Knotty Maps:

A Graphical Calculus for Linear Algebra

Jason M. Erbele
University of California, Riverside

CSU San Bernardino Math Colloquium
11 March 2015

Categories

Categories are devices that are useful for thinking about processes. They are built out of things called *objects* and paths between things, called *morphisms*.

Objects are boring. Morphisms are interesting.

Morphisms can be pictured as "buttons" on a string:

These string diagrams are read from top to bottom.

Categories

Categories are devices that are useful for thinking about processes. They are built out of things called *objects* and paths between things, called *morphisms*.

Objects are boring. Morphisms are interesting.

Morphisms can be pictured as "buttons" on a string:

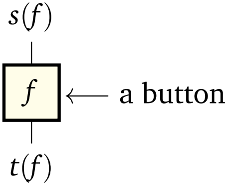
These string diagrams are read from top to bottom.

Categories

Categories are devices that are useful for thinking about processes. They are built out of things called *objects* and paths between things, called *morphisms*.

Objects are boring. Morphisms are interesting.

Morphisms can be pictured as “buttons” on a string:



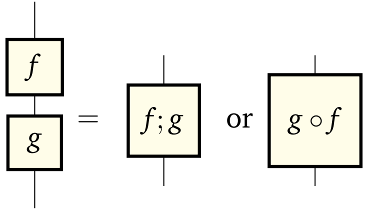
These string diagrams are read from top to bottom.

Morphisms can be composed to make new morphisms.

In pictures, that is two buttons on the same string getting glued together to make a new button:

This gives a way of thinking about sequential processes, but it is also useful to think about parallel processes.

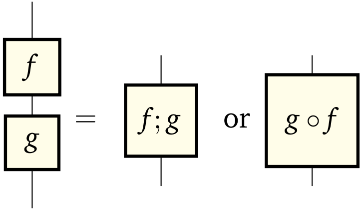
Morphisms can be composed to make new morphisms.
In pictures, that is two buttons on the same string getting glued together to make a new button:



This gives a way of thinking about sequential processes, but it is also useful to think about parallel processes.

Categories

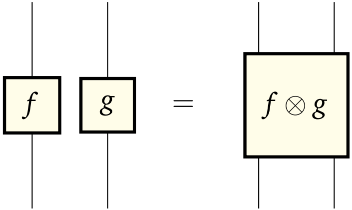
Morphisms can be composed to make new morphisms.
In pictures, that is two buttons on the same string getting glued together to make a new button:



This gives a way of thinking about sequential processes, but it is also useful to think about parallel processes.

Monoidal categories

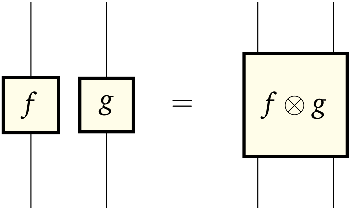
Monoidal categories allow us to glue buttons together that are sitting side by side:



Any number of strings, including zero, can be attached to the top or bottom of a button in a monoidal category.

Monoidal categories

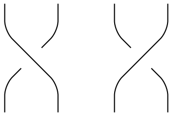
Monoidal categories allow us to glue buttons together that are sitting side by side:



Any number of strings, including zero, can be attached to the top or bottom of a button in a monoidal category.

Braided monoidal categories

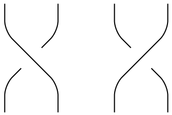
To get a string from the left of a diagram to the right, it will have to cross over other strings. In a braided monoidal category there are two special buttons to do this and its inverse:



These “braidings” interact with other buttons the way you would expect twisted strings to go across physical buttons. They also interact with each other with rules familiar to knot theorists as *Reidemeister moves*.

Braided monoidal categories

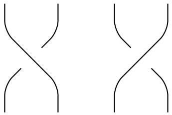
To get a string from the left of a diagram to the right, it will have to cross over other strings. In a braided monoidal category there are two special buttons to do this and its inverse:



These “braidings” interact with other buttons the way you would expect twisted strings to go across physical buttons. They also interact with each other with rules familiar to knot theorists as *Reidemeister moves*.

Braided monoidal categories

To get a string from the left of a diagram to the right, it will have to cross over other strings. In a braided monoidal category there are two special buttons to do this and its inverse:



These “braidings” interact with other buttons the way you would expect twisted strings to go across physical buttons. They also interact with each other with rules familiar to knot theorists as *Reidemeister moves*.

Symmetric monoidal categories

If the braiding is its own inverse, swapping a pair of strings *twice* will always be "the same" as doing nothing at all. This gives a symmetry between "over" crossings and "under" crossings, turning the braided monoidal category into a symmetric monoidal category.

Theorem (John Baez, JE)

(Part 1) Every linear map between vector spaces can be expressed as a string diagram in a symmetric monoidal category that has a special set of extra buttons.

Symmetric monoidal categories

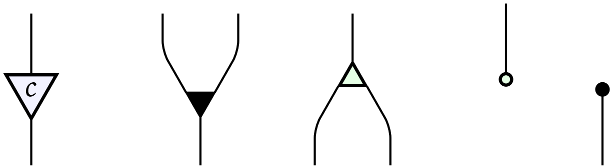
If the braiding is its own inverse, swapping a pair of strings *twice* will always be "the same" as doing nothing at all. This gives a symmetry between "over" crossings and "under" crossings, turning the braided monoidal category into a symmetric monoidal category.

Theorem (John Baez, JE)

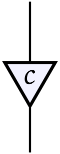
(Part 1) Every linear map between vector spaces can be expressed as a string diagram in a symmetric monoidal category that has a special set of extra buttons.

Extra buttons

These special buttons come in five forms:



Many buttons come in the form of one string in, one string out:



$$c: \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & cx \end{array}$$

This set of buttons allows us to multiply any strand by an arbitrary element of our field.

Extra buttons

The copy room

Varying the number of strings attached below the button...

$$\begin{aligned} \Delta: \quad \mathbb{R} &\to \mathbb{R} \oplus \mathbb{R} \\ x &\mapsto (x, x) \end{aligned}$$

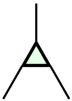
$$!: \mathbb{R} \to \{0\}$$

... we can duplicate or delete the value on a string.

Extra buttons

The copy room

Varying the number of strings attached below the button...



$$\begin{aligned} \Delta\colon \quad \mathbb{R} &\to \mathbb{R} \oplus \mathbb{R} \\ x &\mapsto (x, x) \end{aligned}$$

$$!\colon \mathbb{R} \to \{0\}$$

... we can duplicate or delete the value on a string.

Extra buttons

Additionally...

Varying the number of strings attached above the button...

$$\begin{aligned} +: \quad \mathbb{R} \oplus \mathbb{R} &\to \mathbb{R} \\ (x,y) &\mapsto x+y \end{aligned}$$

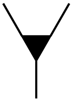
$$0 : \{0\} \to \mathbb{R}$$

... we can add strings together or make zero from nothing.

Extra buttons

Additionally...

Varying the number of strings attached above the button...



$$+\colon \quad \begin{array}{ccc} \mathbb{R} \oplus \mathbb{R} & \to & \mathbb{R} \\ (x, y) & \mapsto & x + y \end{array}$$

$$0\colon \{0\} \to \mathbb{R}$$

... we can add strings together or make zero from nothing.

Example

The linear transformation $T\colon \mathbb{R}^3 \to \mathbb{R}^2$ given by:

$$(x_1, x_2, x_3) \mapsto (y_1, y_2) = (3x_1 + 7x_2 + 2x_3, 9x_1 + x_2)$$

corresponds to the matrix:

$$\begin{pmatrix} 3 & 7 & 2 \\ 9 & 1 & 0 \end{pmatrix}$$

and has a standard form string diagram that looks like this:

Example

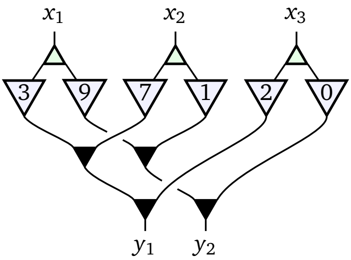
The linear transformation $T\colon \mathbb{R}^3 \to \mathbb{R}^2$ given by:

$$(x_1, x_2, x_3) \mapsto (y_1, y_2) = (3x_1 + 7x_2 + 2x_3, 9x_1 + x_2)$$

corresponds to the matrix:

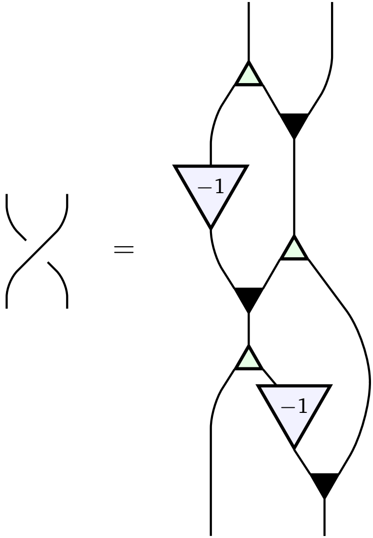
$$\begin{pmatrix} 3 & 7 & 2 \\ 9 & 1 & 0 \end{pmatrix}$$

and has a standard form string diagram that looks like this:



We don't even need to start with the braiding button, as it can be built from three of the "extra" buttons:

We don't even need to start with the braiding button, as it can be built from three of the "extra" buttons:



Rewrite rules

This raises an important question: *How can we tell when different combinations of buttons do the same thing?* In other words, what are the rules for rewriting these string diagrams?

Theorem (John Baez, JE)

(Part 2) Any two string diagrams that do the same thing can be related through a sequence of rewrite rules, where each step in the sequence uses exactly one of 18 special rules.

Rewrite rules

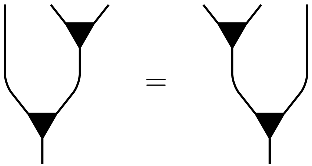
This raises an important question: *How can we tell when different combinations of buttons do the same thing?* In other words, what are the rules for rewriting these string diagrams?

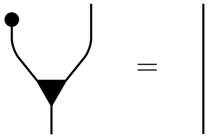
Theorem (John Baez, JE)

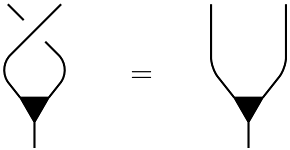
(Part 2) Any two string diagrams that do the same thing can be related through a sequence of rewrite rules, where each step in the sequence uses exactly one of 18 special rules.

Rewrite rules

Dark buttons
(Algebra)

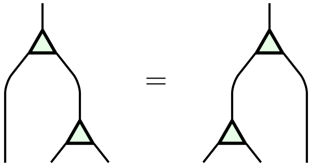


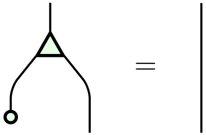


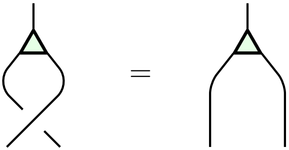


Rewrite rules

Light buttons
(Coalgebra)

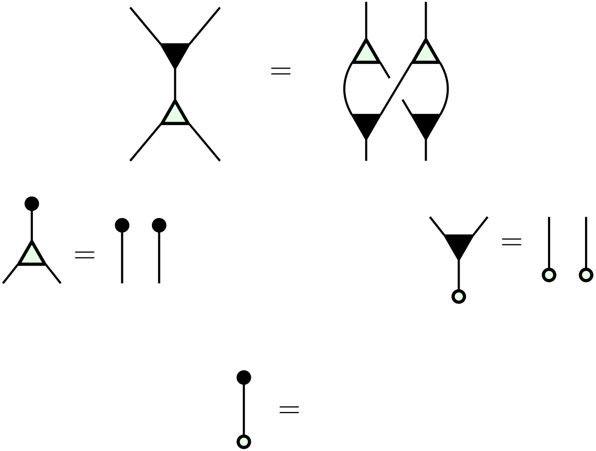






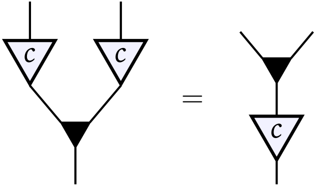
Rewrite rules

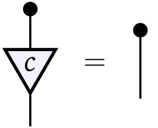
Mixing dark and light buttons
(Bialgebra)



Rewrite rules

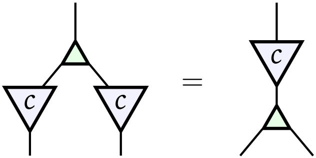
Mixing dark and scalar buttons (Linearity)

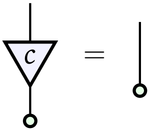




Rewrite rules

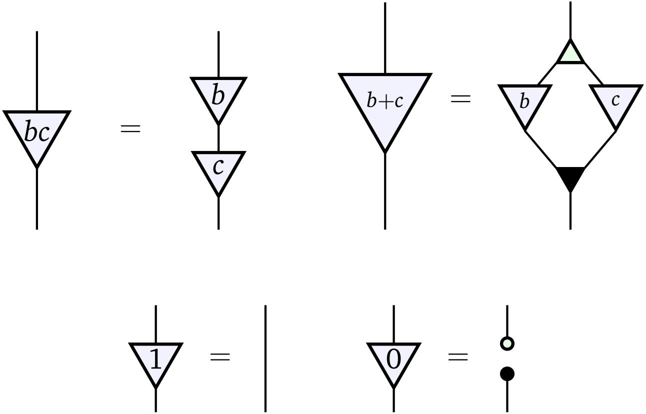
Mixing light and scalar buttons
(Colinearity)





Rewrite rules

Ring properties of the scalar buttons



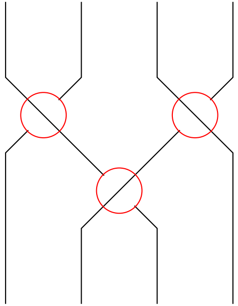
This slide left intentionally blank.

This slide left intentionally blank.

(Right isn't as blank)

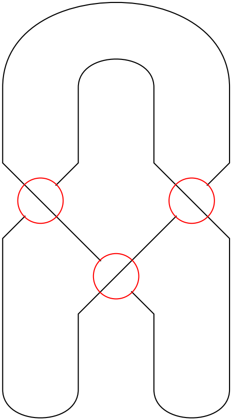
Tangles and knots

A string diagram where the only buttons are the two braidings of a braided monoidal category is the same as a diagram of a braid.



Tangles and knots

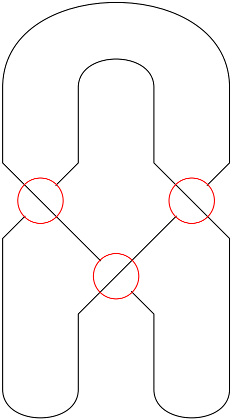
To get a knot, we need a way to loop strings back around:

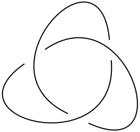


This means we need a button that looks like ∩ and one that looks like ∪.

Tangles and knots

To get a knot, we need a way to loop strings back around:

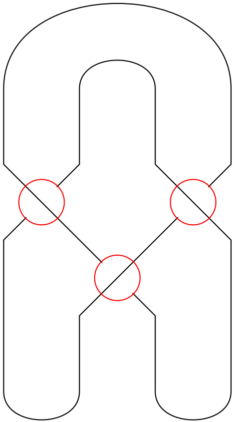


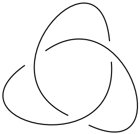


This means we need a button that looks like ∩ and one that looks like ∪.

Tangles and knots

To get a knot, we need a way to loop strings back around:

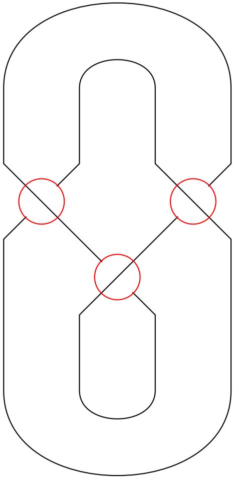




This means we need a button that looks like ∩ and one that looks like ∪.

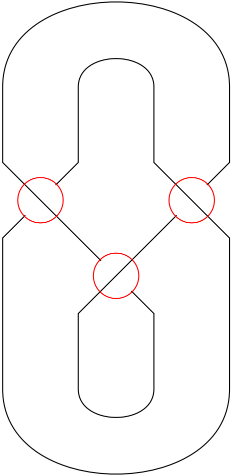
Tangles and knots

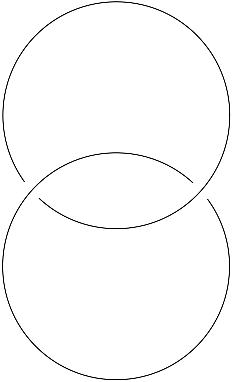
There is a bigger structure that comes from adding these two buttons, since we don't only get knots:



Tangles and knots

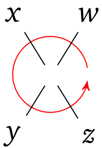
There is a bigger structure that comes from adding these two buttons, since we don’t only get knots:





Idea:

- Label each downward strand with a group element
- Add relations at each button



given by

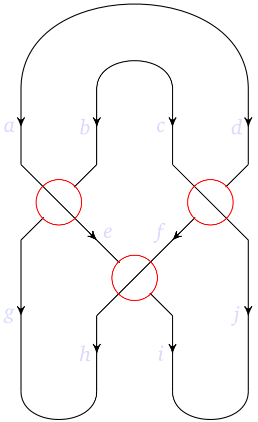
$$wxy^{-1}z^{-1} = 1,$$

and continuous downward strands are the same group element.

- In particular, the strings leaving $\cap$ are inverses. Likewise for strings entering $\cup$.

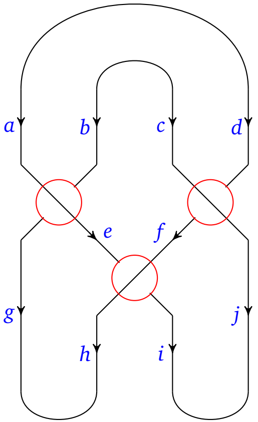
Tangles and knots

Knot groups



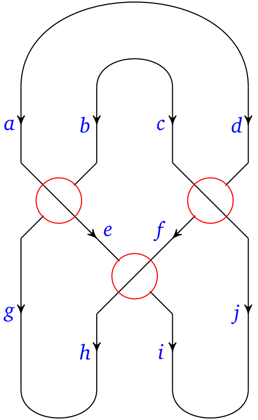
$$bag^{-1}e^{-1} = 1$$
$$a = e$$
$$dcf^{-1}j^{-1} = 1$$
$$c = j$$
$$feh^{-1}i^{-1} = 1$$
$$f = h$$
$$a^{-1}d^{-1} = 1$$
$$b^{-1}c^{-1} = 1$$
$$hg = 1$$
$$ji = 1$$

The group generated by $\{a, b, c, d, e, f, g, h, i, j\}$ subject to the relations above is the knot group for the trefoil.



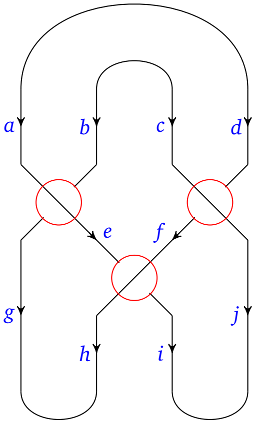
$$bag^{-1}e^{-1} = 1$$
$$a = e$$
$$dcf^{-1}j^{-1} = 1$$
$$c = j$$
$$feh^{-1}i^{-1} = 1$$
$$f = h$$
$$a^{-1}d^{-1} = 1$$
$$b^{-1}c^{-1} = 1$$
$$hg = 1$$
$$ji = 1$$

The group generated by $\{a, b, c, d, e, f, g, h, i, j\}$ subject to the relations above is the knot group for the trefoil.



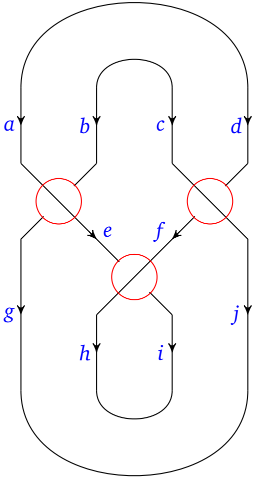
$$bag^{-1}e^{-1} = 1$$
$$a = e$$
$$dcf^{-1}j^{-1} = 1$$
$$c = j$$
$$feh^{-1}i^{-1} = 1$$
$$f = h$$
$$a^{-1}d^{-1} = 1$$
$$b^{-1}c^{-1} = 1$$
$$hg = 1$$
$$ji = 1$$

The group generated by $\{a, b, c, d, e, f, g, h, i, j\}$ subject to the relations above is the knot group for the trefoil.



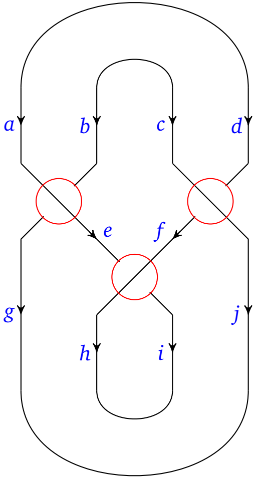
$$bag^{-1}e^{-1} = 1$$
$$a = e$$
$$dcf^{-1}j^{-1} = 1$$
$$c = j$$
$$feh^{-1}i^{-1} = 1$$
$$f = h$$
$$a^{-1}d^{-1} = 1$$
$$b^{-1}c^{-1} = 1$$
$$hg = 1$$
$$ji = 1$$

The group generated by $\{a, b, c, d, e, f, g, h, i, j\}$ subject to the relations above is the knot group for the trefoil.



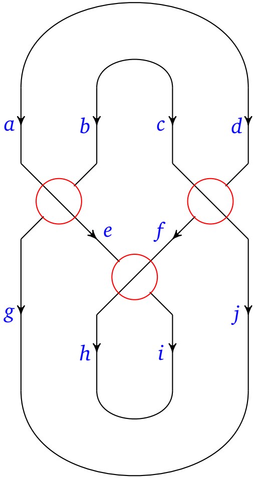
$$bag^{-1}e^{-1} = 1$$
$$a = e$$
$$dcf^{-1}j^{-1} = 1$$
$$c = j$$
$$feh^{-1}i^{-1} = 1$$
$$f = h$$
$$a^{-1}d^{-1} = 1$$
$$b^{-1}c^{-1} = 1$$
$$jg = 1$$
$$ih = 1$$

The group generated by $\{a, b, c, d, e, f, g, h, i, j\}$ subject to the relations above is the link group for the Hopf link.



$$bag^{-1}e^{-1} = 1$$
$$a = e$$
$$dcf^{-1}j^{-1} = 1$$
$$c = j$$
$$feh^{-1}i^{-1} = 1$$
$$f = h$$
$$a^{-1}d^{-1} = 1$$
$$b^{-1}c^{-1} = 1$$
$$jg = 1$$
$$ih = 1$$

The group generated by $\{a, b, c, d, e, f, g, h, i, j\}$ subject to the relations above is the link group for the Hopf link.



$$bag^{-1}e^{-1} = 1$$
$$a = e$$
$$dcf^{-1}j^{-1} = 1$$
$$c = j$$
$$feh^{-1}i^{-1} = 1$$
$$f = h$$
$$a^{-1}d^{-1} = 1$$
$$b^{-1}c^{-1} = 1$$
$$jg = 1$$
$$ih = 1$$

The group generated by $\{a, b, c, d, e, f, g, h, i, j\}$ subject to the relations above is the link group for the Hopf link.

Let's see those relations together:

Trefoil knot	**Hopf link**
$bag^{-1}e^{-1} = 1$	$bag^{-1}e^{-1} = 1$
$a = e$	$a = e$
$dcf^{-1}j^{-1} = 1$	$dcf^{-1}j^{-1} = 1$
$c = j$	$c = j$
$feh^{-1}i^{-1} = 1$	$feh^{-1}i^{-1} = 1$
$f = h$	$f = h$
$a^{-1}d^{-1} = 1$	$a^{-1}d^{-1} = 1$
$b^{-1}c^{-1} = 1$	$b^{-1}c^{-1} = 1$
$hg = 1$	$jg = 1$
$ji = 1$	$ih = 1$

The only difference is where the ∪s are different. So maybe we can define the tangle group for the common tangle portion...

Let's see those relations together:

Trefoil knot	**Hopf link**
$bag^{-1}e^{-1} = 1$	$bag^{-1}e^{-1} = 1$
$a = e$	$a = e$
$dcf^{-1}j^{-1} = 1$	$dcf^{-1}j^{-1} = 1$
$c = j$	$c = j$
$feh^{-1}i^{-1} = 1$	$feh^{-1}i^{-1} = 1$
$f = h$	$f = h$
$a^{-1}d^{-1} = 1$	$a^{-1}d^{-1} = 1$
$b^{-1}c^{-1} = 1$	$b^{-1}c^{-1} = 1$
$hg = 1$	$jg = 1$
$ji = 1$	$ih = 1$

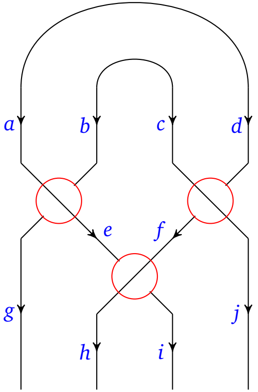
The only difference is where the $\cup$s are different. So maybe we can define the tangle group for the common tangle portion...

... as the group generated by $\{a, b, c, d, e, f, g, h, i, j\}$ subject only to the relations that come from the buttons that are in common!

$$bag^{-1}e^{-1} = 1$$
$$a = e$$
$$dcf^{-1}j^{-1} = 1$$
$$c = j$$
$$feh^{-1}i^{-1} = 1$$
$$f = h$$
$$a^{-1}d^{-1} = 1$$
$$b^{-1}c^{-1} = 1$$

The tangle group for this tangle is the group generated by $\{a, b, c, d, e, f, g, h, i, j\}$, subject to the relations above. Tongue tangling fact: the tangle group of a tangle is a tangle invariant.

... as the group generated by $\{a, b, c, d, e, f, g, h, i, j\}$ subject only to the relations that come from the buttons that are in common!



$$bag^{-1}e^{-1} = 1$$
$$a = e$$
$$dcf^{-1}j^{-1} = 1$$
$$c = j$$
$$feh^{-1}i^{-1} = 1$$
$$f = h$$
$$a^{-1}d^{-1} = 1$$
$$b^{-1}c^{-1} = 1$$

The tangle group for this tangle is the group generated by $\{a, b, c, d, e, f, g, h, i, j\}$, subject to the relations above. Tongue tangling fact: the tangle group of a tangle is a tangle invariant.

Thank you.