MonoidalCategories

Monoidal and monoidal (co)closed categories

2022.08-03

15 August 2022

Mohamed Barakat

Sebastian Gutsche

Sebastian Posur

Tom Kuhmichel

Mohamed Barakat
Email: mohamed.barakat@uni-siegen.de
Homepage: http://algebra.mathematik.uni-siegen.de/barakat/
Address: Walter-Flex-Str. 3
57068 Siegen
Germany

Sebastian Gutsche
Email: gutsche@mathematik.uni-siegen.de
Homepage: http://algebra.mathematik.uni-siegen.de/gutsche/
Address: Department Mathematik
Universität Siegen
Walter-Flex-Straße 3
57068 Siegen
Germany

Sebastian Posur
Email: sebastian.posur@uni-siegen.de
Homepage: http://algebra.mathematik.uni-siegen.de/posur/
Address: Department Mathematik
Universität Siegen
Walter-Flex-Straße 3
57068 Siegen
Germany

Tom Kuhmichel
Email: tom.kuhmichel@student.uni-siegen.de
Homepage: https://github.com/TKuh
Address: Department Mathematik
Universität Siegen
Walter-Flex-Straße 3
57068 Siegen
Germany

Contents

Chapter 1

Monoidal Categories

1.1 Monoidal Categories

A 6-tuple $(\mathbf{C}, \otimes, 1, \alpha, \lambda, \rho)$ consisting of

- a category $\mathbf{C}$,
- a functor $\otimes : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ compatible with the congruence of morphisms,
- an object $1 \in \mathbf{C}$,
- a natural isomorphism $\alpha_{a,b,c} : a \otimes (b \otimes c) \cong (a \otimes b) \otimes c$,
- a natural isomorphism $\lambda_a : 1 \otimes a \cong a$,
- a natural isomorphism $\rho_a : a \otimes 1 \cong a$,

is called a *monoidal category*, if

- for all objects a, b, c, d, the pentagon identity holds:

$(\alpha_{a,b,c} \otimes \mathrm{id}_d) \circ \alpha_{a,b\otimes c,d} \circ (\mathrm{id}_a \otimes \alpha_{b,c,d}) \sim \alpha_{a\otimes b,c,d} \circ \alpha_{a,b,c\otimes d}$,

- for all objects a, c, the triangle identity holds:

$(\rho_a \otimes \mathrm{id}_c) \circ \alpha_{a,1,c} \sim \mathrm{id}_a \otimes \lambda_c$.

The corresponding GAP property is given by `IsMonoidalCategory`.

1.1.1 TensorProductOnMorphisms (for IsCapCategoryMorphism, IsCapCategoryMorphism)

▷ `TensorProductOnMorphisms(alpha, beta)` (operation)

Returns: a morphism in $\mathrm{Hom}(a \otimes b, a' \otimes b')$

The arguments are two morphisms $\alpha : a \to a', \beta : b \to b'$. The output is the tensor product $\alpha \otimes \beta$.

1.1.2 TensorProductOnMorphismsWithGivenTensorProducts (for IsCapCategoryObject, IsCapCategoryMorphism, IsCapCategoryMorphism, IsCapCategoryObject)

▷ `TensorProductOnMorphismsWithGivenTensorProducts(s, alpha, beta, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a \otimes b, a' \otimes b')$

The arguments are an object $s = a \otimes b$, two morphisms $\alpha : a \to a', \beta : b \to b'$, and an object $r = a' \otimes b'$. The output is the tensor product $\alpha \otimes \beta$.

1.1.3 AssociatorRightToLeft (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `AssociatorRightToLeft(a, b, c)` (operation)

Returns: a morphism in $\mathrm{Hom}(a \otimes (b \otimes c), (a \otimes b) \otimes c)$.

The arguments are three objects a, b, c. The output is the associator $\alpha_{a,(b,c)} : a \otimes (b \otimes c) \to (a \otimes b) \otimes c$.

1.1.4 AssociatorRightToLeftWithGivenTensorProducts (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `AssociatorRightToLeftWithGivenTensorProducts(s, a, b, c, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a \otimes (b \otimes c), (a \otimes b) \otimes c)$.

The arguments are an object $s = a \otimes (b \otimes c)$, three objects a, b, c, and an object $r = (a \otimes b) \otimes c$. The output is the associator $\alpha_{a,(b,c)} : a \otimes (b \otimes c) \to (a \otimes b) \otimes c$.

1.1.5 AssociatorLeftToRight (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `AssociatorLeftToRight(a, b, c)` (operation)

Returns: a morphism in $\mathrm{Hom}((a \otimes b) \otimes c \to a \otimes (b \otimes c))$.

The arguments are three objects a, b, c. The output is the associator $\alpha_{(a,b),c} : (a \otimes b) \otimes c \to a \otimes (b \otimes c)$.

1.1.6 AssociatorLeftToRightWithGivenTensorProducts (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `AssociatorLeftToRightWithGivenTensorProducts(s, a, b, c, r)` (operation)

Returns: a morphism in $\mathrm{Hom}((a \otimes b) \otimes c \to a \otimes (b \otimes c))$.

The arguments are an object $s = (a \otimes b) \otimes c$, three objects a, b, c, and an object $r = a \otimes (b \otimes c)$. The output is the associator $\alpha_{(a,b),c} : (a \otimes b) \otimes c \to a \otimes (b \otimes c)$.

1.1.7 LeftUnitor (for IsCapCategoryObject)

▷ `LeftUnitor(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(1 \otimes a, a)$

The argument is an object a. The output is the left unitor $\lambda_a : 1 \otimes a \to a$.

1.1.8 LeftUnitorWithGivenTensorProduct (for IsCapCategoryObject, IsCapCategoryObject)

▷ `LeftUnitorWithGivenTensorProduct(a, s)` (operation)

Returns: a morphism in $\mathrm{Hom}(1 \otimes a, a)$

The arguments are an object a and an object $s = 1 \otimes a$. The output is the left unitor $\lambda_a : 1 \otimes a \rightarrow a$.

1.1.9 LeftUnitorInverse (for IsCapCategoryObject)

▷ `LeftUnitorInverse(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(a, 1 \otimes a)$

The argument is an object a. The output is the inverse of the left unitor $\lambda_a^{-1} : a \rightarrow 1 \otimes a$.

1.1.10 LeftUnitorInverseWithGivenTensorProduct (for IsCapCategoryObject, IsCapCategoryObject)

▷ `LeftUnitorInverseWithGivenTensorProduct(a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a, 1 \otimes a)$

The argument is an object a and an object $r = 1 \otimes a$. The output is the inverse of the left unitor $\lambda_a^{-1} : a \rightarrow 1 \otimes a$.

1.1.11 RightUnitor (for IsCapCategoryObject)

▷ `RightUnitor(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(a \otimes 1, a)$

The argument is an object a. The output is the right unitor $\rho_a : a \otimes 1 \rightarrow a$.

1.1.12 RightUnitorWithGivenTensorProduct (for IsCapCategoryObject, IsCapCategoryObject)

▷ `RightUnitorWithGivenTensorProduct(a, s)` (operation)

Returns: a morphism in $\mathrm{Hom}(a \otimes 1, a)$

The arguments are an object a and an object $s = a \otimes 1$. The output is the right unitor $\rho_a : a \otimes 1 \rightarrow a$.

1.1.13 RightUnitorInverse (for IsCapCategoryObject)

▷ `RightUnitorInverse(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(a, a \otimes 1)$

The argument is an object a. The output is the inverse of the right unitor $\rho_a^{-1} : a \rightarrow a \otimes 1$.

1.1.14 RightUnitorInverseWithGivenTensorProduct (for IsCapCategoryObject, IsCapCategoryObject)

▷ `RightUnitorInverseWithGivenTensorProduct(a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a, a \otimes 1)$

The arguments are an object a and an object $r = a \otimes 1$. The output is the inverse of the right unitor $\rho_a^{-1} : a \rightarrow a \otimes 1$.

1.1.15 TensorProductOnObjects (for IsCapCategoryObject, IsCapCategoryObject)

▷ `TensorProductOnObjects(a, b)` (operation)

Returns: an object

The arguments are two objects a, b. The output is the tensor product $a \otimes b$.

1.1.16 AddTensorProductOnObjects (for IsCapCategory, IsFunction)

▷ `AddTensorProductOnObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorProductOnObjects`. $F : (a,b) \mapsto a \otimes b$.

1.1.17 TensorUnit (for IsCapCategory)

▷ `TensorUnit(C)` (attribute)

Returns: an object

The argument is a category **C**. The output is the tensor unit 1 of **C**.

1.1.18 AddTensorUnit (for IsCapCategory, IsFunction)

▷ `AddTensorUnit(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorUnit`. $F : () \mapsto 1$.

1.2 Additive Monoidal Categories

1.2.1 LeftDistributivityExpanding (for IsCapCategoryObject, IsList)

▷ `LeftDistributivityExpanding(a, L)` (operation)

Returns: a morphism in $\mathrm{Hom}(a \otimes (b_1 \oplus \cdots \oplus b_n), (a \otimes b_1) \oplus \cdots \oplus (a \otimes b_n))$

The arguments are an object a and a list of objects $L = (b_1, \dots, b_n)$. The output is the left distributivity morphism $a \otimes (b_1 \oplus \cdots \oplus b_n) \to (a \otimes b_1) \oplus \cdots \oplus (a \otimes b_n)$.

1.2.2 LeftDistributivityExpandingWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsList, IsCapCategoryObject)

▷ `LeftDistributivityExpandingWithGivenObjects(s, a, L, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(s, r)$

The arguments are an object $s = a \otimes (b_1 \oplus \cdots \oplus b_n)$, an object a, a list of objects $L = (b_1, \dots, b_n)$, and an object $r = (a \otimes b_1) \oplus \cdots \oplus (a \otimes b_n)$. The output is the left distributivity morphism $s \to r$.

1.2.3 LeftDistributivityFactoring (for IsCapCategoryObject, IsList)

▷ `LeftDistributivityFactoring(a, L)` (operation)

Returns: a morphism in $\mathrm{Hom}((a \otimes b_1) \oplus \cdots \oplus (a \otimes b_n), a \otimes (b_1 \oplus \cdots \oplus b_n))$

The arguments are an object a and a list of objects $L = (b_1, \dots, b_n)$. The output is the left distributivity morphism $(a \otimes b_1) \oplus \cdots \oplus (a \otimes b_n) \to a \otimes (b_1 \oplus \cdots \oplus b_n)$.

1.2.4 LeftDistributivityFactoringWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsList, IsCapCategoryObject)

▷ `LeftDistributivityFactoringWithGivenObjects(s, a, L, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(s,r)$

The arguments are an object $s = (a \otimes b_1) \oplus \cdots \oplus (a \otimes b_n)$, an object a, a list of objects $L = (b_1, \dots, b_n)$, and an object $r = a \otimes (b_1 \oplus \cdots \oplus b_n)$. The output is the left distributivity morphism $s \to r$.

1.2.5 RightDistributivityExpanding (for IsList, IsCapCategoryObject)

▷ `RightDistributivityExpanding(L, a)` (operation)

Returns: a morphism in $\mathrm{Hom}((b_1 \oplus \cdots \oplus b_n) \otimes a, (b_1 \otimes a) \oplus \cdots \oplus (b_n \otimes a))$

The arguments are a list of objects $L = (b_1, \dots, b_n)$ and an object a. The output is the right distributivity morphism $(b_1 \oplus \cdots \oplus b_n) \otimes a \to (b_1 \otimes a) \oplus \cdots \oplus (b_n \otimes a)$.

1.2.6 RightDistributivityExpandingWithGivenObjects (for IsCapCategoryObject, IsList, IsCapCategoryObject, IsCapCategoryObject)

▷ `RightDistributivityExpandingWithGivenObjects(s, L, a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(s,r)$

The arguments are an object $s = (b_1 \oplus \cdots \oplus b_n) \otimes a$, a list of objects $L = (b_1, \dots, b_n)$, an object a, and an object $r = (b_1 \otimes a) \oplus \cdots \oplus (b_n \otimes a)$. The output is the right distributivity morphism $s \to r$.

1.2.7 RightDistributivityFactoring (for IsList, IsCapCategoryObject)

▷ `RightDistributivityFactoring(L, a)` (operation)

Returns: a morphism in $\mathrm{Hom}((b_1 \otimes a) \oplus \cdots \oplus (b_n \otimes a), (b_1 \oplus \cdots \oplus b_n) \otimes a)$

The arguments are a list of objects $L = (b_1, \dots, b_n)$ and an object a. The output is the right distributivity morphism $(b_1 \otimes a) \oplus \cdots \oplus (b_n \otimes a) \to (b_1 \oplus \cdots \oplus b_n) \otimes a$.

1.2.8 RightDistributivityFactoringWithGivenObjects (for IsCapCategoryObject, IsList, IsCapCategoryObject, IsCapCategoryObject)

▷ `RightDistributivityFactoringWithGivenObjects(s, L, a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(s,r)$

The arguments are an object $s = (b_1 \otimes a) \oplus \cdots \oplus (b_n \otimes a)$, a list of objects $L = (b_1, \dots, b_n)$, an object a, and an object $r = (b_1 \oplus \cdots \oplus b_n) \otimes a$. The output is the right distributivity morphism $s \to r$.

1.3 Braided Monoidal Categories

A monoidal category $\mathbf{C}$ equipped with a natural isomorphism $B_{a,b} : a \otimes b \cong b \otimes a$ is called a *braided monoidal category* if

- $\lambda_a \circ B_{a,1} \sim \rho_a$,

- $(B_{c,a} \otimes \mathrm{id}_b) \circ \alpha_{c,a,b} \circ B_{a \otimes b,c} \sim \alpha_{a,c,b} \circ (\mathrm{id}_a \otimes B_{b,c}) \circ \alpha^{-1}_{a,b,c}$,
- $(\mathrm{id}_b \otimes B_{c,a}) \circ \alpha^{-1}_{b,c,a} \circ B_{a,b \otimes c} \sim \alpha^{-1}_{b,a,c} \circ (B_{a,b} \otimes \mathrm{id}_c) \circ \alpha_{a,b,c}$.

The corresponding GAP property is given by `IsBraidedMonoidalCategory`.

1.3.1 Braiding (for IsCapCategoryObject, IsCapCategoryObject)

▷ `Braiding(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(a \otimes b, b \otimes a)$.

The arguments are two objects a, b. The output is the braiding $B_{a,b} : a \otimes b \to b \otimes a$.

1.3.2 BraidingWithGivenTensorProducts (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `BraidingWithGivenTensorProducts(s, a, b, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a \otimes b, b \otimes a)$.

The arguments are an object $s = a \otimes b$, two objects a, b, and an object $r = b \otimes a$. The output is the braiding $B_{a,b} : a \otimes b \to b \otimes a$.

1.3.3 BraidingInverse (for IsCapCategoryObject, IsCapCategoryObject)

▷ `BraidingInverse(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(b \otimes a, a \otimes b)$.

The arguments are two objects a, b. The output is the inverse braiding $B^{-1}_{a,b} : b \otimes a \to a \otimes b$.

1.3.4 BraidingInverseWithGivenTensorProducts (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `BraidingInverseWithGivenTensorProducts(s, a, b, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(b \otimes a, a \otimes b)$.

The arguments are an object $s = b \otimes a$, two objects a, b, and an object $r = a \otimes b$. The output is the inverse braiding $B^{-1}_{a,b} : b \otimes a \to a \otimes b$.

1.4 Symmetric Monoidal Categories

A braided monoidal category $\mathbf{C}$ is called *symmetric monoidal category* if $B^{-1}_{a,b} \sim B_{b,a}$. The corresponding GAP property is given by `IsSymmetricMonoidalCategory`.

1.5 Closed Monoidal Categories

A monoidal category $\mathbf{C}$ which has for each functor $- \otimes b : \mathbf{C} \to \mathbf{C}$ a right adjoint (denoted by $\underline{\mathrm{Hom}}(b, -)$) is called a *closed monoidal category*.

If no operations involving duals are installed manually, the dual objects will be derived as $a^\vee := \underline{\mathrm{Hom}}(a, 1)$.

The corresponding GAP property is called `IsClosedMonoidalCategory`.

1.5.1 InternalHomOnObjects (for IsCapCategoryObject, IsCapCategoryObject)

▷ `InternalHomOnObjects(a, b)` (operation)

Returns: an object

The arguments are two objects a, b. The output is the internal hom object $\underline{\mathrm{Hom}}(a,b)$.

1.5.2 InternalHomOnMorphisms (for IsCapCategoryMorphism, IsCapCategoryMorphism)

▷ `InternalHomOnMorphisms(alpha, beta)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a',b), \underline{\mathrm{Hom}}(a,b'))$

The arguments are two morphisms $\alpha : a \rightarrow a', \beta : b \rightarrow b'$. The output is the internal hom morphism $\underline{\mathrm{Hom}}(\alpha,\beta) : \underline{\mathrm{Hom}}(a',b) \rightarrow \underline{\mathrm{Hom}}(a,b')$.

1.5.3 InternalHomOnMorphismsWithGivenInternalHoms (for IsCapCategoryObject, IsCapCategoryMorphism, IsCapCategoryMorphism, IsCapCategoryObject)

▷ `InternalHomOnMorphismsWithGivenInternalHoms(s, alpha, beta, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a',b), \underline{\mathrm{Hom}}(a,b'))$

The arguments are an object $s = \underline{\mathrm{Hom}}(a',b)$, two morphisms $\alpha : a \rightarrow a', \beta : b \rightarrow b'$, and an object $r = \underline{\mathrm{Hom}}(a,b')$. The output is the internal hom morphism $\underline{\mathrm{Hom}}(\alpha,\beta) : \underline{\mathrm{Hom}}(a',b) \rightarrow \underline{\mathrm{Hom}}(a,b')$.

1.5.4 EvaluationMorphism (for IsCapCategoryObject, IsCapCategoryObject)

▷ `EvaluationMorphism(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a,b) \otimes a, b)$.

The arguments are two objects a, b. The output is the evaluation morphism $\mathrm{ev}_{a,b} : \underline{\mathrm{Hom}}(a,b) \otimes a \rightarrow b$, i.e., the counit of the tensor hom adjunction.

1.5.5 EvaluationMorphismWithGivenSource (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `EvaluationMorphismWithGivenSource(a, b, s)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a,b) \otimes a, b)$.

The arguments are two objects a, b and an object $s = \underline{\mathrm{Hom}}(a,b) \otimes a$. The output is the evaluation morphism $\mathrm{ev}_{a,b} : \underline{\mathrm{Hom}}(a,b) \otimes a \rightarrow b$, i.e., the counit of the tensor hom adjunction.

1.5.6 CoevaluationMorphism (for IsCapCategoryObject, IsCapCategoryObject)

▷ `CoevaluationMorphism(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(a, \underline{\mathrm{Hom}}(b, a \otimes b))$.

The arguments are two objects a, b. The output is the coevaluation morphism $\mathrm{coev}_{a,b} : a \rightarrow \underline{\mathrm{Hom}}(b, a \otimes b)$, i.e., the unit of the tensor hom adjunction.

1.5.7 CoevaluationMorphismWithGivenRange (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `CoevaluationMorphismWithGivenRange(a, b, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a, \underline{\mathrm{Hom}}(b, a \otimes b))$.

The arguments are two objects a, b and an object $r = \underline{\mathrm{Hom}}(b, a \otimes b)$. The output is the coevaluation morphism $\mathrm{coev}_{a,b} : a \to \underline{\mathrm{Hom}}(b, a \otimes b)$, i.e., the unit of the tensor hom adjunction.

1.5.8 TensorProductToInternalHomAdjunctionMap (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryMorphism)

▷ `TensorProductToInternalHomAdjunctionMap(a, b, f)` (operation)

Returns: a morphism in $\mathrm{Hom}(a, \underline{\mathrm{Hom}}(b, c))$.

The arguments are objects a, b and a morphism $f : a \otimes b \to c$. The output is a morphism $g : a \to \underline{\mathrm{Hom}}(b, c)$ corresponding to f under the tensor hom adjunction.

1.5.9 InternalHomToTensorProductAdjunctionMap (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryMorphism)

▷ `InternalHomToTensorProductAdjunctionMap(b, c, g)` (operation)

Returns: a morphism in $\mathrm{Hom}(a \otimes b, c)$.

The arguments are objects b, c and a morphism $g : a \to \underline{\mathrm{Hom}}(b, c)$. The output is a morphism $f : a \otimes b \to c$ corresponding to g under the tensor hom adjunction.

1.5.10 MonoidalPreComposeMorphism (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MonoidalPreComposeMorphism(a, b, c)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a,b) \otimes \underline{\mathrm{Hom}}(b,c), \underline{\mathrm{Hom}}(a,c))$.

The arguments are three objects a, b, c. The output is the precomposition morphism $\mathrm{MonoidalPreComposeMorphismWithGivenObjects}_{a,b,c} : \underline{\mathrm{Hom}}(a,b) \otimes \underline{\mathrm{Hom}}(b,c) \to \underline{\mathrm{Hom}}(a,c)$.

1.5.11 MonoidalPreComposeMorphismWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MonoidalPreComposeMorphismWithGivenObjects(s, a, b, c, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a,b) \otimes \underline{\mathrm{Hom}}(b,c), \underline{\mathrm{Hom}}(a,c))$.

The arguments are an object $s = \underline{\mathrm{Hom}}(a,b) \otimes \underline{\mathrm{Hom}}(b,c)$, three objects a, b, c, and an object $r = \underline{\mathrm{Hom}}(a,c)$. The output is the precomposition morphism $\mathrm{MonoidalPreComposeMorphismWithGivenObjects}_{a,b,c} : \underline{\mathrm{Hom}}(a,b) \otimes \underline{\mathrm{Hom}}(b,c) \to \underline{\mathrm{Hom}}(a,c)$.

1.5.12 MonoidalPostComposeMorphism (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MonoidalPostComposeMorphism(a, b, c)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(b,c) \otimes \underline{\mathrm{Hom}}(a,b), \underline{\mathrm{Hom}}(a,c))$.

The arguments are three objects a,b,c. The output is the postcomposition morphism $\mathrm{MonoidalPostComposeMorphismWithGivenObjects}_{a,b,c} : \underline{\mathrm{Hom}}(b,c) \otimes \underline{\mathrm{Hom}}(a,b) \to \underline{\mathrm{Hom}}(a,c)$.

1.5.13 MonoidalPostComposeMorphismWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MonoidalPostComposeMorphismWithGivenObjects(s, a, b, c, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(b,c) \otimes \underline{\mathrm{Hom}}(a,b), \underline{\mathrm{Hom}}(a,c))$.

The arguments are an object $s = \underline{\mathrm{Hom}}(b,c) \otimes \underline{\mathrm{Hom}}(a,b)$, three objects a,b,c, and an object $r = \underline{\mathrm{Hom}}(a,c)$. The output is the postcomposition morphism $\mathrm{MonoidalPostComposeMorphismWithGivenObjects}_{a,b,c} : \underline{\mathrm{Hom}}(b,c) \otimes \underline{\mathrm{Hom}}(a,b) \to \underline{\mathrm{Hom}}(a,c)$.

1.5.14 DualOnObjects (for IsCapCategoryObject)

▷ `DualOnObjects(a)` (attribute)

Returns: an object

The argument is an object a. The output is its dual object $a^{\vee}$.

1.5.15 DualOnMorphisms (for IsCapCategoryMorphism)

▷ `DualOnMorphisms(alpha)` (attribute)

Returns: a morphism in $\mathrm{Hom}(b^{\vee}, a^{\vee})$.

The argument is a morphism $\alpha : a \to b$. The output is its dual morphism $\alpha^{\vee} : b^{\vee} \to a^{\vee}$.

1.5.16 DualOnMorphismsWithGivenDuals (for IsCapCategoryObject, IsCapCategoryMorphism, IsCapCategoryObject)

▷ `DualOnMorphismsWithGivenDuals(s, alpha, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(b^{\vee}, a^{\vee})$.

The argument is an object $s = b^{\vee}$, a morphism $\alpha : a \to b$, and an object $r = a^{\vee}$. The output is the dual morphism $\alpha^{\vee} : b^{\vee} \to a^{\vee}$.

1.5.17 EvaluationForDual (for IsCapCategoryObject)

▷ `EvaluationForDual(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(a^{\vee} \otimes a, 1)$.

The argument is an object a. The output is the evaluation morphism $\mathrm{ev}_a : a^{\vee} \otimes a \to 1$.

1.5.18 EvaluationForDualWithGivenTensorProduct (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `EvaluationForDualWithGivenTensorProduct(s, a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a^{\vee} \otimes a, 1)$.

The arguments are an object $s = a^{\vee} \otimes a$, an object a, and an object $r = 1$. The output is the evaluation morphism $\mathrm{ev}_a : a^{\vee} \otimes a \to 1$.

1.5.19 MorphismToBidual (for IsCapCategoryObject)

▷ `MorphismToBidual(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(a, (a^\vee)^\vee)$.

The argument is an object a. The output is the morphism to the bidual $a \to (a^\vee)^\vee$.

1.5.20 MorphismToBidualWithGivenBidual (for IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismToBidualWithGivenBidual(a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a, (a^\vee)^\vee)$.

The arguments are an object a, and an object $r = (a^\vee)^\vee$. The output is the morphism to the bidual $a \to (a^\vee)^\vee$.

1.5.21 TensorProductInternalHomCompatibilityMorphism (for IsList)

▷ `TensorProductInternalHomCompatibilityMorphism(list)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a,a') \otimes \underline{\mathrm{Hom}}(b,b'), \underline{\mathrm{Hom}}(a \otimes b, a' \otimes b'))$.

The argument is a list of four objects $[a,a',b,b']$. The output is the natural morphism $\mathrm{TensorProductInternalHomCompatibilityMorphismWithGivenObjects}_{a,a',b,b'} : \underline{\mathrm{Hom}}(a,a') \otimes \underline{\mathrm{Hom}}(b,b') \to \underline{\mathrm{Hom}}(a \otimes b, a' \otimes b')$.

1.5.22 TensorProductInternalHomCompatibilityMorphismWithGivenObjects (for IsCapCategoryObject, IsList, IsCapCategoryObject)

▷ `TensorProductInternalHomCompatibilityMorphismWithGivenObjects(s, list, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a,a') \otimes \underline{\mathrm{Hom}}(b,b'), \underline{\mathrm{Hom}}(a \otimes b, a' \otimes b'))$.

The arguments are a list of four objects $[a,a',b,b']$, and two objects $s = \underline{\mathrm{Hom}}(a,a') \otimes \underline{\mathrm{Hom}}(b,b')$ and $r = \underline{\mathrm{Hom}}(a \otimes b, a' \otimes b')$. The output is the natural morphism $\mathrm{TensorProductInternalHomCompatibilityMorphismWithGivenObjects}_{a,a',b,b'} : \underline{\mathrm{Hom}}(a,a') \otimes \underline{\mathrm{Hom}}(b,b') \to \underline{\mathrm{Hom}}(a \otimes b, a' \otimes b')$.

1.5.23 TensorProductDualityCompatibilityMorphism (for IsCapCategoryObject, IsCapCategoryObject)

▷ `TensorProductDualityCompatibilityMorphism(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(a^\vee \otimes b^\vee, (a \otimes b)^\vee)$.

The arguments are two objects a,b. The output is the natural morphism $\mathrm{TensorProductDualityCompatibilityMorphismWithGivenObjects} : a^\vee \otimes b^\vee \to (a \otimes b)^\vee$.

1.5.24 TensorProductDualityCompatibilityMorphismWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `TensorProductDualityCompatibilityMorphismWithGivenObjects(s, a, b, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a^\vee \otimes b^\vee, (a \otimes b)^\vee)$.

The arguments are an object $s = a^\vee \otimes b^\vee$, two objects a,b, and an object $r = (a \otimes b)^\vee$. The output is the natural morphism TensorProductDualityCompatibilityMorphismWithGivenObjects$_{a,b}$: $a^\vee \otimes b^\vee \to (a \otimes b)^\vee$.

1.5.25 MorphismFromTensorProductToInternalHom (for IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismFromTensorProductToInternalHom(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(a^\vee \otimes b, \underline{\mathrm{Hom}}(a,b))$.

The arguments are two objects a,b. The output is the natural morphism MorphismFromTensorProductToInternalHomWithGivenObjects$_{a,b}$: $a^\vee \otimes b \to \underline{\mathrm{Hom}}(a,b)$.

1.5.26 MorphismFromTensorProductToInternalHomWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismFromTensorProductToInternalHomWithGivenObjects(s, a, b, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a^\vee \otimes b, \underline{\mathrm{Hom}}(a,b))$.

The arguments are an object $s = a^\vee \otimes b$, two objects a,b, and an object $r = \underline{\mathrm{Hom}}(a,b)$. The output is the natural morphism MorphismFromTensorProductToInternalHomWithGivenObjects$_{a,b}$: $a^\vee \otimes b \to \underline{\mathrm{Hom}}(a,b)$.

1.5.27 IsomorphismFromDualObjectToInternalHomIntoTensorUnit (for IsCapCategoryObject)

▷ `IsomorphismFromDualObjectToInternalHomIntoTensorUnit(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(a^\vee, \underline{\mathrm{Hom}}(a,1))$.

The argument is an object a. The output is the isomorphism IsomorphismFromDualObjectToInternalHomIntoTensorUnit$_a$: $a^\vee \to \underline{\mathrm{Hom}}(a,1)$.

1.5.28 IsomorphismFromInternalHomIntoTensorUnitToDualObject (for IsCapCategoryObject)

▷ `IsomorphismFromInternalHomIntoTensorUnitToDualObject(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a,1), a^\vee)$.

The argument is an object a. The output is the isomorphism IsomorphismFromInternalHomIntoTensorUnitToDualObject$_a$: $\underline{\mathrm{Hom}}(a,1) \to a^\vee$.

1.5.29 UniversalPropertyOfDual (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryMorphism)

▷ `UniversalPropertyOfDual(t, a, alpha)` (operation)

Returns: a morphism in $\mathrm{Hom}(t, a^\vee)$.

The arguments are two objects t,a, and a morphism $\alpha : t \otimes a \to 1$. The output is the morphism $t \to a^\vee$ given by the universal property of $a^\vee$.

1.5.30 LambdaIntroduction (for IsCapCategoryMorphism)

▷ `LambdaIntroduction(alpha)` (attribute)

Returns: a morphism in $\mathrm{Hom}(1,\underline{\mathrm{Hom}}(a,b))$.

The argument is a morphism $\alpha : a \to b$. The output is the corresponding morphism $1 \to \underline{\mathrm{Hom}}(a,b)$ under the tensor hom adjunction.

1.5.31 LambdaElimination (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryMorphism)

▷ `LambdaElimination(a, b, alpha)` (operation)

Returns: a morphism in $\mathrm{Hom}(a,b)$.

The arguments are two objects a,b, and a morphism $\alpha : 1 \to \underline{\mathrm{Hom}}(a,b)$. The output is a morphism $a \to b$ corresponding to α under the tensor hom adjunction.

1.5.32 IsomorphismFromObjectToInternalHom (for IsCapCategoryObject)

▷ `IsomorphismFromObjectToInternalHom(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(a,\underline{\mathrm{Hom}}(1,a))$.

The argument is an object a. The output is the natural isomorphism $a \to \underline{\mathrm{Hom}}(1,a)$.

1.5.33 IsomorphismFromObjectToInternalHomWithGivenInternalHom (for IsCapCategoryObject, IsCapCategoryObject)

▷ `IsomorphismFromObjectToInternalHomWithGivenInternalHom(a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a,\underline{\mathrm{Hom}}(1,a))$.

The argument is an object a, and an object $r = \underline{\mathrm{Hom}}(1,a)$. The output is the natural isomorphism $a \to \underline{\mathrm{Hom}}(1,a)$.

1.5.34 IsomorphismFromInternalHomToObject (for IsCapCategoryObject)

▷ `IsomorphismFromInternalHomToObject(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(1,a),a)$.

The argument is an object a. The output is the natural isomorphism $\underline{\mathrm{Hom}}(1,a) \to a$.

1.5.35 IsomorphismFromInternalHomToObjectWithGivenInternalHom (for IsCapCategoryObject, IsCapCategoryObject)

▷ `IsomorphismFromInternalHomToObjectWithGivenInternalHom(a, s)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(1,a),a)$.

The argument is an object a, and an object $s = \underline{\mathrm{Hom}}(1,a)$. The output is the natural isomorphism $\underline{\mathrm{Hom}}(1,a) \to a$.

1.6 Coclosed Monoidal Categories

A monoidal category $\mathbf{C}$ which has for each functor $-\otimes b : \mathbf{C} \to \mathbf{C}$ a left adjoint (denoted by $\underline{\mathrm{coHom}}(-,b)$) is called a *coclosed monoidal category*.

If no operations involving coduals are installed manually, the codual objects will be derived as $a_\vee := \underline{\mathrm{coHom}}(1,a)$.

The corresponding GAP property is called `IsCoclosedMonoidalCategory`.

1.6.1 InternalCoHomOnObjects (for IsCapCategoryObject, IsCapCategoryObject)

▷ `InternalCoHomOnObjects(a, b)` (operation)

Returns: an object

The arguments are two objects a,b. The output is the internal cohom object $\underline{\mathrm{coHom}}(a,b)$.

1.6.2 InternalCoHomOnMorphisms (for IsCapCategoryMorphism, IsCapCategoryMorphism)

▷ `InternalCoHomOnMorphisms(alpha, beta)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,b'),\underline{\mathrm{coHom}}(a',b))$

The arguments are two morphisms $\alpha : a \rightarrow a', \beta : b \rightarrow b'$. The output is the internal cohom morphism $\underline{\mathrm{coHom}}(\alpha,\beta) : \underline{\mathrm{coHom}}(a,b') \rightarrow \underline{\mathrm{coHom}}(a',b)$.

1.6.3 InternalCoHomOnMorphismsWithGivenInternalCoHoms (for IsCapCategoryObject, IsCapCategoryMorphism, IsCapCategoryMorphism, IsCapCategoryObject)

▷ `InternalCoHomOnMorphismsWithGivenInternalCoHoms(s, alpha, beta, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,b'),\underline{\mathrm{coHom}}(a',b))$

The arguments are an object $s = \underline{\mathrm{coHom}}(a,b')$, two morphisms $\alpha : a \rightarrow a', \beta : b \rightarrow b'$, and an object $r = \underline{\mathrm{coHom}}(a',b)$. The output is the internal cohom morphism $\underline{\mathrm{coHom}}(\alpha,\beta) : \underline{\mathrm{coHom}}(a,b') \rightarrow \underline{\mathrm{coHom}}(a',b)$.

1.6.4 CoclosedEvaluationMorphism (for IsCapCategoryObject, IsCapCategoryObject)

▷ `CoclosedEvaluationMorphism(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(a,\underline{\mathrm{coHom}}(a,b) \otimes b)$.

The arguments are two objects a,b. The output is the coclosed evaluation morphism $\mathrm{coclev}_{a,b} : a \rightarrow \underline{\mathrm{coHom}}(a,b) \otimes b$, i.e., the unit of the cohom tensor adjunction.

1.6.5 CoclosedEvaluationMorphismWithGivenRange (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `CoclosedEvaluationMorphismWithGivenRange(a, b, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a,\underline{\mathrm{coHom}}(a,b) \otimes b)$.

The arguments are two objects a,b and an object $r = \underline{\mathrm{coHom}}(a,b) \otimes b$. The output is the coclosed evaluation morphism $\mathrm{coclev}_{a,b} : a \rightarrow \underline{\mathrm{coHom}}(a,b) \otimes b$, i.e., the unit of the cohom tensor adjunction.

1.6.6 CoclosedCoevaluationMorphism (for IsCapCategoryObject, IsCapCategoryObject)

▷ `CoclosedCoevaluationMorphism(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a \otimes b, b), a)$.

The arguments are two objects a, b. The output is the coclosed coevaluation morphism $\mathrm{coclcoev}_{a,b} : \underline{\mathrm{coHom}}(a \otimes b, b) \to a$, i.e., the counit of the cohom tensor adjunction.

1.6.7 CoclosedCoevaluationMorphismWithGivenSource (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `CoclosedCoevaluationMorphismWithGivenSource(a, b, s)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a \otimes b, b), b)$.

The arguments are two objects a, b and an object $s = \underline{\mathrm{coHom}}(\mathrm{a} \otimes \mathrm{b}, \mathrm{b})$. The output is the coclosed coevaluation morphism $\mathrm{coclcoev}_{a,b} : \underline{\mathrm{coHom}}(a \otimes b, b) \to a$, i.e., the unit of the cohom tensor adjunction.

1.6.8 TensorProductToInternalCoHomAdjunctionMap (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryMorphism)

▷ `TensorProductToInternalCoHomAdjunctionMap(c, b, g)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a, b), c)$.

The arguments are objects c, b and a morphism $g : a \to c \otimes b$. The output is a morphism $f : \underline{\mathrm{coHom}}(a, b) \to c$ corresponding to g under the cohom tensor adjunction.

1.6.9 InternalCoHomToTensorProductAdjunctionMap (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryMorphism)

▷ `InternalCoHomToTensorProductAdjunctionMap(a, b, f)` (operation)

Returns: a morphism in $\mathrm{Hom}(a, c \otimes b)$.

The arguments are objects a, b and a morphism $f : \underline{\mathrm{coHom}}(a, b) \to c$. The output is a morphism $g : a \to c \otimes b$ corresponding to f under the cohom tensor adjunction.

1.6.10 MonoidalPreCoComposeMorphism (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MonoidalPreCoComposeMorphism(a, b, c)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a, c), \underline{\mathrm{coHom}}(b, c) \otimes \underline{\mathrm{coHom}}(a, b))$.

The arguments are three objects a, b, c. The output is the precocomposition morphism $\mathrm{MonoidalPreCoComposeMorphismWithGivenObjects}_{a,b,c} : \underline{\mathrm{coHom}}(a, c) \to \underline{\mathrm{coHom}}(b, c) \otimes \underline{\mathrm{coHom}}(a, b)$.

1.6.11 MonoidalPreCoComposeMorphismWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MonoidalPreCoComposeMorphismWithGivenObjects(s, a, b, c, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a, c), \underline{\mathrm{coHom}}(b, c) \otimes \underline{\mathrm{coHom}}(a, b))$.

The arguments are an object $s = \underline{\mathrm{coHom}}(a,c)$, three objects a,b,c, and an object $r = \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(b,c)$. The output is the precocomposition morphism $\mathrm{MonoidalPreCoComposeMorphismWithGivenObjects}_{a,b,c} : \underline{\mathrm{coHom}}(a,c) \to \underline{\mathrm{coHom}}(b,c) \otimes \underline{\mathrm{coHom}}(a,b)$.

1.6.12 MonoidalPostCoComposeMorphism (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MonoidalPostCoComposeMorphism(a, b, c)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,c), \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(b,c))$.

The arguments are three objects a,b,c. The output is the postcocomposition morphism $\mathrm{MonoidalPostCoComposeMorphismWithGivenObjects}_{a,b,c} : \underline{\mathrm{coHom}}(a,c) \to \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(b,c)$.

1.6.13 MonoidalPostCoComposeMorphismWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MonoidalPostCoComposeMorphismWithGivenObjects(s, a, b, c, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,c), \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(b,c))$.

The arguments are an object $s = \underline{\mathrm{coHom}}(a,c)$, three objects a,b,c, and an object $r = \underline{\mathrm{coHom}}(b,c) \otimes \underline{\mathrm{coHom}}(a,b)$. The output is the postcocomposition morphism $\mathrm{MonoidalPostCoComposeMorphismWithGivenObjects}_{a,b,c} : \underline{\mathrm{coHom}}(a,c) \to \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(b,c)$.

1.6.14 CoDualOnObjects (for IsCapCategoryObject)

▷ `CoDualOnObjects(a)` (attribute)

Returns: an object

The argument is an object a. The output is its codual object $a_{\vee}$.

1.6.15 CoDualOnMorphisms (for IsCapCategoryMorphism)

▷ `CoDualOnMorphisms(alpha)` (attribute)

Returns: a morphism in $\mathrm{Hom}(b_{\vee}, a_{\vee})$.

The argument is a morphism $\alpha : a \to b$. The output is its codual morphism $\alpha_{\vee} : b_{\vee} \to a_{\vee}$.

1.6.16 CoDualOnMorphismsWithGivenCoDuals (for IsCapCategoryObject, IsCapCategoryMorphism, IsCapCategoryObject)

▷ `CoDualOnMorphismsWithGivenCoDuals(s, alpha, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(b_{\vee}, a_{\vee})$.

The argument is an object $s = b_{\vee}$, a morphism $\alpha : a \to b$, and an object $r = a_{\vee}$. The output is the dual morphism $\alpha_{\vee} : b^{\vee} \to a^{\vee}$.

1.6.17 CoclosedEvaluationForCoDual (for IsCapCategoryObject)

▷ `CoclosedEvaluationForCoDual(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(1, a_\vee \otimes a)$.

The argument is an object a. The output is the coclosed evaluation morphism $\mathrm{coclev}_a : 1 \to a_\vee \otimes a$.

1.6.18 CoclosedEvaluationForCoDualWithGivenTensorProduct (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `CoclosedEvaluationForCoDualWithGivenTensorProduct(s, a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(1, a_\vee \otimes a)$.

The arguments are an object $s = 1$, an object a, and an object $r = a_\vee \otimes a$. The output is the coclosed evaluation morphism $\mathrm{coclev}_a : 1 \to a_\vee \otimes a$.

1.6.19 MorphismFromCoBidual (for IsCapCategoryObject)

▷ `MorphismFromCoBidual(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}((a_\vee)_\vee, a)$.

The argument is an object a. The output is the morphism from the cobidual $(a_\vee)_\vee \to a$.

1.6.20 MorphismFromCoBidualWithGivenCoBidual (for IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismFromCoBidualWithGivenCoBidual(a, s)` (operation)

Returns: a morphism in $\mathrm{Hom}((a_\vee)_\vee, a)$.

The arguments are an object a, and an object $s = (a_\vee)_\vee$. The output is the morphism from the cobidual $(a_\vee)_\vee \to a$.

1.6.21 InternalCoHomTensorProductCompatibilityMorphism (for IsList)

▷ `InternalCoHomTensorProductCompatibilityMorphism(list)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a \otimes a', b \otimes b'), \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(a',b'))$.

The argument is a list of four objects $[a,a',b,b']$. The output is the natural morphism $\mathrm{InternalCoHomTensorProductCompatibilityMorphismWithGivenObjects}_{a,a',b,b'} : \underline{\mathrm{coHom}}(a \otimes a', b \otimes b') \to \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(a',b')$.

1.6.22 InternalCoHomTensorProductCompatibilityMorphismWithGivenObjects (for IsCapCategoryObject, IsList, IsCapCategoryObject)

▷ `InternalCoHomTensorProductCompatibilityMorphismWithGivenObjects(s, list, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a \otimes a', b \otimes b'), \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(a',b'))$.

The arguments are a list of four objects $[a,a',b,b']$, and two objects $s = \underline{\mathrm{coHom}}(a \otimes a', b \otimes b')$ and $r = \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(a',b')$. The output is the natural morphism $\mathrm{InternalCoHomTensorProductCompatibilityMorphismWithGivenObjects}_{a,a',b,b'} : \underline{\mathrm{coHom}}(a \otimes a', b \otimes b') \to \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(a',b')$.

1.6.23 CoDualityTensorProductCompatibilityMorphism (for IsCapCategoryObject, IsCapCategoryObject)

▷ `CoDualityTensorProductCompatibilityMorphism(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}((a \otimes b)_\vee, a_\vee \otimes b_\vee)$.

The arguments are two objects a, b. The output is the natural morphism CoDualityTensorProductCompatibilityMorphismWithGivenObjects : $(a \otimes b)_\vee \to a_\vee \otimes b_\vee$.

1.6.24 CoDualityTensorProductCompatibilityMorphismWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `CoDualityTensorProductCompatibilityMorphismWithGivenObjects(s, a, b, r)` (operation)

Returns: a morphism in $\mathrm{Hom}((a \otimes b)_\vee, a_\vee \otimes b_\vee)$.

The arguments are an object $s = (a \otimes b)_\vee$, two objects a, b, and an object $r = a_\vee \otimes b_\vee$. The output is the natural morphism $\mathrm{CoDualityTensorProductCompatibilityMorphismWithGivenObjects}_{a,b} : (a \otimes b)_\vee \to a_\vee \otimes b_\vee$.

1.6.25 MorphismFromInternalCoHomToTensorProduct (for IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismFromInternalCoHomToTensorProduct(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,b), b_\vee \otimes a)$.

The arguments are two objects a, b. The output is the natural morphism $\mathrm{MorphismFromInternalCoHomToTensorProductWithGivenObjects}_{a,b} : \underline{\mathrm{coHom}}(a,b) \to b_\vee \otimes a$.

1.6.26 MorphismFromInternalCoHomToTensorProductWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismFromInternalCoHomToTensorProductWithGivenObjects(s, a, b, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,b), a \otimes b_\vee)$.

The arguments are an object $s = \underline{\mathrm{coHom}}(a,b)$, two objects a, b, and an object $r = b_\vee \otimes a$. The output is the natural morphism $\mathrm{MorphismFromInternalCoHomToTensorProductWithGivenObjects}_{a,b} : \underline{\mathrm{coHom}}(a,b) \to a \otimes b_\vee$.

1.6.27 IsomorphismFromCoDualObjectToInternalCoHomFromTensorUnit (for IsCapCategoryObject)

▷ `IsomorphismFromCoDualObjectToInternalCoHomFromTensorUnit(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(a_\vee, \underline{\mathrm{coHom}}(1, a))$.

The argument is an object a. The output is the isomorphism $\mathrm{IsomorphismFromCoDualObjectToInternalCoHomFromTensorUnit}_a : a_\vee \to \underline{\mathrm{coHom}}(1, a)$.

1.6.28 IsomorphismFromInternalCoHomFromTensorUnitToCoDualObject (for IsCapCategoryObject)

▷ `IsomorphismFromInternalCoHomFromTensorUnitToCoDualObject(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(1,a),a_\vee)$.

The argument is an object a. The output is the isomorphism $\mathrm{IsomorphismFromInternalCoHomFromTensorUnitToCoDualObject}_a : \underline{\mathrm{coHom}}(1,a) \to a_\vee$.

1.6.29 UniversalPropertyOfCoDual (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryMorphism)

▷ `UniversalPropertyOfCoDual(t, a, alpha)` (operation)

Returns: a morphism in $\mathrm{Hom}(a_\vee,t)$.

The arguments are two objects t,a, and a morphism $\alpha : 1 \to t \otimes a$. The output is the morphism $a_\vee \to t$ given by the universal property of $a_\vee$.

1.6.30 CoLambdaIntroduction (for IsCapCategoryMorphism)

▷ `CoLambdaIntroduction(alpha)` (attribute)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,b),1)$.

The argument is a morphism $\alpha : a \to b$. The output is the corresponding morphism $\underline{\mathrm{coHom}}(a,b) \to 1$ under the cohom tensor adjunction.

1.6.31 CoLambdaElimination (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryMorphism)

▷ `CoLambdaElimination(a, b, alpha)` (operation)

Returns: a morphism in $\mathrm{Hom}(a,b)$.

The arguments are two objects a,b, and a morphism $\alpha : \underline{\mathrm{coHom}}(a,b) \to 1$. The output is a morphism $a \to b$ corresponding to α under the cohom tensor adjunction.

1.6.32 IsomorphismFromObjectToInternalCoHom (for IsCapCategoryObject)

▷ `IsomorphismFromObjectToInternalCoHom(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(a,\underline{\mathrm{coHom}}(a,1))$.

The argument is an object a. The output is the natural isomorphism $a \to \underline{\mathrm{coHom}}(a,1)$.

1.6.33 IsomorphismFromObjectToInternalCoHomWithGivenInternalCoHom (for IsCapCategoryObject, IsCapCategoryObject)

▷ `IsomorphismFromObjectToInternalCoHomWithGivenInternalCoHom(a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a,\underline{\mathrm{coHom}}(a,1))$.

The argument is an object a, and an object $r = \underline{\mathrm{coHom}}(a,1)$. The output is the natural isomorphism $a \to \underline{\mathrm{coHom}}(a,1)$.

1.6.34 IsomorphismFromInternalCoHomToObject (for IsCapCategoryObject)

▷ `IsomorphismFromInternalCoHomToObject(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,1),a)$.

The argument is an object a. The output is the natural isomorphism $\underline{\mathrm{coHom}}(a,1) \to a$.

1.6.35 IsomorphismFromInternalCoHomToObjectWithGivenInternalCoHom (for IsCapCategoryObject, IsCapCategoryObject)

▷ `IsomorphismFromInternalCoHomToObjectWithGivenInternalCoHom(a, s)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,1),a)$.

The argument is an object a, and an object $s = \underline{\mathrm{coHom}}(a,1)$. The output is the natural isomorphism $\underline{\mathrm{coHom}}(a,1) \to a$.

1.7 Symmetric Closed Monoidal Categories

A monoidal category **C** which is symmetric and closed is called a *symmetric closed monoidal category*.

The corresponding GAP property is given by `IsSymmetricClosedMonoidalCategory`.

1.8 Symmetric Coclosed Monoidal Categories

A monoidal category **C** which is symmetric and coclosed is called a *symmetric coclosed monoidal category*.

The corresponding GAP property is given by `IsSymmetricCoclosedMonoidalCategory`.

1.9 Rigid Symmetric Closed Monoidal Categories

A symmetric closed monoidal category **C** satisfying

- the natural morphism

$\underline{\mathrm{Hom}}(a,a') \otimes \underline{\mathrm{Hom}}(b,b') \to \underline{\mathrm{Hom}}(a \otimes b, a' \otimes b')$ is an isomorphism,

- the natural morphism

$a \to \underline{\mathrm{Hom}}(\underline{\mathrm{Hom}}(a,1),1)$ is an isomorphism is called a *rigid symmetric closed monoidal category*.

If no operations involving the closed structure are installed manually, the internal hom objects will be derived as $\underline{\mathrm{Hom}}(a,b) := a^\vee \otimes b$ and, in particular, $\underline{\mathrm{Hom}}(a,1) := a^\vee \otimes 1$.

The corresponding GAP property is given by `IsRigidSymmetricClosedMonoidalCategory`.

1.9.1 IsomorphismFromTensorProductWithDualObjectToInternalHom (for IsCapCategoryObject, IsCapCategoryObject)

▷ `IsomorphismFromTensorProductWithDualObjectToInternalHom(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(a^\vee \otimes b, \underline{\mathrm{Hom}}(a,b))$.

The arguments are two objects a,b. The output is the natural morphism $\mathrm{IsomorphismFromTensorProductWithDualObjectToInternalHom}_{a,b} : a^\vee \otimes b \to \underline{\mathrm{Hom}}(a,b)$.

1.9.2 IsomorphismFromInternalHomToTensorProductWithDualObject (for IsCapCategoryObject, IsCapCategoryObject)

▷ `IsomorphismFromInternalHomToTensorProductWithDualObject(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a,b), a^{\vee} \otimes b)$.

The arguments are two objects a,b. The output is the inverse of IsomorphismFromTensorProductWithDualObjectToInternalHom, namely IsomorphismFromInternalHomToTensorProductWithDualObject$_{a,b} : \underline{\mathrm{Hom}}(a,b) \to a^{\vee} \otimes b$.

1.9.3 MorphismFromInternalHomToTensorProduct (for IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismFromInternalHomToTensorProduct(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a,b), a^{\vee} \otimes b)$.

The arguments are two objects a,b. The output is the inverse of MorphismFromTensorProductToInternalHomWithGivenObjects, namely MorphismFromInternalHomToTensorProductWithGivenObjects$_{a,b} : \underline{\mathrm{Hom}}(a,b) \to a^{\vee} \otimes b$.

1.9.4 MorphismFromInternalHomToTensorProductWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismFromInternalHomToTensorProductWithGivenObjects(s, a, b, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a,b), a^{\vee} \otimes b)$.

The arguments are an object $s = \underline{\mathrm{Hom}}(a,b)$, two objects a,b, and an object $r = a^{\vee} \otimes b$. The output is the inverse of MorphismFromTensorProductToInternalHomWithGivenObjects, namely MorphismFromInternalHomToTensorProductWithGivenObjects$_{a,b} : \underline{\mathrm{Hom}}(a,b) \to a^{\vee} \otimes b$.

1.9.5 TensorProductInternalHomCompatibilityMorphismInverse (for IsList)

▷ `TensorProductInternalHomCompatibilityMorphismInverse(list)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a \otimes b, a' \otimes b'), \underline{\mathrm{Hom}}(a,a') \otimes \underline{\mathrm{Hom}}(b,b'))$.

The argument is a list of four objects $[a,a',b,b']$. The output is the natural morphism TensorProductInternalHomCompatibilityMorphismInverseWithGivenObjects$_{a,a',b,b'} : \underline{\mathrm{Hom}}(a \otimes b, a' \otimes b') \to \underline{\mathrm{Hom}}(a,a') \otimes \underline{\mathrm{Hom}}(b,b')$.

1.9.6 TensorProductInternalHomCompatibilityMorphismInverseWithGivenObjects (for IsCapCategoryObject, IsList, IsCapCategoryObject)

▷ `TensorProductInternalHomCompatibilityMorphismInverseWithGivenObjects(s, list, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{Hom}}(a \otimes b, a' \otimes b'), \underline{\mathrm{Hom}}(a,a') \otimes \underline{\mathrm{Hom}}(b,b'))$.

The arguments are a list of four objects $[a,a',b,b']$, and two objects $s = \underline{\mathrm{Hom}}(a \otimes b, a' \otimes b')$ and $r = \underline{\mathrm{Hom}}(a,a') \otimes \underline{\mathrm{Hom}}(b,b')$. The output is the natural morphism TensorProductInternalHomCompatibilityMorphismInverseWithGivenObjects$_{a,a',b,b'}$: $\underline{\mathrm{Hom}}(a \otimes b, a' \otimes b') \to \underline{\mathrm{Hom}}(a,a') \otimes \underline{\mathrm{Hom}}(b,b')$.

1.9.7 CoevaluationForDual (for IsCapCategoryObject)

▷ `CoevaluationForDual(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(1, a \otimes a^{\vee})$.

The argument is an object a. The output is the coevaluation morphism $\mathrm{coev}_a : 1 \to a \otimes a^{\vee}$.

1.9.8 CoevaluationForDualWithGivenTensorProduct (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `CoevaluationForDualWithGivenTensorProduct(s, a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(1, a \otimes a^{\vee})$.

The arguments are an object $s = 1$, an object a, and an object $r = a \otimes a^{\vee}$. The output is the coevaluation morphism $\mathrm{coev}_a : 1 \to a \otimes a^{\vee}$.

1.9.9 TraceMap (for IsCapCategoryMorphism)

▷ `TraceMap(alpha)` (attribute)

Returns: a morphism in $\mathrm{Hom}(1, 1)$.

The argument is an endomorphism $\alpha : a \to a$. The output is the trace morphism $\mathrm{trace}_{\alpha} : 1 \to 1$.

1.9.10 RankMorphism (for IsCapCategoryObject)

▷ `RankMorphism(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(1, 1)$.

The argument is an object a. The output is the rank morphism $\mathrm{rank}_a : 1 \to 1$.

1.9.11 MorphismFromBidual (for IsCapCategoryObject)

▷ `MorphismFromBidual(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}((a^{\vee})^{\vee}, a)$.

The argument is an object a. The output is the inverse of the morphism to the bidual $(a^{\vee})^{\vee} \to a$.

1.9.12 MorphismFromBidualWithGivenBidual (for IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismFromBidualWithGivenBidual(a, s)` (operation)

Returns: a morphism in $\mathrm{Hom}((a^{\vee})^{\vee}, a)$.

The argument is an object a, and an object $s = (a^{\vee})^{\vee}$. The output is the inverse of the morphism to the bidual $(a^{\vee})^{\vee} \to a$.

1.10 Rigid Symmetric Coclosed Monoidal Categories

A symmetric coclosed monoidal category **C** satisfying

- the natural morphism

$\underline{\mathrm{coHom}}(a \otimes a', b \otimes b') \to \underline{\mathrm{coHom}}(a, b) \otimes \underline{\mathrm{coHom}}(a', b')$ is an isomorphism,

- the natural morphism

$\underline{\mathrm{coHom}}(1,\underline{\mathrm{coHom}}(1,a)) \rightarrow a$ is an isomorphism is called a *rigid symmetric coclosed monoidal category*.

If no operations involving the coclosed structure are installed manually, the internal cohom objects will be derived as $\underline{\mathrm{coHom}}(a,b) := a \otimes b_\vee$ and, in particular, $\underline{\mathrm{coHom}}(1,a) := 1 \otimes a_\vee$.

The corresponding GAP property is given by `IsRigidSymmetricCoclosedMonoidalCategory`.

1.10.1 IsomorphismFromInternalCoHomToTensorProductWithCoDualObject (for IsCapCategoryObject, IsCapCategoryObject)

▷ `IsomorphismFromInternalCoHomToTensorProductWithCoDualObject(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,b), b_\vee \otimes a)$.

The arguments are two objects a,b. The output is the natural morphism $\mathrm{IsomorphismFromInternalCoHomToTensorProductWithCoDualObjectWithGivenObjects}_{a,b} : \underline{\mathrm{coHom}}(a,b) \rightarrow b_\vee \otimes a$.

1.10.2 IsomorphismFromTensorProductWithCoDualObjectToInternalCoHom (for IsCapCategoryObject, IsCapCategoryObject)

▷ `IsomorphismFromTensorProductWithCoDualObjectToInternalCoHom(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(a_\vee \otimes b, \underline{\mathrm{coHom}}(b,a)$.

The arguments are two objects a,b. The output is the inverse of IsomorphismFromInternalCoHomToTensorProductWithCoDualObject, namely $\mathrm{IsomorphismFromTensorProductWithCoDualObjectToInternalCoHom}_{a,b} : a_\vee \otimes b \rightarrow \underline{\mathrm{coHom}}(b,a)$.

1.10.3 MorphismFromTensorProductToInternalCoHom (for IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismFromTensorProductToInternalCoHom(a, b)` (operation)

Returns: a morphism in $\mathrm{Hom}(a_\vee \otimes b, \underline{\mathrm{coHom}}(b,a))$.

The arguments are two objects a,b. The output is the inverse of MorphismFromInternalCoHomToTensorProductWithGivenObjects, namely $\mathrm{MorphismFromTensorProductToInternalCoHomWithGivenObjects}_{a,b} : a_\vee \otimes b \rightarrow \underline{\mathrm{coHom}}(b,a)$.

1.10.4 MorphismFromTensorProductToInternalCoHomWithGivenObjects (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismFromTensorProductToInternalCoHomWithGivenObjects(s, a, b, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a_\vee \otimes b, \underline{\mathrm{coHom}}(b,a)$.

The arguments are an object $s_\vee = a \otimes b$, two objects a,b, and an object $r = \underline{\mathrm{coHom}}(b,a)$. The output is the inverse of MorphismFromInternalCoHomToTensorProductWithGivenObjects, namely $\mathrm{MorphismFromTensorProductToInternalCoHomWithGivenObjects}_{a,b} : a_\vee \otimes b \rightarrow \underline{\mathrm{coHom}}(b,a)$.

1.10.5 InternalCoHomTensorProductCompatibilityMorphismInverse (for IsList)

▷ `InternalCoHomTensorProductCompatibilityMorphismInverse(list)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(a',b'), \underline{\mathrm{coHom}}(a \otimes a', b \otimes b'))$.

The argument is a list of four objects $[a,a',b,b']$. The output is the natural morphism $\mathrm{InternalCoHomTensorProductCompatibilityMorphismInverseWithGivenObjects}_{a,a',b,b'} : \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(a',b') \to \underline{\mathrm{coHom}}(a \otimes a', b \otimes b')$.

1.10.6 InternalCoHomTensorProductCompatibilityMorphismInverseWithGivenObjects (for IsCapCategoryObject, IsList, IsCapCategoryObject)

▷ `InternalCoHomTensorProductCompatibilityMorphismInverseWithGivenObjects(s, list, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(\underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(a',b'), \underline{\mathrm{coHom}}(a \otimes a', b \otimes b')$.

The arguments are a list of four objects $[a,a',b,b']$, and two objects $s = \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(a',b')$ and $r = \underline{\mathrm{coHom}}(a \otimes a', b \otimes b')$. The output is the natural morphism $\mathrm{InternalCoHomTensorProductCompatibilityMorphismInverseWithGivenObjects}_{a,a',b,b'} : \underline{\mathrm{coHom}}(a,b) \otimes \underline{\mathrm{coHom}}(a',b') \to \underline{\mathrm{coHom}}(a \otimes a', b \otimes b')$.

1.10.7 CoclosedCoevaluationForCoDual (for IsCapCategoryObject)

▷ `CoclosedCoevaluationForCoDual(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(a \otimes a_{\vee}, 1)$.

The argument is an object a. The output is the coclosed coevaluation morphism $\mathrm{coclcoev}_a : a \otimes a_{\vee} \to 1$.

1.10.8 CoclosedCoevaluationForCoDualWithGivenTensorProduct (for IsCapCategoryObject, IsCapCategoryObject, IsCapCategoryObject)

▷ `CoclosedCoevaluationForCoDualWithGivenTensorProduct(s, a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a \otimes a_{\vee}, 1)$.

The arguments are an object $s = a \otimes a_{\vee}$, an object a, and an object $r = 1$. The output is the coclosed coevaluation morphism $\mathrm{coclcoev}_a : a \otimes a_{\vee} \to 1$.

1.10.9 CoTraceMap (for IsCapCategoryMorphism)

▷ `CoTraceMap(alpha)` (attribute)

Returns: a morphism in $\mathrm{Hom}(1,1)$.

The argument is an endomorphism $\alpha : a \to a$. The output is the cotrace morphism $\mathrm{cotrace}_{\alpha} : 1 \to 1$.

1.10.10 CoRankMorphism (for IsCapCategoryObject)

▷ `CoRankMorphism(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(1,1)$.

The argument is an object a. The output is the corank morphism $\mathrm{corank}_a : 1 \to 1$.

1.10.11 MorphismToCoBidual (for IsCapCategoryObject)

▷ `MorphismToCoBidual(a)` (attribute)

Returns: a morphism in $\mathrm{Hom}(a, (a_{\vee})_{\vee})$.

The argument is an object a. The output is the inverse of the morphism from the cobidual $a \to (a_\vee)_\vee$.

1.10.12 MorphismToCoBidualWithGivenCoBidual (for IsCapCategoryObject, IsCapCategoryObject)

▷ `MorphismToCoBidualWithGivenCoBidual(a, r)` (operation)

Returns: a morphism in $\mathrm{Hom}(a,(a_\vee)_\vee)$.

The argument is an object a, and an object $r = (a_\vee)_\vee$. The output is the inverse of the morphism from the cobidual $a \to (a_\vee)_\vee$.

1.11 Convenience Methods

1.11.1 InternalHom (for IsCapCategoryCell, IsCapCategoryCell)

▷ `InternalHom(a, b)` (operation)

Returns: a cell

This is a convenience method. The arguments are two cells a,b. The output is the internal hom cell. If a,b are two CAP objects the output is the internal Hom object $\underline{\mathrm{Hom}}(a,b)$. If at least one of the arguments is a CAP morphism the output is a CAP morphism, namely the internal hom on morphisms, where any object is replaced by its identity morphism.

1.11.2 InternalCoHom (for IsCapCategoryCell, IsCapCategoryCell)

▷ `InternalCoHom(a, b)` (operation)

Returns: a cell

This is a convenience method. The arguments are two cells a,b. The output is the internal cohom cell. If a,b are two CAP objects the output is the internal cohom object $\underline{\mathrm{coHom}}(a,b)$. If at least one of the arguments is a CAP morphism the output is a CAP morphism, namely the internal cohom on morphisms, where any object is replaced by its identity morphism.

1.12 Add-methods

1.12.1 AddLeftDistributivityExpanding (for IsCapCategory, IsFunction)

▷ `AddLeftDistributivityExpanding(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `LeftDistributivityExpanding`. $F : (a,L) \mapsto$ `LeftDistributivityExpanding`(a,L).

1.12.2 AddLeftDistributivityExpandingWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddLeftDistributivityExpandingWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `LeftDistributivityExpandingWithGivenObjects`. $F : (s,a,L,r) \mapsto$ `LeftDistributivityExpandingWithGivenObjects`(s,a,L,r).

1.12.3 AddLeftDistributivityFactoring (for IsCapCategory, IsFunction)

▷ `AddLeftDistributivityFactoring(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `LeftDistributivityFactoring`. $F : (a,L) \mapsto$ `LeftDistributivityFactoring`(a,L).

1.12.4 AddLeftDistributivityFactoringWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddLeftDistributivityFactoringWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `LeftDistributivityFactoringWithGivenObjects`. $F : (s,a,L,r) \mapsto$ `LeftDistributivityFactoringWithGivenObjects`(s,a,L,r).

1.12.5 AddRightDistributivityExpanding (for IsCapCategory, IsFunction)

▷ `AddRightDistributivityExpanding(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `RightDistributivityExpanding`. $F : (L,a) \mapsto$ `RightDistributivityExpanding`(L,a).

1.12.6 AddRightDistributivityExpandingWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddRightDistributivityExpandingWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `RightDistributivityExpandingWithGivenObjects`. $F : (s,L,a,r) \mapsto$ `RightDistributivityExpandingWithGivenObjects`(s,L,a,r).

1.12.7 AddRightDistributivityFactoring (for IsCapCategory, IsFunction)

▷ `AddRightDistributivityFactoring(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `RightDistributivityFactoring`. $F : (L,a) \mapsto$ `RightDistributivityFactoring`(L,a).

1.12.8 AddRightDistributivityFactoringWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddRightDistributivityFactoringWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `RightDistributivityFactoringWithGivenObjects`. $F : (s,L,a,r) \mapsto$ `RightDistributivityFactoringWithGivenObjects`(s,L,a,r).

1.12.9 AddBraiding (for IsCapCategory, IsFunction)

▷ `AddBraiding(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `Braiding`. $F : (a,b) \mapsto$ `Braiding`(a,b).

1.12.10 AddBraidingInverse (for IsCapCategory, IsFunction)

▷ `AddBraidingInverse(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `BraidingInverse`. $F : (a,b) \mapsto$ `BraidingInverse`(a,b).

1.12.11 AddBraidingInverseWithGivenTensorProducts (for IsCapCategory, IsFunction)

▷ `AddBraidingInverseWithGivenTensorProducts(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `BraidingInverseWithGivenTensorProducts`. $F : (s,a,b,r) \mapsto$ `BraidingInverseWithGivenTensorProducts`(s,a,b,r).

1.12.12 AddBraidingWithGivenTensorProducts (for IsCapCategory, IsFunction)

▷ `AddBraidingWithGivenTensorProducts(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `BraidingWithGivenTensorProducts`. $F : (s,a,b,r) \mapsto$ `BraidingWithGivenTensorProducts`(s,a,b,r).

1.12.13 AddCoevaluationMorphism (for IsCapCategory, IsFunction)

▷ `AddCoevaluationMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoevaluationMorphism`. $F : (a,b) \mapsto$ `CoevaluationMorphism`(a,b).

1.12.14 AddCoevaluationMorphismWithGivenRange (for IsCapCategory, IsFunction)

▷ `AddCoevaluationMorphismWithGivenRange(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoevaluationMorphismWithGivenRange`. $F : (a,b,r) \mapsto$ `CoevaluationMorphismWithGivenRange`(a,b,r).

1.12.15 AddDualOnMorphisms (for IsCapCategory, IsFunction)

▷ `AddDualOnMorphisms(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `DualOnMorphisms`. $F : (alpha) \mapsto$ `DualOnMorphisms`$(alpha)$.

1.12.16 AddDualOnMorphismsWithGivenDuals (for IsCapCategory, IsFunction)

▷ `AddDualOnMorphismsWithGivenDuals(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `DualOnMorphismsWithGivenDuals`. $F : (s,alpha,r) \mapsto$ `DualOnMorphismsWithGivenDuals`$(s,alpha,r)$.

1.12.17 AddDualOnObjects (for IsCapCategory, IsFunction)

▷ `AddDualOnObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `DualOnObjects`. $F : (a) \mapsto$ `DualOnObjects`(a).

1.12.18 AddEvaluationForDual (for IsCapCategory, IsFunction)

▷ `AddEvaluationForDual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `EvaluationForDual`. $F : (a) \mapsto$ `EvaluationForDual`(a).

1.12.19 AddEvaluationForDualWithGivenTensorProduct (for IsCapCategory, IsFunction)

▷ `AddEvaluationForDualWithGivenTensorProduct(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `EvaluationForDualWithGivenTensorProduct`. $F : (s,a,r) \mapsto$ `EvaluationForDualWithGivenTensorProduct`(s,a,r).

1.12.20 AddEvaluationMorphism (for IsCapCategory, IsFunction)

▷ `AddEvaluationMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `EvaluationMorphism`. $F : (a,b) \mapsto$ `EvaluationMorphism`(a,b).

1.12.21 AddEvaluationMorphismWithGivenSource (for IsCapCategory, IsFunction)

▷ `AddEvaluationMorphismWithGivenSource(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `EvaluationMorphismWithGivenSource`. $F : (a,b,s) \mapsto$ `EvaluationMorphismWithGivenSource`(a,b,s).

1.12.22 AddInternalHomOnMorphisms (for IsCapCategory, IsFunction)

▷ `AddInternalHomOnMorphisms(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalHomOnMorphisms`. $F : (alpha,beta) \mapsto$ `InternalHomOnMorphisms`$(alpha,beta)$.

1.12.23 AddInternalHomOnMorphismsWithGivenInternalHoms (for IsCapCategory, IsFunction)

▷ `AddInternalHomOnMorphismsWithGivenInternalHoms(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalHomOnMorphismsWithGivenInternalHoms`. $F : (s,alpha,beta,r) \mapsto$ `InternalHomOnMorphismsWithGivenInternalHoms`$(s,alpha,beta,r)$.

1.12.24 AddInternalHomOnObjects (for IsCapCategory, IsFunction)

▷ `AddInternalHomOnObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalHomOnObjects`. $F : (a,b) \mapsto$ `InternalHomOnObjects`(a,b).

1.12.25 AddInternalHomToTensorProductAdjunctionMap (for IsCapCategory, IsFunction)

▷ `AddInternalHomToTensorProductAdjunctionMap(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalHomToTensorProductAdjunctionMap`. $F : (b,c,g) \mapsto$ `InternalHomToTensorProductAdjunctionMap`(b,c,g).

1.12.26 AddIsomorphismFromDualObjectToInternalHomIntoTensorUnit (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromDualObjectToInternalHomIntoTensorUnit(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromDualObjectToInternalHomIntoTensorUnit`. $F : (a) \mapsto$ `IsomorphismFromDualObjectToInternalHomIntoTensorUnit`(a).

1.12.27 AddIsomorphismFromInternalHomIntoTensorUnitToDualObject (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromInternalHomIntoTensorUnitToDualObject(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromInternalHomIntoTensorUnitToDualObject`. $F : (a) \mapsto$ `IsomorphismFromInternalHomIntoTensorUnitToDualObject`(a).

1.12.28 AddIsomorphismFromInternalHomToObject (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromInternalHomToObject(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromInternalHomToObject`. $F : (a) \mapsto$ `IsomorphismFromInternalHomToObject`(a).

1.12.29 AddIsomorphismFromInternalHomToObjectWithGivenInternalHom (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromInternalHomToObjectWithGivenInternalHom(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromInternalHomToObjectWithGivenInternalHom`. $F : (a,s) \mapsto$ `IsomorphismFromInternalHomToObjectWithGivenInternalHom`(a,s).

1.12.30 AddIsomorphismFromObjectToInternalHom (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromObjectToInternalHom(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromObjectToInternalHom`. $F : (a) \mapsto$ `IsomorphismFromObjectToInternalHom`(a).

1.12.31 AddIsomorphismFromObjectToInternalHomWithGivenInternalHom (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromObjectToInternalHomWithGivenInternalHom(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromObjectToInternalHomWithGivenInternalHom`. $F : (a,r) \mapsto$ `IsomorphismFromObjectToInternalHomWithGivenInternalHom`(a,r).

1.12.32 AddLambdaElimination (for IsCapCategory, IsFunction)

▷ `AddLambdaElimination(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `LambdaElimination`. $F : (a,b,alpha) \mapsto$ `LambdaElimination`$(a,b,alpha)$.

1.12.33 AddLambdaIntroduction (for IsCapCategory, IsFunction)

▷ `AddLambdaIntroduction(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `LambdaIntroduction`. $F : (alpha) \mapsto$ `LambdaIntroduction`$(alpha)$.

1.12.34 AddMonoidalPostComposeMorphism (for IsCapCategory, IsFunction)

▷ `AddMonoidalPostComposeMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MonoidalPostComposeMorphism`. $F : (a,b,c) \mapsto$ `MonoidalPostComposeMorphism`(a,b,c).

1.12.35 AddMonoidalPostComposeMorphismWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddMonoidalPostComposeMorphismWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MonoidalPostComposeMorphismWithGivenObjects`. $F : (s,a,b,c,r) \mapsto$ `MonoidalPostComposeMorphismWithGivenObjects`(s,a,b,c,r).

1.12.36 AddMonoidalPreComposeMorphism (for IsCapCategory, IsFunction)

▷ `AddMonoidalPreComposeMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MonoidalPreComposeMorphism`. $F : (a,b,c) \mapsto$ `MonoidalPreComposeMorphism`(a,b,c).

1.12.37 AddMonoidalPreComposeMorphismWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddMonoidalPreComposeMorphismWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MonoidalPreComposeMorphismWithGivenObjects`. $F : (s,a,b,c,r) \mapsto$ `MonoidalPreComposeMorphismWithGivenObjects`(s,a,b,c,r).

1.12.38 AddMorphismFromTensorProductToInternalHom (for IsCapCategory, IsFunction)

▷ `AddMorphismFromTensorProductToInternalHom(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismFromTensorProductToInternalHom`. $F : (a,b) \mapsto$ `MorphismFromTensorProductToInternalHom`(a,b).

1.12.39 AddMorphismFromTensorProductToInternalHomWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddMorphismFromTensorProductToInternalHomWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismFromTensorProductToInternalHomWithGivenObjects`. $F : (s,a,b,r) \mapsto$ `MorphismFromTensorProductToInternalHomWithGivenObjects`(s,a,b,r).

1.12.40 AddMorphismToBidual (for IsCapCategory, IsFunction)

▷ `AddMorphismToBidual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismToBidual`. $F : (a) \mapsto$ `MorphismToBidual`(a).

1.12.41 AddMorphismToBidualWithGivenBidual (for IsCapCategory, IsFunction)

▷ `AddMorphismToBidualWithGivenBidual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismToBidualWithGivenBidual`. $F : (a,r) \mapsto$ `MorphismToBidualWithGivenBidual`(a,r).

1.12.42 AddTensorProductDualityCompatibilityMorphism (for IsCapCategory, IsFunction)

▷ `AddTensorProductDualityCompatibilityMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorProductDualityCompatibilityMorphism`. $F : (a,b) \mapsto$ `TensorProductDualityCompatibilityMorphism`(a,b).

1.12.43 AddTensorProductDualityCompatibilityMorphismWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddTensorProductDualityCompatibilityMorphismWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorProductDualityCompatibilityMorphismWithGivenObjects`. $F : (s,a,b,r) \mapsto$ `TensorProductDualityCompatibilityMorphismWithGivenObjects`(s,a,b,r).

1.12.44 AddTensorProductInternalHomCompatibilityMorphism (for IsCapCategory, IsFunction)

▷ `AddTensorProductInternalHomCompatibilityMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorProductInternalHomCompatibilityMorphism`. $F : (list) \mapsto$ `TensorProductInternalHomCompatibilityMorphism`$(list)$.

1.12.45 AddTensorProductInternalHomCompatibilityMorphismWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddTensorProductInternalHomCompatibilityMorphismWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorProductInternalHomCompatibilityMorphismWithGivenObjects`. $F : (source, list, range) \mapsto$ `TensorProductInternalHomCompatibilityMorphismWithGivenObjects`$(source, list, ran$

1.12.46 AddTensorProductToInternalHomAdjunctionMap (for IsCapCategory, IsFunction)

▷ `AddTensorProductToInternalHomAdjunctionMap(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorProductToInternalHomAdjunctionMap`. $F : (a,b,f) \mapsto$ `TensorProductToInternalHomAdjunctionMap`(a,b,f).

1.12.47 AddUniversalPropertyOfDual (for IsCapCategory, IsFunction)

▷ `AddUniversalPropertyOfDual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `UniversalPropertyOfDual`. $F : (t, a, alpha) \mapsto$ `UniversalPropertyOfDual`$(t, a, alpha)$.

1.12.48 AddCoDualOnMorphisms (for IsCapCategory, IsFunction)

▷ `AddCoDualOnMorphisms(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoDualOnMorphisms`. $F : (alpha) \mapsto$ `CoDualOnMorphisms`$(alpha)$.

1.12.49 AddCoDualOnMorphismsWithGivenCoDuals (for IsCapCategory, IsFunction)

▷ `AddCoDualOnMorphismsWithGivenCoDuals(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoDualOnMorphismsWithGivenCoDuals`. $F : (s, alpha, r) \mapsto$ `CoDualOnMorphismsWithGivenCoDuals`$(s, alpha, r)$.

1.12.50 AddCoDualOnObjects (for IsCapCategory, IsFunction)

▷ `AddCoDualOnObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoDualOnObjects`. $F : (a) \mapsto$ `CoDualOnObjects`(a).

1.12.51 AddCoDualityTensorProductCompatibilityMorphism (for IsCapCategory, IsFunction)

▷ `AddCoDualityTensorProductCompatibilityMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoDualityTensorProductCompatibilityMorphism`. $F : (a, b) \mapsto$ `CoDualityTensorProductCompatibilityMorphism`(a, b).

1.12.52 AddCoDualityTensorProductCompatibilityMorphismWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddCoDualityTensorProductCompatibilityMorphismWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation

`CoDualityTensorProductCompatibilityMorphismWithGivenObjects`. $F : (s,a,b,r) \mapsto$ `CoDualityTensorProductCompatibilityMorphismWithGivenObjects`(s,a,b,r).

1.12.53 AddCoLambdaElimination (for IsCapCategory, IsFunction)

▷ `AddCoLambdaElimination(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoLambdaElimination`. $F : (a,b,alpha) \mapsto$ `CoLambdaElimination`$(a,b,alpha)$.

1.12.54 AddCoLambdaIntroduction (for IsCapCategory, IsFunction)

▷ `AddCoLambdaIntroduction(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoLambdaIntroduction`. $F : (alpha) \mapsto$ `CoLambdaIntroduction`$(alpha)$.

1.12.55 AddCoclosedCoevaluationMorphism (for IsCapCategory, IsFunction)

▷ `AddCoclosedCoevaluationMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoclosedCoevaluationMorphism`. $F : (a,b) \mapsto$ `CoclosedCoevaluationMorphism`(a,b).

1.12.56 AddCoclosedCoevaluationMorphismWithGivenSource (for IsCapCategory, IsFunction)

▷ `AddCoclosedCoevaluationMorphismWithGivenSource(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoclosedCoevaluationMorphismWithGivenSource`. $F : (a,b,s) \mapsto$ `CoclosedCoevaluationMorphismWithGivenSource`(a,b,s).

1.12.57 AddCoclosedEvaluationForCoDual (for IsCapCategory, IsFunction)

▷ `AddCoclosedEvaluationForCoDual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoclosedEvaluationForCoDual`. $F : (a) \mapsto$ `CoclosedEvaluationForCoDual`(a).

1.12.58 AddCoclosedEvaluationForCoDualWithGivenTensorProduct (for IsCapCategory, IsFunction)

▷ `AddCoclosedEvaluationForCoDualWithGivenTensorProduct(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoclosedEvaluationForCoDualWithGivenTensorProduct`. $F : (s,a,r) \mapsto$ `CoclosedEvaluationForCoDualWithGivenTensorProduct`(s,a,r).

1.12.59 AddCoclosedEvaluationMorphism (for IsCapCategory, IsFunction)

▷ `AddCoclosedEvaluationMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoclosedEvaluationMorphism`. $F : (a,b) \mapsto$ `CoclosedEvaluationMorphism`(a,b).

1.12.60 AddCoclosedEvaluationMorphismWithGivenRange (for IsCapCategory, IsFunction)

▷ `AddCoclosedEvaluationMorphismWithGivenRange(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoclosedEvaluationMorphismWithGivenRange`. $F : (a,b,r) \mapsto$ `CoclosedEvaluationMorphismWithGivenRange`(a,b,r).

1.12.61 AddInternalCoHomOnMorphisms (for IsCapCategory, IsFunction)

▷ `AddInternalCoHomOnMorphisms(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalCoHomOnMorphisms`. $F : (alpha,beta) \mapsto$ `InternalCoHomOnMorphisms`$(alpha,beta)$.

1.12.62 AddInternalCoHomOnMorphismsWithGivenInternalCoHoms (for IsCapCategory, IsFunction)

▷ `AddInternalCoHomOnMorphismsWithGivenInternalCoHoms(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalCoHomOnMorphismsWithGivenInternalCoHoms`. $F : (s,alpha,beta,r) \mapsto$ `InternalCoHomOnMorphismsWithGivenInternalCoHoms`$(s,alpha,beta,r)$.

1.12.63 AddInternalCoHomOnObjects (for IsCapCategory, IsFunction)

▷ `AddInternalCoHomOnObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalCoHomOnObjects`. $F : (a,b) \mapsto$ `InternalCoHomOnObjects`(a,b).

1.12.64 AddInternalCoHomTensorProductCompatibilityMorphism (for IsCapCategory, IsFunction)

▷ `AddInternalCoHomTensorProductCompatibilityMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalCoHomTensorProductCompatibilityMorphism`. $F : (list) \mapsto$ `InternalCoHomTensorProductCompatibilityMorphism`$(list)$.

1.12.65 AddInternalCoHomTensorProductCompatibilityMorphismWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddInternalCoHomTensorProductCompatibilityMorphismWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalCoHomTensorProductCompatibilityMorphismWithGivenObjects`. $F : (source, list, range) \mapsto$ `InternalCoHomTensorProductCompatibilityMorphismWithGivenObjects`$(source, list, r$

1.12.66 AddInternalCoHomToTensorProductAdjunctionMap (for IsCapCategory, IsFunction)

▷ `AddInternalCoHomToTensorProductAdjunctionMap(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalCoHomToTensorProductAdjunctionMap`. $F : (a,b,f) \mapsto$ `InternalCoHomToTensorProductAdjunctionMap`(a,b,f).

1.12.67 AddIsomorphismFromCoDualObjectToInternalCoHomFromTensorUnit (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromCoDualObjectToInternalCoHomFromTensorUnit(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromCoDualObjectToInternalCoHomFromTensorUnit`. $F : (a) \mapsto$ `IsomorphismFromCoDualObjectToInternalCoHomFromTensorUnit`(a).

1.12.68 AddIsomorphismFromInternalCoHomFromTensorUnitToCoDualObject (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromInternalCoHomFromTensorUnitToCoDualObject(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromInternalCoHomFromTensorUnitToCoDualObject`. $F : (a) \mapsto$ `IsomorphismFromInternalCoHomFromTensorUnitToCoDualObject`(a).

1.12.69 AddIsomorphismFromInternalCoHomToObject (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromInternalCoHomToObject(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromInternalCoHomToObject`. $F : (a) \mapsto$ `IsomorphismFromInternalCoHomToObject`(a).

1.12.70 AddIsomorphismFromInternalCoHomToObjectWithGivenInternalCoHom (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromInternalCoHomToObjectWithGivenInternalCoHom(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromInternalCoHomToObjectWithGivenInternalCoHom`. $F : (a,s) \mapsto$ `IsomorphismFromInternalCoHomToObjectWithGivenInternalCoHom`(a,s).

1.12.71 AddIsomorphismFromObjectToInternalCoHom (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromObjectToInternalCoHom(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromObjectToInternalCoHom`. $F : (a) \mapsto$ `IsomorphismFromObjectToInternalCoHom`(a).

1.12.72 AddIsomorphismFromObjectToInternalCoHomWithGivenInternalCoHom (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromObjectToInternalCoHomWithGivenInternalCoHom(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromObjectToInternalCoHomWithGivenInternalCoHom`. $F : (a,r) \mapsto$ `IsomorphismFromObjectToInternalCoHomWithGivenInternalCoHom`(a,r).

1.12.73 AddMonoidalPostCoComposeMorphism (for IsCapCategory, IsFunction)

▷ `AddMonoidalPostCoComposeMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MonoidalPostCoComposeMorphism`. $F : (a,b,c) \mapsto$ `MonoidalPostCoComposeMorphism`(a,b,c).

1.12.74 AddMonoidalPostCoComposeMorphismWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddMonoidalPostCoComposeMorphismWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MonoidalPostCoComposeMorphismWithGivenObjects`. $F : (s,a,b,c,r) \mapsto$ `MonoidalPostCoComposeMorphismWithGivenObjects`(s,a,b,c,r).

1.12.75 AddMonoidalPreCoComposeMorphism (for IsCapCategory, IsFunction)

▷ `AddMonoidalPreCoComposeMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MonoidalPreCoComposeMorphism`. $F : (a,b,c) \mapsto$ `MonoidalPreCoComposeMorphism`(a,b,c).

1.12.76 AddMonoidalPreCoComposeMorphismWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddMonoidalPreCoComposeMorphismWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MonoidalPreCoComposeMorphismWithGivenObjects`. $F : (s,a,b,c,r) \mapsto$ `MonoidalPreCoComposeMorphismWithGivenObjects`(s,a,b,c,r).

1.12.77 AddMorphismFromCoBidual (for IsCapCategory, IsFunction)

▷ `AddMorphismFromCoBidual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismFromCoBidual`. $F : (a) \mapsto$ `MorphismFromCoBidual`(a).

1.12.78 AddMorphismFromCoBidualWithGivenCoBidual (for IsCapCategory, IsFunction)

▷ `AddMorphismFromCoBidualWithGivenCoBidual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismFromCoBidualWithGivenCoBidual`. $F : (a,s) \mapsto$ `MorphismFromCoBidualWithGivenCoBidual`(a,s).

1.12.79 AddMorphismFromInternalCoHomToTensorProduct (for IsCapCategory, IsFunction)

▷ `AddMorphismFromInternalCoHomToTensorProduct(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismFromInternalCoHomToTensorProduct`. $F : (a,b) \mapsto$ `MorphismFromInternalCoHomToTensorProduct`(a,b).

1.12.80 AddMorphismFromInternalCoHomToTensorProductWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddMorphismFromInternalCoHomToTensorProductWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismFromInternalCoHomToTensorProductWithGivenObjects`. $F : (s,a,b,r) \mapsto$ `MorphismFromInternalCoHomToTensorProductWithGivenObjects`(s,a,b,r).

1.12.81 AddTensorProductToInternalCoHomAdjunctionMap (for IsCapCategory, IsFunction)

▷ `AddTensorProductToInternalCoHomAdjunctionMap(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorProductToInternalCoHomAdjunctionMap`. $F : (c,b,g) \mapsto$ `TensorProductToInternalCoHomAdjunctionMap`(c,b,g).

1.12.82 AddUniversalPropertyOfCoDual (for IsCapCategory, IsFunction)

▷ `AddUniversalPropertyOfCoDual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `UniversalPropertyOfCoDual`. $F : (t,a,alpha) \mapsto$ `UniversalPropertyOfCoDual`$(t,a,alpha)$.

1.12.83 AddAssociatorLeftToRight (for IsCapCategory, IsFunction)

▷ `AddAssociatorLeftToRight(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `AssociatorLeftToRight`. $F : (a,b,c) \mapsto$ `AssociatorLeftToRight`(a,b,c).

1.12.84 AddAssociatorLeftToRightWithGivenTensorProducts (for IsCapCategory, IsFunction)

▷ `AddAssociatorLeftToRightWithGivenTensorProducts(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `AssociatorLeftToRightWithGivenTensorProducts`. $F : (s,a,b,c,r) \mapsto$ `AssociatorLeftToRightWithGivenTensorProducts`(s,a,b,c,r).

1.12.85 AddAssociatorRightToLeft (for IsCapCategory, IsFunction)

▷ `AddAssociatorRightToLeft(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `AssociatorRightToLeft`. $F : (a,b,c) \mapsto$ `AssociatorRightToLeft`(a,b,c).

1.12.86 AddAssociatorRightToLeftWithGivenTensorProducts (for IsCapCategory, IsFunction)

▷ `AddAssociatorRightToLeftWithGivenTensorProducts(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `AssociatorRightToLeftWithGivenTensorProducts`. $F : (s,a,b,c,r) \mapsto$ `AssociatorRightToLeftWithGivenTensorProducts`(s,a,b,c,r).

1.12.87 AddLeftUnitor (for IsCapCategory, IsFunction)

▷ `AddLeftUnitor(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `LeftUnitor`. $F : (a) \mapsto$ `LeftUnitor`(a).

1.12.88 AddLeftUnitorInverse (for IsCapCategory, IsFunction)

▷ `AddLeftUnitorInverse(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `LeftUnitorInverse`. $F : (a) \mapsto$ `LeftUnitorInverse`(a).

1.12.89 AddLeftUnitorInverseWithGivenTensorProduct (for IsCapCategory, IsFunction)

▷ `AddLeftUnitorInverseWithGivenTensorProduct(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `LeftUnitorInverseWithGivenTensorProduct`. $F : (a,r) \mapsto$ `LeftUnitorInverseWithGivenTensorProduct`(a,r).

1.12.90 AddLeftUnitorWithGivenTensorProduct (for IsCapCategory, IsFunction)

▷ `AddLeftUnitorWithGivenTensorProduct(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `LeftUnitorWithGivenTensorProduct`. $F : (a,s) \mapsto$ `LeftUnitorWithGivenTensorProduct`(a,s).

1.12.91 AddRightUnitor (for IsCapCategory, IsFunction)

▷ `AddRightUnitor(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `RightUnitor`. $F : (a) \mapsto$ `RightUnitor`(a).

1.12.92 AddRightUnitorInverse (for IsCapCategory, IsFunction)

▷ `AddRightUnitorInverse(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `RightUnitorInverse`. $F : (a) \mapsto$ `RightUnitorInverse`(a).

1.12.93 AddRightUnitorInverseWithGivenTensorProduct (for IsCapCategory, IsFunction)

▷ `AddRightUnitorInverseWithGivenTensorProduct(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `RightUnitorInverseWithGivenTensorProduct`. $F : (a,r) \mapsto$ `RightUnitorInverseWithGivenTensorProduct`(a,r).

1.12.94 AddRightUnitorWithGivenTensorProduct (for IsCapCategory, IsFunction)

▷ `AddRightUnitorWithGivenTensorProduct(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `RightUnitorWithGivenTensorProduct`. $F : (a,s) \mapsto$ `RightUnitorWithGivenTensorProduct`(a,s).

1.12.95 AddTensorProductOnMorphisms (for IsCapCategory, IsFunction)

▷ `AddTensorProductOnMorphisms(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorProductOnMorphisms`. $F : (alpha,beta) \mapsto$ `TensorProductOnMorphisms`$(alpha,beta)$.

1.12.96 AddTensorProductOnMorphismsWithGivenTensorProducts (for IsCapCategory, IsFunction)

▷ `AddTensorProductOnMorphismsWithGivenTensorProducts(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorProductOnMorphismsWithGivenTensorProducts`. $F : (s, alpha, beta, r) \mapsto$ `TensorProductOnMorphismsWithGivenTensorProducts`$(s, alpha, beta, r)$.

1.12.97 AddCoevaluationForDual (for IsCapCategory, IsFunction)

▷ `AddCoevaluationForDual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoevaluationForDual`. $F : (a) \mapsto$ `CoevaluationForDual`(a).

1.12.98 AddCoevaluationForDualWithGivenTensorProduct (for IsCapCategory, IsFunction)

▷ `AddCoevaluationForDualWithGivenTensorProduct(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoevaluationForDualWithGivenTensorProduct`. $F : (s, a, r) \mapsto$ `CoevaluationForDualWithGivenTensorProduct`(s, a, r).

1.12.99 AddIsomorphismFromInternalHomToTensorProductWithDualObject (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromInternalHomToTensorProductWithDualObject(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromInternalHomToTensorProductWithDualObject`. $F : (a, b) \mapsto$ `IsomorphismFromInternalHomToTensorProductWithDualObject`(a, b).

1.12.100 AddIsomorphismFromTensorProductWithDualObjectToInternalHom (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromTensorProductWithDualObjectToInternalHom(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromTensorProductWithDualObjectToInternalHom`. $F : (a, b) \mapsto$ `IsomorphismFromTensorProductWithDualObjectToInternalHom`(a, b).

1.12.101 AddMorphismFromBidual (for IsCapCategory, IsFunction)

▷ `AddMorphismFromBidual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismFromBidual`. $F : (a) \mapsto$ `MorphismFromBidual`(a).

1.12.102 AddMorphismFromBidualWithGivenBidual (for IsCapCategory, IsFunction)

▷ `AddMorphismFromBidualWithGivenBidual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismFromBidualWithGivenBidual`. $F : (a,s) \mapsto$ `MorphismFromBidualWithGivenBidual`(a,s).

1.12.103 AddMorphismFromInternalHomToTensorProduct (for IsCapCategory, IsFunction)

▷ `AddMorphismFromInternalHomToTensorProduct(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismFromInternalHomToTensorProduct`. $F : (a,b) \mapsto$ `MorphismFromInternalHomToTensorProduct`(a,b).

1.12.104 AddMorphismFromInternalHomToTensorProductWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddMorphismFromInternalHomToTensorProductWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `MorphismFromInternalHomToTensorProductWithGivenObjects`. $F : (s,a,b,r) \mapsto$ `MorphismFromInternalHomToTensorProductWithGivenObjects`(s,a,b,r).

1.12.105 AddRankMorphism (for IsCapCategory, IsFunction)

▷ `AddRankMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `RankMorphism`. $F : (a) \mapsto$ `RankMorphism`(a).

1.12.106 AddTensorProductInternalHomCompatibilityMorphismInverse (for IsCapCategory, IsFunction)

▷ `AddTensorProductInternalHomCompatibilityMorphismInverse(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorProductInternalHomCompatibilityMorphismInverse`. $F : (list) \mapsto$ `TensorProductInternalHomCompatibilityMorphismInverse`$(list)$.

1.12.107 AddTensorProductInternalHomCompatibilityMorphismInverseWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddTensorProductInternalHomCompatibilityMorphismInverseWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TensorProductInternalHomCompatibilityMorphismInverseWithGivenObjects`. $F : (source, list, range) \mapsto$ `TensorProductInternalHomCompatibilityMorphismInverseWithGivenObjects`$(sourc$

1.12.108 AddTraceMap (for IsCapCategory, IsFunction)

▷ `AddTraceMap(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `TraceMap`. $F : (alpha) \mapsto$ `TraceMap`$(alpha)$.

1.12.109 AddCoRankMorphism (for IsCapCategory, IsFunction)

▷ `AddCoRankMorphism(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoRankMorphism`. $F : (a) \mapsto$ `CoRankMorphism`(a).

1.12.110 AddCoTraceMap (for IsCapCategory, IsFunction)

▷ `AddCoTraceMap(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoTraceMap`. $F : (alpha) \mapsto$ `CoTraceMap`$(alpha)$.

1.12.111 AddCoclosedCoevaluationForCoDual (for IsCapCategory, IsFunction)

▷ `AddCoclosedCoevaluationForCoDual(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoclosedCoevaluationForCoDual`. $F : (a) \mapsto$ `CoclosedCoevaluationForCoDual`(a).

1.12.112 AddCoclosedCoevaluationForCoDualWithGivenTensorProduct (for IsCapCategory, IsFunction)

▷ `AddCoclosedCoevaluationForCoDualWithGivenTensorProduct(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `CoclosedCoevaluationForCoDualWithGivenTensorProduct`. $F : (s,a,r) \mapsto$ `CoclosedCoevaluationForCoDualWithGivenTensorProduct`(s,a,r).

1.12.113 AddInternalCoHomTensorProductCompatibilityMorphismInverse (for IsCapCategory, IsFunction)

▷ `AddInternalCoHomTensorProductCompatibilityMorphismInverse(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalCoHomTensorProductCompatibilityMorphismInverse`. $F : (list) \mapsto$ `InternalCoHomTensorProductCompatibilityMorphismInverse`$(list)$.

1.12.114 AddInternalCoHomTensorProductCompatibilityMorphismInverseWithGivenObjects (for IsCapCategory, IsFunction)

▷ `AddInternalCoHomTensorProductCompatibilityMorphismInverseWithGivenObjects(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `InternalCoHomTensorProductCompatibilityMorphismInverseWithGivenObjects`. $F : (source, list, range) \mapsto$ `InternalCoHomTensorProductCompatibilityMorphismInverseWithGivenObjects`$(sou$

1.12.115 AddIsomorphismFromInternalCoHomToTensorProductWithCoDualObject (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromInternalCoHomToTensorProductWithCoDualObject(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation `IsomorphismFromInternalCoHomToTensorProductWithCoDualObject`. $F : (a,b) \mapsto$ `IsomorphismFromInternalCoHomToTensorProductWithCoDualObject`(a,b).

1.12.116 AddIsomorphismFromTensorProductWithCoDualObjectToInternalCoHom (for IsCapCategory, IsFunction)

▷ `AddIsomorphismFromTensorProductWithCoDualObjectToInternalCoHom(C, F)` (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation

IsomorphismFromTensorProductWithCoDualObjectToInternalCoHom. $F : (a,b) \mapsto$ IsomorphismFromTensorProductWithCoDualObjectToInternalCoHom(a,b).

1.12.117 AddMorphismFromTensorProductToInternalCoHom (for IsCapCategory, IsFunction)

▷ AddMorphismFromTensorProductToInternalCoHom(C, F) (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation MorphismFromTensorProductToInternalCoHom. $F : (a,b) \mapsto$ MorphismFromTensorProductToInternalCoHom(a,b).

1.12.118 AddMorphismFromTensorProductToInternalCoHomWithGivenObjects (for IsCapCategory, IsFunction)

▷ AddMorphismFromTensorProductToInternalCoHomWithGivenObjects(C, F) (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation MorphismFromTensorProductToInternalCoHomWithGivenObjects. $F : (s,a,b,r) \mapsto$ MorphismFromTensorProductToInternalCoHomWithGivenObjects(s,a,b,r).

1.12.119 AddMorphismToCoBidual (for IsCapCategory, IsFunction)

▷ AddMorphismToCoBidual(C, F) (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation MorphismToCoBidual. $F : (a) \mapsto$ MorphismToCoBidual(a).

1.12.120 AddMorphismToCoBidualWithGivenCoBidual (for IsCapCategory, IsFunction)

▷ AddMorphismToCoBidualWithGivenCoBidual(C, F) (operation)

Returns: nothing

The arguments are a category C and a function F. This operation adds the given function F to the category for the basic operation MorphismToCoBidualWithGivenCoBidual. $F : (a,r) \mapsto$ MorphismToCoBidualWithGivenCoBidual(a,r).

Chapter 2

Examples and Tests

2.1 Test functions

2.1.1 AdditiveMonoidalCategoriesTest

▷ `AdditiveMonoidalCategoriesTest(cat, a, L)` (function)

The arguments are

- a CAP category *cat*
- an object *a*
- a list *L* of objects

This function checks for every operation declared in AdditiveMonoidalCategories.gd if it is computable in the CAP category *cat*. If yes, then the operation is executed with the parameters given above and compared to the equivalent computation in the opposite category of *cat*. Pass the options

- `verbose := true` to output more information.
- `only_primitive_operations := true`, which is passed on to Opposite(), to only primitively install dual operations for primitively installed operations in *cat*. The advantage is, that more derivations might be tested. On the downside, this might test fewer dual_pre/postprocessor_funcs.

2.1.2 BraidedMonoidalCategoriesTest

▷ `BraidedMonoidalCategoriesTest(cat, a, b)` (function)

The arguments are

- a CAP category *cat*
- objects a, b

This function checks for every operation declared in BraidedMonoidalCategories.gd if it is computable in the CAP category *cat*. If yes, then the operation is executed with the parameters given above and compared to the equivalent computation in the opposite category of *cat*. Pass the options

- `verbose := true` to output more information.
- `only_primitive_operations := true`, which is passed on to Opposite(), to only primitively install dual operations for primitively installed operations in *cat*. The advantage is, that more derivations might be tested. On the downside, this might test fewer dual_pre/postprocessor_funcs.

2.1.3 ClosedMonoidalCategoriesTest

▷ `ClosedMonoidalCategoriesTest(`*cat, a, b, c, d, alpha, beta, gamma, delta, epsilon, zeta*`)` (function)

The arguments are

- a CAP category *cat*
- objects a,b,c,d
- a morphism $\alpha : a \to b$
- a morphism $\beta : c \to d$
- a morphism $\gamma : a \otimes b \to 1$
- a morphism $\delta : c \otimes d \to 1$
- a morphism $\varepsilon : 1 \to \mathrm{Hom}(a,b)$
- a morphism $\zeta : 1 \to \mathrm{Hom}(c,d)$

This function checks for every operation declared in ClosedMonoidalCategories.gd if it is computable in the CAP category *cat*. If yes, then the operation is executed with the parameters given above and compared to the equivalent computation in the opposite category of *cat*. Pass the options

- `verbose := true` to output more information.
- `only_primitive_operations := true`, which is passed on to Opposite(), to only primitively install dual operations for primitively installed operations in *cat*. The advantage is, that more derivations might be tested. On the downside, this might test fewer dual_pre/postprocessor_funcs.

2.1.4 CoclosedMonoidalCategoriesTest

▷ `CoclosedMonoidalCategoriesTest(`*cat, a, b, c, d, alpha, beta, gamma, delta, epsilon, zeta*`)` (function)

The arguments are a CAP category *cat* objects a,b,c,d

- a morphism $\alpha : a \to b$
- a morphism $\beta : c \to d$
- a morphism $\gamma : 1 \to a \otimes b$

- a morphism $\delta : 1 \to c \otimes d$
- a morphism $\varepsilon : \mathrm{coHom}(a,b) \to 1$
- a morphism $\zeta : \mathrm{coHom}(c,d) \to 1$

This function checks for every operation declared in CoclosedMonoidalCategories.gd if it is computable in the CAP category *cat*. If yes, then the operation is executed with the parameters given above and compared to the equivalent computation in the opposite category of *cat*. Pass the options

- `verbose := true` to output more information.
- `only_primitive_operations := true`, which is passed on to Opposite(), to only primitively install dual operations for primitively installed operations in *cat*. The advantage is, that more derivations might be tested. On the downside, this might test fewer dual_pre/postprocessor_funcs.

2.1.5 MonoidalCategoriesTensorProductAndUnitTest

▷ `MonoidalCategoriesTensorProductAndUnitTest(cat, a, b)` (function)

The arguments are

- a CAP category *cat*
- objects a,b

This function checks for every operation declared in MonoidalCategoriesTensorProductAndUnit.gd if it is computable in the CAP category *cat*. If yes, then the operation is executed with the parameters given above and compared to the equivalent computation in the opposite category of *cat*. Pass the options

- `verbose := true` to output more information.
- `only_primitive_operations := true`, which is passed on to Opposite(), to only primitively install dual operations for primitively installed operations in *cat*. The advantage is, that more derivations might be tested. On the downside, this might test fewer dual_pre/postprocessor_funcs.

2.1.6 MonoidalCategoriesTest

▷ `MonoidalCategoriesTest(cat, a, b, c, alpha, beta)` (function)

The arguments are

- a CAP category *cat*
- objects a,b,c
- a morphism $\alpha : a \to b$
- a morphism $\beta : c \to d$

This function checks for every operation declared in MonoidalCategories.gd if it is computable in the CAP category *cat*. If yes, then the operation is executed with the parameters given above and compared to the equivalent computation in the opposite category of *cat*. Pass the options

- `verbose := true` to output more information.
- `only_primitive_operations := true`, which is passed on to Opposite(), to only primitively install dual operations for primitively installed operations in *cat*. The advantage is, that more derivations might be tested. On the downside, this might test fewer dual_pre/postprocessor_funcs.

2.1.7 RigidSymmetricClosedMonoidalCategoriesTest

▷ `RigidSymmetricClosedMonoidalCategoriesTest(cat, a, b, c, d, alpha)` (function)

The arguments are

- a CAP category *cat*
- objects a, b, c, d
- an endomorphism $\alpha : a \to a$

This function checks for every object and morphism declared in RigidSymmetricClosedMonoidalCategories.gd if it is computable in the CAP category *cat*. If yes, then the operation is executed with the parameters given above and compared to the equivalent computation in the opposite category of *cat*. Pass the options

- `verbose := true` to output more information.
- `only_primitive_operations := true`, which is passed on to Opposite(), to only primitively install dual operations for primitively installed operations in *cat*. The advantage is, that more derivations might be tested. On the downside, this might test fewer dual_pre/postprocessor_funcs.

2.1.8 RigidSymmetricCoclosedMonoidalCategoriesTest

▷ `RigidSymmetricCoclosedMonoidalCategoriesTest(cat, a, b, c, d, alpha)` (function)

The arguments are

- a CAP category *cat*
- objects a, b, c, d
- an endomorphism $\alpha : a \to a$

This function checks for every object and morphism declared in RigidSymmetricCoclosedMonoidalCategories.gd if it is computable in the CAP category *cat*. If yes, then the operation is executed with the parameters given above and compared to the equivalent computation in the opposite category of *cat*. Pass the options

- `verbose := true` to output more information.
- `only_primitive_operations := true`, which is passed on to Opposite(), to only primitively install dual operations for primitively installed operations in *cat*. The advantage is, that more derivations might be tested. On the downside, this might test fewer dual_pre/postprocessor_funcs.

2.2 Basics

Example

```
gap> LoadPackage( "MonoidalCategories" );
true
gap> vecspaces := CreateCapCategory( "VectorSpaces" );
VectorSpaces
gap> ReadPackage( "MonoidalCategories",
>          "examples/VectorSpacesMonoidalCategory.gi" );
true
gap> z := ZeroObject( vecspaces );
<A rational vector space of dimension 0>
gap> a := QVectorSpace( 1 );
<A rational vector space of dimension 1>
gap> b := QVectorSpace( 2 );
<A rational vector space of dimension 2>
gap> c := QVectorSpace( 3 );
<A rational vector space of dimension 3>
gap> alpha := VectorSpaceMorphism( a, [ [ 1, 0 ] ], b );
A rational vector space homomorphism with matrix:
[ [  1,  0 ] ]
gap> beta := VectorSpaceMorphism( b,
>                  [ [ 1, 0, 0 ], [ 0, 1, 0 ] ], c );
A rational vector space homomorphism with matrix:
[ [  1,  0,  0 ],
  [  0,  1,  0 ] ]
gap> gamma := VectorSpaceMorphism( c,
>                   [ [ 0, 1, 1 ], [ 1, 0, 1 ], [ 1, 1, 0 ] ], c );
A rational vector space homomorphism with matrix:
[ [  0,  1,  1 ],
  [  1,  0,  1 ],
  [  1,  1,  0 ] ]
gap> IsCongruentForMorphisms(
>          TensorProductOnMorphisms( alpha, beta ),
>          TensorProductOnMorphisms( beta, alpha ) );
false
gap> IsOne( AssociatorRightToLeft( a, b, c ) );
true
gap> IsCongruentForMorphisms(
>          gamma, LambdaElimination( c, c, LambdaIntroduction( gamma ) ) );
true
gap> IsZero( TraceMap( gamma ) );
true
gap> IsCongruentForMorphisms(
>          RankMorphism( DirectSum( a, b ) ), RankMorphism( c ) );
```

```
true
gap> IsOne( Braiding( b, c ) );
false
gap> IsOne( PreCompose( Braiding( b, c ), Braiding( c, b ) ) );
true
```

Chapter 3

Code Generation for Monodial Categories

3.1 Monoidal Categories

3.1.1 WriteFileForMonoidalStructure

▷ `WriteFileForMonoidalStructure(key_val_rec, package_name, files_rec)` (function)

Returns: nothing

This functions uses the dictionary *key_val_rec* to create a new monoidal structure. It generates the necessary files in the package *package_name* using the file-correspondence table *files_rec*. See the implementation for details.

3.2 Closed Monoidal Categories

3.2.1 WriteFileForClosedMonoidalStructure

▷ `WriteFileForClosedMonoidalStructure(key_val_rec, package_name, files_rec)` (function)

Returns: nothing

This functions uses the dictionary *key_val_rec* to create a new closed monoidal structure. It generates the necessary files in the package *package_name* using the file-correspondence table *files_rec*. See the implementation for details.

3.3 Coclosed Monoidal Categories

3.3.1 WriteFileForCoclosedMonoidalStructure

▷ `WriteFileForCoclosedMonoidalStructure(key_val_rec, package_name, files_rec)` (function)

Returns: nothing

This functions uses the dictionary *key_val_rec* to create a new coclosed monoidal structure. It generates the necessary files in the package *package_name* using the file-correspondence table *files_rec*. See the implementation for details.

Chapter 4

The terminal category with multiple objects

This is an example of a category which is created using `CategoryConstructor` out of no input.

This category "lies" in all doctrines and can hence be used (in conjunction with `LazyCategory`) in order to check the type-correctness of the various derived methods provided by CAP or any CAP-based package.

4.1 Constructors

4.1.1 TerminalCategory

▷ `TerminalCategory()` (function)

Construct a terminal category possibly with multiple objects.

Example

```
gap> T := TerminalCategory( );
TerminalCategory( )
gap> InfoOfInstalledOperationsOfCategory( T );
68 primitive operations were used to derive 317 operations for this category
which constructively
* IsEquippedWithHomomorphismStructure
* IsLinearCategoryOverCommutativeRing
* IsAbelianCategoryWithEnoughInjectives
* IsAbelianCategoryWithEnoughProjectives
* IsRigidSymmetricClosedMonoidalCategory
* IsRigidSymmetricCoclosedMonoidalCategory
gap> i := InitialObject( T );
<A zero object in TerminalCategory( )>
gap> t := TerminalObject( T );
<A zero object in TerminalCategory( )>
gap> z := ZeroObject( T );
<A zero object in TerminalCategory( )>
gap> Display( i );
A zero object in TerminalCategory( ).
gap> Display( t );
A zero object in TerminalCategory( ).
```

```
gap> Display( z );
A zero object in TerminalCategory( ).
gap> IsIdenticalObj( i, z );
true
gap> IsIdenticalObj( t, z );
true
gap> IsWellDefined( z );
true
gap> id_z := IdentityMorphism( z );
<A zero, identity morphism in TerminalCategory( )>
gap> fn_z := ZeroObjectFunctorial( T );
<A zero, isomorphism in TerminalCategory( )>
gap> IsWellDefined( fn_z );
true
gap> IsEqualForMorphisms( id_z, fn_z );
true
gap> IsCongruentForMorphisms( id_z, fn_z );
true
```

4.1.2 TerminalCategoryWithMultipleObjects

▷ TerminalCategoryWithMultipleObjects() (function)

Construct a terminal category with multiple objects.

Example
```
gap> T := TerminalCategoryWithMultipleObjects( );
TerminalCategoryWithMultipleObjects( )
gap> InfoOfInstalledOperationsOfCategory( T );
68 primitive operations were used to derive 317 operations for this category
which constructively
* IsEquippedWithHomomorphismStructure
* IsLinearCategoryOverCommutativeRing
* IsAbelianCategoryWithEnoughInjectives
* IsAbelianCategoryWithEnoughProjectives
* IsRigidSymmetricClosedMonoidalCategory
* IsRigidSymmetricCoclosedMonoidalCategory
gap> i := InitialObject( T );
<A zero object in TerminalCategoryWithMultipleObjects( )>
gap> t := TerminalObject( T );
<A zero object in TerminalCategoryWithMultipleObjects( )>
gap> z := ZeroObject( T );
<A zero object in TerminalCategoryWithMultipleObjects( )>
gap> Display( i );
ZeroObject
gap> Display( t );
ZeroObject
gap> Display( z );
ZeroObject
gap> IsIdenticalObj( i, z );
true
gap> IsIdenticalObj( t, z );
true
```

```
gap> id_z := IdentityMorphism( z );
<A zero, identity morphism in TerminalCategoryWithMultipleObjects( )>
gap> fn_z := ZeroObjectFunctorial( T );
<A zero, isomorphism in TerminalCategoryWithMultipleObjects( )>
gap> IsEqualForMorphisms( id_z, fn_z );
false
gap> IsCongruentForMorphisms( id_z, fn_z );
true
gap> a := "a" / T;
<A zero object in TerminalCategoryWithMultipleObjects( )>
gap> Display( a );
a
gap> IsWellDefined( a );
true
gap> aa := ObjectConstructor( T, "a" );
<A zero object in TerminalCategoryWithMultipleObjects( )>
gap> Display( aa );
a
gap> a = aa;
true
gap> b := "b" / T;
<A zero object in TerminalCategoryWithMultipleObjects( )>
gap> Display( b );
b
gap> a = b;
false
gap> t := TensorProduct( a, b );
<A zero object in TerminalCategoryWithMultipleObjects( )>
gap> Display( t );
TensorProductOnObjects
gap> a = t;
false
gap> TensorProduct( a, a ) = t;
true
gap> m := MorphismConstructor( a, "m", b );
<A zero, isomorphism in TerminalCategoryWithMultipleObjects( )>
gap> Display( m );
a
|
| m
v
b
gap> IsWellDefined( m );
true
gap> n := MorphismConstructor( a, "n", b );
<A zero, isomorphism in TerminalCategoryWithMultipleObjects( )>
gap> Display( n );
a
|
| n
v
b
```

```
gap> IsEqualForMorphisms( m, n );
false
gap> IsCongruentForMorphisms( m, n );
true
gap> m = n;
true
gap> id := IdentityMorphism( a );
<A zero, identity morphism in TerminalCategoryWithMultipleObjects( )>
gap> Display( id );
a
|
| IdentityMorphism
v
a
gap> m = id;
false
gap> id = MorphismConstructor( a, "xyz", a );
true
gap> z := ZeroMorphism( a, a );
<A zero, isomorphism in TerminalCategoryWithMultipleObjects( )>
gap> Display( z );
a
|
| ZeroMorphism
v
a
gap> id = z;
true
```

4.1.3 / (for IsString, IsTerminalCategoryWithMultipleObjects)

▷ `/(T, str)` (operation)

Create an object a in the terminal category T with multiple objects with `String(str)` $= a$.

4.2 GAP Categories

4.2.1 IsTerminalCategoryWithMultipleObjects (for IsCapCategory)

▷ `IsTerminalCategoryWithMultipleObjects(T)` (filter)

Returns: `true` or `false`

The GAP type of a terminal category with multiple objects.

4.2.2 IsCellInTerminalCategoryWithMultipleObjects (for IsCapCategoryCell)

▷ `IsCellInTerminalCategoryWithMultipleObjects(T)` (filter)

Returns: `true` or `false`

The GAP type of a cell in a terminal category with multiple objects.

4.2.3 IsObjectInTerminalCategoryWithMultipleObjects (for IsCellInTerminalCategoryWithMultipleObjects andIsCapTerminalCategoryObjectRep)

▷ `IsObjectInTerminalCategoryWithMultipleObjects(T)` (filter)

Returns: `true` or `false`

The GAP type of an object in a terminal category with multiple objects.

4.2.4 IsMorphismInTerminalCategoryWithMultipleObjects (for IsCellInTerminalCategoryWithMultipleObjects andIsCapTerminalCategoryMorphismRep)

▷ `IsMorphismInTerminalCategoryWithMultipleObjects(T)` (filter)

Returns: `true` or `false`

The GAP type of a morphism in a terminal category with multiple objects.

Index