

Fundamentals of CuTe Layout Algebra and Category-theoretic Interpretation

Colfax Research:

Jack Carlisle, Jay Shah, Reuben Stern, Paul VanKoughnett

April 18, 2026

1 CuTe layouts

2 CuTe layout algebra

3 Categories

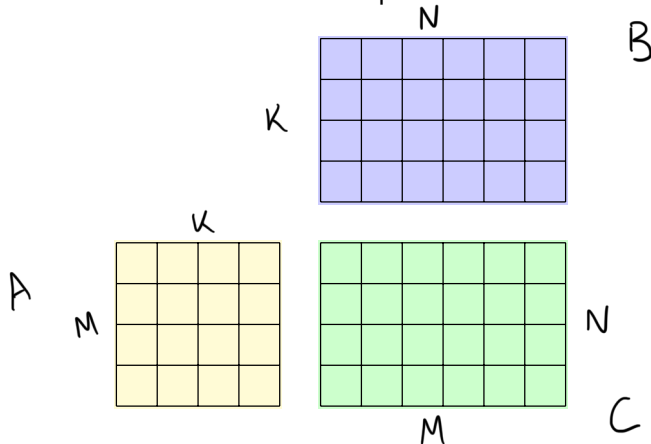
4 Categories of layouts

5 The composition algorithm

Expository

Original

Consider a GEMM kernel which computes $C = AB$.



The operands A , B and output C are *tensors*.

Tensor	Shape	
A	(M, K)	} Matrices
B	(K, N)	
C	(M, N)	

Higher rank tensors are common, too:

Tensor	Shape	
Q	(B, M, H, D)	
K	(B, N, H, D)	
V	(B, N, H, D)	← headdim
	↑ batch ↑ seqLen ↑ nheads	

In CuTe, a tensor is specified by an iterator and a layout:

$$L = S : D$$

\uparrow \uparrow
 Shape Stride

Example:

$$L = (4, 6) : (6, 1) \leftarrow 4 \times 6 \text{ matrix}$$

$$L = (8, 8) : (1, 0)$$

$$L = (5, 5) : (20, 2)$$

$$L = ((2, 3), (4, 2)) : ((4, 16), (1, 8))$$

Row major

$+1$ $+1$ contiguous

	0	1	2	3	4	5
$+b$	6	7	8	9	10	11
$+b$	12	13	14	15	16	17
$+b$	18	19	20	21	22	23

$$L = (4, 6) : (6, 1)$$

Column Major

contiguous $+4$ $+4$

	0	4	8	12	16	20
$+b$	1	5	9	13	17	21
	2	6	10	14	18	22
	3	7	11	15	19	23

$$L = (4, 6) : (1, 4)$$

Broadcast layout

0	0	0	0	0	0
1	1	1	1	1	1
2	2	2	2	2	2
3	3	3	3	3	3

Strided layout

0	1	2	3	4	5
12	13	14	15	16	17
24	25	26	27	28	29
36	37	38	39	40	41

Handwritten annotations: $+1$ above the first two columns with arrows pointing to the second and third columns, and $+12$ to the left of the first column with a curved arrow pointing to the second row.

$$L = (4, 6) : (1, 0)$$

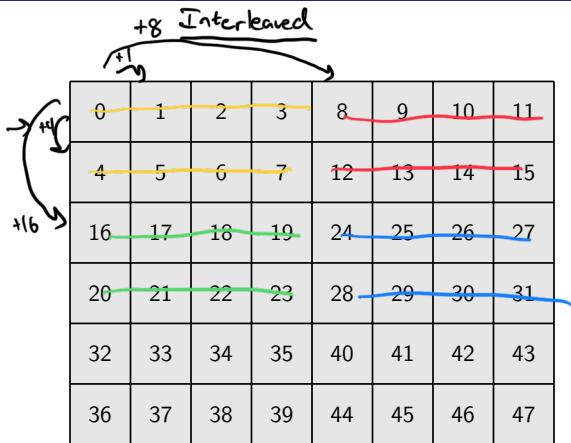
vector of
shape 4

\rightsquigarrow

mat. of
shape
 4×6

$$L = (4, 6) : (12, 1)$$

Each 2×4
tile is
contiguous in
memory



$$L = ((\underline{2}, \underline{3}), (\underline{4}, \underline{2})) : ((\underline{4}, \underline{16}), (\underline{1}, \underline{8}))$$

"Logical shape
 6×8 "

$$32 \text{ threads} \quad 4 \text{ values}$$

$$L = (\overbrace{(4, 8)}, \overbrace{(2, 2)}) : ((32, 1), (16, 8))$$

Row\Col	0	1	2	3	4	5	6	7	
0	T0: {c0, c1}		T1: {c0, c1}		T2: {c0, c1}		T3: {c0, c1}		
1	T4: {c0, c1}		T5: {c0, c1}		T6: {c0, c1}		T7: {c0, c1}		
2	→								
..	←								
7	T28: {c0, c1}		T29: {c0, c1}		T30: {c0, c1}		T31: {c0, c1}		
8	T0: {c2, c3}		T1: {c2, c3}		T2: {c2, c3}		T3: {c2, c3}		
9	T4: {c2, c3}		T5: {c2, c3}		T6: {c2, c3}		T7: {c2, c3}		
10	→								
..	←								
15	T28: {c2, c3}		T29: {c2, c3}		T30: {c2, c3}		T31: {c2, c3}		

%laneid:{fragments}

Figure: Accumulator layout for a `mma.sync` instruction

128 threads q_{wgs}

$$L = ((4, 8, 4), (2, 2, 8)) : ((128, 1, 16), (64, 8, 512))$$

(Thread, Value) \rightsquigarrow logical (M, N)

Thread
0

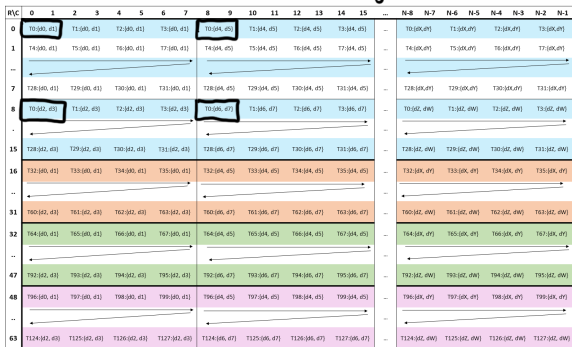
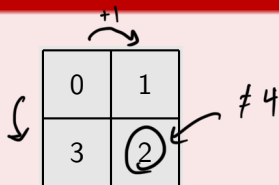


Figure: Accumulator layout for a wgmma instruction

Not all labellings correspond to a valid layout.

Non-example:

Consider



$$L = (2, 2) : \begin{matrix} 3 & 1 \\ & 1 \end{matrix} \begin{matrix} (?) \\ (?) \end{matrix}$$

Need layout with swizzle function to represent this mapping.

"Composed Layout"

These images depict the *coordinate function* of L .

Notation:

If S is a nested tuple, we write $\text{prof}\left(\left(\left(3,2\right),2\right)\right) = \text{prof}\left(\left(\left(1,1\right),4\right)\right)$

$$\underline{[0, S]} = \{X \mid \text{prof}(X) = \text{prof}(S) \text{ and } 0 \leq x_i < s_i\}.$$

Definition: (Coordinate function)

If $L = S : D$ is a layout, then the *coordinate function* of L is given by

$$[0, S] \xrightarrow{D} \mathbb{Z}$$

$$X \longmapsto D \cdot X$$

$$((4,0),2)$$

$$L = ((3,2),2) : ((4,0),2)$$

$$[0, S] \xrightarrow{D} \mathbb{Z}$$

$$((0,0),0) \longmapsto 4 * 0 + 0 * 0 + 2 * 0 = 0$$

$$((1,0),0) \longmapsto 4 * 1 + 0 * 0 + 2 * 0 = 4$$

$$((2,0),0) \longmapsto 4 * 2 + 0 * 1 + 2 * 0 = 8$$

$$((0,1),0) \longmapsto 4 * 0 + 0 * 1 + 2 * 0 = 0$$

$$((1,1),0) \longmapsto 4 * 1 + 0 * 0 + 2 * 0 = 4$$

$$((2,1),0) \longmapsto 4 * 2 + 0 * 0 + 2 * 0 = 8$$

$$((0,0),1) \longmapsto 4 * 0 + 0 * 1 + 2 * 1 = 2$$

$$((1,0),1) \longmapsto 4 * 1 + 0 * 1 + 2 * 1 = 6$$

$$((2,0),1) \longmapsto 4 * 2 + 0 * 0 + 2 * 1 = 10$$

$$((0,1),1) \longmapsto 4 * 0 + 0 * 0 + 2 * 1 = 2$$

$$((1,1),1) \longmapsto 4 * 1 + 0 * 1 + 2 * 1 = 6$$

$$((2,1),1) \longmapsto 4 * 2 + 0 * 1 + 2 * 1 = 10$$

$[0, S)$

Remark:

Coordinate function has

- Inputs: nested tuples,
- Outputs: integers.

Can't compose these!

Can provide more general inputs. First, we need a vocabulary for working with nested tuples.

Definition (by example):

Consider $S = (4, (5, 5), (2, (3, 3)))$:

- 1 The modes of S are $4, (5, 5), (2, (3, 3))$.
- 2 The entries of S are $4, 5, 5, 2, 3, 3$,
- 3 The rank of S is the number of modes $rank(S) = 3$
- 4 The length of S is the number of entries: $len(S) = 6$.
- 5 The size of S is the product of its entries: $size(S) = 1800$.

"compatible with"

Definition:

- 1 We say S refines S' , and write

$$S \longrightarrow S'$$

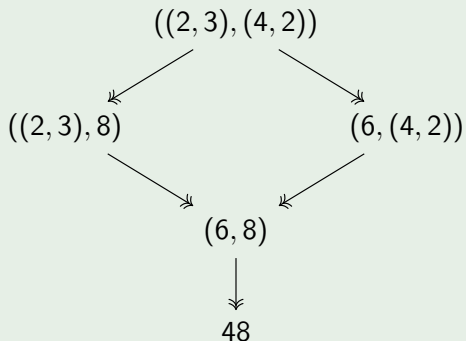
if S may be obtained from S' by replacing some of its entries with nested tuples of the same size.

$$\left((2,2), (5,5) \right) \longrightarrow (4, 25)$$

$$S \longrightarrow S'$$

Example: (Nested tuple refinements)

Nested tuples refined by $S = ((2, 3), (4, 2))$:



Definition:

If $S = (s_1, \dots, s_m)$ is a (nested) tuple of size N , then the *colexicographic isomorphism* is

$$[0, S) \xrightarrow{\cong} [0, N)$$

$$(x_1, \dots, x_m) \xrightarrow{\text{colex}} \sum_{i=1}^m s_1 \cdots s_{i-1} x_i$$

$$\left(\left[\frac{x}{s_1 \cdots s_{i-1}} \right] \bmod s_i \right)_{i=1}^m \xleftarrow{\text{colex}^{-1}} x$$

$$S = ((3, 2), 2)$$

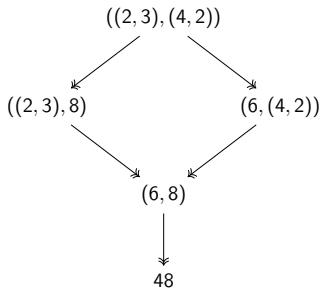
$$\begin{array}{l}
 [0, S) \xrightarrow{\text{colex}} [0, 12) \\
 ((0, 0), 0) \longmapsto 0 \\
 ((1, 0), 0) \longmapsto 1 \\
 ((2, 0), 0) \longmapsto 2 \\
 ((0, 1), 0) \longmapsto 3 \\
 ((1, 1), 0) \longmapsto 4 \\
 ((2, 1), 0) \longmapsto 5 \\
 ((0, 0), 1) \longmapsto 6 \\
 ((1, 0), 1) \longmapsto 7 \\
 ((2, 0), 1) \longmapsto 8 \\
 ((0, 1), 1) \longmapsto 9 \\
 ((1, 1), 1) \longmapsto 10 \\
 ((2, 1), 1) \longmapsto 11
 \end{array}$$

More generally, if S refines S' , then there is a colexicographic isomorphism

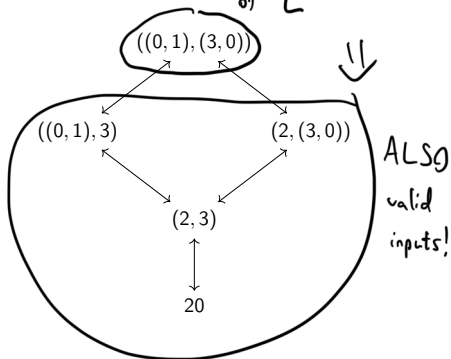
$$[0, S) \xrightarrow[\text{colex}]{\cong} [0, S')$$

Same formula applied “entrywise”.

Refinements
of $S = ((2,3), (4,2))$
y



valid input
↓ for coordinate fn
of L



We especially care about integer inputs.

Definition:

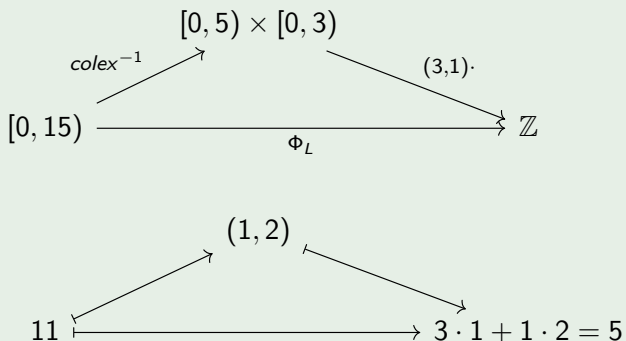
The *layout function* of $L = S : D$ is the composite

$$\begin{array}{ccc}
 & [0, S) & \\
 \text{colex}^{-1} \nearrow & & \searrow D \cdot \\
 [0, \text{size}(L)) & \xrightarrow{\Phi_L} & \mathbb{Z}
 \end{array}$$

$$\begin{array}{ccc}
 & (x_1, \dots, x_m) & \\
 \nearrow & & \searrow \\
 x & \xrightarrow{\quad} & d_1 x_1 + \dots + d_m x_m
 \end{array}$$

Example:

$$L = (5, 3) : (3, 1)$$



Interleaved layouts arise naturally via algebraic operations.

- 1 Logical division $A \oslash B$
- 2 Logical product $A \otimes B$
- 3 Variants of these

$$A = (6, 8) : (1, 6)$$

0	6	12	18	24	30	36	42
1	7	13	19	25	31	37	43
2	8	14	20	26	32	38	44
3	9	15	21	27	33	39	45
4	10	16	22	28	34	40	46
5	11	17	23	29	35	41	47

$$B = (2, 4) : (1, 6)$$

0	6	12	18
1	7	13	19

0	2	4	24	26	28
1	3	5	25	27	29
6	8	10	30	32	34
7	9	11	31	33	35
12	14	16	36	38	40
13	15	17	37	39	41
18	20	22	42	44	46
19	21	23	43	45	47

$$A \otimes B = ((2, 4), (3, 2)) : ((1, 6), (2, 24))$$

Tile



$$A = (3, 3) : (3, 1)$$

0	1	2
3	4	5
6	7	8

Arrangements
of Tiles

$$B = (2, 2) : (1, 4)$$

0	4
1	5

0	9	36	45
3	12	39	48
6	15	42	51
1	10	37	46
4	13	40	49
7	16	43	52
2	11	38	47
5	14	41	50
8	17	44	53

← entries

← within

← each tile

$$A \otimes B = ((3, 3) (2, 2)) : ((3, 1), (9, 36))$$

Tiles

These are defined in terms of more fundamental layout operations.

Definition:

- ① The logical division of A by B is

$$A \oslash B = (A \circ B, A \circ B^c)$$

where $B^c = \text{comp}(B, \text{size}(A))$

- ② The logical product of A and B is

$$A \otimes B = (A, A^c \circ B)$$

where $A^c = \text{comp}(A, \text{size}(A) * \text{cosize}(B))$.

Need to understand composition, complements, coalesce, concatenation.

What is composition?

Definition:

If A and B are layouts, then the composition of A and B is a layout

$$B \circ A$$

satisfying

- 1 The shape of $B \circ A$ refines the shape of A , and
- 2 $\Phi_{B \circ A} = \Phi_B \circ \Phi_A$.

Example: (Composition)

$$A = (4, 6) : (6, 1) \quad \leftarrow \text{Row major}$$

$$B = (12, 4) : (4, 1) \quad \leftarrow \text{Row major}$$

$$B \circ A = ((2, 2), 6) : ((24, 1), 4)$$

Check

$$\begin{array}{l}
 \text{colex}^{-1} \\
 \Phi_A(3) \xrightarrow{\text{colex}^{-1}} (3, 0) \cdot (6, 1) = 18 \\
 \Phi_B(18) = (6, 1) \cdot (4, 1) = \boxed{25} \\
 \hline
 \Phi_{B \circ A}(3) = ((1, 1), 0) \cdot ((24, 1), 4) = \boxed{25} \quad \checkmark \\
 \text{colex}^{-1}
 \end{array}$$

$$(L, L', L'') \neq (L, (L', L'')) \\ \neq ((L, L'), L'')$$

What is concatenation?

Definition:

If $L = S : D$ and $L' = S' : D'$ are layouts, then their concatenation is

$$(L, L') = (S, S') : (D, D').$$

Example: (Concatenation)

If $L = (2, 2) : (2, 1)$ and $L' = (3, 5) : (1, 3)$, then

$$(L, L') = ((2, 2), (3, 5)) : ((2, 1), (1, 3)).$$

$$\text{size} \left((L, L') \right) = \text{size}(L) \cdot \text{size}(L').$$

What is complementation?

Definition:

If L is a layout and N is an integer, then

$$L' = \underline{\text{comp}(L, N)}$$

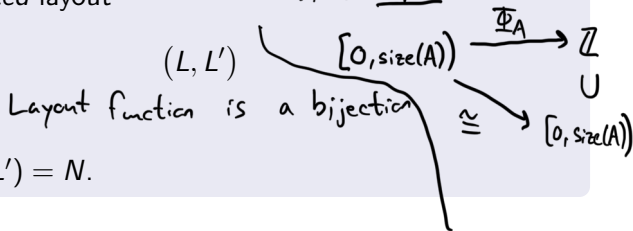
is a layout such that

- 1 the *concatenated* layout

is compact.

- 2 $\text{size}(L) * \text{size}(L') = N$.

Def: If A is a layout, then
 A is compact if



Example: (Complement)

If $L = (3, 7) : (2, 30)$ and $N = 210$, then

$$\text{comp}(L, N) = (2, 5) : (1, 6).$$

since

$$(L, L') = ((2, 5), (3, 7)) : ((1, 6), (2, 30))$$

is compact of size 210.

What is coalesce?

Definition:

If L is a layout, then $coal(L)$ is obtained by

- 1 flattening L , and
- 2 consolidating adjacent modes when possible:

$$s_i, s_{i+1} : d_i, d_{i+1} \rightsquigarrow s_i s_{i+1} : d_i$$

when $s_i d_i = d_{i+1}$.

This “simplifies” the layout without altering the layout function Φ_L .

Example: (Coalesce)

If $L = (2, 3, 2, 3) : (12, 6, 1, 2)$, then

$$\text{coal}(L) = (2, 3, 6) : (12, 6, 1)$$

Computing these layout operations, especially composition, is subtle:

Composition subtleties:

If A is a layout, then Φ_A has the form

$$\begin{array}{ccc}
 & [0, S] & \\
 \text{colex}^{-1} \nearrow & & \searrow D \cdot \\
 [0, M] & \xrightarrow{\Phi_A} & [0, N]
 \end{array}$$

If we compose Φ_A with some other Φ_B we get

$$\begin{array}{ccccc}
 & [0, S] & & [0, T] & \\
 \text{colex}^{-1} \nearrow & & \searrow D \cdot & \text{colex}^{-1} \nearrow & \searrow E \cdot \\
 [0, M] & \xrightarrow{\Phi_A} & [0, N] & \xrightarrow{\Phi_B} & [0, K]
 \end{array}$$

Composition subtleties:

Q: Can we factor $\Phi_B \circ \Phi_A$ as shown below?

$$\begin{array}{ccc} & ? & \\ \text{colex}^{-1} \nearrow & & \searrow ? \\ [0, M) & \xrightarrow{\Phi_B \circ \Phi_A} & [0, K) \end{array}$$

A: Not always! Want to identify conditions that guarantee that $B \circ A$ exists.

Recap:

- 1 Layouts are important:
 - Specify how data is stored in memory
 - Specify how data is partitioned among agents
- 2 Need to work with
 - Logical division,
 - Logical product,
 - Composition,
 - Complement.

Can develop an intuitive and powerful mathematical theory of layouts by restricting to *tractable* layouts. These are

- 1 Expressive enough to support the operations we care about (permutations, slicing, broadcasting, partitioning, etc.), and
- 2 Simple enough to admit a clean mathematical theory.

Definition:

We say a layout L is *tractable* if L may be obtained from a column major layout by

- 1 permuting modes,
- 2 removing modes,
- 3 inserting modes with stride 0,
- 4 reparenthesizing.

Remark:

Can also give an explicit divisibility condition which characterizes tractable layouts.

Example: (Compact layouts)

If L is *compact*, then L is tractable. For instance,

$$L = ((2, 2), (2, 2)) : ((8, 1), (4, 2))$$

Example: (Strided layouts)

If L is obtained from a tractable layout by removing modes, then L is tractable. For instance,

$$L = (8, 8) : (1, 16).$$

Example: (Broadcast layouts)

If L is obtained from a tractable layout by appending modes with stride 0, then L is tractable. For instance,

$$L = (128, 128) : (1, 0)$$

Our goal is to connect the theory of layouts to some well-established concepts in abstract algebra, especially *categories*. These arise naturally in mathematics, physics, and computer science.

Definition:

A **category \mathbf{C}** consists of

- 1 a collection of objects

$$ob(\mathbf{C}) = \{X, Y, Z, \dots\},$$

- 2 a collection of morphisms

$$mor(\mathbf{C}) = \{f, g, h, \dots\},$$

- 3 a composition rule

$$X \xrightarrow{f} Y \xrightarrow{g} Z \quad \rightsquigarrow \quad X \xrightarrow{g \circ f} Z$$

Example 1:

In the category **FinSet**,

- ① an object is a finite set $[0, n) = \{0, \dots, n - 1\}$ ($n \geq 0$),
- ② a morphism is a function

$$[0, n) \xrightarrow{f} [0, m)$$

- ③ Composition is functional composition

$$[0, n) \xrightarrow{f} [0, m) \xrightarrow{g} [0, p) \quad \rightsquigarrow \quad [0, n) \xrightarrow{g \circ f} [0, p)$$

Example 2:

In the category **Vect**,

- 1 an object is a vector space \mathbb{R}^n ($n \geq 0$),
- 2 a morphism is a matrix

$$\mathbb{R}^n \xrightarrow{A} \mathbb{R}^m$$

- 3 Composition is matrix multiplication

$$\mathbb{R}^n \xrightarrow{A} \mathbb{R}^m \xrightarrow{B} \mathbb{R}^p \quad \rightsquigarrow \quad \mathbb{R}^n \xrightarrow{BA} \mathbb{R}^p$$

Example 3:

In the category $\mathbf{Z}_{>0}$,

- ① an object is a positive integer a .
- ② There is a unique morphism

$$a \rightarrow b$$

if a divides b , otherwise none.

- ③ Composition \Leftrightarrow divisibility is transitive.

$$a \longrightarrow b \longrightarrow c \quad \rightsquigarrow \quad a \longrightarrow c$$

There are also morphisms between categories called *functors*.

Definition:

If \mathbf{C} and \mathbf{D} are categories, then a *functor* is an assignment

$$\mathbf{C} \xrightarrow{F} \mathbf{D}$$

$$\begin{array}{ccc} X & & FX \\ f \downarrow & & \downarrow Ff \\ Y & & FY \end{array}$$

that is compatible with composition:

Example

$$\mathbf{FinSet} \xrightarrow{F} \mathbf{Vect}$$

$$\begin{array}{ccc} [0, n) & & \mathbb{R}^n \\ f \downarrow & & \downarrow Ff \\ [0, m) & & \mathbb{R}^m \end{array}$$

$$(Ff)_{i,j} = \begin{cases} 1 & f(j) = i \\ 0 & \text{else} \end{cases}$$

Want to define a category whose morphisms encode layouts.
We begin with the flat case.
Later, we will extend to the nested case.

Definition:

In the category **Tuple**,

- 1 an object is a tuple (s_1, \dots, s_m) .
- 2 A morphism $f : (s_1, \dots, s_m) \rightarrow (t_1, \dots, t_n)$ is specified by a function

$$\alpha : \{*, 1, \dots, m\} \rightarrow \{*, 1, \dots, n\}$$

satisfying

- $\alpha(*) = *$,
 - If $\alpha(i) = \alpha(i') \neq *$, then $i = i'$,
 - If $\alpha(i) = j$, then $s_i = t_j$.
- 3 Composition is functional composition of underlying maps:

$$S \xrightarrow[\alpha]{f} T \xrightarrow[\beta]{g} U \quad \rightsquigarrow \quad S \xrightarrow[\beta \circ \alpha]{g \circ f} U$$

Example:

Here are some **Tuple** morphisms.

$$\begin{array}{ccc} 6 & \longrightarrow & 6 \\ 4 & \longrightarrow & 4 \end{array}$$

f

$$\begin{array}{ccc} 6 & & 4 \\ & \searrow & \nearrow \\ 4 & & 6 \end{array}$$

g

$$\begin{array}{ccc} & & 4 \\ & \nearrow & \\ 4 & & 2 \\ 6 & \longrightarrow & 6 \end{array}$$

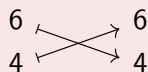
h

$$\begin{array}{ccc} & & 3 \\ & & \\ 8 & \longrightarrow & 8 \\ 8 & \longrightarrow & 8 \end{array}$$

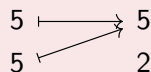
k

Non-example:

These diagrams do not depict valid **Tuple**-morphisms.



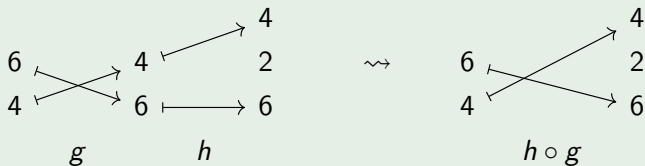
f



g

We compose morphisms by pasting diagrams.

Example:



Mathematical Remark:

Construction of **Tuple** is motivated by the theory of *operads*.

- 1 Associative operad **Assoc**,
- 2 Commutative operad **Comm**,
- 3 Lie operad **Lie**.

More precisely, our category **Tuple** (and variants we will consider later) occur naturally as subcategories of the category of operators of certain (colored) operads.

Each **Tuple**-morphism encodes a layout:

Construction:

If

$$(s_1, \dots, s_m) \xrightarrow[\alpha]{} (t_1, \dots, t_n)$$

is a **Tuple**-morphism we define

$$L_f = (s_1, \dots, s_m) : (d_1, \dots, d_m)$$

where

$$d_i = \begin{cases} t_1 \cdots t_{\alpha(i)-1} & \alpha(i) \neq * \\ 0 & \alpha(i) = *. \end{cases}$$

Example:

$$6 \longrightarrow 6$$

$$4 \longrightarrow 4$$

f

$$6 \longrightarrow 4$$

$$4 \longrightarrow 6$$

g

$$4 \longrightarrow 4$$

$$6 \longrightarrow 6$$

h

3

$$8 \longrightarrow 8$$

$$8 \longrightarrow 8$$

k

Not all layouts arise from this construction.

Example:

The layout

$$L = (4, 6) : (1, 5)$$

does not arise as L_f for a **Tuple**-morphism f :

$$\begin{array}{ccc} & & 6 \\ & \nearrow & \\ 6 & & ? \\ & \longmapsto & 4 \end{array}$$

Want to characterize which layouts arise.

Recall the notion of a tractable layout:

Definition:

We say a layout L is *tractable* if L may be obtained from a column major layout by

- 1 permuting modes,
- 2 removing modes,
- 3 inserting modes with stride 0,
- 4 reparenthesizing.

Correspondence Theorem:

- ① The layouts which arise from **Tuple**-morphisms are exactly the flat tractable layouts.
- ② There is a one-to-one correspondence

$$\left\{ \begin{array}{l} \text{Non-degenerate flat} \\ \text{tractable layouts} \end{array} \right\} \leftrightarrow \left\{ \begin{array}{l} \text{Non-degenerate} \\ \text{**Tuple**-morphisms of} \\ \text{standard form} \end{array} \right\}$$

Compact layout

\leftrightarrow

Permutation

$$L = (4, 6) : (6, 1)$$

\leftrightarrow

Strided layout

\leftrightarrow

Insertion

$$L = (4, 6) : (2, 64)$$

\leftrightarrow

Broadcast layout

\leftrightarrow

Projection

$$L = (128, 128) : (128, 0)$$

\leftrightarrow

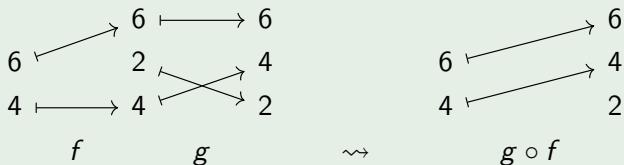
Why do we care?

Operations on
Tuple-morphisms



Operations on
layouts

Example: (Composition)



Theorem:

If f and g are composable, then $L_{g \circ f} = L_g \circ L_f$.

Categorical formulation:

Theorem:

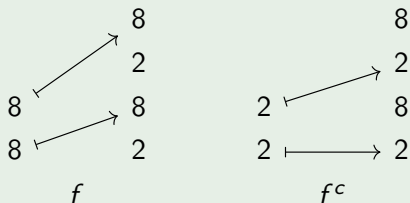
There is a realization functor

$$\begin{array}{ccc}
 \mathbf{Tuple} & \longrightarrow & \mathbf{Set} \\
 \\
 S & & [0, \text{size}(S)) \\
 f \downarrow & & \downarrow L_f \\
 T & & [0, \text{size}(T))
 \end{array}$$

In particular, if f and g are composable, then

$$L_g \circ L_f = L_{g \circ f}.$$

Example: (Complement)

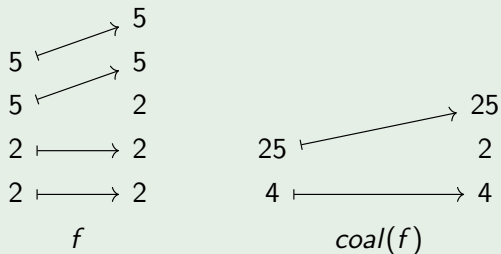


Theorem:

If $f : S \rightarrow T$ is an injective **Tuple**-morphism, then

$$L_{f^c} = \text{comp}(L_f, \text{size}(T)).$$

Example: (Coalesce)



Theorem:

If $f : S \rightarrow T$ is a **Tuple**-morphism, then

$$L_{coal(f)} = coal(L_f).$$

Want to express nested layouts, too.

Notation:

If S is a nested tuple, we write

$$S = (s_1, \dots, s_m)_P$$

where $(s_1, \dots, s_m) = S^b$ is the flattening of S , and P is the profile of S .

Definition:

In the category **Nest**,

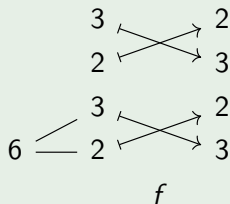
- 1 an object is a nested tuple $S = (s_1, \dots, s_m)_P$.
- 2 A morphism $f : (s_1, \dots, s_m)_P \rightarrow (t_1, \dots, t_n)_Q$ is specified by a function

$$\alpha : \{*, 1, \dots, m\} \rightarrow \{*, 1, \dots, n\}$$

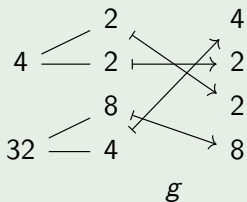
satisfying

- 1 $\alpha(*) = *$,
 - 2 if $\alpha(i) = \alpha(i') \neq *$, then $i = i'$,
 - 3 if $\alpha(i) = j$, then $s_i = t_j$.
- 3 Composition is functional composition of underlying maps.

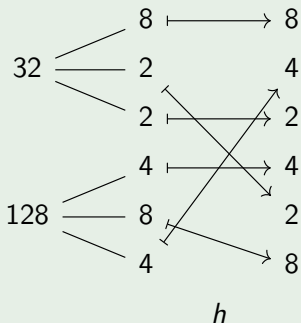
Example:



Example: (mma.sync accumulator thread-value layout)

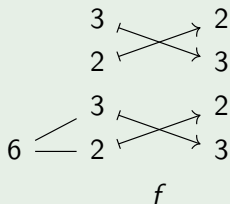


Example: (wgmma accumulator thread-value layout)



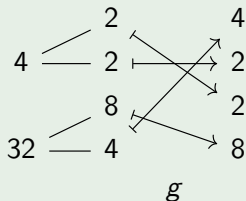
We can use the same formula as before to define the layout encoded by any **Nest**-morphism.

Example:



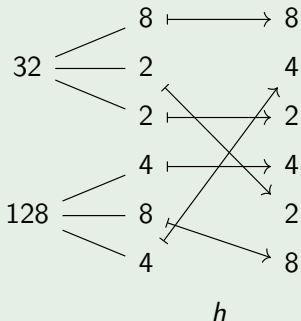
$$L = ((2, 3), 2, 3) : ((3, 1), 18, 6)$$

Example: (mma.sync accumulator thread-value layout)



$$L = ((4, 8), (2, 2)) : ((32, 1), (16, 8))$$

Example: (wgmma accumulator thread-value layout)



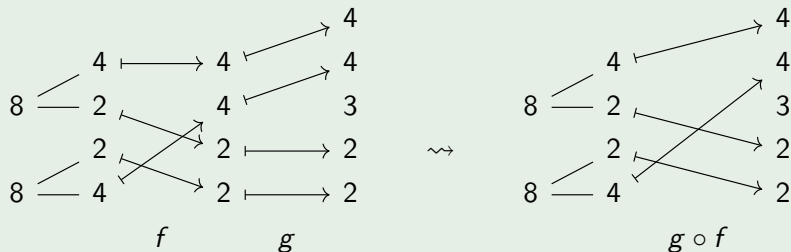
$$L = ((4, 8, 4), (2, 2, 8)) : ((128, 1, 16), (64, 8, 512))$$

Theorem:

- 1 The layouts which arise from **Nest**-morphisms are exactly the tractable layouts.
- 2 There is a one-to-one correspondence

$$\left\{ \begin{array}{l} \text{Non-degenerate} \\ \text{tractable layouts} \end{array} \right\} \leftrightarrow \left\{ \begin{array}{l} \text{Non-degenerate} \\ \textbf{Nest-morphisms of} \\ \text{standard form} \end{array} \right\}$$

Example: (Composition)

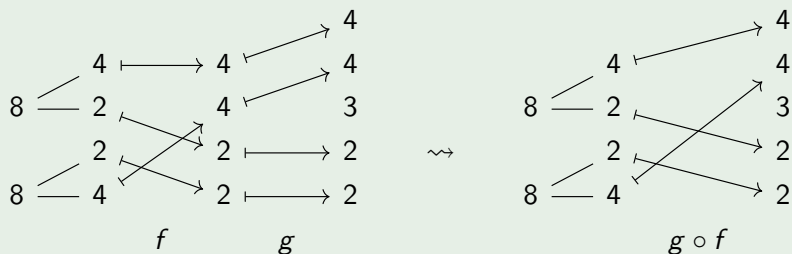


$$L_f = ((4, 2), (2, 4)) : ((4, 1), (2, 16))$$

$$L_g = (2, 2, 4, 4) : (1, 2, 12, 48)$$

$$\rightsquigarrow L_{g \circ f} = ((4, 2), (2, 4)) : ((12, 1), (2, 48))$$

Example: (Composition)



Theorem:

If $f : S \rightarrow T$ and $g : T \rightarrow U$ are non-degenerate **Nest**-morphisms, then

$$L_g \circ L_f = L_{g \circ f}.$$

Categorical formulation:

Theorem:

There is a realization functor

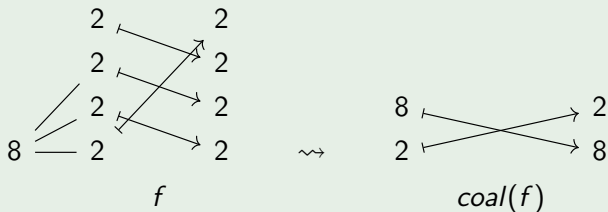
$$\mathbf{Nest} \longrightarrow \mathbf{Set}$$

$$\begin{array}{ccc}
 S & & [0, \text{size}(S)) \\
 f \downarrow & & \downarrow L_f \\
 T & & [0, \text{size}(T))
 \end{array}$$

In particular, if f and g are composable, then

$$L_g \circ L_f = L_{g \circ f}.$$

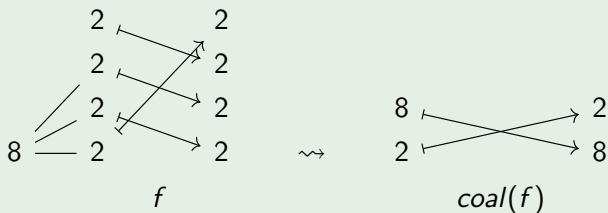
Example: (Coalesce)



$$L_f = ((2, 2, 2), 2) : ((8, 1, 2), 4)$$

$$L_{coal(f)} = (2, 8) : (8, 1)$$

Example: (Coalesce)

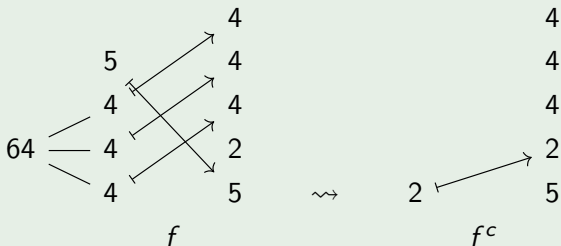


Theorem:

If $f : S \rightarrow T$ is a **Nest**-morphism, then

$$L_{coal(f)} = coal(L_f).$$

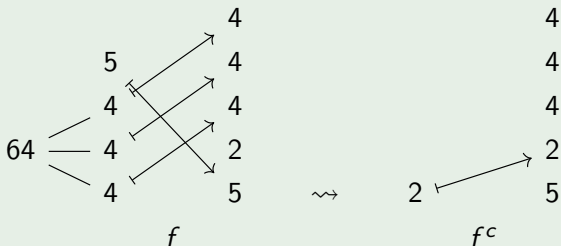
Example: (Complement)



$$L_f = ((4, 4, 4), 5) : ((10, 40, 160), 1)$$

$$L_{f^c} = (2) : (5)$$

Example: (Complement)

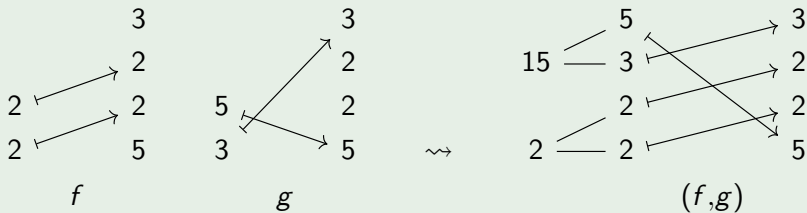


Theorem:

If $f : S \rightarrow T$ is an injective **Nest**-morphism and $N = \text{size}(T)$, then

$$\text{coal}(L_{f^c}) = \text{comp}(L_f, N).$$

Example: (Concatenation)

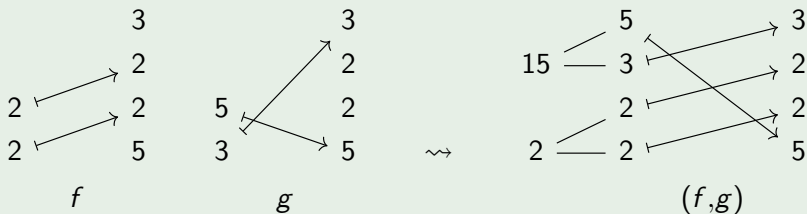


$$L_f = (2, 2) : (5, 10)$$

$$L_g = (3, 5) : (20, 1)$$

$$L_{(f, g)} = ((2, 2), (3, 5)) : ((5, 10), (20, 1))$$

Example: (Concatenation)



Theorem:

If $f : S \rightarrow T$ and $g : U \rightarrow T$ are **Nest**-morphisms with disjoint images, then

$$L_{(f,g)} = (L_f, L_g)$$

When defined, we can set

$$f \oslash g = (f \circ g, f \circ g^c)$$

$$f \otimes g = (f, f^c \circ g)$$

Theorem:

If f and g are non-degenerate **Nest**-morphisms, and g divides f , then

$$\text{coal}(L_{f \oslash g}) = \text{coal}(L_f \oslash L_g)$$

Theorem:

If f and g are non-degenerate **Nest**-morphisms and f and g are product admissible, then

$$L_{f \otimes g} = L_f \otimes L_g$$

We can use our theory to compute the composition $B \circ A$ of tractable layouts A and B .

Example 1: (Composition Algorithm)

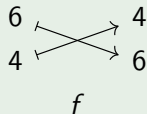
Consider the layouts

$$A = (4, 6) : (6, 1) \quad B = (12, 4) : (4, 1)$$

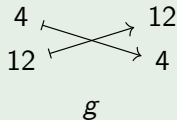
Example 1: (Composition Algorithm)

Take their standard representations

$$A = (4, 6) : (6, 1)$$

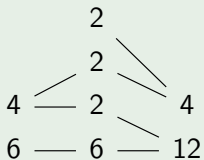


$$B = (12, 4) : (4, 1)$$



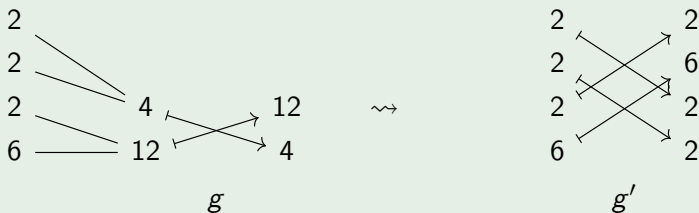
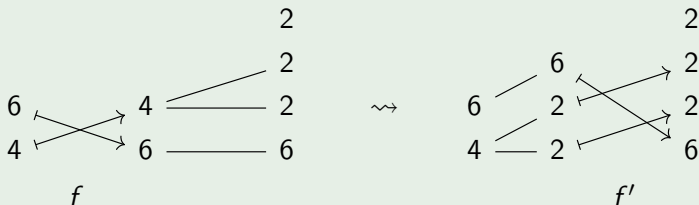
Example 1: (Composition Algorithm)

Modify the codomain of f and domain of g :



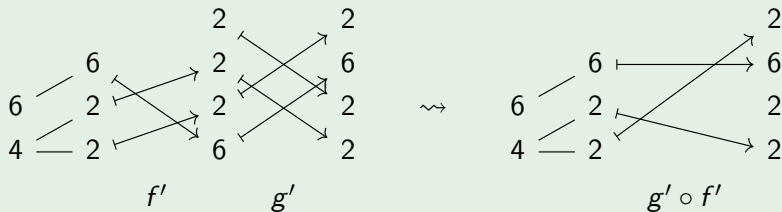
Example 1: (Composition Algorithm)

Make a corresponding modification to f and g :



Example 1: (Composition Algorithm)

Compose!



$$B \circ A = L_{g' \circ f'} = ((2, 2), 6) : ((24, 1), 4).$$

Example 2: (Composition Algorithm)

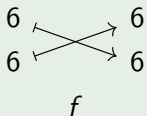
Consider the layouts

$$A = (6, 6) : (6, 1) \quad B = (12, 3, 6) : (1, 72, 12).$$

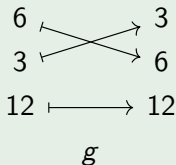
Example 2: (Composition Algorithm)

Take the standard representations

$$A = (6, 6) : (6, 1)$$

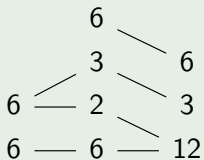


$$B = (12, 3, 6) : (1, 72, 12).$$



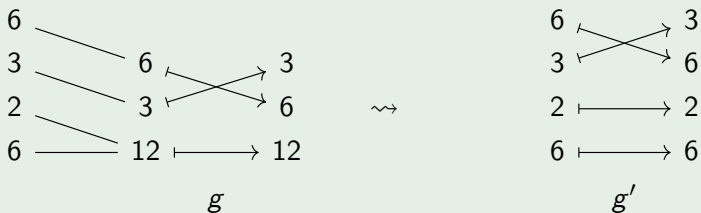
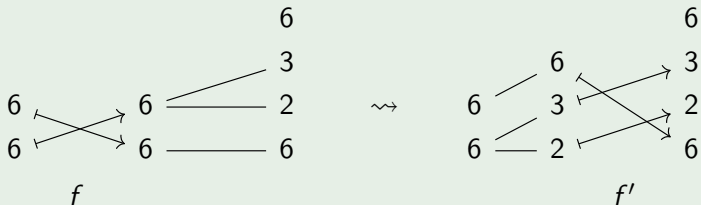
Example 2: (Composition Algorithm)

Modify the codomain of f and domain of g



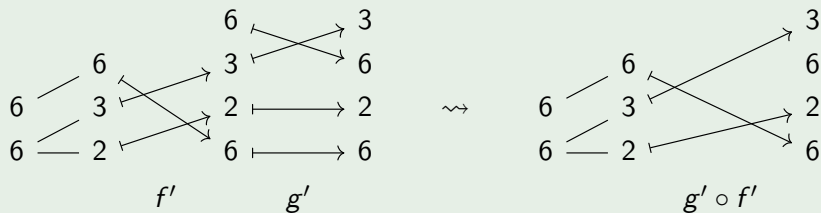
Example 2: (Composition Algorithm)

Make a corresponding modification to f and g



Example 2: (Composition Algorithm)

Compose!



$$B \circ A = L_{g' \circ f'} = ((2, 3), 6) : ((6, 72), 1).$$

The preceding examples are instances of a more general algorithm.

Composition Algorithm

Input: Tractable layouts A and B .

- Take the standard representations of A and $coal(B)$.

$$f : S \rightarrow T, \quad g : U \rightarrow V$$

- Compute a mutual refinement (T', U') of T and U ,
- Pull back f along $T' \rightarrow T$ to obtain $f' : S' \rightarrow T' \rightarrow U'$,
- Push forward $U' \rightarrow U$ along g to obtain $g' : U' \rightarrow V'$,

Composition Algorithm (cont.)

- Compose f' and g' to obtain

$$S' \xrightarrow{g' \circ f'} V'.$$

- Take the encoded layout $C = L_{g' \circ f'}$.

Output: Layout $C = B \circ A$.

Remark:

With appropriate modifications, we can use the same algorithm to compute $B \circ A$ when A is tractable and B is *arbitrary*.

Theorem: (Composition algorithm)

The algorithm described above is well-defined, and is guaranteed to produce the composition $B \circ A$ of tractable layouts A and B , if it exists.

Future directions:

- Span category formulation, further connections to operad theory.
- Characterize algebraic relations satisfied by logical division, logical product, etc.
- Interaction with other layout abstractions: swizzles, linear layouts, integer set relations.

References and related work:

- *CuTe Layout Representation and Algebra* (Cecka)
<https://arxiv.org/abs/2603.02298>
- *Categorical foundations for CuTe layouts* (Colfax)
<https://arxiv.org/abs/2601.05972> (blog post:
<https://research.colfax-intl.com/categorical-foundations-for-cute-layouts/>)
- *Modeling Layout Abstractions using Integer Set Relations*
(Bhaskaracharya et. al.)
<https://arxiv.org/html/2511.10374v1>
- *Linear Layouts: Robust Code Generation of Efficient Tensor Computation Using \mathbb{F}_2* (Zhou et. al.)
<https://arxiv.org/abs/2505.23819>
- CuTe documentation: (<https://docs.nvidia.com/cutlass/latest/media/docs/cpp/cute/index.html>)
- Cris Cecka's GPU mode talk
(<https://www.youtube.com/watch?v=ufa4pmB0BT8>)

Thanks for listening!