Tensor

A Tensor represents a multi-dimensional array of elements.

ATensor represents a multi-dimensional array of elements and is depicted as a struct containing both the tensor's shape and a flattened array of its data. The generic Tensor is defined as follows:

```

Copy structTensor { shape:Span, data:Span }

```

Data types

Orion supports currently these tensor types.

?

TensorTrait

```

Copy useorion::operators::tensor::TensorTrait;

```

TensorTrait defines the operations that can be performed on a Tensor.

?

Arithmetic Operations

Tensor implements arithmetic traits. This allows you to perform basic arithmetic operations using the associated operators. (+ ,- ,* ,/). Tensors arithmetic operations supports broadcasting.

Two tensors are “broadcastable” if the following rules hold:

- Each tensor has at least one dimension.
- When iterating over the dimension sizes, starting at the trailing dimension, the dimension sizes must either be equal, one of them is 1, or one of them does not exist.
-

Examples

Element-wise add.

```

Copy usecore::array::{ArrayTrait,SpanTrait}; useorion::operators::tensor::{TensorTrait,Tensor,U32Tensor,U32TensorAdd};

fnelement_wise_add_example()->Tensor { // We instantiate two 3D Tensors here. lettensor_1=TensorTrait::new( shape:array![2,2,2].span(), data:array![0,1,2,3,4,5,6,7].span(), ); lettensor_2=TensorTrait::new( shape:array![2,2,2].span(), data:array![0,1,2,3,4,5,6,7].span(), );

// We can add two tensors as follows. returntensor_1+tensor_2; }

                    [[[0,2],[4,6]],[[8,10],[12,14]]]

```

Add two tensors of different shapes but compatible in broadcasting.

```

Copy usecore::array::{ArrayTrait,SpanTrait}; useorion::operators::tensor::{TensorTrait,Tensor,U32Tensor,U32TensorAdd};

fnbroadcasting_add_example()->Tensor { // We instantiate two 3D Tensors here. lettensor_1=TensorTrait::new( shape:array![2,2,2].span(), data:array![0,1,2,3,4,5,6,7].span(), ); lettensor_2=TensorTrait::new( shape:array![1,2,1].span(), data:array![10,100].span(), );
```

// We can add two tensors as follows. returntensor_1+tensor_2; }

[[[10,11],[102,103]],[[14,15],[106,107]]]

```

Previous Operators Next tensor.new

Last updated2 months ago
```