CONTAINERS: COMPOSITIONALITY FOR TENSORS

Neil Ghani, Pierre Hyvernat, Artjoms Šinkarovs

University of Southampton

13 June 2025

CREATION OF ARRAYS

CREATION OF ARRAYS

- `a : X`

CREATION OF ARRAYS

- `a : X`
- `a : Vec n X`

CREATION OF ARRAYS

- `a : X`
- `a : Vec n X`
- `a : Ar s X`

ARRAY TYPE

ARRAY TYPE

```
data FV (X : Set) : List ℕ → Set where
  scal : X → FV X []
  nest : Vec (FV X s) n → FV X (n :: s)
```

ARRAY TYPE

```
data FV (X : Set) : List ℕ → Set where
  scal : X → FV X []
  nest : Vec (FV X s) n → FV X (n :: s)
```

```
S = List ℕ
P = All Fin

Ar : S → Set → Set
Ar s X = P s → X
```

EXAMPLES

```
transpose : Ar s X → Ar (rev s) X
transpose a i = a (p-rev i)
```

EXAMPLES

```
transpose : Ar s X → Ar (rev s) X
transpose a i = a (p-rev i)
```

```
map : (X → Y) → Ar s X → Ar s Y
map f a i = f (a i)

zipWith : (X → Y → Z) → Ar s X → Ar s Y → Ar s Z
zipWith f a b i = f (a i) (b i)

K : X → Ar s X
K x y = x
```

EXAMPLES

```
module Matmul
  (σ : ∀ {X s} → (X → X → X) → X → Ar s X → X)
  (_⊠_ : X → Y → Z) (ε : Z) (_⊞_ : Z → Z → Z)
  where

  mm : Ar s (Ar p X)
     → Ar p (Ar q Y)
     → Ar s (Ar q Z)
  mm a b i j = σ _⊞_ ε λ k → a i k ⊠ b k j
```

CONTAINERS

```
record Con : Set₁ where
  constructor _◃_
  field
    S : Set
    P : S → Set

⟦_⟧ : Con → Set → Set
⟦ A ◃ B ⟧ X = Σ[ a ∈ A ] (B a → X)
```

CONTAINER OPERATIONS

```
𝟘 : Con;               𝟙 : Con
𝟘 = ⊥ ◃ λ ();          𝟙 = ⊤ ◃ λ _ → ⊤

_⊕_ : Con → Con → Con
(A ◃ B) ⊕ (C ◃ D) = (A ⊎ C)
                  ◃ λ { (inj₁ a) → B a;
                        (inj₂ c) → D c }

_×ᶜ_ : Con → Con → Con
(A ◃ B) ×ᶜ (C ◃ D) = A × C
                   ◃ λ (a , c) → B a ⊎ D c

_⊗_ : Con → Con → Con
(A ◃ B) ⊗ (C ◃ D) = A × C
                  ◃ λ (a , c) → B a × D c
```

CONTAINER MORPHISMS

```
record Mor (A B : Con) : Set where
  constructor _◃_
  field
    MS : A .S → B .S
    MP : ∀ {a} → B .P (MS a) → A .P a
```

ARRAYS AS CONTAINERS

```
Π : (A : Set) → (A → Set) → Set
Π A B = (a : A) → B a

⊗ : Con → Con → Con
⊗ (A ◃ B) (C ◃ D) = ⟦ A ◃ B ⟧ C
                    ◃ λ (a , f) → Π (B a) (D ∘ f)

-- (2 , f:[5 , 6]) ◃ ((i : Fin 2) → Fin (f i))
Ar = ⟦ ⊗ (ℕ ◃ Fin) (ℕ ◃ Fin) ⟧
```

GENERALISATION

Let us use container `(A ◃ I)` as a shape, where:

1. `A` corresponds to the set of axes;
2. `I` assigns to a specific axis `a` indices `I a`.

GENERALISATION

Let us use container `(A ◃ I)` as a shape, where:

1. `A` corresponds to the set of axes;
2. `I` assigns to a specific axis `a` indices `I a`.

```
Mat : (m n : ℕ) → Con
Mat m n = 2 ◃ λ { ff → Fin m; tt → Fin n }
```

ARRAY TYPE

```
⟦_⟧a : Con → Set → Set
⟦ A ◃ B ⟧a X = Π A B → X

Ar s = ⟦ s ⟧a
```

ARRAY TYPE

```
⟦_⟧a : Con → Set → Set
⟦ A ◃ B ⟧a X = Π A B → X

Ar s = ⟦ s ⟧a
```

Gives rise to:

```
record Con′ : Set where
  constructor _◃′_
  field
    S : Cat
    P : S ⇒ Setᵒᵖ
```

In the sense that `Ar` as `Con′` is `Con ◃′ Π`.

PROPERTIES

```
nest : Ar (s ⊕ p ) X → Ar s (Ar p X)
nest a i j = a λ { (inj₁ l) → i l;
                   (inj₂ r) → j r  }

unnest : Ar s (Ar p X) → Ar (s ⊕ p) X
unnest a ij = a (ij ∘ inj₁) (ij ∘ inj₂)

pair : Ar s X → Ar p Y → Ar (s ⊕ p) (X × Y)
pair a b = unnest λ i j → a i , b j
```

PROPERTIES

```
map : (X → Y) → Ar s X → Ar s Y
map f a i = f (a i)

reshape : Mor s p → Ar s X → Ar p X
reshape m a i = a (m .MP ∘ i ∘ m .MS)
```

TENSORS

```
_•_ : Ar (s ⊕ p) X → Ar (p ⊕ q) X → Ar (s ⊕ q) X
```

gives rise to a category of tensors: `Ob = Con`; morphsisms `s -> p = Ar (s ⊕ p) X`, composition is `_•_`.

`Ar (s ⊕ p) X` is a tensor of `s` covectors and `p` vectors; `𝟘` and `𝟙` are singletons; `Ar s X` can be turned into a "column vector" `Ar (s ⊕ 𝟙) X` or "row vector" `Ar (𝟙 ⊕ s) X`.

CONCLUSIONS

- New way to understand arrays as containers
- Combinators derived from compositional properties
- Interpretation of tensors
- WIP formalisation
  https://github.com/ashinkarov/2025-types

THANK YOU!

I am hiring PhD students in Southampton a.sinkarovs@soton.ac.uk.

Neil is hiring people who are interested in Types/CT/FP and AI/AI Agents neil@kodamai.com, toby@kodamai.com.