{length } (z#zs))*(\text{length } y))) \rightarrow (i - (\text{length } y)) < (\text{length } zs)*(\text{length } y) \)
proof-
have \( \text{length } (z#zs) = (\text{length } zs)+1 \)
  by auto
then have \( i < ((\text{length } (z#zs))*(\text{length } y)) \)
  \( \rightarrow \) \( i < ((\text{length } zs)+1)*(\text{length } y) \)
  by auto
then have 1: \( i < ((\text{length } (z#zs))*(\text{length } y)) \)
  \( \rightarrow \) \( (i - (\text{length } y)) < ((\text{length } zs)*(\text{length } y)) \)
  by auto
then show ?thesis by auto
qed

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

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

18
lemma vec-vec-Tensor-simpl2:
\((i \geq (\text{length } y))\) 
\(\rightarrow ((\text{vec-vec-Tensor } (z#zs) y)!i = (\text{vec-vec-Tensor } zs y)!((i - (\text{length } y))))\)
using vec-vec-Tensor.simps(2) append-simpl2 preserving-length 
by \text{metis}

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

lemma int-nat-div:
\((\text{int } a) \text{ div } (\text{int } b) = \text{int } ((a::\text{nat}) \text{ div } b)\)
by \(\text{metis zdiv-int}\)

lemma int-nat-eq:
assumes \(\text{int } (a::\text{nat}) = \text{int } b\) 
shows \(a = b\) 
using assms of-nat-eq-iff by auto 

lemma nat-div:
assumes \((b::\text{nat}) > 0\) 
and \(a > b\) 
shows \((a \text{ div } b) = ((a - b) \text{ div } b) + 1\)
proof- 
fix \(x\) 
have 1:\((\text{int } b)>0\) 
using assms(1) division-product by auto 
moreover have \((\text{int } a)>(\text{int } b)\) 
using assms(2) by auto 
with 1 have 2: \(((\text{int } a) \text{ div } (\text{int } b))\)
\[(\text{int}\ a - \text{int}\ b) \div \text{int}\ b) + 1\]

using division-product by auto

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

from int-nat-div assms(2) have 4:
    (((\text{int}\ a) - \text{int}\ b)) \div \text{int}\ (b)) = \text{int}\ ((a - b) \div b)
by (metis (full-types) less-asm not-less of-nat-diff)

have \((\text{int}\ x) + 1 = \text{int}\ (x + 1)\)
by auto

with 2 \& 4 have \((\text{int}\ a) \div \text{int}\ b) = \text{int}\ (((a - b) \div b) + 1)\)
by auto

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

qed

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

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

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

lemma logic:
assumes \(A \rightarrow B\)
and \(\neg A \rightarrow B\)
sshows \(B\)
using assms(1) assms(2) by auto

theorem vec-vec-Tensor-elements:
assumes \((y \neq [])\)
sshows \(\forall i.((i < ((\text{length}\ x) \ast (\text{length}\ y))))\)
\(\rightarrow ((\text{vec-vec-Tensor}\ x\ y)!i)\)
\(= f\ (x!(i \div (\text{length}\ y)))\ (y!(i \mod (\text{length}\ y))))\)
apply (rule allI)
proof (induct x)
case Nil
have \((\text{length} \ [] = 0)\)
by auto
also have \((\text{length} \ (\text{vec-vec-Tensor} \ []\ y) = 0)\)
using vec-vec-Tensor.simps(1) by auto
then have \(i < (\text{length} \ (\text{vec-vec-Tensor} \ []\ y)) \Longrightarrow\ False\)
by auto
moreover have \((\text{vec-vec-Tensor} \; \emptyset \; y) = \emptyset\)
  by auto
moreover have
\[(i < (\text{length} \; (\text{vec-vec-Tensor} \; \emptyset \; y))) \rightarrow
  ((\text{vec-vec-Tensor} \; x \; y)!i) = f \; (x!(i \text{ div} \; (\text{length} \; y))) \; (y!(i \text{ mod} \; (\text{length} \; y)))\]
  by auto
then show \(?\text{case}\)
  by auto
next
  case \((\text{Cons} \; z \; zs)\)
  have \(1: \text{vec-vec-Tensor} \; (z \# zs) \; y = (\text{times} \; z \; y)@(\text{vec-vec-Tensor} \; zs \; y)\)
    by auto
  have \(2:i < (\text{length} \; y) \rightarrow ((\text{times} \; z \; y)!i = f \; z \; (y!i))\)
    using \text{times-elements} \; \text{by auto}
moreover have \(3: \)
\[i < (\text{length} \; y) \rightarrow (\text{vec-vec-Tensor} \; (z \# zs) \; y)!i = (\text{times} \; z \; y)!i\]
  using \text{vec-vec-Tensor-simpl} \; \text{by auto}
moreover have \(35: \)
\[i < (\text{length} \; y) \rightarrow (\text{vec-vec-Tensor} \; (z \# zs) \; y)!i = f \; z \; (y!i)\]
  using \text{calculation(1)} \; \text{calculation(2)} \; \text{by metis}
  have \(4: (y \neq \emptyset) \rightarrow (\text{length} \; y) > 0\)
    by auto
  have \((i < (\text{length} \; y)) \rightarrow ((i \text{ div} \; (\text{length} \; y)) = 0)\)
    by auto
then have \(6: (i < (\text{length} \; y)) \rightarrow (z \# zs)!(i \text{ div} \; (\text{length} \; y)) = z\)
  using \text{nth-Cons-0} \; \text{by auto}
then have \(7: (i < (\text{length} \; y)) \rightarrow (i \text{ mod} \; (\text{length} \; y)) = i\)
  by auto
with \(2 \; 6\)
  have \((i < (\text{length} \; y)) \rightarrow (\text{times} \; z \; y)!i = f \; ((z \# zs)!(i \text{ div} \; (\text{length} \; y))) \; (y! \; (i \text{ mod} \; (\text{length} \; y)))\)
    by auto
with \(3\) have \text{step1}: \((i < (\text{length} \; y)) \rightarrow ((i < ((\text{length} \; x) \text{*(length} \; y))) \rightarrow ((\text{vec-vec-Tensor} \; (z \# zs) \; y)!i = f \; ((z \# zs)!(i \text{ div} \; (\text{length} \; y))) \! (y! \; (i \text{ mod} \; (\text{length} \; y))))))\)
    by auto
  have \((\text{length} \; y) \leq i \rightarrow (i - (\text{length} \; y)) \geq 0\)
    by auto
  have \text{step2}: \((\text{length} \; y) < i \rightarrow ((i < (\text{length} \; (z \# zs) \text{*(length} \; y))) \rightarrow ((\text{vec-vec-Tensor} \; (z \# zs) \; y)!i = f)\)

21
proof—

have \((\text{length } y) > 0\)
  using assms by auto
then have 1:
  \((i > (\text{length } y))\)
  \((i \div (\text{length } y)) = ((i - (\text{length } y)) \div (\text{length } y)) + 1\)
  using \(\text{nat-div}\) by auto
have \((z\#zs)!((j+1))\)
  by auto
then have
  \((z\#zs)!((i - (\text{length } y)) \div (\text{length } y)))\)
  \((z\#zs)!((i - (\text{length } y)) \div (\text{length } y)) + 1\)
  by auto
with 1 have 2:
  \((i > (\text{length } y))\)
  \((z\#zs)!((i - (\text{length } y)) \div (\text{length } y))\)
  = \((z\#zs)!((i - (\text{length } y)) \div (\text{length } y))\)
  by auto
have \((i > (\text{length } y))\)
  \((i \mod (\text{length } y))\)
  = \(((i - (\text{length } y)) \mod (\text{length } y)))\)
  using \(\text{nat-mod}\) by auto
then have 3:
  \((i > (\text{length } y))\)
  \(((y! (i \mod (\text{length } y)))\)
  = \((y! ((i - (\text{length } y)) \mod (\text{length } y)))\))
  by auto
have 4: \((i > (\text{length } y))\)
  \((\text{vec-vec-Tensor } z\#zs) y)!i\)
  = \((\text{vec-vec-Tensor } zs y)!((i - (\text{length } y))\)
  using \(\text{vec-vec-Tensor-simpl2}\) by auto
have 5: \((i > (\text{length } y))\)
  \((i < ((\text{length } z\#zs) \ast (\text{length } y)))\)
  = \((i - (\text{length } y)< (\text{length } zs) \ast (\text{length } y))\)
  by auto
then have 6:
  \(\forall i.((i < ((\text{length } zs) \ast (\text{length } y)))\)
  \(((\text{vec-vec-Tensor } zs y)!i)\)
  = \(f\)
  \((z\#zs)!((i - (\text{length } y)) \div (\text{length } y)))\)
  \((y! (i \mod (\text{length } y)))\))
  using \text{Cons.hyps} by auto
with 5 have \((i > (\text{length } y))\)
  \(((i < ((\text{length } z\#zs) \ast (\text{length } y)))\)
  \(((\text{vec-vec-Tensor } zs y)!((i - (\text{length } y))\)
  = \(f\)
  \((z\#zs)!((i - (\text{length } y)) \div (\text{length } y)))\)

22
\[(y!(i-(\text{length } y) \mod (\text{length } y)))\]
\[= (((i<((\text{length } zs)\ast(\text{length } y))))\]
\[\rightarrow ((\text{vec-vec-Tensor } zs y)!i))\]
\[= f\]
\[(zs!((i-(\text{length } y)) \div (\text{length } y)))\]
\[(y!(i-(\text{length } y)) \mod (\text{length } y)))\]

by auto

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

by auto

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

by auto

then show \textit{thesis} by auto

qed

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

proof-

have 1: \((i = (\text{length } y))\]
\[\rightarrow ((\text{vec-vec-Tensor } (z\#zs) y)!i)\]
\[= (\text{vec-vec-Tensor } zs y)!0\]
using \textit{vec-vec-Tensor-simpl2} by auto

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

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

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

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

with 3 have 5: \((i = \text{length } y)\]
\[\rightarrow ((z\#zs)!(i \ \text{div} \ (\text{length } y))) = (zs!0)\]

23
by auto
have \( \forall i. (i < (\text{length } zs) \ast (\text{length } y)) \)
\[ \rightarrow ((\text{vec-vec-Tensor } (zs) y)!i) = f ((zs)!((i \div (\text{length } y)))
\] (\text{y}!(i \mod (\text{length } y))))

using Cons.hyps by auto
with 4 have 6: \((i = \text{length } y) \rightarrow ((0 < ((\text{length } zs) \ast (\text{length } y)))
\[ \rightarrow (((\text{vec-vec-Tensor } (zs) y)!0)
\] = f ((zs)!0) (y!0))
\[ = f ((i < ((\text{length } zs) \ast (\text{length } y)))
\] \rightarrow (((\text{vec-vec-Tensor } zs y)!i)
\] = f ((zs)!((i \div (\text{length } y)))
\] (\text{y}!(i \mod (\text{length } y))))

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)))
\[ \rightarrow (((\text{vec-vec-Tensor } zs y)!0)
\] = f ((zs)!0) (y!0))
using 7 8 Cons.hyps by auto
with 4 5 8 have \((i = \text{length } y) \rightarrow ((i < ((\text{length } (z\#zs)) \ast (\text{length } y)))
\[ \rightarrow (((\text{vec-vec-Tensor } (zs) y)!0)
\] = f ((zs)!0) (y!0)))
by auto
with 1 2 5 have \((i = \text{length } y) \rightarrow ((i < ((\text{length } (z\#zs)) \ast (\text{length } y)))
\[ \rightarrow (((\text{vec-vec-Tensor } ((z\#zs)) y)!i)
\] = f ((z\#zs)!((i \div (\text{length } y)))
\] (\text{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)))
\[ \rightarrow (((\text{vec-vec-Tensor } ((z\#zs)) y)!i)
\] = f ((z\#zs)!((i \div (\text{length } y)))
\] (\text{y}!(i \mod (\text{length } y))))
by auto
have \((i < (\text{length } y)) \lor (i \geq (\text{length } y)) \)
by auto
with step1 step4 have

\((i < \text{length } (z#zs)) \cdot (\text{length } y)\)
\rightarrow (((\text{vec-vec-Tensor } (z#zs) y)i)
\rightarrow f
((z#zs)! (i \text{ div } (\text{length } y)))(y! (i \text{ mod } (\text{length } y))))

using logic by (metis 6 7 35)
then show ?case by auto
qed

a few more results that will be used later on

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

lemma int-nat-equiv: \((x > 0) \rightarrow (\text{nat } (\text{int } x + -1)+1) = x\)
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)
  \rightarrow 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 \((x > 0) \rightarrow \text{nat } ((\text{int } x) + -1) + 1 = x\)
  by auto
then show ?thesis by auto
qed

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

fix \(j\)
have \(((x#xs)! (k+1) = xs! k)\)
  by auto
have \(j = (k+1) \rightarrow (\text{nat } ((\text{int } j)+-1)) = k\)
  by auto
moreover have \((\text{nat } ((\text{int } j)+-1)) = k\)
  \rightarrow ((\text{nat } ((\text{int } j)+-1)) + 1) = (k +1)
  by auto
moreover have \((j > 0) \rightarrow ((\text{nat } ((\text{int } j)+-1)) + 1) = j)\)

25
using \texttt{int-nat-equiv} by \texttt{(auto)}
moreover have \((k>0) \rightarrow ((\#xs)!k = zs!(\texttt{nat} ((\texttt{int} k)+-1)))\)
using \texttt{Suc-eq-plus1 int-nat-equiv nth-Cons-Suc} by \texttt{(metis)}
from this show \texttt{thesis} by \texttt{auto}
qed

\textbf{lemma} \texttt{row-length-eq:}
\begin{align*}
(mat \quad (\texttt{row-length} \quad (a\#b\#N)) \quad (\texttt{length} \quad (a\#b\#N)) \quad (a\#b\#N)) \\
\rightarrow \\
(\texttt{row-length} \quad (a\#b\#N) = (\texttt{row-length} \quad (b\#N)))
\end{align*}

\textbf{proof–}

have \( (mat \quad (\texttt{row-length} \quad (a\#b\#N)) \quad (\texttt{length} \quad (a\#b\#N)) \quad (a\#b\#N)) \\
\rightarrow (b \in \texttt{set} \quad (a\#b\#M)) \)
by \texttt{auto}
moreover have \( (mat \quad (\texttt{row-length} \quad (a\#b\#N)) \quad (\texttt{length} \quad (a\#b\#N)) \quad (a\#b\#N)) \\
\rightarrow (\texttt{Ball} \quad \texttt{set} \quad (a\#b\#N)) \quad (\texttt{vec} \quad (\texttt{row-length} \quad (a\#b\#N))) \)
using \texttt{mat-def} by \texttt{metis}
moreover have \( (mat \quad (\texttt{row-length} \quad (a\#b\#N)) \quad (\texttt{length} \quad (a\#b\#N)) \quad (a\#b\#N)) \\
\rightarrow (b \in \texttt{set} \quad (a\#b\#N)) \\
\rightarrow (\texttt{vec} \quad (\texttt{row-length} \quad (a\#b\#N)) \quad b) \)
by \texttt{(metis calculation(2))}
then have \( (mat \quad (\texttt{row-length} \quad (a\#b\#N)) \quad (\texttt{length} \quad (a\#b\#N)) \quad (a\#b\#N)) \\
\rightarrow (\texttt{length} \quad b) = (\texttt{row-length} \quad (a\#b\#N)) \)
using \texttt{vec-def} by \texttt{auto}
then have \( (mat \quad (\texttt{row-length} \quad (a\#b\#N)) \quad (\texttt{length} \quad (a\#b\#N)) \quad (a\#b\#N)) \\
\rightarrow (\texttt{row-length} \quad (\texttt{b}\#N)) = (\texttt{row-length} \quad (a\#b\#N)) \)
using \texttt{row-length-def} by \texttt{auto}
then show \texttt{thesis} by \texttt{auto}
qed

The following theorem tells us the relationship between entries of \texttt{vec \_ mat \_ Tensor}
v \(\texttt{M}\) and entries of \(v\) and \(\texttt{M}\) respectively

\textbf{theorem} \texttt{vec-mat-Tensor-elements:}
\begin{align*}
\forall i. \forall j. \\
(((i<(\texttt{length} \quad v)\star (\texttt{row-length} \quad M))) \\
\land (j < (\texttt{length} \quad M))) \\
\land (mat \quad (\texttt{row-length} \quad M) \quad (\texttt{length} \quad M) \quad M) \\
\rightarrow ((\texttt{vec-mat-Tensor} \quad v \quad M)!j!i) \\
= f \quad (v!((i \div (\texttt{row-length} \quad M))) \quad (M)!j!(i \mod (\texttt{row-length} \quad M)))
\end{align*}
apply \texttt{(rule allI)}
apply \texttt{(rule allI)}
proof \texttt{(induct M)}
case \texttt{Nil}
have \texttt{row-length} \[0\]
using \texttt{row-length-def} by \texttt{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 []))) (((M!i mod (row-length []))))
by auto
from this
show ?thesis 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+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 \texttt{Cons.hyps} by \texttt{auto}

have 2: \((\text{row-length } (a\#M)) = (\text{length } a)\)
using \texttt{row-length-def} by \texttt{auto}

then have 3:\(i < (\text{row-length } (a\#M)) \cdot (\text{length } v)\)
\(= (i < (\text{length } a) \cdot (\text{length } v))\)
by \texttt{auto}
have \(a \neq []\)
using \texttt{Cons} by \texttt{auto}

then have 4:
\[\forall i. ((i < (\text{length } a) \cdot (\text{length } v)) \rightarrow ((\text{vec-vec-Tensor } v \ a)!i) = f)
= (v!(i \text{ div } (\text{length } a)))
= (a!(i \text{ mod } (\text{length } a)))\]
using \texttt{vec-vec-Tensor-elements Cons.hyps mult.commute}
by (\texttt{simp add: mult.commute vec-vec-Tensor-elements})

have \((\text{vec-mat-Tensor } v \ (a\#M))!0 = (\text{vec-vec-Tensor } v \ a)\)
using \texttt{vec-mat-Tensor.simps(2)} by \texttt{auto}

with 2 4 have 5:
\[\forall i. ((i < (\text{row-length } (a\#M)) \cdot (\text{length } v)) \rightarrow ((\text{vec-mat-Tensor } v \ (a\#M))!0!i)
= f)
= (v!(i \text{ div } (\text{row-length } (a\#M))))
= ((a\#M)!0!(i \text{ mod } (\text{row-length } (a\#M))))\]
by \texttt{auto}

have \((\text{length } (a\#M)) > 0\)
by \texttt{auto}

with 5 have 6:
\([j = 0] \rightarrow ((\text{vec-mat-Tensor } v \ (a\#M))!j!i)
= f)
= (v!(i \text{ div } (\text{row-length } (a\#M))))
= ((a\#M)!j!(i \text{ mod } (\text{row-length } (a\#M))))\]
by \texttt{auto}

have \(((i < (\text{row-length } (a\#M)) \cdot (\text{length } v))
\land (j < (\text{length } (a\#M))))
\land (\text{mat } (\text{row-length } (a\#M)) \cdot (\text{length } (a\#M)) \cdot (a\#M)) \rightarrow ((\text{vec-mat-Tensor } v \ (a\#M))!j!i)
= f)
= (v!(i \text{ div } (\text{row-length } (a\#M))))
= ((a\#M)!j!(i \text{ mod } (\text{row-length } (a\#M))))\]

\textbf{proof}(cases \(M\))
\textbf{case} \texttt{Nil}
have \((\text{length } (a\#[])) = 1\)

28
by auto
then have $(j < (\text{length (a#[])})) = (j = 0)$
by auto
then have $((\langle i < (\text{row-length (a#[])})* (\text{length v}) \rangle \\
& (j < (\text{length (a#[])}))) \\
& \langle \text{mat (row-length (a#[])) (length (a#[])) (a#[])} \rangle \\
\rightarrow \langle \text{vec-mat-Tensor v (a#[])} \rangle !i) = f$
\hspace{1cm} ((a#[])!j!i mod (row-length (a#[]))))
by auto
using 6 Nil
then show ?thesis using Nil by auto
next
\begin{itemize}
\item \text{case (Cons b N)}
\item \text{have 7: (mat (row-length (a#b#N)) (length (a#b#N)) (a#b#N))} \\
\hspace{1cm} \rightarrow \text{row-length (a#b#N) = (row-length (b#N))}
\hspace{1cm} \text{using row-length-eq bymetis}
\item \text{have 8: (j > 0)} \\
\hspace{1cm} \rightarrow \langle \text{vec-mat-Tensor v (b#N)} !(\text{nat ((int j) + -1)}) \rangle \\
\hspace{1cm} = \langle \text{vec-mat-Tensor v (a#b#N)} \rangle !j
\hspace{1cm} \text{using vec-mat-Tensor.simps(2) using list-int-nat bymetis}
\item \text{have 9: (j > 0)} \\
\hspace{1cm} \rightarrow \langle \text{vec-mat-Tensor v (b#N)} !(\text{nat ((int j) + -1)}) !i \rangle = f$
\hspace{1cm} ((b#N)!(\text{nat ((int j) + -1)})!i mod (row-length (b#N))))
\hspace{1cm} \text{using Cons.hyps Cons mult.commute bymetis}
\item \text{have (j > 0) \rightarrow (\text{nat ((int j) + -1)}) < (\text{length (b#N)})} \\
\hspace{1cm} \rightarrow (\text{nat ((int j) + -1) + 1}) < \text{length (a#b#N)}
\hspace{1cm} \text{by auto}
\item then have \hspace{1cm} (j > 0) \\
\hspace{1cm} \rightarrow \langle \text{nat ((int j) + -1)} \rangle < \text{length (b#N)} = (j < \text{length (a#b#N)})
\hspace{1cm} \text{by auto}
\item then have \hspace{1cm} (j > 0) \\
\hspace{1cm} \rightarrow ((\langle i < (\text{row-length (b#N)})* (\text{length v}) \rangle \\
\hspace{1.4cm} \& (j < \text{length (a#b#N)})) \\
\hspace{1.4cm} \& (\text{mat (row-length (b#N)) (length (b#N)) (b#N)}) \rightarrow \langle \text{vec-mat-Tensor v (b#N)} \rangle !(\text{nat ((int j) + -1)}) !i) = f$
\hspace{1.4cm} ((b#N)!(\text{nat ((int j) + -1)})!i mod (row-length (b#N))))
\hspace{1.4cm} \text{using Cons.hyps Cons mult.commute bymetis}
\item with 8 have \hspace{1cm} (j > 0) \\
\hspace{1cm} \rightarrow ((\langle i < (\text{row-length (b#N)})* (\text{length v}) \rangle \
\hspace{1.4cm} \& (j < \text{length (a#b#N)})) \\
\hspace{1cm} \& (\text{mat (row-length (b#N)) (length (b#N)) (b#N)}) \rightarrow \langle \text{vec-mat-Tensor v (b#N)} \rangle !(\text{nat ((int j) + -1)}) !i) = f$
\hspace{1.4cm} ((b#N)!(\text{nat ((int j) + -1)})!i mod (row-length (b#N))))
\hspace{1cm} \text{using Cons.hyps Cons mult.commute bymetis}
\end{itemize}
\end{document}
\[\begin{align*}
&\land (j < \text{length } (a\#b\#N)) \\
&\land (\text{mat } (\text{row-length } (b\#N)) (\text{length } (b\#N)) (b\#N)) \\
\rightarrow \\
&((\text{vec-mat-Tensor } v (a\#b\#N))!j!i) \\
&= f \\
&(v!(i \text{ div } (\text{row-length } (b\#N)))) \\
&((a\#b\#N)!l!(i \text{ mod } (\text{row-length } (b\#N)))) \\
&\text{by auto} \\
\text{also have } (j > 0) \rightarrow (b\#N)!l!(\text{nat } ((\text{int } j)+1)) = (a\#b\#N)!j \\
&\text{using list-int-nat by metis} \\
\text{moreover have } (j > 0) \rightarrow \\
&(((i < (\text{row-length } (b\#N)))*(\text{length } v)) \\
&\land (j < \text{length } (a\#b\#N)) \\
&\land (\text{mat } (\text{row-length } (a\#b\#N)) (\text{length } (a\#b\#N)) (a\#b\#N)) \\
\rightarrow \\
&((\text{vec-mat-Tensor } v (a\#b\#N))!j!i) \\
&= f \\
&(v!(i \text{ div } (\text{row-length } (b\#N)))) \\
&((a\#b\#N)!l!(i \text{ mod } (\text{row-length } (b\#N)))) \\
&\text{using reduct-matrix by (metis)} \\
\text{moreover have } (\text{mat } (\text{row-length } (a\#b\#N)) (\text{length } (a\#b\#N)) (a\#b\#N)) \\
\rightarrow (\text{row-length } (b\#N)) = (\text{row-length } (a\#b\#N)) \\
&\text{by (metis 7 Cons)} \\
\text{moreover have } 10: (j > 0) \\
\rightarrow \\
&(((i < (\text{row-length } (a\#b\#N)))*(\text{length } v)) \\
&\land (j < \text{length } (a\#b\#N)) \\
&\land (\text{mat } (\text{row-length } (a\#b\#N)) (\text{length } (a\#b\#N)) (a\#b\#N)) \\
\rightarrow \\
&((\text{vec-mat-Tensor } v (a\#b\#N))!j!i) \\
&= f \\
&(v!(i \text{ div } (\text{row-length } (a\#b\#N)))) \\
&((a\#b\#N)!l!(i \text{ mod } (\text{row-length } (a\#b\#N)))) \\
&\text{by (metis calculation(3) calculation(4))} \\
\text{have } (j = 0) \lor (j > 0) \\
&\text{by auto} \\
\text{with } 6 \text{ 10 logic have} \\
&(((i < (\text{row-length } (a\#b\#N)))*(\text{length } v)) \\
&\land (j < \text{length } (a\#b\#N)) \\
&\land (\text{mat } (\text{row-length } (a\#b\#N)) (\text{length } (a\#b\#N)) (a\#b\#N))
\end{align*}\]
\[ \rightarrow ((\text{vec-mat-Tensor } v (a \# b \# N))!j!i) \]
\[ = f \]
\[ (v! (i \text{ div } (\text{row-length } (a \# b \# N)))) \]
\[ ((a \# b \# N)!j! (i \text{ mod } (\text{row-length } (a \# b \# N)))) \]

using Cons by metis

from this show ?thesis by (metis Cons)

qed

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

qed

from this show ?case by auto

qed

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

**Theorem matrix-Tensor-elements:**

**fixes** M1 M2

**shows**

\[ \forall i. \forall j. (((i < (\text{row-length } M1) \cdot (\text{row-length } M2))) \]
\[ \land (j < (\text{length } M1) \cdot (\text{length } M2))) \]
\[ \land (\text{mat } (\text{row-length } M1) \cdot (\text{length } M1) \cdot M1) \]
\[ \land (\text{mat } (\text{row-length } M2) \cdot (\text{length } M2) \cdot M2) \]

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

apply (rule allI)

apply (rule allI)

proof (induct M1)

case Nil

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

using row-length-def by auto

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

by auto

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

\[ \rightarrow \text{False} \]

by auto

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

by auto

moreover have \(((i < ((\text{row-length } []) \cdot (\text{row-length } M2))) \]
\[ \land (j < (\text{length } []) \cdot (\text{length } M2))) \]
\[ \land (\text{mat } (\text{row-length } []) \cdot (\text{length } []) \cdot []) \]
\[ \land (\text{mat } (\text{row-length } M2) \cdot (\text{length } M2) \cdot M2) \]

\[ \rightarrow ([[] \otimes M2]!j!i) = f \]
(\[(j \div \text{length } \mathbb{M})!(i \div \text{row-length } \mathbb{M}2))
(M2!(j \mod \text{length } \mathbb{M})!(i \mod \text{row-length } \mathbb{M}2))

by auto
then show \?case by auto
next
case (\text{Cons } v \ M)
fix a
have 0:(v#M) \otimes \mathbb{M}2 = (\text{vec-mat-Tensor } v \ M2)@((\text{Tensor } \mathbb{M} \ M2))
by auto
then have 1:

\((j < \text{length } \mathbb{M}2) \rightarrow (v#M)!j = (\text{vec-mat-Tensor } v \ M2)!j)\)

using append-simpl vec-mat-Tensor-length by metis
have \((j < \text{length } \mathbb{M}2)) \wedge (\text{mat } \text{row-length } \mathbb{M}2) \text{ (length } \mathbb{M}2) \ M2

\rightarrow ((\text{Tensor } a \ M2)!j)!i = f (a!(i \div \text{row-length } \mathbb{M}2)) (M2)!j!(i \mod \text{row-length } \mathbb{M}2)))

using vec-mat-Tensor-elements by auto
have \((j < \text{length } \mathbb{M}2) \rightarrow (j \div \text{length } \mathbb{M}2) = 0)\)
by auto
then have 2:\((j < \text{length } \mathbb{M}2) \rightarrow (v#M)!(j \div \text{length } \mathbb{M}2) = v)\)
by auto
have \((j < \text{length } \mathbb{M}2) \rightarrow (j \mod \text{length } \mathbb{M}2) = j)\)
by auto
moreover have \((j < \text{length } \mathbb{M}2) \rightarrow (v#M)!(j \mod \text{length } \mathbb{M}2) = (v#M)!j)\)
by auto
have step0:

\((j < \text{length } \mathbb{M}2) \rightarrow ((i < \text{(length } v)\ast\text{(row-length } \mathbb{M}2)))
\wedge (j < \text{length } \mathbb{M}2) \ast ((\text{length } v#M)))
\wedge (\text{mat } \text{row-length } \mathbb{M}2) \text{ (length } \mathbb{M}2) \ M2

\rightarrow ((\text{Tensor } v#M)!j)!i) = f

((v#M)!(j \div \text{length } \mathbb{M}2)!(i \div \text{row-length } \mathbb{M}2)) (M2)!j!(i \mod \text{row-length } \mathbb{M}2)))))

using 2 1 calculation(1) vec-mat-Tensor-elements by auto
have step1:

\((j < \text{length } \mathbb{M}2) \rightarrow ((i < \text{(row-length } v#M)\ast\text{(row-length } \mathbb{M}2)))
\wedge (j < \text{length } v#M)\ast(\text{length } \mathbb{M}2))
\wedge (\text{mat } \text{row-length } (v#M)) \text{ (length } v#M) \ (v#M))
\wedge (\text{mat } \text{row-length } \mathbb{M}2) \text{ (length } \mathbb{M}2) \ M2

\rightarrow ((\text{Tensor } v#M)!j)!i) = f

((v#M)!(j \div \text{length } \mathbb{M}2)!(i \div \text{row-length } \mathbb{M}2)) (M2)!j!(i \mod \text{row-length } \mathbb{M}2)))))

using row-length-def step0 by auto
from 0 have 3:

\((j \geq \text{length } \mathbb{M}2) \rightarrow ((v#M) \otimes \mathbb{M}2)!j = (\mathbb{M} \otimes \mathbb{M}2)!(j - \text{length } \mathbb{M}2))\)
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 } M) (\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 (\text{div}(\text{row-length } M2)))
(M2!((j - (\text{length } M2)) \mod (\text{length } M2))!i (\text{mod}(\text{row-length } M2)))
)
using Cons.hyps by auto

moreover have (\text{mat}(\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M))
\rightarrow (\text{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 (\text{div}(\text{row-length } M2)))
(M2!((j - (\text{length } M2)) \mod (\text{length } M2))!i (\text{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:
\( j \geq (\text{length } M2) \)
→

(((i < (row-length (v#M) ) * (row-length M2))
 ∧ (j < (length (v#M) ) * (length M2))
 ∧ (mat (row-length (v#M) ) (length (v#M) ) (v#M))
 ∧ (mat (row-length M2) (length M2) M2))

→ (((v#M) ⊗ M2)!i) =

\( f \)

((v#M)!(j div (length M2))!(i div (row-length M2)))
(M2!(j mod length M2)!(i mod (row-length M2)))

proof (cases M2)
case Nil

have (0 = ((row-length (v#M) ) * (row-length M2)))

using row-length-def Nil malt-0-right by auto

then have (i < ((row-length (v#M) ) * (row-length M2))) → False

by auto

then show ?thesis by auto

next
case (Cons w N)

fix k

have (k < (length M)) ∧ (k ≥ 1) → M!(k - 1) = (v#M)!k

using not-one-le-zero nth-Cons’ by auto

have (j ≥ (length (w#N))) → (j div (length (w#N))) ≥ 1

using div-le-mono div-self length-0-conv neq-Nil-conv by metis

moreover have (j ≥ (length (w#N))) → (j div (length (w#N))) - 1 ≥ 0

by auto

moreover have (j ≥ (length (w#N)))

→ M!(j div (length (w#N))) - 1

= (v#M)!(j div (length (w#N)))

using calculation(1) not-one-le-zero nth-Cons’ by auto

from this 7 have 9: (j ≥ (length (w#N)))

→ M!(j - (length (w#N))) div (length (w#N))

= (v#M)!(j div (length (w#N)))

using Cons by auto

have 10: (j ≥ (length (w#N)))

→ ((j - (length (w#N))) mod (length (w#N)))

= (j mod (length (w#N)))

using mod-if not-less by auto

with 5 9 have

(j ≥ (length (w#N))) →

((i < (row-length M) * (row-length (w#N))))

∧ (j - (length (w#N)) < (length M) * (length (w#N)))

∧ (mat (row-length (v#M)) (length (v#M)) (v#M))
\( \land (\text{mat} \ (\text{row-length} \ (w \#N)) \ (\text{length} \ (w \#N)) \ (w \#N)) \\\n\rightarrow ((M \otimes (w \#N)))((j - (\text{length} \ (w \#N)))i) \\\n= f \\\n((v \#M[((j \text{ div} \ (\text{length} \ (w \#N)))i \text{ div} \ (\text{row-length} \ (w \#N)))) \\\n((w \#N)((j \text{ mod} \ (\text{length} \ (w \#N)))i \text{ mod} \ (\text{row-length} \ (w \#N)))))) \\\n\text{using} \ \text{Cons} \ \text{by} \ \text{auto} \\\n\text{then have} \ (j \geq \ (\text{length} \ (w \#N))) \rightarrow \\\n(((i < ((\text{row-length} \ M) + (\text{row-length} \ (w \#N)))) \\\n\land (j < (\text{length} \ (v \#M))) + (\text{length} \ (w \#N)))) \\\n\land (\text{mat} \ (\text{row-length} \ (v \#M)) \ (\text{length} \ (v \#M)) \ (v \#M)) \\\n\land (\text{mat} \ (\text{row-length} \ (w \#N)) \ (\text{length} \ (w \#N)) \ (w \#N)) \rightarrow ((((M \otimes (w \#N)))((j - (\text{length} \ (w \#N)))i) \\\n= f \\\n((v \#M)[(j \text{ div} \ (\text{length} \ (w \#N)))i \text{ div} \ (\text{row-length} \ (w \#N)))) \\\n((w \#N)[(j \text{ mod} \ (\text{length} \ (w \#N)))i \text{ mod} \ (\text{row-length} \ (w \#N)))))) \\\n\text{using} \ 6 \ \text{by} \ \text{auto} \\\n\text{then have} \ 11: \ (j \geq \ (\text{length} \ (w \#N))) \rightarrow \\\n(((i < ((\text{row-length} \ M) + (\text{row-length} \ (w \#N)))) \\\n\land (j < (\text{length} \ (v \#M))) + (\text{length} \ (w \#N)))) \\\n\land (\text{mat} \ (\text{row-length} \ (v \#M)) \ (\text{length} \ (v \#M)) \ (v \#M)) \\\n\land (\text{mat} \ (\text{row-length} \ (w \#N)) \ (\text{length} \ (w \#N)) \ (w \#N)) \rightarrow ((((M \otimes (w \#N)))((j - (\text{length} \ (w \#N)))i) \\\n= f \\\n((v \#M)[(j \text{ div} \ (\text{length} \ (w \#N)))i \text{ div} \ (\text{row-length} \ (w \#N)))) \\\n((w \#N)[(j \text{ mod} \ (\text{length} \ (w \#N)))i \text{ mod} \ (\text{row-length} \ (w \#N)))))) \\\n\text{using} \ 3 \ \text{by} \ \text{auto} \\\n\text{have} \ (j \geq \ (\text{length} \ (w \#N))) \rightarrow \\\n(((i < ((\text{row-length} \ (v \#M)) + (\text{row-length} \ (w \#N)))) \\\n\land (j < (\text{length} \ (v \#M))) + (\text{length} \ (w \#N)))) \\\n\land (\text{mat} \ (\text{row-length} \ (v \#M)) \ (\text{length} \ (v \#M)) \ (v \#M)) \\\n\land (\text{mat} \ (\text{row-length} \ (w \#N)) \ (\text{length} \ (w \#N)) \ (w \#N)) \rightarrow ((((M \otimes (w \#N)))((j - (\text{length} \ (w \#N)))i) \\\n= f \\\n((v \#M)[(j \text{ div} \ (\text{length} \ (w \#N)))i \text{ div} \ (\text{row-length} \ (w \#N)))) \\\n((w \#N)[(j \text{ mod} \ (\text{length} \ (w \#N)))i \text{ mod} \ (\text{row-length} \ (w \#N)))))) \\\n\text{proof}(\text{cases} \ M) \\\n\text{case} \ \text{Nil} \\\n\text{have} \ \text{Nil0}:(\text{length} \ (v \#[])) = 1 \\\n\text{by} \ \text{auto} \\\n\text{then have} \ \text{Nil1}: \ (j < ((\text{length} \ (v \#[])) + (\text{length} \ (w \#N)))) = (j < (\text{length} \ (w \#N))) \\\n\text{by} \ \text{metis} \ \text{Nil} \ \text{nat-mul-1} \\\n\text{have} \ (\text{row-length} \ (v \#[])) = (\text{length} \ v) \\\n\text{using} \ \text{row-length-def} \ \text{by} \ \text{auto} \\\n\text{then have} \ \text{Nil2}: }
\[(i < ((\text{row-length } (v \# M)) \times (\text{row-length } (w \# N))))
\]
\[= (i < ((\text{length } v) \times (\text{row-length } (w \# N))))\]
using Nil by auto

then have \((j < (\text{length } (w \# N))) \rightarrow (j \div (\text{length } (w \# N))) = 0\)
by auto

from this have Nil3:
\[(j < (\text{length } (w \# N))) \rightarrow (v \# M)! (j \div (\text{length } (w \# N))) = v\]
using Nil by auto

then have Nil4:
\[(j < (\text{length } (w \# N))) \rightarrow (j \mod (\text{length } (w \# N))) = j\]
by auto

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

from vec-mat-Tensor-elements have
\[((i < ((\text{length } v) \times (\text{row-length } (w \# N))))
\& (j < (\text{length } (w \# N))))
\& (\text{mat } (\text{row-length } (w \# N)) (\text{length } (w \# N)) (w \# N))

\rightarrow (((\text{vec-mat-Tensor } v (w \# N))! j! i)
\quad = f
\quad (v!(i \div \text{row-length } (w \# N)))
\quad (w \# N)! (i \mod (\text{row-length } (w \# N))))\]
by metis

then have
\[((i < ((\text{row-length } (v \# M)) \times (\text{row-length } (w \# N))))
\& (j < ((\text{length } (v \# M)) \times (\text{length } (w \# N))))
\& (\text{mat } (\text{row-length } (w \# N)) (\text{length } (w \# N)) (w \# N))

\rightarrow (((v \# M)! (\text{length } (w \# N))! (j \div (\text{row-length } (w \# N))))
\quad (w \# N)! (j \mod (\text{length } (w \# N)))))\]
using Nil1 Nil2 Nil by auto

then have
\[((i < ((\text{row-length } (v \# M)) \times (\text{row-length } (w \# N))))
\& (j < ((\text{length } (v \# M)) \times (\text{length } (w \# N))))
\& (\text{mat } (\text{row-length } (w \# N)) (\text{length } (w \# N)) (w \# M))
\& (\text{mat } (\text{row-length } (w \# N)) (\text{length } (w \# N)) (w \# M))

\rightarrow (((v \# M)! (\text{length } (w \# N))! (j \div (\text{row-length } (w \# N))))
\quad (w \# N)! (j \mod (\text{length } (w \# N))))\]
using Nil3 Nil4 Nil5 Nil by auto

then have
\[((i < ((\text{row-length } (v \# M)) \times (\text{row-length } (w \# N))))
\& (j < ((\text{length } (v \# M)) \times (\text{length } (w \# N))))
\& (\text{mat } (\text{row-length } (v \# M)) (\text{length } (v \# M)) (v \# M))
\& (\text{mat } (\text{row-length } (w \# N)) (\text{length } (w \# N)) (w \# M))

\rightarrow (((v \# M)! (\text{length } (w \# N))! (j \div (\text{row-length } (w \# N))))
\quad (w \# N)! (j \mod (\text{length } (w \# N))))\]
using Nil3 Nil4 Nil5 Nil by auto

36
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 div (length M2))!(i div (row-length M2)))
	  (M2!(j mod length M2))!(i mod (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: \(\neg((x \div z) < y)\)
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-mul-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: \(\div z \geq (y \ast z) \div z\)
  by auto
then have \((y \ast z) \div z \leq 1\)
  by auto
with 6 have 1: \(1 \geq y\)
  using 1 3 assms div-self less-nat-zero-code multi-zero-left
  multi.commute mod-div-mult-eq
  by auto
then have 7: \((y = 0) \vee (y = 1)\)
  by auto
have \((y = 0) \Longrightarrow x < 0\)
  using assms by auto
moreover have \(x \geq 0\)
  by auto
then have 8: \((y = 0) \Longrightarrow False\)
  using calculation less-nat-zero-code by auto
moreover have \((y = 1) \Longrightarrow (x < z)\)
  using assms by auto
then have \((y = 1) \Longrightarrow (x \div z) = 0\)
  by (metis div-less)
then have \((y = 1) \Longrightarrow (x \div z) < y\)
  by auto
then have \((y = 1) \Longrightarrow False\)
  using 0 by auto
then show False using 7 8 by auto
qed

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

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

**Lemma** col-vec-mat-Tensor-prelim:
\[ \forall j. \langle j < (\text{length } M) \rightarrow \text{col (vec-mat-Tensor v M) } j = \text{vec-vec-Tensor v (col M j)} \rangle \]

**Unfolding** col-def
**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) = \((\text{vec-vec-Tensor v w})\#(\text{vec-mat-Tensor v N})\)
**Using** vec-mat-Tensor.simps Cons by auto
**Then show** ?case
**Proof** (cases j)
**Case** 0
**Have** vec-mat-Tensor v (w#N)!0 = \((\text{vec-vec-Tensor v w})\)
**By** auto
**Then show** ?thesis using 0 by auto
**Next**
**Case** (Suc k)
**Have** vec-mat-Tensor v (w#N)!j = \((\text{vec-mat-Tensor v N})!(k)\)
**Using** Cons Suc by auto
**Moreover have** j < length \((w\#N) \implies k < length N \)
**Using** Suc by (metis length-Suc-conv not-less-eq)
**Moreover then have** k < length \(N\)
**implies** \((\text{vec-mat-Tensor v N})!k = \text{vec-vec-Tensor v (N\!k)}\)
**Using** Cons.simps 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** \( \forall j. \langle (j < (\text{length } M1)\ast(\text{length } M2)) \\wedge (\text{mat (row-length } M1) \ast(\text{length } M1) \ast M1) \wedge (\text{mat (row-length } M2) \ast(\text{length } M2) \ast M2) \rightarrow \text{col (M1 } \otimes M2) j = \text{vec-vec-Tensor} (\text{col } M1 (j \div\text{ length } M2)) (\text{col } M2 (j \mod\text{ length } M2)) \rangle \)
**Apply** (rule allI)
proof \( \text{induct } M_1 \)

case Nil

show ?case using Nil by auto

next

case (Cons \( v \) \( M \))

have \( j < (\text{length } (v\#M)) \times (\text{length } M_2) \)
\( \land \) mat (\( \text{row-length } (v \# M) \)) (\( \text{length } (v \# M) \)) (\( v \# M \))
\( \land \) mat (\( \text{row-length } M_2 \)) (\( \text{length } M_2 \)) \( M_2 \) \( \Rightarrow \)
(\( \text{col } (v \# M \otimes M_2) \) \( j \))
\( = \) vec-vec-Tensor
(\( \text{col } (v \# M) (j \div \text{length } M_2) \))
(\( \text{col } M_2 (j \mod (\text{length } M_2)) \))

proof−

fix \( k \)

assume \( 0 < j < (\text{length } (v\#M)) \times (\text{length } M_2) \)
\( \land \) mat (\( \text{row-length } (v \# M) \)) (\( \text{length } (v \# M) \)) (\( v \# M \))
\( \land \) mat (\( \text{row-length } M_2 \)) (\( \text{length } M_2 \)) \( M_2 \)

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

by (metis reduct-matrix)

have \( j < (1 + \text{length } M) \times \text{length } M_2 \)

using 0 by auto

then have \( j < \text{length } M_2 + \text{length } M \times \text{length } M_2 \)

by auto

then have \( 2 : j \geq \text{length } M_2 \)

\( \Rightarrow \) \( j - \text{length } M_2 < \text{length } M \times \text{length } M_2 \)

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 \( 3 : (v\#M) \otimes M_2 = (\text{vec-mat-Tensor } v \otimes M_2) @ (M \otimes M_2) \)

using Tensor.simps by auto

have (\( \text{col } ((v\#M) \otimes M_2) \) \( j \)) = (\( \text{col } ((\text{vec-mat-Tensor } v \otimes M_2) @ (M \otimes M_2)) \) \( j \))

using col-def by auto

then have \( j < \text{length } (\text{vec-mat-Tensor } v \otimes M_2) \)

\( \Rightarrow \) \( \text{col } ((v\#M) \otimes M_2) \) \( j \) = (\( \text{col } (\text{vec-mat-Tensor } v \otimes M_2) \) \( j \))

unfolding col-def using append-simpl by auto

then have \( 4 : j < \text{length } M_2 \)

\( \Rightarrow \) \( \text{col } ((v\#M) \otimes M_2) \) \( j \) = (\( \text{col } (\text{vec-mat-Tensor } v \otimes M_2) \) \( j \))

using vec-mat-Tensor-length by simp

then have \( j < \text{length } M_2 \)

\( \Rightarrow \) \( \text{col } (\text{vec-mat-Tensor } v \otimes M_2) \) \( j \)

\( = \) vec-vec-Tensor \( v \) \( (\text{col } M_2 \) \( j \))

using col-vec-mat-Tensor by auto

then have

\( j < \text{length } M_2 \)

\( \Rightarrow \)
(\( \text{col } (\text{vec-mat-Tensor } v \otimes M_2) \) \( j \))
\( = \) vec-vec-Tensor
((\( v\#M \) \( ! (j \div \text{length } M_2) \))
(\( \text{col } M_2 (j \mod (\text{length } M_2)) \))
by auto
then have step-1: \( j < \text{length } M2 \implies (\text{col } ((v\#M) \otimes M2) \ j) = \text{vec-vec-Tensor} \ ((v\#M)!((j \ div \ \text{length } M2)) \ (\text{col } M2 \ (j \ mod \ (\text{length } M2)))) \)

using 4 by auto
have 4: \( j \geq \text{length } M2 \implies (\text{col } ((v\#M) \otimes M2) \ j) = (M \otimes M2)!((j - (\text{length } M2)) \)

unfolding col-def using 3 append-simpl2 vec-mat-Tensor-length by metis
then have 5: \( j \geq \text{length } M2 \implies \text{col } (M \otimes M2) \ (j - \text{length } M2) = \text{vec-vec-Tensor} \ (\text{col } M \ ((j - \text{length } M2) \ div \ \text{length } M2)) \ (\text{col } M2 \ ((j - \text{length } M2) \ mod \ \text{length } M2)) \)

using 1 0 2 Cons by auto
then have 6: \( j \geq \text{length } M2 \implies (j - \text{length } M2) \ div \ (\text{length } M2) + 1 = j \ div \ (\text{length } M2) \)

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

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

using 5 by auto
moreover have \( k < (\text{length } M) \implies (\text{col } M \ k) = (\text{col } (v\#M) \ (k+1)) \)

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

proof—
assume temp: \( j \geq \text{length } M2 \)
have \( j - (\text{length } M2) < (\text{length } M)+(\text{length } M2) \)
using 2 temp by auto
then have \( (j - (\text{length } M2)) \ div \ (\text{length } M2) < (\text{length } M) \)

using div-right-inapp mult.commute by metis
moreover have
\[
(j - (\text{length } M2)) \div (\text{length } M2) < (\text{length } M) \\
\quad \quad \rightarrow \ (\text{col } M ((j - (\text{length } M2)) \div (\text{length } M2))) \\
\quad \quad \quad = (\text{col } (v \# M) ((j - (\text{length } M2)) \div (\text{length } M2) + 1)))
\]

unfolding col-def by auto

ultimately have temp1:
\[
(\text{col } (v \# M) (((j - \text{length } M2) \div \text{length } M2) + 1)) \\
\quad \quad = (\text{col } M (((j - \text{length } M2) \div \text{length } M2))))
\]

by auto

then have (\text{col } (v \# M) (((j - \text{length } M2) \div \text{length } M2) + 1)) \\
\quad \quad = (\text{col } (v \# M) (j \div \text{length } M2))

using 6 temp by auto

then show \( \text{thesis} \) using temp1 7 by (metis temp)

qed

then have \( j \geq \text{length } M2 \implies \text{col } ((v \# M) \otimes M2) j \)
\[
\quad \quad = \text{vec-vec-Tensor } (\text{col } (v \# M) (j \div \text{length } M2)) \\
\quad \quad \quad \quad = (\text{col } M2 (j \mod \text{length } M2))
\]

using col-def 4 by metis

then show \( \text{thesis} \)

using step-1 col-def le-refl nat-less-le nat-neq-iff

by (metis)

qed

then show \( \text{?thesis} \) using step-1 col-def le-refl nat-less-le nat-neq-iff

by (metis)

qed

lemma row-Cons:row \( (v \# M) i = (v!i)\#(\text{row } M i) \)

unfolding row-def map-def by auto

lemma row-append:row \( (A@B)i = (\text{row } A i)@(\text{row } B i) \)

unfolding row-def map-append by auto

lemma row-empty:row \([\,] i = [\,] \)

unfolding row-def by auto

lemma vec-vec-Tensor-right-empty:vec-vec-Tensor \( x [\,] = [\,] \)

using vec-vec-Tensor.simps times.simps length-0-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 \implies [\,[i] = [\] \\

by auto

lemma row-vec-mat-Tensor-prelim:
\[
\forall i. \\
\quad ((i < (\text{length } v)) \land (\text{row-length } M)) \land (\text{mat nr } (\text{length } M) M) \\
\quad \quad \rightarrow \text{row } (\text{vec-mat-Tensor } v M) i
\]

42
apply (rule allI)

proof (induct M)

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

next

  case (Cons w N)
  have row (vec-mat-Tensor v (w#N)) i = row ((vec-vec-Tensor v w)#(vec-mat-Tensor v N)) i
    using vec-mat-Tensor.simps by auto
  then have 1:... = ((vec-vec-Tensor v w)!i)#(row (vec-mat-Tensor v N) i)
    using row-Cons by auto
  have 2:row-length (w#N) = length w
    using row-length-def by auto
  then have 3:(mat nr (length (w#N)) (w#N)) \(\Rightarrow\) nr = length w
    using hd-in-set list.distinct(1) mat-uniqueness matrix-row-length by metis
  then have \(((i < (length v)\ast(row-length (w#N)))
    \land (mat nr (length (w#N)) (w#N))
    \Rightarrow row (vec-mat-Tensor v (w#N)) i = times
                    (v!(i div row-length (w#N)))
                    (row (w#N) (i mod row-length (w#N))))

proof –

  assume \assms: i < (length v)\ast(row-length (w#N))
    \land (mat nr (length (w#N)) (w#N))

  show ?thesis

  proof (cases N)

    case Nil
    have row (vec-mat-Tensor v (w#N)) i = [(vec-vec-Tensor v w)!i]
      using vec-mat-Tensor.simps Nil row-empty by auto
    then show ?thesis

    proof (cases w)

      case Nil
      have (vec-vec-Tensor v w) = []
        using Nil vec-vec-Tensor-right-empty by auto
      moreover have (length v)\ast(row-length (w#N)) = 0
        using Nil row-length-def by auto
      then have [(vec-vec-Tensor v [])!i] = []
        using \assms less-nat-zero-code by metis
      ultimately show \thesis

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

next

  case (Cons a w1)
  have 1:w \neq []
    using Cons by auto
  then have i < (length v)\ast(length w)
    using \assms row-length-def by auto
  then have (vec-vec-Tensor v w)!i
\[ f = v! \{ i \text{ div} (\text{length} w) \} \] (w! \{ i \text{ mod} (\text{length} w) \} ) \]

using \textit{vec-vec-Tensor-elements} 1 all/ by auto

then have \((\text{row} (\text{vec-mat-Tensor} v (w\#N)) \cdot i) = \text{times} \)
\((v! \{ i \text{ div} \text{row-length} (w\#N) \}) \) (row \((w\#N) \cdot i \text{ mod} (\text{length} w) \)) \)

using \textit{Cons vec-mat-Tensor.simps row-def row-length-def 2 Nil row-Cons \text{row-empty times.simps(1) times.simps(2) by metis \text{then show \?thesis using row-def 2 by metis \text{qed next case (Cons w1 N1)}} \)}

have \text{Cons-0:row-length} N \cdot \text{length} w1
using \textit{Cons row-length-def by auto} have \text{mat nr} \cdot \text{length} (w\#w1\#N1) \cdot (w\#w1\#N1)
using \textit{assms Cons by auto} then have \text{Cons-1:}
\text{mat} \cdot \text{row-length} (w\#w1\#N1) \cdot \text{length} (w\#w1\#N1) \cdot (w\#w1\#N1)
\begin{align*}
\text{by (metis matrix-row-length)}
\text{then have Cons-2:}
\text{mat} \cdot \text{row-length} (w1\#N1) \cdot \text{length} (w1\#N1) \cdot (w1\#N1)
\end{align*}
\begin{align*}
\text{by (metis reduct-matrix)}
\text{then have Cons-3:}
\text{length} w1 \cdot \text{length} w
\end{align*}
\begin{align*}
\text{using Cons-1 unfolding mat-def row-length-def Ball-def vec-def}
\text{by (metis 2 Cons-0 Cons-1 local.Cons row-length-eq)}
\text{then have Cons-4:}
\text{mat nr} \cdot \text{length} (w1\#N1) \cdot (w1\#N1)
\end{align*}
\begin{align*}
\text{using 3 Cons-2 assms hd-conv-nth list.distinct(1) nth-Cons-0 row-length-def}
\text{by metis}
\text{moreover have i \cdot \text{less-nat-zero-code}}
\text{ultimately have Cons-5:row} \cdot \text{vec-mat-Tensor} v \cdot N \cdot i = \text{times}
\end{align*}
\begin{align*}
\text{\,(v! \{ i \text{ div} \text{row-length} N \})}
\end{align*}
\begin{align*}
\text{\,(row N \cdot i \text{ mod} \text{row-length} N)}
\end{align*}
\begin{align*}
\text{using Cons Cons.hyps by auto}
\text{then show \?thesis}
\text{proof(cases w)}
\text{case Nil}
\text{have} \,(\text{vec-vec-Tensor} v \cdot w) = []
\text{using Nil vec-vec-Tensor-right-empty by auto}
\text{moreover have} \,(\text{length} v) \cdot (\text{row-length} (w\#N)) = 0
\text{using Nil row-length-def by auto}
\text{then have} \,[\text{vec-vec-Tensor} v \cdot []!i] = []
\text{using assms by (metis less-nat-zero-code)}
\end{align*}
ultimately show \( ?thesis \)
using \texttt{vec-vec-Tensor.simps row-empty Nil assms}
by (metis list.distinct(1))

next
case (Cons a w2)
  have 1\(\vdash w \neq []\)
    using Cons by auto
  then have \(i < (\text{length } v) \ast (\text{length } w)\)
    using assms row-length-def by auto
  then have ConsCons-2:
    \((\text{vec-vec-Tensor } v \ w)!i = f\)
    \((v!(i \div (\text{length } w)))\)
    \((w!(i \mod (\text{length } w)))\)
    using vec-vec-Tensor-elements 1 allI by auto
  moreover have
    times
      \((v!(i \div \text{row-length } (w \# N)))\)
      \((\text{row } (w \# N) (i \mod \text{row-length } (w \# N)))\)
      = \((f\)
      \((v!(i \div (\text{length } w)))\)
      \((w!(i \mod (\text{length } w))))\)
      \(\#(\text{times } (v ! (i \div \text{row-length } N)))\)
      \((\text{row } N (i \mod \text{row-length } N)))\)

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 \text{row-length } (w \# N)))
    = (w!(i \mod (row-length (w \# N))))
    \(\#(\text{row } N (i \mod \text{row-length } (w \# N)))\)
    unfolding row-def by auto
  then have ...
    = (w!(i \mod (\text{length } w))))
    \(\#(\text{row } N (i \mod \text{row-length } N)))\)
    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
    \((v!(i \div \text{row-length } (w \# N)))\)
    \((\text{row } N (i \mod \text{row-length } (w \# N)))\)
    \(\#(\text{times } (v ! (i \div \text{row-length } (w \# N))))\)
    \((\text{row } N (i \mod \text{row-length } N)))\)
    by auto
  then have ...
    = \((f\)
    \((v!(i \div \text{length } w)))\)
    \((w!(i \mod (\text{length } w))))\)
    \(\#(\text{times } (v ! (i \div \text{row-length } N)))\)
(row N (i mod row-length N)))

using 3 Cons-3 assms temp row-length-def by auto
then show ?thesis using times.simps 2 row-Cons temp by metis
qed
then show ?thesis using Cons-5 ConsCons-2 1
  row-Cons vec-mat-Tensor.simps(2) by (metis)
qed
qed
qed
then show ?case by auto
qed

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

lemma row-formula:
fixes M1 and M2
shows \( \forall i. ((i < (\text{row-length } M1) \times (\text{row-length } M2)) \land (\text{mat} (\text{row-length } M1) (\text{length } M1) M1) \land (\text{mat} (\text{row-length } M2) (\text{length } M2) M2) \rightarrow \text{row} ((M1 \otimes M2) i) = \text{vec-vec-Tensor} (\text{row } M1 (i \div \text{row-length } M2), \text{row } M2 (i \mod \text{row-length } M2))) \)

apply rule allI
proof (induct M1)
case Nil
  show ?case using Nil by (metis less-nat-zero-code mult-0 row-length-Nil)
next
case (Cons v M)
  have \( ((i < (\text{row-length } (v\#M)) \times (\text{row-length } M2)) \land (\text{mat} (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M)) \land (\text{mat} (\text{row-length } M2) (\text{length } M2) M2) \rightarrow \text{row} ((v\#M) \otimes M2) i = \text{vec-vec-Tensor} (\text{row } (v\#M) (i \div \text{row-length } M2), \text{row } M2 (i \mod \text{row-length } M2))) \)
    by auto
proof-
  assume assms:
    \( (i < (\text{row-length } (v\#M)) \times (\text{row-length } M2)) \land (\text{mat} (\text{row-length } (v\#M)) (\text{length } (v\#M)) (v\#M)) \land (\text{mat} (\text{row-length } M2) (\text{length } M2) M2) \)
  have 0:i < (length v)\times(\text{row-length } M2)
    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 M2) i = row ((vec-mat-Tensor v M2)\otimes(M \otimes M2)) i
    by auto
  then have 2:... = (row (vec-mat-Tensor v M2) i)\otimes(row (M \otimes M2) i)
    using row-append by auto

46
then show \(?thesis\)
proof(cases \(M\))
case \(Nil\)
  have \(\text{row } ((v\#M) \otimes M2)\ i = (\text{row } \text{vec-mat-Tensor} v\ M2)\ i\)
    using \(Nil\) 2 by auto
  moreover have \(\text{row } (\text{vec-mat-Tensor} v\ M2)\ i = \text{times}\ (v! (i\ \text{div} \text{row-length} M2))\ (\text{row } M2\ (i\ \text{mod} \text{row-length} M2))\)
    using \(\text{row-vec-mat-Tensor-prelim}\ \text{assms}\ 0\) by auto
ultimately show \(?thesis\) using \(\text{vec-vec-Tensor-def}\ \text{Nil append-Nil2\ vec-vec-Tensor.simps(1)}\)
  \(\text{vec-vec-Tensor.simps(2) row-Cons\ row-empty}\) by (metis)
next
case \((\text{Cons } w\ N)\)
  have \(\text{Cons-Cons-1}\):\(\text{mat}\ (\text{row-length} M)\ (\text{length} M)\ M\)
    using \(\text{assms}\ \text{reduct-matrix}\) by auto
  then have \(\text{row-length } (w\#N) = \text{row-length } (v\#M)\)
    using \(\text{assms}\ \text{Cons}\ \text{unfolding}\ \text{mat-def}\ \text{Ball-def vec-def}\)
    using \(\text{append-Cons}\ \text{hd-in-set list.distinct(1)}\)
    \(\text{rotate1.simps(2) set-rotate1}\)
    by auto
  then have \(\text{Cons-Cons-2}\):\(\text{i} < (\text{row-length} M)\ast(\text{row-length} M2)\)
    using \(\text{assms}\ \text{Cons}\) by auto
  then have \(\text{Cons-Cons-3}\):\(\text{row } (M \otimes M2)\ i = \text{vec-vec-Tensor}\ (\text{row } M\ (i\ \text{div} \text{row-length} M2))\ (\text{row } M2\ (i\ \text{mod} \text{row-length} M2))\)
    using \(\text{Cons.hyps}\ \text{Cons-Cons-1}\ \text{assms}\) by auto
  moreover have \(\text{row } (\text{vec-mat-Tensor} v\ M2)\ i = \text{times}\ (v! (i\ \text{div} \text{row-length} M2))\ (\text{row } M2\ (i\ \text{mod} \text{row-length} M2))\)
    using \(\text{row-vec-mat-Tensor-prelim}\ \text{assms}\ 0\) by auto
  then have \(\text{row } ((v\#M) \otimes M2)\ i = \text{times}\ (v! (i\ \text{div} \text{row-length} M2))\ (\text{row } M2\ (i\ \text{mod} \text{row-length} M2))\)
    using \(\text{cons}\ \text{Cons-Cons-3}\) by auto
  moreover have \(\ldots = (\text{vec-vec-Tensor}\ ((v! (i\ \text{div} \text{row-length} M2)))\ #(\text{row } M\ (i\ \text{div} \text{row-length} M2)))\ (\text{row } M2\ (i\ \text{mod} \text{row-length} M2)))\)
    using \(\text{vec-vec-Tensor.simps(2)}\) by auto
  moreover have \(\ldots = (\text{vec-vec-Tensor}\ \text{row } (v\#M)\ (i\ \text{div} \text{row-length} M2))\ (\text{row } M2\ (i\ \text{mod} \text{row-length} M2)))\)
    using \(\text{row-Cons}\) by (metis)
ultimately show \(?thesis\) by (metis)
qed

then show case by auto

lemma effective-row-formula:
  fixes M1 and M2
  assumes i < (row-length M1) *(row-length M2)
    and (mat (row-length M1) (length M1) M1)
    and (mat (row-length M2) (length M2) M2)
  shows row (M1 ⊗ M2) i
    = vec-vec-Tensor
      (row M1 (i div row-length M2))
      (row M2 (i mod row-length M2))
  using assms row-formula by auto

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

lemma trans-impl:
  (∀ i j. (P i j −→ Q i j)) ∧ (∀ i j. (Q i j −→ R i j))
  ⇒ (∀ i j. (P i j −→ R i j))
  by auto

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

lemma mod-less-divisor:
  (¬((a::nat) < b)) ⇒ (a ≥ b)
  by auto

lemma not-null:
  xs ≠ [] ⇒ ∃ y ys. xs = y#ys
  by (metis neq-Nil-conv)

lemma (y::nat) ≠ 0 ⇒ (x mod y) < y
  using mod-less-divisor by auto

lemma mod-prop1:
  ((a::nat) mod (b*c)) mod c = (a mod c)
proof (cases c = 0)
  case True
  have b*c = 0
    by (metis True mult-0-right)
  then have (a::nat) mod (b*c) = a

48
by auto
then have \(((a::nat) \mod (b*c)) \mod c = a \mod c\)
by auto

then show \(?thesis\) by auto
next
case False
let \(?x = (a::nat) \mod (b*c)\)
let \(?z = ?x \mod c\)
have \(\exists m. a = m*(b*c) + ?x\)
  by (metis div-mult-mod-eq)
then obtain \(m1\) where \(a = m1*(b*c) + ?x\)
  by auto
then have \(?x = (a - m1*(b*c))\)
  by auto
then have \(\exists m. (?x = m*c + ?z)\)
  using mod-div-decomp by blast
then obtain \(m\) where \( (?x = m*c + ?z)\)
  by auto
then have \(a - m1*(b*c) = m*c + ?z\)
  using \((a \mod (b \times c)) = a - m1 \times (b \times c)\) by (metis)
then have \(a = m1*b*c + m*c + ?z\)
  using \((a = m1 \times (b \times c) + a \mod (b \times c)) \mod (b \times c)\)
  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 \(\exists n. a = n*(c) + ?y\)
  by (metis \(1 \mod (b \times c) = m \times c + a \mod (b \times c) \mod c\) mod-mult-self3)
then obtain \(n\) where \(a = n*(c) + ?y\)
  by auto
with \(1\) have \((m1*b + m)*c + ?z = n*c + ?y\)
  by auto
then have \((m1*b + m)*c - (n*c) = ?y - ?z\)
  by auto
then have \((m1*b + m - n)*c = (?y - ?z)\)
  by (metis diff-mult-distrib2 mult.commute)
then have \(c \ dvd \ (?y - ?z)\)
  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 $\text{thesis using False}$
by (metis 1 mod-add-right-eq mod-mult-self2 add.commute mult.commute)

qed

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

next
case False
have $F-1: (b > 0) \land (c > 0)$
  using False by auto
have $\exists x. a = x*(b*c) + (a \mod (b*c))$
  using mod-div-decomp by blast
then obtain $x$ where $a = x*(b*c) + (a \mod (b*c))$
  by auto
then have $a \div c = ((x*(b*c)) \div c) + ((a \mod (b*c)) \div c)$
  using div-add1-eq mod-add-self1 mod-add-self2
  mod-by-0 mod-div-trivial mod-prop1 mod-self
  by (metis)
then have $a \div c = (((x*b)\ast c) \div c) + ((a \mod (b*c)) \div c)$
  by auto
then have $F-2: a \div c = (x*b) + ((a \mod (b*c)) \div c)$
  by (metis $F-1$ nonzero-mult-div-cancel-left mult.commute neq0-conv)
have $\exists y. a \div c = (y*b) + ((a \div c) \mod b)$
  by (metis add.commute mod-div-mult-eq)
then obtain $y$ where $a \div c = (y*b) + ((a \div c) \mod b)$
  by auto
with $F-2$ have $F-3: (x*b) + ((a \mod (b*c)) \div c) = (y*b) + ((a \div c) \mod b)$
  by auto
then have $(x*b) - (y*b) = ((a \div c) \mod b) - ((a \mod (b*c)) \div c)$
  by auto
then have $(x - y)*b = ((a \div c) \mod b) - ((a \mod (b*c)) \div c)$
  by (metis diff-mult-distrib2 mult.commute)
then have $F-4:b \text{ dvd } ( ((a \div c) \mod b) - ((a \mod (b*c)) \div c) )$
by (metis dvd-eq-mod-eq-0 mod-mult-self1-is-0 mult.commute)
have F-5: \( b > ((a \div c) \mod b) \)
  by (metis F-1 mod-less-divisor)
have \( b \cdot c > (a \mod (b \cdot c)) \)
  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-ineq nonzero-mult-cancel-right neq0-conv)
then have \( b > (a \mod (b \cdot c)) \div c \)
  by (metis calculation div-right-ineq mult.commute)
with F-4 F-5
have F-6: \( ((a \mod (b \cdot c)) \div c) = 0 \)
  using less-imp-diff-less nat-dvd-not-less by blast
with F-3 have \((y \cdot b) - (x \cdot b)\) = \( ((a \div c) \mod b) - ((a \div c) \mod b) \)
  by auto
then have \((y - x) \cdot b = ((a \mod (b \cdot c)) \div c) - ((a \div c) \mod b)\)
  by (metis diff-mult-distrib2 mult.commute)
then have F-7: \( b \div ( ((a \mod (b \cdot c)) \div c) - ((a \div c) \mod b) ) \)
  by (metis dvd-eq-mod-eq-0 mod-mult-self1-is-0 mult.commute)
have F-8: \( b > ((a \div c) \mod b) \)
  by (metis F-1 mod-less-divisor)
then 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-ineq nonzero-mult-cancel-right neq0-conv)
then have \( b > (a \mod (b \cdot c)) \div c \)
  by (metis calculation div-right-ineq 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 \( M1 \ M2 \ M3 \)
sows
(\( (\text{mat} (\text{row-length } M1) (\text{length } M1) M1) \wedge (\text{mat} (\text{row-length } M2) (\text{length } M2) M2) \wedge (\text{mat} (\text{row-length } M3) (\text{length } M3) M3) \rightarrow M1 \otimes (M2 \otimes M3) = (M1 \otimes M2) \otimes M3 \) (is \(?x \rightarrow ?l = ?r)\)
proof-
fix \( j \)
assume 0: \( (\text{mat} (\text{row-length } M1) (\text{length } M1) M1) \wedge (\text{mat} (\text{row-length } M2) (\text{length } M2) M2) \wedge (\text{mat} (\text{row-length } M3) (\text{length } M3) M3) \)
have 1: \( \text{length} ((M_1 \otimes M_2) \otimes M_3) = (\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3) \)

proof
- have \( \text{length} (M_2 \otimes M_3) = (\text{length } M_2) \ast (\text{length } M_3) \)
  by (metis length-Tensor)
then have \( \text{length} (M_1 \otimes (M_2 \otimes M_3)) = (\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3) \)
  using mult.assoc length-Tensor by auto
moreover have \( \text{length} (M_1 \otimes M_2) = (\text{length } M_1) \ast (\text{length } M_2) \)
  by (metis length-Tensor)
ultimately show \( \text{thesis} \) using mult.assoc length-Tensor by auto qed

have 2: \( \text{row-length} ((M_1 \otimes M_2) \otimes M_3) = (\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3) \)

proof
- have \( \text{row-length} (M_2 \otimes M_3) = (\text{row-length } M_2) \ast (\text{row-length } M_3) \)
  using row-length-mat assoc by auto
then have \( \text{row-length} (M_1 \otimes (M_2 \otimes M_3)) = (\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3) \)
  using row-length-mat assoc by auto
moreover have \( \text{row-length} (M_1 \otimes M_2) = (\text{row-length } M_1) \ast (\text{row-length } M_2) \)
  using row-length-mat by auto
ultimately show \( \text{thesis} \) using row-length-mat assoc by auto qed

have 3: \( \forall i. \forall j. ((i < ((\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3)))) \wedge (j < (\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3))) \rightarrow \)
  \(((((M_1 \otimes M_2) \otimes M_3) \backslash j \mid i) = f ((M_1 \otimes M_2) \mid (j \text{ div (row-length } M_3)) \mid (j \text{ div (row-length } M_3))) \mid (M_3 \mid (j \text{ mod length } M_3) \mid (i \text{ mod (row-length } M_3)))) \)
  using 0 matrix-Tensor-elements 1 2 effective-well-defined-Tensor length-Tensor row-length-mat by auto
moreover have \( \forall j. (j < (\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3)) \rightarrow (j \text{ div (row-length } M_3)) < (\text{length } M_1) \ast (\text{length } M_2) \)
  apply(rule allI)
  apply(simp add:div-left-ineq)
  done
moreover have \( \forall i. (i < (\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3)) \rightarrow (i \text{ div (row-length } M_3)) < (\text{row-length } M_1) \ast (\text{row-length } M_2) \)
  apply(rule allI)
  apply(simp add:div-left-ineq)
  done
ultimately have 4: \( \forall i. \forall j. ((i < ((\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3)))) \wedge (j < (\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3))) \rightarrow \)
  \(((M_1 \otimes M_2) \otimes M_3) \backslash j \mid i) = f ((M_1 \otimes M_2) \mid (j \text{ div (row-length } M_3)) \mid (j \text{ div (row-length } M_3))) \mid (M_3 \mid (j \text{ mod length } M_3) \mid (i \text{ mod (row-length } M_3)))) \)
  using 0 matrix-Tensor-elements 1 2 effective-well-defined-Tensor length-Tensor row-length-mat by auto 
M3))
∧(j < (length M1)*((length M2)*(length M3)))
→
((i div (row-length M3)) < (row-length M1)*((length M2)))
∧ ((j div (length M3)) < (length M1)*((length M2)))

using all 0 by auto

have (mat (row-length M1) (length M1) M1)
∧ (mat (row-length M2) (length M2) M2)

using 0 by auto
then have ∀ i.∀ j.(((i div (row-length M3)) < (row-length M1)*((length M2)))
∧ ((j div (length M3)) < (length M1)*((length M2)))
→
(((M1 ⊗ M2))!((i div (length M3)))!((j div (row-length M3)))

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

using effective-matrix-tensor-elements by auto
with j have 5:∀ i.∀ j.(((i <((row-length M1)*((row-length M2)*(row-length M3))))
∧ ((j < (length M1)*((length M2)*(length M3))))
→
(((M1 ⊗ M2))!((i div (length M3)))!((j div (row-length M3)))

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

by auto
with 3 have 6:
∀ i.∀ j.(((i <((row-length M1)*((row-length M2)*(row-length M3))))
∧ (j < (length M1)*((length M2)*(length M3))))
→
(((M1 ⊗ M2) ⊗ M3)!j!i)

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

by auto

have (j div (length M3))div (length M2) = (j div ((length M3)*((length M2))))
using div-mult2-eq by auto
moreover have ((i div (row-length M3)) div (row-length M2)) = (i div ((row-length M3)*((row-length M2))))
using div-mult2-eq by auto
ultimately have step1:∀ i.∀ j.(((i <((row-length M1)*((row-length M2)*(row-length M3))))
∧ (j < (length M1)*((length M2)*(length M3))))
\[(M_1 \otimes M_2 \otimes M_3)!j!i\]

\[= f\]

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

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

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

using 6 by (metis 3 5 div-mult2-eq)

then have step1: \(\forall i \forall j.(((i < ((\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3)))
\wedge (j < (\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3))\)

\[\rightarrow ((M_1 \otimes M_2 \otimes M_3)!j!i)\]

\[= f\]

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

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

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

by (metis mult.commute)

have 7:

\(\forall i \forall j.(((i < ((\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3)))
\wedge (j < (\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3))\)

\[\rightarrow ((M_1 \otimes (M_2 \otimes M_3))!j!i)\]

\[= f\]

\[
((M_1)!j \text{ div } (\text{length } M_2) \otimes M_3))!\text{ div } (\text{row-length } M_2) \otimes M_3)))))
\]

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

using 0 matrix-Tensor-elements 1 2 effective-well-defined-Tensor

length-Tensor row-length-mat

by auto

then have

\(\forall i \forall j.(((i < ((\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3)))
\wedge (j < (\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3))\)

\[\rightarrow ((M_1 \otimes (M_2 \otimes M_3))!j!i)\]

\[= f\]

\[
((M_1)!j \text{ div } ((\text{length } M_2) \ast (\text{length } M_3)))!\text{ div } (\text{row-length } M_2) \ast (\text{row-length } M_3)))))
\]

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

using length-Tensor row-length-mat by auto

then have

\(\forall i \forall j.(((i < ((\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3)))
\wedge (j < (\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3))\)
\[
\rightarrow \\
((\mathbf{M}_1 \otimes (\mathbf{M}_2 \otimes \mathbf{M}_3)!j!i))
= f \\
((\mathbf{M}_1)!((\text{length } \mathbf{M}_3)*(\text{length } \mathbf{M}_2)))
!(i \div ((\text{row-length } \mathbf{M}_3)\ast(\text{row-length } \mathbf{M}_2)))) \\
((\mathbf{M}_2 \otimes \mathbf{M}_3)!((j \mod \text{length } (\mathbf{M}_2 \otimes \mathbf{M}_3)))
\!(i \mod (\text{row-length } (\mathbf{M}_2 \otimes \mathbf{M}_3))))
\]
using mult.commute by (metis)

have 8: 
\(\forall j.((j < (\text{length } \mathbf{M}_1)\ast(\text{length } \mathbf{M}_2)\ast(\text{length } \mathbf{M}_3)))
\rightarrow ((j \mod (\text{length } (\mathbf{M}_2 \otimes \mathbf{M}_3))) < (\text{length } (\mathbf{M}_2 \otimes \mathbf{M}_3)))\)

proof(cases length \((\mathbf{M}_2 \otimes \mathbf{M}_3) = 0\))
case True
have \((\text{length } \mathbf{M}_2)\ast(\text{length } \mathbf{M}_3) = 0\)
using length-Tensor True by auto
then have \((\text{length } \mathbf{M}_1)\ast(\text{length } \mathbf{M}_2)\ast(\text{length } \mathbf{M}_3) = 0\)
by auto
then show \(\text{thesis} \) by (metis less-nat-zero-code)
next
case False
have \((\text{length } (\mathbf{M}_2 \otimes \mathbf{M}_3)) > 0\)
using False by auto
then show \(\text{thesis} \) using mod-less-divisor by auto
qed

then have 9: 
\(\forall i.((i < (\text{row-length } \mathbf{M}_1)\ast(\text{row-length } \mathbf{M}_2)\ast(\text{row-length } \mathbf{M}_3)))
\rightarrow ((i \mod (\text{row-length } (\mathbf{M}_2 \otimes \mathbf{M}_3))) < (\text{row-length } (\mathbf{M}_2 \otimes \mathbf{M}_3)))\)

proof(cases row-length \((\mathbf{M}_2 \otimes \mathbf{M}_3) = 0\))
case True
have \((\text{row-length } \mathbf{M}_2)\ast(\text{row-length } \mathbf{M}_3) = 0\)
using True by (metis row-length-mat)
then have \((\text{row-length } \mathbf{M}_1)\ast(\text{row-length } \mathbf{M}_2)\ast(\text{row-length } \mathbf{M}_3) = 0\)
by auto
then show \(\text{thesis} \) by (metis less-nat-zero-code)
next
case False
have \((\text{row-length } (\mathbf{M}_2 \otimes \mathbf{M}_3)) > 0\)
using False by auto
then show \(\text{thesis} \) using mod-less-divisor by auto
qed

with 8 have 10:\(\forall i.\forall j.((i < ((\text{row-length } \mathbf{M}_1)\ast(\text{row-length } \mathbf{M}_2)\ast(\text{row-length } \mathbf{M}_3)))
\land (j < (\text{length } \mathbf{M}_1)\ast(\text{length } \mathbf{M}_2)\ast(\text{length } \mathbf{M}_3)))
\rightarrow \)
\((i \mod (\text{row-length } (\mathbf{M}_2 \otimes \mathbf{M}_3))) < (\text{row-length } (\mathbf{M}_2 \otimes \mathbf{M}_3))
\land (j \mod (\text{length } (\mathbf{M}_2 \otimes \mathbf{M}_3))) < (\text{length } (\mathbf{M}_2 \otimes \mathbf{M}_3))\)
by auto
then have 11:\(\forall i. j.((i < ((\text{row-length } \mathbf{M}_1)\ast(\text{row-length } \mathbf{M}_2)\ast(\text{row-length } \mathbf{M}_3)))
\land (j < (\text{length } \mathbf{M}_1)\ast(\text{length } \mathbf{M}_2)\ast(\text{length } \mathbf{M}_3)))
\rightarrow \)

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 \( \forall \ i. ((i \ mod \ ((\row-length \ M2)\times(\row-length \ M3))) \mod \ (\row-length \ M3) = (i \ mod \ (\row-length \ M3))) \)
using mod-prop1 by auto
ultimately show \( \text{thesis by auto} \)
qed
ultimately have 12: \( \forall \ i \ j. ((i < (\row-length \ M1) \times(\row-length \ M2) \times(\row-length \ M3)) \) \( \wedge (j < (\length \ M1)\times(\length \ M2)\times(\length \ M3)) \) \( \rightarrow \) \( (((\M2 \otimes \M3))!(j \ mod \ (\length \ (\M2 \otimes \M3)))) \) \( !((i \ mod \ \row-length \ (\M2 \otimes \M3))) \) \( = f \) \( (((\M2)!(((j \ mod \ (\length \ (\M2 \otimes \M3)))) \mod \ (\length \ M3)))) \) \( !((i \ mod \ (\row-length \ (\M2 \otimes \M3)))) \) \( \) \( (\M3)!((j \ mod \ (\length \ M3)))!(i \ mod \ (\row-length \ M3))) \) by auto
moreover have \( \forall \ j. ((j \ mod \ (\length \ (\M2 \otimes \M3))) \) \( = \) \( (j \ mod \ ((\length \ M2)\times(\length \ M3)))) \) using length-Tensor by auto
then show \( \text{thesis by auto} \)
qed
moreover have \( \forall \ i. ((i \ mod \ (\row-length \ (\M2 \otimes \M3))) \) \( = \) \( (i \ mod \ (\row-length \ M3)) \mod \ (\row-length \ M2) \) proof–
have \( \forall \ j. ((j \ mod \ (\length \ (\M2 \otimes \M3))) \) \( = \) \( (j \ mod \ ((\length \ M2)\times(\length \ M3)))) \) using length-Tensor by auto
then show \( \text{thesis by auto} \)
qed
moreover have \( \forall \ i. ((i \ mod \ (\row-length \ (\M2 \otimes \M3))) \) \( = \) \( (i \ mod \ ((\row-length \ M2)\times(\row-length \ M3)))) \) using row-length-mat by auto
then show \( \text{thesis by auto} \)
qed
ultimately have \( \forall \ i \ j. \)
\( (((\M2 \otimes \M3))!(j \ mod \ (\length \ (\M2 \otimes \M3)))) \) \( !((i \ mod \ \row-length \ (\M2 \otimes \M3))) \) \( = f \) \( (((\M2)!((j \ div \ (\length \ M3)) \mod \ (\length \ M2)))) \) \( !((i \ div \ (\row-length \ M3)) \mod \ (\row-length \ M2))) \) \( (\M3)!((j \ mod \ (\length \ M3)))!(i \ mod \ (\row-length \ M3))) \) by auto
with 7 have 13: \( \forall \ i \ j. (((i < ((\row-length \ M1)\times(\row-length \ M2)\times(\row-length \ M3))) \) \( \wedge (j < (\length \ M1)\times(\length \ M2)\times(\length \ M3))) \) \( \rightarrow \)
\[(M_1 \otimes (M_2 \otimes M_3))!j!i) = f (M_1)!j ((\text{length } M_2) \ast (\text{length } M_3))!
\text{!}i ((\text{row-length } M_2) \ast (\text{row-length } M_3)))
(f ((M_2)!((\text{length } M_3))!j \mod (\text{length } M_2)))!((\text{div } (\text{row-length } M_3)) \mod (\text{row-length } M_2)))
(M_3)!j \mod (\text{length } M_3)
!i \text{ mod (row-length } M_3)))
\]
\[\text{using length-Tensor row-length-mat by auto}
\]
\[
\text{moreover have } \forall \ i \ j . \ f ((M_1)!j ((\text{length } M_2) \ast (\text{length } M_3)))!
\text{!}i ((\text{row-length } M_2) \ast (\text{row-length } M_3)))
(f ((M_2)!((\text{length } M_3)) \mod (\text{length } M_2)))!((\text{div } (\text{row-length } M_3)) \mod (\text{row-length } M_2)))
(M_3)!j \mod (\text{length } M_3)
!i \text{ mod (row-length } M_3)))
\]
\[\text{using assoc by auto}
\]
\[
\text{with 13 have } \forall \ i . j . ((i < ((\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3)))
\wedge (j < (\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3)))
\rightarrow
((M_1 \otimes (M_2 \otimes M_3))!j!i)
= f (f ((M_1)!j ((\text{length } M_2) \ast (\text{length } M_3)))!
\text{!}i ((\text{row-length } M_2) \ast (\text{row-length } M_3)))
((M_2)!((\text{length } M_3)) \mod (\text{length } M_2))
!((\text{div } (\text{row-length } M_3)) \mod (\text{row-length } M_2)))
(M_3)!j \mod (\text{length } M_3)
!i \text{ mod (row-length } M_3)))
\]
\[\text{by auto}
\]
\[
\text{with step1 have step2:}
\forall \ i . j . ((i < ((\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3)))
\wedge (j < (\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3)))
\rightarrow
((M_1 \otimes (M_2 \otimes M_3))!j!i) = (((M_1 \otimes M_2) \otimes M_3)!j!i))
\text{by auto}
\]
\[
\text{moreover have } \text{mat } ((\text{row-length } M_1) \ast (\text{row-length } M_2) \ast (\text{row-length } M_3))
((\text{length } M_1) \ast (\text{length } M_2) \ast (\text{length } M_3))
(M_1 \otimes (M_2 \otimes M_3))
\]
\[\text{proof-}
\text{have } \text{mat } ((\text{row-length } M_2) \ast (\text{row-length } M_3)) ((\text{length } M_2) \ast (\text{length } M_3)) (M_2
\]
58
\( M3 \)

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

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

using \( 0 \) effective-well-defined-Tensor row-length-mat length-Tensor by metis

ultimately show ?thesis using row-length-mat length-Tensor \( \text{mult} \).assoc by (simp add: length-Tensor row-length-mat semigroup-mult-class \( \text{mult} \).assoc)

qed

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

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

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

using \( 0 \) effective-well-defined-Tensor row-length-mat length-Tensor by metis

ultimately show ?thesis using row-length-mat length-Tensor \( \text{mult} \).assoc by (metis \( \text{mult} \).assoc)

qed

ultimately show ?thesis using \( \text{mat-eqI} \) by blast

qed

end

lemma \( \forall (a::\text{nat})\ b. (\text{times } a\ b) = (\text{times } b\ a) \)

by auto

1.2 Associativity and Distributive properties

locale plus-mult =

\( \text{mult} + \)

fixes \( \text{zer} : 'a \)

fixes \( g: 'a \Rightarrow 'a \Rightarrow 'a \) (infixl \( + 60 \))

fixes \( \text{inver}: 'a \Rightarrow 'a \)

assumes \( \text{plus-comm}: g \ a\ b = g\ b\ a \)

assumes \( \text{plus-assoc}: (g\ (g\ a\ b)\ c) = (g\ a\ (g\ b\ c)) \)

assumes \( \text{plus-left-id}: g\ \text{zer}\ x = x \)

assumes \( \text{plus-right-id}: g\ x\ \text{zer} = x \)

assumes \( \text{plus-left-distributivity}: f\ a\ (g\ b\ c) = g\ (f\ a\ b)\ (f\ a\ c) \)

assumes \( \text{plus-right-distributivity}: f\ (g\ a\ b)\ c = g\ (f\ a\ c)\ (f\ b\ c) \)

assumes \( \text{plus-left-inverse}: (g\ x\ (\text{inver}\ x)) = \text{zer} \)

assumes \( \text{plus-right-inverse}: (g\ (\text{inver}\ x)\ x) = \text{zer} \)

59
context plus-mult

begin

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

matrix_mult refers to multiplication of matrices in the locale plus_mult

abbreviation matrix-mult::'a mat ⇒ 'a mat ⇒ 'a mat (infixl ◦ 65)
  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 ! i ! 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 ! i ! 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
then have length $v = nr1$
  using assms(1) mat-def Ball-def vec-def Cons by metis
moreover then have length $v = nr2$
  using 1 assms(2) mat-def Ball-def vec-def Cons by metis
ultimately show $nr1 = nr2$
  by auto
next
have length $M = nc1$
  using mat-def assms(1) by auto
moreover have length $M = nc2$
  using mat-def assms(2) by auto
ultimately show $nc1 = nc2$
  by auto
qed

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

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

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

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

lemma tensor-compose-distribution1:
assumes wf1:mat (row-length $A_1$) (length $A_1$) $A_1$
  and wf2:mat (row-length $A_2$) (length $A_2$) $A_2$
  and wf3:mat (row-length $B_1$) (length $B_1$) $B_1$
  and wf4:mat (row-length $B_2$) (length $B_2$) $B_2$
  and matchAA:length $A_1 = \text{row-length } A_2$
  and matchBB:length $B_1 = \text{row-length } B_2$
  and non-nil:($A_1 \neq [] \land (A_2 \neq []) \land (B_1 \neq []) \land (B_2 \neq [])$
shows mat ((row-length $A_1$)* (row-length $B_1$))
  $((\text{length } A_2) * (\text{length } B_2))$
  $((A_1 \circ A_2) \otimes (B_1 \circ B_2))$
proof –
  have 0:mat (row-length $A_1$) (length $A_2$) (matrix-mult $A_1$ $A_2$)
    using wf1 wf2 mat-mult matchAA by auto
  then have 1:mat (row-length $(A_1 \circ A_2)$) (length $(A_1 \circ A_2)$) (matrix-mult $A_1$ $A_2$)
    by (metis matrix-row-length)
  then have 2: (row-length $(A_1 \circ A_2)$) = (row-length $A_1$) and length $(A_1 \circ A_2) = \text{length } A_2$
    using non-empty-mat-mult unique-row-col 0
    apply (metis length-0-conv mat-empty-column-length non-nil)
    by (metis 0 1 mat-empty-column-length unique-row-col(2))
  moreover have 3:mat (row-length $B_1$) (length $B_2$) (matrix-mult $B_1$ $B_2$)
    using wf3 wf4 matchBB mat-mult by auto
  then have 4:mat (row-length $(B_1 \circ B_2)$) (length $(B_1 \circ B_2)$) (matrix-mult $B_1$ $B_2$)
    using non-empty-mat-mult unique-row-col 3
    apply (metis length-0-conv mat-empty-column-length non-nil)
    by (metis 3 4 mat-empty-column-length unique-row-col(2))
  then show ?thesis using 1 4 5 well-defined-Tensor
    by (metis 2 calculation(2))
qed

lemma effective-tensor-compose-distribution1:
matrix-match $A_1 \ A_2 \ B_1 \ B_2 \implies \text{mat}\ ((\text{row-length } A_1)\ast(\text{row-length } B_1))\((\text{length } A_2)\ast(\text{length } B_2))\((A_1 \odot A_2)\otimes(B_1 \odot B_2))$

using tensor-compose-distribution1 unfolding matrix-match-def by auto

lemma tensor-compose-distribution2:
assumes wf1:\text{mat}\ ((\text{row-length } A_1)\ (\text{length } A_1)\ A_1\ 
and \ wf2:\text{mat}\ ((\text{row-length } A_2)\ (\text{length } A_2)\ A_2\ 
and \ wf3:\text{mat}\ ((\text{row-length } B_1)\ (\text{length } B_1)\ B_1\ 
and \ wf4:\text{mat}\ ((\text{row-length } B_2)\ (\text{length } B_2)\ B_2\ 
and \ matchAA:\text{length } A_1 = \text{row-length } A_2\ 
and \ matchBB:\text{length } B_1 = \text{row-length } B_2\ 
and \ non-\text{Nil}:(A_1 \neq []\wedge A_2 \neq []\wedge B_1 \neq []\wedge 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{wf1} \text{wf3} \text{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{wf2} \text{wf4} \text{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 mat-mult 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) \text{length-0-cone length-Tensor mult-is-0 by metis

theorem non-empty-distribution:
assumes \text{mat} nr1 n1 A_1\ 
and \ \text{mat} n1 nc1 A_2\ 
and \ \text{mat} nr2 n2 B_1\ 
and \ \text{mat} n2 nc2 B_2\ 
and \ A_1 \neq [] \ and \ B_1 \neq [] \ and \ A_2 \neq [] \ and \ B_2 \neq []
shows ((A_1 \odot A_2)\otimes(B_1 \odot B_2)) \neq []$
proof−
have $A_1 \circ A_2 \neq []$
using \text{assms} \text{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
  ⇒ mat ((row-length A1)*i(row-length B1))
  (length A2)*i(row-length B2))
  ((A1 ⊗ B1) o (A2 ⊗ B2))
  using tensor-compose-distribution2 unfolding matrix-match-def by auto

theorem effective-matrix-Tensor-elements:
fixes M1 M2 i j
assumes i<((row-length M1)*i(row-length M2))
  and j< (length M1)*i(length M2)
  and mat (row-length M1) (length M1) M1
  and mat (row-length M2) (length M2) M2
shows ((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 assms by auto

theorem effective-matrix-Tensor-elements2:
fixes M1 M2
assumes mat (row-length M1) (length M1) M1
  and mat (row-length M2) (length M2) M2
shows (∀ i<((row-length M1)*i(row-length M2)).
  ∀ j< (length M1)*i(length 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 assms by auto

definition matrix-compose-cond:: 'a mat ⇒ 'a mat ⇒ 'a mat ⇒ nat ⇒ nat ⇒ bool
where
matrix-compose-cond A1 A2 B1 B2 i j ≡
  (mat (row-length A1) (length A1) A1)
  ∧ (mat (row-length A2) (length A2) A2)
  ∧ (mat (row-length B1) (length B1) B1)
  ∧ (mat (row-length B2) (length B2) B2)
  ∧ (length A1 = row-length A2)
  ∧ (length B1 = row-length B2)
  ∧ (A1 ≠ []) ∧ (A2 ≠ []) ∧ (B1 ≠ []) ∧ (B2 ≠ [])
  ∧ (i< (row-length A1)*i(row-length B1)) ∧ (j< (length A2)*i(length B2))

theorem elements-matrix-distribution-1:
\textbf{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 \(\text{matchAA: length } A1 = \text{row-length } A2\) \\
and \(\text{matchBB: length } B1 = \text{row-length } B2\) \\
and \(\text{non-nil} :(A1 \neq \emptyset) \land (A2 \neq \emptyset) \land (B1 \neq \emptyset) \land (B2 \neq \emptyset)\) \\
and \(i < (\text{row-length } A1) \ast (\text{row-length } B1)\) \text{ and } \(j < (\text{length } A2) \ast (\text{length } B2)\)

\textbf{shows} \\
\((\text{matrix-mult } A1 A2) \circ (\text{matrix-mult } B1 B2)) i j i = f (\text{scalar-product} (\text{row } A1 (i \text{ div } (\text{row-length } B1))) \\
(\text{col } A2 (j \text{ div } (\text{length } B2)))) \\
(\text{scalar-product} (\text{row } B1 (i \text{ mod } (\text{row-length } B1))) \\
(\text{col } B2 (j \text{ mod } (\text{length } B2))))\

\textbf{proof–} \\
\text{have } 0 : ((\text{matrix-mult } A1 A2) \circ (\text{matrix-mult } B1 B2)) \neq \emptyset \\
\text{using } \text{non-empty-distribution} \text{ assms} \text{ by } \text{auto} \\
\text{then have } 1 : \text{mat } ((\text{row-length } A1) \ast (\text{row-length } B1)) \\
((\text{length } A2) \ast (\text{length } B2)) \\
((\text{A1} \circ \text{A2}) \circ (\text{B1} \circ \text{B2})) \\
\text{using } \text{tensor-compose-distribution1} \text{ assms} \text{ by } \text{auto} \\
\text{then have } 2 : \text{mat } (\text{row-length } ((\text{A1} \circ \text{A2}) \circ (\text{B1} \circ \text{B2}))) \\
((\text{A1} \circ \text{A2}) \circ (\text{B1} \circ \text{B2})) \\
\text{by } \text{metis} \text{ matrix-row-length} \\
\text{then have } 3 : ((\text{row-length } A1) \ast (\text{row-length } B1)) \\
= (\text{row-length } ((\text{A1} \circ \text{A2}) \circ (\text{B1} \circ \text{B2}))) \\
\text{and } ((\text{length } A2) \ast (\text{length } B2)) = (\text{length } ((\text{A1} \circ \text{A2}) \circ (\text{B1} \circ \text{B2}))) \\
\text{using } 0 1 \text{ unique-row-col} \\
\text{apply } \text{metis} \\
\text{using } 0 1 2 \text{ unique-row-col by } \text{metis} \\
\text{then have } i : (i < ((\text{row-length } A1) \ast (\text{row-length } B1))) \\
= (i < (\text{row-length } ((\text{A1} \circ \text{A2}) \circ (\text{B1} \circ \text{B2})))) \\
\text{by } \text{auto} \\
\text{moreover have } j : (j < ((\text{length } A2) \ast (\text{length } B2))) \\
= (j < (\text{length } ((\text{A1} \circ \text{A2}) \circ (\text{B1} \circ \text{B2})))) \\
\text{using } 3 \text{ length } A2 \ast \text{ length } B2 = \text{ length } (\text{A1} \circ \text{A2} \circ \text{B1} \circ \text{B2}) \\
\text{by } \text{metis} \\
\text{have } 4 : \text{mat } (\text{row-length } A1) (\text{length } A2) (A1 \circ A2) \\
\text{using } \text{assms} \text{ mat-mult} \text{ by } \text{auto} \\
\text{then have } 5 : \text{mat } (\text{row-length } (A1 \circ A2)) (\text{length } (A1 \circ A2)) (A1 \circ A2) \\
\text{using } \text{matrix-row-length} \text{ by } \text{metis} \\
\text{with } 4 \text{ have } 6 : \text{row-length } A1 = \text{row-length } (A1 \circ A2) \\
\text{by } \text{metis } 0 \text{ Tensor.simps}(1) \text{ unique-row-col}(1) \\
\text{with } 4 \text{ 5 have } 7 : \text{length } A2 = \text{length } (A1 \circ A2) \\
\text{by } \text{metis } \text{mat-empty-column-length} \text{ unique-row-col}(2) \\
\text{then have } 8 : \text{mat } (\text{row-length } B1) (\text{length } B2) (B1 \circ B2) \\
\text{using } \text{assms} \text{ mat-mult} \text{ by } \text{auto} \\
\text{then have } 9 : \text{mat } (\text{row-length } (B1 \circ B2)) (\text{length } (B1 \circ B2)) (B1 \circ B2)
using matrix-row-length by (metis)

with 7 8 have 10: row-length B1 = row-length (B1 ⨀ 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 = length (B1 ⨀ B2)
  by (metis mat-empty-column-length unique-row-col(2))

from 6 10 have 12:
  \( i < ((\text{row-length } A1) \times (\text{row-length } B1)) \)
  \( i < (\text{row-length } (A1 \circ A2)) \times (\text{row-length } (B1 \circ B2)) \)
  by auto

then have 13: \( i < (\text{row-length } (A1 \circ A2)) \times (\text{row-length } (B1 \circ B2)) \)
  using assms by auto

from 7 11 have 14:
  \( j < ((\text{length } A2) \times (\text{length } B2)) \)
  \( j < (\text{length } (A1 \circ A2)) \times (\text{length } (B1 \circ B2)) \)
  by auto

then have 15: \( j < (\text{length } (A1 \circ A2)) \times (\text{length } (B1 \circ B2)) \)
  using assms by auto

then have step-1: \((A1 \circ A2) \otimes (B1 \circ B2))\!
  = (\text{scalar-product} (\text{row } A1 (i \text{ div } (\text{row-length } B1))) \text{ (col } A2 (j \text{ div } (\text{length } B2)))\)

proof
  have j div (length B2) < (length A2)
    using div-left-ineq assms by auto
  moreover have i div (row-length B1) < (row-length A1)
    using assms div-left-ineq by auto
  moreover have mat (length A1) (length A2) A2
    using wf2 matchAA by auto
  ultimately show \(?thesis \) using wf1 non-Nil matrix-mult-index by blast

qed

moreover have \((B1 \circ B2)!((j \text{ mod } (\text{length } B2)))\)
  = (\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 < (length A2) \times (length B2)
    using assms by auto
  then have j mod (length B2) < (length B2)
    by (metis calculation less-nat-zero-code mod-less-divisor mult-is-0)
neq0-conv

moreover have \( i \mod (\text{row-length } B1) < (\text{row-length } B1) \)
by (metis assms(8) less-nat-zero-code mod-less-divisor mult-is-0)

neq0-conv

moreover have \( \text{mat} (\text{length } B1) (\text{length } B2) B2 \)
using wf4 matchBB by auto

ultimately show \( \text{thesis} \)
using wf3 non-Nil matrix-mult-index by blast

qed

ultimately show \( \text{thesis} \) by auto

qed

lemma effective-elements-matrix-distribution1:
matrix-compose-cond A1 A2 B1 B2 i j \implies
((matrix-mult A1 A2) \odot (matrix-mult B1 B2))!j!i
= f \( \text{scalar-product} \) \( \text{row } A1 \ (i \div \text{row-length } B1)) \) \( \text{col } A2 \ (j \div \text{length } B2))\)
\( \text{scalar-product} \) \( \text{row } B1 \ (i \mod \text{row-length } B1)) \) \( \text{col } B2 \ (j \mod \text{length } B2))\)

using elements-matrix-distribution-1 matrix-compose-cond-def by auto

lemma matrix-match-condn-1:
matrix-match A1 A2 B1 B2
\(i \prec (\text{row-length } A1) \ast (\text{row-length } B1))\)
\(j \prec (\text{length } A2) \ast (\text{length } B2))\)
\implies ((matrix-mult A1 A2) \odot (matrix-mult B1 B2))!j!i
= f
\( \text{scalar-product} \) \( \text{row } A1 \ (i \div \text{row-length } B1)) \) \( \text{col } A2 \ (j \div \text{length } B2))\)
\( \text{scalar-product} \) \( \text{row } B1 \ (i \mod \text{row-length } B1)) \) \( \text{col } B2 \ (j \mod \text{length } B2))\)

using elements-matrix-distribution-1 unfolding matrix-match-def by auto

lemma effective-matrix-match-condn-1:
assumes (matrix-match A1 A2 B1 B2)
shows \( \forall \ i \ j \ ((i \prec (\text{row-length } A1) \ast (\text{row-length } B1))\)
\(j \prec (\text{length } A2) \ast (\text{length } B2))\)
\implies ((A1 \circ A2) \odot (B1 \circ B2))!j!i
= f
\( \text{scalar-product} \) \( \text{row } A1 \ (i \div \text{row-length } B1)) \) \( \text{col } A2 \ (j \div \text{length } B2))\)
\( \text{scalar-product} \) \( \text{row } B1 \ (i \mod \text{row-length } B1)) \) \( \text{col } B2 \ (j \mod \text{length } B2))\)

using assms matrix-match-condn-1 unfolding matrix-match-def by auto

67
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:i<(row-length A1)*(row-length B1) and j:j< (length A2)*(length B2)
shows
    (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-product (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

68
qed

lemma matrix-match-condn-2):
matrix-match A1 A2 B1 B2
∧((i<(row-length A1)∗(row-length B1))
∧(j<(length A2)∗(length B2)))
⇒ ((A1 ⊗ B1)◦(A2 ⊗ B2))![j]![i]
= scalar-product
  (vec-vec-Tensor
   (row A1 (i div row-length B1))
   (row B1 (i mod row-length B1)))
  (vec-vec-Tensor
   (col A2 (j div length B2))
   (col B2 (j mod length B2)))
using elements-matrix-distribution2 unfolding matrix-match-def by auto

lemma effective-matrix-match-condn-2:
assumes (matrix-match A1 A2 B1 B2)
shows ∀ i j.((i<(row-length A1)∗(row-length B1))
∧(j<(length A2)∗(length B2)))
⇒ ((A1 ⊗ B1)◦(A2 ⊗ B2))![j]![i]
= scalar-product
  (vec-vec-Tensor
   (row A1 (i div row-length B1))
   (row B1 (i mod row-length B1)))
  (vec-vec-Tensor
   (col A2 (j div length B2))
   (col B2 (j mod length B2)))
using assms matrix-match-condn-2 unfolding matrix-match-def by auto

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

lemma zer-left-mult:f zer x = zer
proof
  have g zer zer = zer
    using plus-left-id by auto
  then have f zer x = f (g zer zer) x
    by auto
  then have f zer x = (f zer x) + (f zer x)
    using plus-right-distributivity by auto
  then have (fzer x) + (inver (fzer x)) = (f zer x) + (fzer x) + (inver (fzer x))
    by auto
  then have zer = (fzer x) + zer
    using plus-left-inverse plus-assoc by (metis)
  then show ?thesis
    using plus-right-id by simp
lemma `zip-Cons`:
\[
\text{length } v = \text{length } w \implies \text{zip } (a \# v) (b \# w) = (a, b) \# (\text{zip } v w)
\]

unfolding `zip-def` by `auto`

lemma `scalar-product-times`:
\[
\forall w1 w2. (\text{length } w1 = \text{length } w2) \land (\text{length } w1 = n) \implies
\begin{align*}
(f \cdot (x \cdot y)) \cdot (\text{scalar-product } w1 \cdot w2) \\
\quad = (\text{scalar-product } \times w1) \\
\quad \times (\text{scalar-product } \times w2)
\end{align*}
\]

apply (rule `allI`)  
apply (rule `allI`)  
proof (induct `n`)  

\text{case } 0  
have `(\text{length } w1 = \text{length } w2) \land (\text{length } w1 = 0) \implies ?\text{case}`  

proof−  
assume `assms`: `(\text{length } w1 = \text{length } w2) \land (\text{length } w1 = 0)`  

have 1: `w1 = []`  
using `assms` by `auto`  
moreover have 2: `(\text{length } w1 = \text{length } w2) \land (\text{length } w1 = 0) \implies w2 = []`  
by `auto`  

ultimately have `(\text{length } w1 = \text{length } w2) \land (\text{length } w1 = 0) \implies \text{scalar-product } w1 \cdot w2 = \text{zer}`  

unfolding `scalar-product-def` `scalar-prodI-def` by `auto`  
then have 3: `(\text{length } w1 = \text{length } w2) \land (\text{length } w1 = 0) \implies (f \cdot (x \cdot y)) \cdot (\text{scalar-product } w1 \cdot w2) = \text{zer}`  

using `comm` `zer-left-mult` by `metis`  
then have `\times w1 = []`  
using 1 by `auto`  
moreover have `\times y w2 = []`  
using 2 `assms` by `auto`  

ultimately have `(\text{scalar-product } (\times w1) \cdot (\times y w2)) = \text{zer}`  

unfolding `scalar-product-def` `scalar-prodI-def` by `auto`  
with 3 show `?thesis` by `auto`  

qed  
then show `?case` by `auto`  

next  
\text{case } (\text{Suc } k)  
have `(\text{length } w1 = \text{length } w2) \land (\text{length } w1 = (\text{Suc } k)) \implies ?\text{case}`  

proof−  
assume `assms`: `(\text{length } w1 = \text{length } w2) \land (\text{length } w1 = (\text{Suc } k))`  

have `\exists a1 \cdot w1 = a1 \# a1 \land (\text{length } u1 = k)`  
using `assms` by (metis `length-Suc-conv`)  
then obtain `a1 \cdot u1` where `(w1 = a1 \# u1) \land (\text{length } u1 = k)`  
by `auto`  
then have `Cons-1: (w1 = a1 \# u1) \land (\text{length } u1 = k)`  
by `auto`  

have `\text{length } w2 = (\text{Suc } k)`

70
using \textit{assms} by \textit{auto}
then have \(\exists a_2 u_2. (w_2 = a_2 \# u_2) \land (\text{length } u_2 = k)\)
using \textit{assms} by (metis \text{length-Suc-conv})
then obtain \(a_2 u_2\) where \((w_2 = a_2 \# u_2) \land (\text{length } u_2 = k)\)
by \textit{auto}
then have \(\text{Cons-2}: (w_2 = a_2 \# u_2) \land (\text{length } u_2 = k)\)
by \textit{auto}
then have \(\text{length } a_1 = \text{length } u_2) \land (\text{length } a_1 = k)\)
using \textit{Cons-1} by \textit{auto}
then have \(\text{Cons-3}: (w_2 = a_2 \# u_2) \land (\text{length } u_2 = k)\)
by \textit{auto}
then have \((\text{length } u_1 = \text{length } u_2) \land (\text{length } u_1 = k)\)
using \textit{Cons-1} by \textit{auto}
then have \((\text{length } u_1 = \text{length } u_2) \land (\text{length } u_1 = k)\)
using \textit{Suc assms} by \textit{auto}
have \((\text{scalar-product } (\text{times } x u_1) (\text{times } y u_2) = \text{scalar-product } (\text{times } x u_1) (\text{times } y u_2)\)
unfolding \text{scalar-product-def scalar-prodI-def zip-def} by \textit{auto}
then have \((\text{scalar-product } w_1 w_2) = (\text{scalar-product } w_1 w_2)\)
by \textit{auto}
then have \((x \# y) * \text{scalar-product } u_1 u_2\)
= \(\text{scalar-product } (\text{times } x u_1) (\text{times } y u_2)\)
using \textit{scalar-product-times assms} by \textit{auto}
lemma \textit{effective-scalar-product-times}:
\textit{assumes} (\text{length } w_1 = \text{length } w_2)
\textit{shows} \((f (x \# y) (\text{scalar-product } w_1 w_2))\)
= \((\text{scalar-product } (\text{times } x u_1)) (\text{times } y u_2)\)
\textit{using} \text{scalar-product-times assms} by \textit{auto}
lemma \textit{zip-append}:
\((\text{length } zs = \text{length } ws) \land (\text{length } zs = \text{length } ys)\)
\implies (\text{zip } (\text{times } zs) (\text{times } ws)) = (\text{zip } zs ys) \# (\text{zip } zs ws)\)
using `zip-append1 zip-append2` by `auto`

**lemma** `scalar-product-append`:
\[
\forall \text{xs ys zs ws}. \left( \text{length zs} = \text{length ws} \right) \land \left( \text{length xs} = \text{length ys} \right) \land \left( \text{length xs} = n \right) \rightarrow \\
\left( \text{scalar-product} \left( \text{xs@zs} \right) \left( \text{ys@ws} \right) \right) = \left( \text{scalar-product} \right. \text{xs} \text{ys}) \left. + \left( \text{scalar-product} \right. \text{zs} \text{ws} \right)
\]

**apply** `(rule allI)`
**apply** `(rule allI)`
**apply** `(rule allI)`
**apply** `(rule allI)`
**proof** `(induct n)`

**case** `0`

**have** \( \left( \text{length zs} = \text{length ws} \right) \land \left( \text{length xs} = \text{length ys} \right) \land \left( \text{length xs} = 0 \right) \) \rightarrow \\
\left( \text{scalar-product} \left( \text{xs@zs} \right) \left( \text{ys@ws} \right) \right) = \left( \text{scalar-product} \right. \text{xs} \text{ys}) \left. + \left( \text{scalar-product} \right. \text{zs} \text{ws} \right)

**proof**
**assume** `assms`: \( \left( \text{length zs} = \text{length ws} \right) \land \left( \text{length xs} = \text{length ys} \right) \land \left( \text{length xs} = 0 \right) \)

**have** 1: \( \text{xs} = [] \) 
**using** `assms` **by** `auto`
**moreover have** 2: \( \text{ys} = [] \) 
**using** `assms` **by** `auto`

**ultimately have** `scalar-product xs ys = zer`
**unfolding** `scalar-product-def scalar-prodI-def zip-def` **by** `auto`

**then have** \( \left( \text{scalar-product} \right. \text{xs} \text{ys}) + \left( \text{scalar-product} \right. \text{zs} \text{ws} \right) = \left( \text{scalar-product} \right. \text{zs} \text{ws} \right)

**using** `plus-left-id` **by** `auto`
**moreover have** \( \left( \text{scalar-product} \left( \text{xs@zs} \right) \left( \text{ys@ws} \right) \right) = \left( \text{scalar-product} \right. \text{zs} \text{ws} \right) \)
**using** 1 2 **by** `auto`

**ultimately show** `?thesis` **by** `auto`
**qed**
**then show** `?case` **by** `auto`
**next**
**case** `(Suc k)`

**have** \( \left( \text{length zs} = \text{length ws} \right) \land \left( \text{length xs} = \text{length ys} \right) \land \left( \text{length xs} = (Suc k) \right) \) \rightarrow \\
\left( \text{scalar-product} \left( \text{xs@zs} \right) \left( \text{ys@ws} \right) \right) = \left( \text{scalar-product} \right. \text{xs} \text{ys}) \left. + \left( \text{scalar-product} \right. \text{zs} \text{ws} \right)

**proof**
**assume** `assms`: \( \left( \text{length zs} = \text{length ws} \right) \land \left( \text{length xs} = \text{length ys} \right) \land \left( \text{length xs} = (Suc k) \right) \)
**have** \( \exists x \text{zxs}. \left( \text{xs} = x\#\text{zxs} \right) \land \left( \text{length zxs} = k \right) \)
using assms by (metis Suc-length-conv)
then obtain \( x \) \( xss \) where \((xs = x\#xss) \land (\text{length } xss = k)\)
  by auto
then have \((xs = x\#xss) \land (\text{length } xss = k)\)
  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 \( 1\) : \((xs = x\#xss) \land (\text{length } xss = k)\)
  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 \( 1 \ \ 2 \ \ \text{assms} \) Suc by auto
then have \( 4\) : \((\text{scalar-product } (((x\#xss)@zs) ((y\#yss)@ws))) = \((\text{scalar-product } (x\#(xss@zs)) (y\#(yss@ws)))\)
  by auto
then have \( ... = (x*y) + (\text{scalar-product } (xss@zs) (yss@ws)) \)
  unfolding scalar-product-def scalar-prodI-def
  using zip-Conss scalar-prodI-def scalar-prod-cons
  by (metis)
with \( 4 \) have \( 5\) : \((\text{scalar-product } ((x\#zs)@zs) ((y\#ys)@ws))\)
  = \((x*y) + (\text{scalar-product } (xss@zs) (yss@ws))\)
  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-Conss
  by (metis \( 1 \ \ 2 \) scalar-prodI-def scalar-prod-cons)
moreover then have \((\text{scalar-product } xs ys) + (\text{scalar-product } zs ws)\)
  = \((x*y) + (\text{scalar-product } xss yss) + (\text{scalar-product } zs ws)\)
  by auto
ultimately show \( ?\text{thesis} \) using \( 3 \ \ \text{plus-assoc} \) by auto
qed
then show \( ?\text{case} \) by auto
qed

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

lemma scalar-product-distributivity:
\( \forall \ v1 \ \ v2 \ \ w1 \ \ w2. ((\text{length } v1 = \text{length } v2) \land (\text{length } v1 = n) \land (\text{length } w1 = \text{length } w2) \land \)
\[
\begin{align*}
\rightarrow (\text{scalar-product } v_1 v_2) \ast (\text{scalar-product } w_1 w_2) \\
= \text{scalar-product } (\text{vec-vec-Tensor } v_1 w_1) (\text{vec-vec-Tensor } v_2 w_2)
\end{align*}
\]

apply (rule allI)
apply (rule allI)
apply (rule allI)
apply (rule allI)
proof (induct \( n \))
case 0

have \((\text{length } v_1 = \text{length } v_2) \land (\text{length } v_1 = 0) \land (\text{length } w_1 = \text{length } w_2)\)
  \(\rightarrow \text{length } v_1 = 0\)
using 0 by auto

then have 1:\((\text{length } v_1 = \text{length } v_2) \land (\text{length } v_1 = 0) \land (\text{length } w_1 = \text{length } w_2)\)
  \(\rightarrow v_1 = []\)
by auto
moreover have \((\text{length } v_1 = \text{length } v_2) \land (\text{length } v_1 = 0) \land (\text{length } w_1 = \text{length } w_2)\)
  \(\rightarrow \text{length } v_2 = 0\)
using 0 by auto
moreover then have 2:\((\text{length } v_1 = \text{length } v_2) \land (\text{length } v_1 = 0) \land (\text{length } w_1 = \text{length } w_2)\)
  \(\rightarrow v_2 = []\)
by auto
ultimately have 3:
\((\text{length } v_1 = \text{length } v_2) \land (\text{length } v_1 = 0) \land (\text{length } w_1 = \text{length } w_2)\)
  \(\rightarrow \text{scalar-product } v_1 v_2 = 0\)
unfolding scalar-product-def scalar-prodI-def using zip-Nil by auto
then have 4: \(f \text{zer} (\text{scalar-product } w_1 w_2) = 0\)
using zer-left-mult by auto
have \((\text{length } v_1 = \text{length } v_2) \land (\text{length } v_1 = 0) \land (\text{length } w_1 = \text{length } w_2)\)
  \(\rightarrow \text{vec-vec-Tensor } v_1 w_1 = []\)
using 1 by auto
moreover have \((\text{length } v_1 = \text{length } v_2) \land (\text{length } v_1 = 0) \land (\text{length } w_1 = \text{length } w_2)\)
  \(\rightarrow \text{vec-vec-Tensor } v_2 w_2 = []\)
using 2 by auto
ultimately have \((\text{length } v_1 = \text{length } v_2) \land (\text{length } v_1 = 0) \land (\text{length } w_1 = \text{length } w_2)\)
  \(\rightarrow \text{scalar-product } (\text{vec-vec-Tensor } v_1 w_1) (\text{vec-vec-Tensor } v_2 w_2) = 0\)
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 ((length v1 = length v2) ∧ (length v1 = Suc k)
    ∧ (length w1 = length w2))
    ⟹ f (scalar-product v1 v2) (scalar-product w1 w2)
    = scalar-product (vec-vec-Tensor v1 w1) (vec-vec-Tensor v2 w2)
proof
  assume assms:((length v1 = length v2) ∧ (length v1 = Suc k)
    ∧ (length w1 = length w2))
  have length v1 = Suc
    using Suc assms by auto
  then have (∃a1 u1. (v1 = a1 # u1) ∧ (length u1 = k))
    using assms Suc-length-conv by metis
  then obtain a1 u1 where (v1 = a1 # u1) ∧ (length u1 = k)
    by auto
  moreover have length v2 = Suc
    using assms Suc by auto
  then have (∃a2 u2. (v2 = a2 # u2) ∧ (length u2 = k))
    using Suc-length-conv by metis
  then obtain a2 u2 where (v2 = a2 # u2) ∧ (length u2 = k)
    by auto
  then have Cons-1: (v1 = a1 # u1) ∧ (length u1 = k)
    by simp
  then have length w1 = length w2
    using Cons-1 by auto
  then have Cons-2: (v2 = a2 # u2) ∧ (length u2 = k)
    by simp
  then have length u1 = length u2
    using Cons-1 by auto
  then have Cons-3: (scalar-product u1 u2) * scalar-product w1 w2
    = scalar-product (vec-vec-Tensor u1 w1) (vec-vec-Tensor u2 w2)
    using Suc Cons-1 Cons-2 assms by auto
  then have zip v1 v2 = (a1, a2)#(zip u1 u2)
    using zip-Cons Cons-1 Cons-2 by auto
  then have Cons-4: scalar-product v1 v2 = (a1*a2) + (scalar-product u1 u2)
    unfolding scalar-product-def scalar-prodI-def by auto
  then have f (scalar-product v1 v2) (scalar-product w1 w2)
    = ((a1*a2) + (scalar-product u1 u2)) * (scalar-product v1 w2)
    using(auto)
  then have ... = (((a1*a2) * (scalar-product v1 w2))
    + ((scalar-product u1 u2) * (scalar-product w1 w2))
    using plus-right-distributivity by auto
  then have Cons-5: ... = (((a1*a2) * (scalar-product v1 w2))
    + scalar-product (vec-vec-Tensor u1 w1) (vec-vec-Tensor u2 w2)
    using Cons-3 by auto
  then have Cons-6: ... = (scalar-product (times a1 w1) (times a2 w2))
    + scalar-product (vec-vec-Tensor u1 w1) (vec-vec-Tensor u2 w2)
    using assms effective-scalar-product-times by auto
  then have scalar-product (vec-vec-Tensor v1 w1) (vec-vec-Tensor v2 w2)
    = scalar-product (vec-vec-Tensor (a1 # u1) w1) (vec-vec-Tensor (a2 # u2) w2)
    using Cons-1 Cons-2 by auto
moreover have \((\text{vec-vec-Tensor} (a1 \# u1) w1) = (\text{times} a1 w1) @ (\text{vec-vec-Tensor} u1 w1)\)
using \text{vec-vec-Tensor.simps by auto}
moreover have \((\text{vec-vec-Tensor} (a2 \# u2) w2) = (\text{times} a2 w2) @ (\text{vec-vec-Tensor} u2 w2)\)
using \text{vec-vec-Tensor.simps by auto}
ultimately have Cons-7: scalar-product \((\text{vec-vec-Tensor} v1 w1) (\text{vec-vec-Tensor} v2 w2)\)
= scalar-product (((\text{times} a1 w1) @ (\text{vec-vec-Tensor} u1 w1))
((\text{times} a2 w2) @ (\text{vec-vec-Tensor} u2 w2)))
by auto
moreover have \(\text{length} (\text{vec-vec-Tensor} u2 w2) = \text{length} (\text{vec-vec-Tensor} u1 w1)\)
using \text{assms by (metis Cons-1 Cons-2 vec-vec-Tensor-length)}
moreover have \(\text{length} (\text{times} a1 w1) = \text{length} (\text{times} a2 w2)\)
using \text{assms by (metis preserving-length)}
ultimately have scalar-product (((\text{times} a1 w1) @ (\text{vec-vec-Tensor} u1 w1))
((\text{times} a2 w2) @ (\text{vec-vec-Tensor} u2 w2)) =
(scalar-product (\text{times} a1 w1) (\text{times} a2 w2)) +
scalar-product (\text{vec-vec-Tensor} u1 w1) (\text{vec-vec-Tensor} u2 w2)
using \text{effective-scalar-product-append by auto}
then show \(\text{thesis}\)
using Cons-6 Cons-7 \(a1 * a2 + \text{scalar-product} u1 u2 \ast \text{scalar-product} w1 w2\)
= \(a1 \ast a2 \ast \text{scalar-product} w1 w2\)
+ \(\text{scalar-product} u1 u2 \ast \text{scalar-product} w1 w2\))
by \(\text{metis Cons-3 Cons-4}\)
qed
then show \(\text{case by auto}\)
qed

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

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

76
moreover have \((A!j) \in \text{set } A\)

using \text{assms(2) by auto}

ultimately have \(2:\text{length } (A!j) = \text{nr}\)

by \text{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:}

\text{fixes } A1 A2 B1 B2 i j

\text{assumes wf1: mat (row-length A1) (length A1) A1}

\text{and wf2: mat (row-length A2) (length A2) A2}

\text{and wf3: mat (row-length B1) (length B1) B1}

\text{and wf4: mat (row-length B2) (length B2) B2}

\text{and matchAA: length A1 = row-length A2}

\text{and matchBB: length B1 = row-length B2}

\text{and non-Nil: (A1 \neq \text{[]}) \land (A2 \neq \text{[]} \land (B1 \neq \text{[]} \land (B2 \neq \text{[]}))}

\text{and i:i<(row-length A1)*(row-length B1) and j:j< (length A2)*(length B2)}

\text{shows length (row A1 (i div (row-length B1)))}

= length (\text{col } A2 (j div (length A2))

\text{and length (row B1 (i mod (row-length B1)))}

= length (\text{col } B2 (j mod (length B2))

\text{proof --}

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

using \text{i by (metis div-left-ineq)}

\text{then have 1:length (row A1 (i div (row-length B1))) = length A1}

unfolding \text{row-def by auto}

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

using \text{j by (metis div-left-ineq)}

\text{then have 2:length (col A2 (j div (length B2))) = row-length A2}

using \text{row-length-constant wf2 unfolding \text{col-def by auto}}

with \text{1 matchAA \text{show} length (row A1 (i div (row-length B1)))=length (\text{col A2 (j div (length B2))})}

by \text{auto}

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

using \text{i by (metis less-nat-zero-code mod-less-divisor mult-is-0 neq0-conv)}

\text{then have 2:length (row B1 (i mod (row-length B1))) = length B1}

unfolding \text{row-def by auto}

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

using \text{j by (metis less-nat-zero-code mod-less-divisor mult-is-0 neq0-conv)}

\text{then have length (\text{col B2 (j mod (length B2))}) = row-length B2}

using \text{row-length-constant wf4 unfolding \text{col-def by auto}}

with \text{2 matchBB \text{show} length (row B1 (i mod (row-length B1))) = length (\text{col B2 (j mod (length B2))})
by auto

**Lemma** effective-row-col-match: assumes matrix-match $A_1 A_2 B_1 B_2$

shows $\forall i j . ((i < (\text{row-length } A_1) \ast (\text{row-length } B_1)) \land (j < (\text{length } A_2) \ast (\text{length } B_2)))$

$\longrightarrow \text{length} \ (\text{row } A_1 \ (i \div (\text{row-length } B_1))) = \text{length} \ (\text{col } A_2 \ (j \div (\text{length } B_2)))$

$\forall i j . ((i < (\text{row-length } A_1) \ast (\text{row-length } B_1)) \land (j < (\text{length } A_2) \ast (\text{length } B_2)))$

$\longrightarrow \text{length} \ (\text{row } B_1 \ (i \mod (\text{row-length } B_1))) = \text{length} \ (\text{col } B_2 \ (j \mod (\text{length } B_2)))$

using assms row-col-match unfolding matrix-match-def by auto

**Theorem** prelim-element-match:

matrix-match $A_1 A_2 B_1 B_2 \Longrightarrow (\forall i j . ((i < (\text{row-length } A_1) \ast (\text{row-length } B_1)) \land (j < (\text{length } A_2) \ast (\text{length } B_2)))$

$\longrightarrow (\ (A_1 \circ A_2) \circ (B_1 \circ B_2))_{lj} \ i = ((A_1 \otimes B_1) \circ (A_2 \otimes B_2))_{lj} \ i)$

**Proof**

**Assume** assms: matrix-match $A_1 A_2 B_1 B_2$

**Have** 1: matrix-match $A_1 A_2 B_1 B_2$

using assms matrix-compose-cond-def by auto

then have 2:

$\forall i j . ((i < (\text{row-length } A_1) \ast (\text{row-length } B_1)) \land (j < (\text{length } A_2) \ast (\text{length } B_2)))$

$\longrightarrow (\ (A_1 \circ A_2) \circ (B_1 \circ B_2))_{lj} \ i = (\ (A_1 \otimes B_1) \circ (A_2 \otimes B_2))_{lj} \ i$

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

moreover from 1 have 3: $\forall i j . ((i < (\text{row-length } A_1) \ast (\text{row-length } B_1)) \land (j < (\text{length } A_2) \ast (\text{length } B_2)))$

$\longrightarrow \ (A_1 \otimes B_1) \circ (A_2 \otimes B_2))_{lj} \ i = \text{vec-vec-Tensor} \ (\text{row } A_1 \ (i \div \text{row-length } B_1)) \ (\text{row } B_1 \ (i \mod \text{row-length } B_1))$

using vec-vec-Tensor (col $A_2$ (j div length $B_2$)) (col $B_2$ (j mod length $B_2$)) by auto

have $\forall i j . ((i < (\text{row-length } A_1) \ast (\text{row-length } B_1)) \land (j < (\text{length } A_2) \ast (\text{length } B_2)))$

$\longrightarrow \text{length} \ (\text{row } A_1 \ (i \div (\text{row-length } B_1)))$

$= \text{length} \ (\text{col } A_2 \ (j \div (\text{length } B_2)))$

and $\forall i j . ((i < (\text{row-length } A_1) \ast (\text{row-length } B_1)) \land (j < (\text{length } A_2) \ast (\text{length } B_2)))$

$\longrightarrow \text{length} \ (\text{row } B_1 \ (i \mod (\text{row-length } B_1)))$

$= \text{length} \ (\text{col } B_2 \ (j \mod (\text{length } B_2)))$

78
using assms effective-row-col-match by auto
then have \( \forall i \ j. \ ((i < (\text{row-length } A1) \ast (\text{row-length } B1)) \land (j < (\text{length } A2) \ast (\text{length } B2))) \)
\[ \rightarrow \]
\( (\text{scalar-product} \ (\text{row } A1 \ (i \div (\text{row-length } B1))) \ (\text{col } A2 \ (j \div (\text{length } B2)))) \)
\* (\text{scalar-product} \ (\text{row } B1 \ (i \mod (\text{row-length } B1))) \ (\text{col } B2 \ (j \mod (\text{length } B2)))) \)
\[ = \]
\( \text{scalar-product} \ (\text{vec-vec-Tensor} \ (\text{row } A1 \ (i \div (\text{row-length } B1))) \ (\text{col } B2 \ (j \mod (\text{length } B2)))) \)
\( = \]
\( \text{scalar-product} \ (\text{vec-vec-Tensor} \ (\text{row } B1 \ (i \mod (\text{row-length } B1))) \ (\text{col } B2 \ (j \mod (\text{length } B2)))) \)
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)).
\]
\( (((A1 \circ A2) \circ (B1 \circ B2))!j!i) \)
\[ = \]
\( ((A1 \circ B1) \circ (A2 \circ B2))!j!i \)
using prelim-element-match by auto

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

theorem tensor-compose-condn:
assumes wf1: \( \mat nr nc ((A1 \circ A2) \circ (B1 \circ B2)) \)
and wf2: \( \mat nr nc ((A1 \circ B1) \circ (A2 \circ B2)) \)
and wf3: \( \forall j < nc. \forall i < nr. (((A1 \circ A2) \circ (B1 \circ B2))!j!i) = ((A1 \circ B1) \circ (A2 \circ B2))!j!i \)
shows \( (((A1 \circ A2) \circ (B1 \circ B2)) = ((A1 \circ B1) \circ (A2 \circ B2)) \)
using application wf1 wf2 wf3 by blast

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

theorem distributivity:
assumes matrix-match A1 A2 B1 B2
shows \( ((A1 \circ A2) \circ (B1 \circ B2)) = ((A1 \circ B1) \circ (A2 \circ B2)) \)
proof –
let \(?nr = ((\text{row-length } A1) \ast (\text{row-length } B1)) \)
let \(?nc = ((\text{length } A2) \ast (\text{length } B2)) \)
have \( \mat(?nr ?nc ((A1 \circ A2) \circ (B1 \circ B2))) \)
by (metis assms effective-tensor-compose-distribution1)

moreover have mat ?nr ?nc \((A1 \otimes B1) \circ (A2 \otimes B2)\)
  using assms by (metis effective-tensor-compose-distribution2)

moreover have \(\forall j<\text{?nc}. \forall i<\text{?nr}.\)
  \(((A1 \circ A2) \otimes (B1 \circ B2))![j]![i] = ((A1 \otimes B1) \circ (A2 \otimes B2))![j]![i]\)
  using element-match assms by auto

ultimately show ?thesis
  using application by blast

qed

end