Tensor Product of Matrices

T.V.H. Prathamesh

February 22, 2016

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

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

1 Tensor Product of Matrices

```
theory Matrix-Tensor
imports ../Matrix/Utility ../Matrix/Matrix-Arith
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*
⟨*proof*⟩

lemma *times-vector-id*: *times v* [*id*] = [*v*]
⟨*proof*⟩

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

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*
⟨*proof*⟩

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

theorem *vec-vec-Tensor-length* :
(*length*(*vec-vec-Tensor x y*)) = (*length x*)∗(*length y*)
⟨*proof*⟩

theorem *vec-length*: **assumes** *vec m x* **and** *vec n y*
shows *vec* (*m*∗*n*) (*vec-vec-Tensor x y*)
⟨*proof*⟩

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

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

primrec *vec-mat-Tensor*::'*a vec* ⇒ '*a mat* ⇒'*a mat*
where
vec-mat-Tensor xs [] = []|
vec-mat-Tensor xs (*ys*#*yss*) = (*vec-vec-Tensor xs ys*)#(*vec-mat-Tensor xs yss*)

lemma *vec-mat-Tensor-vector-id*: *vec-mat-Tensor* [*id*] *v* = *v*
⟨*proof*⟩

lemma *vec-mat-Tensor-matrix-id*: *vec-mat-Tensor v* [[*id*]] = [*v*]
⟨*proof*⟩

theorem *vec-mat-Tensor-length*:
length(*vec-mat-Tensor xs ys*) = *length ys*
⟨*proof*⟩

theorem *length-matrix*:
assumes *mat nr nc* (*y*#*ys*) **and** *length v* = *k*
and (*vec-mat-Tensor v* (*y*#*ys*) = *x*#*xs*)
shows (*vec* (*nr*∗*k*) *x*)
⟨*proof*⟩

lemma *matrix-set-list*:
assumes *mat nr nc M*
and *length v* = *k*
and *x* ∈ *set M*
shows ∃ *ys*.∃ *zs*.(*ys*@*x*#*zs* = *M*)
⟨*proof*⟩

primrec *reduct* :: '*a mat* ⇒ '*a mat*
where
reduct [] = []
|*reduct* (*x*#*xs*) = *xs*

lemma *length-reduct*:
assumes *m* ≠ []
shows *length* (*reduct m*) +*1* = (*length m*)
⟨*proof*⟩

lemma *mat-empty-column-length*: **assumes** *mat nr nc M* **and** *M* = []
shows *nc* = *0*
⟨*proof*⟩

lemma *vec-uniqueness*:
assumes *vec m v*
and *vec n v*
shows *m* = *n*
⟨*proof*⟩

lemma *mat-uniqueness*:
assumes *mat nr1 nc M*
and *mat nr2 nc M* **and** $z = hd\ M$ **and** $M \neq []$
shows $(\forall x{\in}(set\ M).(nr1 = nr2))$
$\langle proof \rangle$

lemma *mat-empty-row-length*: **assumes** *mat nr nc M* **and** $M = []$
shows *mat 0 nc M*
$\langle proof \rangle$

abbreviation *null-matrix*::*'a list list*
where
null-matrix $\equiv$ [*Nil*]

lemma *null-mat*:*null-matrix* = [[]]
$\langle proof \rangle$

lemma *zero-matrix*: *mat 0 0* [] $\langle proof \rangle$

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

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

lemma *row-length-Nil*:
row-length [] =*0*
$\langle proof \rangle$

lemma *row-length-Null*:
row-length [[]] =*0*
$\langle proof \rangle$

lemma *row-length-vect-mat*:
row-length (*vec-mat-Tensor v m*) = *length v*$*$(*row-length m*)
$\langle proof \rangle$

Tensor is the tensor product of matrices

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

lemma *Tensor-null*: *xs* $\otimes$[] = []
$\langle proof \rangle$

Tensor commutes with left and right identity

lemma *Tensor-left-id*: [[*id*]] $\otimes$ *xs* = *xs*

$\langle \mathit{proof} \rangle$

lemma *Tensor-right-id*: $\mathit{xs} \otimes [[\mathit{id}]] = \mathit{xs}$
$\langle \mathit{proof} \rangle$

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

lemma *row-length-mat*:
$(\mathit{row\text{-}length}\ (m1 \otimes m2)) = (\mathit{row\text{-}length}\ m1) * (\mathit{row\text{-}length}\ m2)$
$\langle \mathit{proof} \rangle$

lemma *hd-set*:**assumes** $x \in \mathit{set}\ (a \# M)$ **shows** $(x = a) \vee (x \in (\mathit{set}\ M))$
$\langle \mathit{proof} \rangle$

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

theorem *matrix-row-length*:
assumes $\mathit{mat}\ \mathit{nr}\ \mathit{nc}\ M$
shows $\mathit{mat}\ (\mathit{row\text{-}length}\ M)\ (\mathit{length}\ M)\ M$
$\langle \mathit{proof} \rangle$

lemma *reduct-matrix*:
assumes $\mathit{mat}\ (\mathit{row\text{-}length}\ (a \# M))\ (\mathit{length}\ (a \# M))\ (a \# M)$
shows $\mathit{mat}\ (\mathit{row\text{-}length}\ M)\ (\mathit{length}\ M)\ M$
$\langle \mathit{proof} \rangle$

theorem *well-defined-vec-mat-Tensor*:
$(\mathit{mat}\ (\mathit{row\text{-}length}\ M)\ (\mathit{length}\ M)\ M) \Longrightarrow$
$(\mathit{mat}$
$((\mathit{row\text{-}length}\ M) * (\mathit{length}\ v))$
$(\mathit{length}\ M)$
$(\mathit{vec\text{-}mat\text{-}Tensor}\ v\ M))$
$\langle \mathit{proof} \rangle$

The following theorem gives length of tensor product of two matrices

lemma *length-Tensor*: $(\mathit{length}\ (M1 \otimes M2)) = (\mathit{length}\ M1) * (\mathit{length}\ M2)$
$\langle \mathit{proof} \rangle$

lemma *append-reduct-matrix*:
$(\mathit{mat}\ (\mathit{row\text{-}length}\ (M1 @ M2))\ (\mathit{length}\ (M1 @ M2))\ (M1 @ M2))$
$\Longrightarrow (\mathit{mat}\ (\mathit{row\text{-}length}\ M2)\ (\mathit{length}\ M2)\ M2)$
$\langle \mathit{proof} \rangle$

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

theorem *well-defined-Tensor*:
$(mat\ (row\text{-}length\ M1)\ (length\ M1)\ M1)$
$\wedge\ (mat\ (row\text{-}length\ M2)\ (length\ M2)\ M2)$
$\Longrightarrow(mat\ ((row\text{-}length\ M1)*(row\text{-}length\ M2))\ ((length\ M1)*(length\ M2))\ (M1\otimes M2))$
$\langle proof\rangle$

theorem *effective-well-defined-Tensor*:
assumes $(mat\ (row\text{-}length\ M1)\ (length\ M1)\ M1)$
and $(mat\ (row\text{-}length\ M2)\ (length\ M2)\ M2)$
shows mat
$((row\text{-}length\ M1)*(row\text{-}length\ M2))$
$((length\ M1)*(length\ M2))$
$(M1\otimes M2)$
$\langle proof\rangle$

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

theorem *times-elements*:
$\forall\, i.((i<(length\ v)) \longrightarrow (times\ a\ v)!i = f\ a\ (v!i))$
$\langle proof\rangle$

lemma *simpl-times-elements*:
assumes $(i<length\ xs)$
shows $((i<(length\ v)) \longrightarrow (times\ a\ v)!i = f\ a\ (v!i))$
$\langle proof\rangle$

lemma *append-simpl*: $i<(length\ xs) \longrightarrow (xs@ys)!i = (xs!i)$
$\langle proof\rangle$

lemma *append-simpl2*: $i \geq(length\ xs) \longrightarrow (xs@ys)!i = (ys!(i-\ (length\ xs)))$
$\langle proof\rangle$

lemma *append-simpl3*:
assumes $i > (length\ y)$
shows $(i <((length\ (z\#zs))*(length\ y)))$
$\longrightarrow (i - (length\ y))< (length\ zs)*(length\ y)$
$\langle proof\rangle$

lemma *append-simpl4*:
$(i > (length\ y))$
$\longrightarrow ((i <((length\ (z\#zs))*(length\ y))))$
$\longrightarrow ((i - (length\ y))< (length\ zs)*(length\ y))$
$\langle proof\rangle$

lemma *vec-vec-Tensor-simpl*:

i<(*length y*) ⟶ (*vec-vec-Tensor* (*z*#*zs*) *y*)!*i* = (*times z y*)!*i*
⟨*proof*⟩

lemma *vec-vec-Tensor-simpl2*:
(*i* ≥ (*length y*))
⟶ ((*vec-vec-Tensor* (*z*#*zs*) *y*)!*i* = (*vec-vec-Tensor zs y*)!(*i*− (*length y*)))
⟨*proof*⟩

lemma *division-product*:
assumes (*b*::*int*)>*0*
and *a* ≥*b*
shows (*a div b*) = ((*a* − *b*) *div b*) + *1*
⟨*proof*⟩

lemma *int-nat-div*:
(*int a*) *div* (*int b*) = *int* ((*a*::*nat*) *div b*)
⟨*proof*⟩

lemma *int-nat-eq*:
assumes *int* (*a*::*nat*) = *int b*
shows *a* = *b*
⟨*proof*⟩

lemma *nat-div*:
assumes (*b*::*nat*) > *0*
and *a* > *b*
shows (*a div b*) = ((*a* − *b*) *div b*) + *1*
⟨*proof*⟩

lemma *mod-eq*:
(*m*::*int*) *mod n* = (*m* + (−*1*)∗*n*) *mod n*
⟨*proof*⟩

lemma *nat-mod-eq*: (*int* (*m*::*nat*)) *mod* (*int n*) = *int* (*m mod n*)
⟨*proof*⟩

lemma *nat-mod*:
assumes (*m*::*nat*) > *n*
shows (*m*::*nat*) *mod n* = (*m* −*n*) *mod n*
⟨*proof*⟩

lemma *logic*:
assumes *A* ⟶ *B*
and ¬*A* ⟶ *B*
shows *B*
⟨*proof*⟩

theorem *vec-vec-Tensor-elements*:

assumes $(y \neq [])$
shows
$\forall i.((i<((\mathit{length}\ x)*(\mathit{length}\ y)))$
$\longrightarrow ((\mathit{vec\text{-}vec\text{-}Tensor}\ x\ y)!i)$
$= f\ (x!(i\ \mathit{div}\ (\mathit{length}\ y)))\ (y!(i\ \mathit{mod}\ (\mathit{length}\ y))))$
⟨*proof*⟩

a few more results that will be used later on

lemma *nat-int*: $\mathit{nat}\ (\mathit{int}\ x + \mathit{int}\ y) = x + y$
⟨*proof*⟩

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

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

lemma *row-length-eq*:
$(\mathit{mat}\ \ (\mathit{row\text{-}length}\ (a\#b\#N))\ \ (\mathit{length}\ (a\#b\#N))\ (a\#b\#N))$
$\longrightarrow$
$(\mathit{row\text{-}length}\ (a\#b\#N) = (\mathit{row\text{-}length}\ (b\#N)))$
⟨*proof*⟩

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

theorem *vec-mat-Tensor-elements*:
$\forall i.\forall j.$
$(((i<((\mathit{length}\ v)*(\mathit{row\text{-}length}\ M)))$
$\wedge(j < (\mathit{length}\ M)))$
$\wedge(\mathit{mat}\ (\mathit{row\text{-}length}\ M)\ (\mathit{length}\ M)\ M)$
$\longrightarrow ((\mathit{vec\text{-}mat\text{-}Tensor}\ v\ M)!j!i)$
$= f\ (v!(i\ \mathit{div}\ (\mathit{row\text{-}length}\ M)))\ (M!j!(i\ \mathit{mod}\ (\mathit{row\text{-}length}\ M))))$
⟨*proof*⟩

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
$\forall i.\forall j.(((i<((\mathit{row\text{-}length}\ M1)*(\mathit{row\text{-}length}\ M2)))$
$\wedge(j < (\mathit{length}\ M1)*(\mathit{length}\ M2)))$
$\wedge(\mathit{mat}\ (\mathit{row\text{-}length}\ M1)\ (\mathit{length}\ M1)\ M1)$
$\wedge(\mathit{mat}\ (\mathit{row\text{-}length}\ M2)\ (\mathit{length}\ M2)\ M2)$
$\longrightarrow ((M1 \otimes M2)!j!i) =$
f
$(M1!(j\ \mathit{div}\ (\mathit{length}\ M2))!(i\ \mathit{div}\ (\mathit{row\text{-}length}\ M2)))$

```
                    (M2!(j mod length M2)!(i mod (row-length M2))))
 ⟨proof⟩
```

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))))
 ⟨proof⟩

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)))
 ⟨proof⟩
```

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⟩

lemma div-right-ineq:
 assumes (x::nat) < y∗z
 shows  (x div y) < z
 ⟨proof⟩
```

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

```
lemma col-vec-mat-Tensor-prelim:
  ∀ j.(j < (length M)
     ⟶
      col (vec-mat-Tensor v M) j = vec-vec-Tensor v (col M j))
 ⟨proof⟩

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)
  ⟨proof⟩

lemma  col-formula:
```

fixes M1 **and** M2
shows ∀ j.((j < (length M1)∗(length M2))
        ∧ (mat (row-length M1) (length M1) M1)
        ∧ (mat (row-length M2) (length M2) M2)
        ⟶ col (M1 ⊗ M2) j
              = vec-vec-Tensor
                    (col M1 (j div length M2))
                    (col M2 (j mod length M2)))
⟨proof⟩

lemma *row-Cons*:row (v#M) i = (v!i)#(row M i)
⟨proof⟩

lemma *row-append*:row (A@B)i = (row A i)@(row B i)
⟨proof⟩

lemma *row-empty*:row [] i = []
⟨proof⟩

lemma *vec-vec-Tensor-right-empty*:vec-vec-Tensor x [] = []
⟨proof⟩

lemma vec-mat-Tensor v ([]#[]) = [[]]
⟨proof⟩

lemma i<0 ⟶ [[]!i] = []
⟨proof⟩

lemma *row-vec-mat-Tensor-prelim*:
∀ i.
    ((i < (length v)∗(row-length M))∧(mat nr (length M) M)
     ⟶ row (vec-mat-Tensor v M) i
          = times (v!(i div row-length M)) (row M (i mod row-length M)))
⟨proof⟩

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

lemma *row-formula*:
fixes M1 **and** M2
shows ∀ i.((i < (row-length M1)∗(row-length M2))
        ∧(mat (row-length M1) (length M1) M1)
        ∧(mat (row-length M2) (length M2) M2)
           ⟶ row (M1 ⊗ M2) i
                 = vec-vec-Tensor
                       (row M1 (i div row-length M2))
                       (row M2 (i mod row-length M2)))
⟨proof⟩

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*))
⟨*proof*⟩

lemma *alt-effective-matrix-tensor-elements*:
(((*i*<((*row-length M2*)∗(*row-length M3*)))
∧(*j* < (*length M2*)∗(*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*))))
⟨*proof*⟩

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*))
⟨*proof*⟩

lemma ((*x*::*nat*) *div y*) *div z* = (*x div* (*y*∗*z*))
⟨*proof*⟩

lemma (¬((*a*::*nat*) < *b*)) ⟹ (*a* ≥ *b*)
⟨*proof*⟩

lemma *not-null*: *xs* ≠ [] ⟹ ∃ *y ys*. *xs* = *y*#*ys*
⟨*proof*⟩

lemma (*y*::*nat*) ≠ *0* ⟹ (*x mod y*) < *y*
⟨*proof*⟩

lemma *mod-prop1*:((*a*::*nat*) *mod* (*b*∗*c*)) *mod c* = (*a mod c*)
⟨*proof*⟩

lemma *mod-div-relation*:((*a*::*nat*) *mod* (*b*∗*c*)) *div c* = (*a div c*) *mod b*
⟨*proof*⟩

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

lemma *associativity*:
fixes *M1 M2 M3*
shows
(*mat* (*row-length M1*) (*length M1*) *M1*)
∧ (*mat* (*row-length M2*) (*length M2*) *M2*)

∧ (*mat* (*row-length M3*) (*length M3*) *M3*)
⟹
M1 ⊗ (*M2* ⊗ *M3*) = (*M1* ⊗ *M2*) ⊗ *M3* (**is** *?x* ⟹*?l* = *?r*)
⟨*proof*⟩

end

lemma ⋀(*a*::*nat*) *b*.(*times a b*) =(*times b a*)
⟨*proof*⟩

1.2 Associativity and Distributive properties

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

context *plus-mult*
begin

lemma fixes *M1 M2 M3*
shows (*mat* (*row-length M1*) (*length M1*) *M1*)
∧(*mat* (*row-length M2*) (*length M2*) *M2*)
∧(*mat* (*row-length M3*) (*length M3*) *M3*)
⟹ (*M1* ⊗ (*M2* ⊗ *M3*)) = ((*M1* ⊗ *M2*) ⊗ *M3*)
⟨*proof*⟩

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

lemma *matrix-index*:
assumes *wf1*: *mat* (*row-length m1*) *n m1*
and *wf2*: *mat n nc m2*
and *i*: *i* < (*row-length m1*)
and *j*: *j* < *nc*
shows *matrix-mult m1 m2* ! *j* ! *i*
= *scalar-product* (*row m1 i*) (*col m2 j*)
⟨*proof*⟩

lemma *unique-row-col*:
assumes *mat nr1 nc1 M* **and** *mat nr2 nc2 M* **and** *M* ≠ []
shows *nr1* = *nr2* **and** *nc1* = *nc2*
⟨*proof*⟩

lemma *matrix-mult-index*:
assumes *m1* ≠ []
and *wf1*: *mat nr n m1*
and *wf2*: *mat n nc m2*
and *i*: *i* < *nr*
and *j*: *j* < *nc*
shows *matrix-mult m1 m2* ! *j* ! *i* = *scalar-product* (*row m1 i*) (*col m2 j*)
⟨*proof*⟩

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

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

lemma *non-empty-mat-mult*:
assumes *wf1*:*mat nr n A*
and *wf2*:*mat n nc B*
and *A* ≠ [] **and** *B* ≠ []
shows *A* ∘ *B* ≠ []

⟨*proof*⟩

lemma *tensor-compose-distribution1*:
assumes *wf1*:*mat* (*row-length A1*) (*length A1*) *A1*
and *wf2*:*mat* (*row-length A2*) (*length A2*) *A2*
and *wf3*:*mat* (*row-length B1*) (*length B1*) *B1*
and *wf4*:*mat* (*row-length B2*) (*length B2*) *B2*
and *matchAA*:*length A1* = *row-length A2*
and *matchBB*:*length B1* = *row-length B2*
and *non-Nil*:(*A1* ≠ [])∧(*A2* ≠ [])∧(*B1* ≠ [])∧(*B2* ≠ [])
shows *mat* ((*row-length A1*)∗(*row-length B1*))
((*length A2*)∗(*length B2*))
((*A1*∘*A2*)⊗(*B1*∘*B2*))
⟨*proof*⟩

lemma *effective-tensor-compose-distribution1*:
matrix-match A1 A2 B1 B2 ⟹ *mat* ((*row-length A1*)∗(*row-length B1*))
((*length A2*)∗(*length B2*))
((*A1*∘*A2*)⊗(*B1*∘*B2*))
⟨*proof*⟩

lemma *tensor-compose-distribution2*:
assumes *wf1*:*mat* (*row-length A1*) (*length A1*) *A1*
and *wf2*:*mat* (*row-length A2*) (*length A2*) *A2*
and *wf3*:*mat* (*row-length B1*) (*length B1*) *B1*
and *wf4*:*mat* (*row-length B2*) (*length B2*) *B2*
and *matchAA*:*length A1* = *row-length A2*
and *matchBB*:*length B1* = *row-length B2*
and *non-Nil*:(*A1* ≠ [])∧(*A2* ≠ [])∧(*B1* ≠ [])∧(*B2* ≠ [])
shows *mat* ((*row-length A1*)∗(*row-length B1*))
((*length A2*)∗(*length B2*))
((*A1* ⊗ *B1*) ∘(*A2* ⊗*B2*))
⟨*proof*⟩

theorem *tensor-non-empty*: **assumes** *A* ≠ [] **and** *B* ≠ []
shows *A* ⊗ *B* ≠ []
⟨*proof*⟩

theorem *non-empty-distribution*:
assumes *mat nr1 n1 A1*
and *mat n1 nc1 A2*
and *mat nr2 n2 B1*
and *mat n2 nc2 B2*
and *A1* ≠ [] **and** *B1* ≠ [] **and** *A2* ≠ [] **and** *B2* ≠ []
shows ((*A1*∘*A2*)⊗(*B1*∘*B2*)) ≠ []
⟨*proof*⟩

lemma *effective-tensor-compose-distribution2*:*matrix-match A1 A2 B1 B2* ⟹

mat ((*row-length A1*)∗(*row-length B1*))
((*length A2*)∗(*length B2*))
((*A1* ⊗ *B1*) ∘(*A2* ⊗*B2*))
⟨*proof*⟩

theorem *effective-matrix-Tensor-elements*:
fixes *M1 M2 i j*
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*) = *f* (*M1*!(*j div* (*length M2*))!(*i div* (*row-length M2*)))
(*M2*!(*j mod length M2*)!(*i mod* (*row-length M2*)))
⟨*proof*⟩

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*)∗(*row-length M2*)).
∀ *j* < ((*length M1*)∗(*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*))))
⟨*proof*⟩

definition *matrix-compose-cond*::′*a mat* ⇒ ′*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*)∗(*row-length B1*))∧(*j*< (*length A2*)∗(*length B2*))

theorem *elements-matrix-distribution-1*:
assumes *wf1*:*mat* (*row-length A1*) (*length A1*) *A1*
and *wf2*:*mat* (*row-length A2*) (*length A2*) *A2*
and *wf3*:*mat* (*row-length B1*) (*length B1*) *B1*
and *wf4*:*mat* (*row-length B2*) (*length B2*) *B2*
and *matchAA*:*length A1* = *row-length A2*

and *matchBB*:*length B1* = *row-length B2*
and *non-Nil*:(*A1* ≠ [])∧(*A2* ≠ [])∧(*B1* ≠ [])∧(*B2* ≠ [])
and *i*<(*row-length A1*)∗(*row-length B1*) **and** *j*< (*length A2*)∗(*length B2*)
shows
((*matrix-mult A1 A2*)⊗(*matrix-mult B1 B2*))!*j*!*i*
 = *f* (*scalar-product* (*row A1* (*i div* (*row-length B1*)))
      (*col A2* (*j div* (*length B2*))))
   (*scalar-product* (*row B1* (*i mod* (*row-length B1*)))
      (*col B2* (*j mod* (*length B2*))))
⟨*proof*⟩

lemma *effective-elements-matrix-distribution1*:
matrix-compose-cond A1 A2 B1 B2 i j ⟹
((*matrix-mult A1 A2*)⊗(*matrix-mult B1 B2*))!*j*!*i*
 = *f* (*scalar-product* (*row A1* (*i div* (*row-length B1*))) (*col A2* (*j div* (*length B2*))))
   (*scalar-product* (*row B1* (*i mod* (*row-length B1*))) (*col B2* (*j mod* (*length B2*))))
⟨*proof*⟩

lemma *matrix-match-condn-1*:
matrix-match A1 A2 B1 B2
 ∧((*i*<(*row-length A1*)∗(*row-length B1*))
 ∧(*j*<(*length A2*)∗(*length B2*)))
 ⟹ ((*matrix-mult A1 A2*)⊗(*matrix-mult B1 B2*))!*j*!*i*
 = *f*
  (*scalar-product*
    (*row A1* (*i div* (*row-length B1*)))
    (*col A2* (*j div* (*length B2*))))
  (*scalar-product*
    (*row B1* (*i mod* (*row-length B1*)))
    (*col B2* (*j mod* (*length B2*))))
⟨*proof*⟩

lemma *effective-matrix-match-condn-1*:
assumes (*matrix-match A1 A2 B1 B2*)
shows ∀ *i j*.((*i*<(*row-length A1*)∗(*row-length B1*))
  ∧(*j*<(*length A2*)∗(*length B2*))
  ⟶ ((*A1* ∘ *A2*)⊗(*B1* ∘ *B2*))!*j*!*i*
   = *f*
    (*scalar-product*
     (*row A1* (*i div* (*row-length B1*)))
     (*col A2* (*j div* (*length B2*))))
    (*scalar-product*
     (*row B1* (*i mod* (*row-length B1*)))
     (*col B2* (*j mod* (*length B2*)))))
⟨*proof*⟩

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
((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)))
⟨proof⟩

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)))
 ⟨proof⟩


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))))
 ⟨proof⟩
```

lemma *zip-Nil*:*zip* [] [] = []
⟨*proof*⟩

lemma *zer-left-mult*:*f zer x* = *zer*
⟨*proof*⟩

lemma *zip-Cons*:(*length v* = *length w*) ⟹ *zip* (*a*#*v*) (*b*#*w*) = (*a*,*b*)#(*zip v w*)
⟨*proof*⟩

lemma *scalar-product-times*:
∀ *w1 w2*.(*length w1* = *length w2*) ∧(*length w1* = *n*) ⟶
(*f* (*x*∗*y*) (*scalar-product w1 w2*))
= (*scalar-product*
(*times x w1*)
(*times y w2*))
⟨*proof*⟩

lemma *effective-scalar-product-times*:
assumes (*length w1* = *length w2*)
shows (*f* (*x*∗*y*) (*scalar-product w1 w2*))
= (*scalar-product* (*times x w1*) (*times y w2*))
⟨*proof*⟩

lemma *zip-append*:(*length zs* = *length ws*)∧(*length xs* = *length ys*)
⟹ (*zip* (*xs*@*zs*) (*ys*@*ws*)) = (*zip xs ys*)@(*zip zs ws*)
⟨*proof*⟩

lemma *scalar-product-append*:
∀ *xs ys zs ws*.(*length zs* = *length ws*)
∧(*length xs* = *length ys*)
∧(*length xs* = *n*) ⟶
(*scalar-product* (*xs*@*zs*) (*ys*@*ws*))
= (*scalar-product xs ys*)
+(*scalar-product zs ws*)
⟨*proof*⟩

lemma *effective-scalar-product-append*:
assumes *length zs* = *length ws* **and** (*length xs* = *length ys*)
shows (*scalar-product* (*xs*@*zs*) (*ys*@*ws*)) = (*scalar-product xs ys*)+(*scalar-product zs ws*)
⟨*proof*⟩

lemma *scalar-product-distributivity*:
∀ *v1 v2 w1 w2*.((*length v1* = *length v2*)∧(*length v1* = *n*)∧ (*length w1* = *length w2*)

$\longrightarrow$ (*scalar-product v1 v2*)$*$(*scalar-product w1 w2*)
= *scalar-product* (*vec-vec-Tensor v1 w1*) (*vec-vec-Tensor v2 w2*))
⟨*proof*⟩

lemma *effective-scalar-product-distributivity*:
assumes *length v1* = *length v2* **and** *length w1* = *length w2*
shows (*scalar-product v1 v2*)$*$(*scalar-product w1 w2*)
= *scalar-product* (*vec-vec-Tensor v1 w1*) (*vec-vec-Tensor v2 w2*)
⟨*proof*⟩

lemma *row-length-constant*:**assumes** *mat nr nc A* **and** *j* < *length A*
shows *length* (*A*!*j*) = (*row-length A*)
⟨*proof*⟩

theorem *row-col-match*:
fixes *A1 A2 B1 B2 i j*
assumes *wf1*:*mat* (*row-length A1*) (*length A1*) *A1*
and *wf2*:*mat* (*row-length A2*) (*length A2*) *A2*
and *wf3*:*mat* (*row-length B1*) (*length B1*) *B1*
and *wf4*:*mat* (*row-length B2*) (*length B2*) *B2*
and *matchAA*:*length A1* = *row-length A2*
and *matchBB*:*length B1* = *row-length B2*
and *non-Nil*:(*A1* $\neq$ [])$\wedge$(*A2* $\neq$ [])$\wedge$(*B1* $\neq$ [])$\wedge$(*B2* $\neq$ [])
and *i*:*i*<(*row-length A1*)$*$(*row-length B1*) **and** *j*:*j*< (*length A2*)$*$(*length B2*)
shows *length* (*row A1* (*i div* (*row-length B1*)))
= *length* (*col A2* (*j div* (*length B2*)))
and *length* (*row B1* (*i mod* (*row-length B1*)))
= *length* (*col B2* (*j mod* (*length B2*)))
⟨*proof*⟩

lemma *effective-row-col-match*: **assumes** *matrix-match A1 A2 B1 B2*
shows $\forall$ *i j*. ((*i*<(*row-length A1*)$*$(*row-length B1*))$\wedge$(*j*<(*length A2*)$*$(*length B2*)))

$\longrightarrow$*length* (*row A1* (*i div* (*row-length B1*))) = *length* (*col A2* (*j div* (*length B2*)))
$\forall$ *i j*. ((*i*<(*row-length A1*)$*$(*row-length B1*))$\wedge$(*j*<(*length A2*)$*$(*length B2*)))
$\longrightarrow$*length* (*row B1* (*i mod* (*row-length B1*))) = *length* (*col B2* (*j mod* (*length B2*)))
⟨*proof*⟩

theorem *prelim-element-match*:
matrix-match A1 A2 B1 B2 $\Longrightarrow$ ($\forall$ *i j*.((*i*<(*row-length A1*)$*$(*row-length B1*))
$\wedge$(*j*<(*length A2*)$*$(*length B2*)))
$\longrightarrow$

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

theorem *element-match*:
matrix-match $A1\ A2\ B1\ B2 \Longrightarrow (\forall i < ((\textit{row-length}\ A1) * (\textit{row-length}\ B1)).$
$\forall j < ((\textit{length}\ A2) * (\textit{length}\ B2)).$
$(((A1 \circ A2) \otimes (B1 \circ B2))!j!i$
$= ((A1 \otimes B1) \circ (A2 \otimes B2))!j!i))$
$\langle proof \rangle$

lemma *application*: **fixes** *m1 m2*
shows $\forall\, m1\ m2. (\textit{mat}\ nr\ nc\ m1)$
$\wedge (\textit{mat}\ nr\ nc\ m2)$
$\wedge (\forall\ j < nc.\ \forall\ i < nr.\ m1\ !\ j\ !\ i = m2\ !\ j\ !\ i)$
$\longrightarrow (m1 = m2)$
$\langle proof \rangle$

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

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

theorem *distributivity*:
assumes *matrix-match A1 A2 B1 B2*
shows $((A1 \circ A2) \otimes (B1 \circ B2)) = ((A1 \otimes B1) \circ (A2 \otimes B2))$
$\langle proof \rangle$

end

end