wick

$$: \overline{\Psi}_\alpha(x)\gamma^\mu_{\alpha\beta}\overbrace{A_\mu(x)\Psi_\beta(x)}\overline{\Psi}_\eta(y)\gamma^\nu_{\eta\rho}(y)A_\nu\Psi_\rho(y) :$$

```
$ :
wick(id: #1, overline(Psi))_alpha (x)
gamma^mu_(alpha beta)
wick(pos: #top, A)_mu (x)
wick(Psi)_beta (x)
wick(overline(Psi))_eta (y)
gamma^nu_(eta rho) (y)
wick(pos: #top, A)_nu
wick(id: #1, Psi)_rho (y)
: $
```

This small Typst package handles the typesetting of Wick contractions $\overbrace{\phi^\dagger}\phi$. We shall not fall into the same typographical limitations that stopped the italian physicist Gian Carlo Wick from using this notation in his papers.

> *"Houriet and Kind's symbol is a line connecting the two factors like a string attached at both ends. It is very convenient for handwriting, but has been abandoned here for typographical reasons."*
>
> — Gian Carlo Wick [PhysRev.80.268]

Basic usage

In the examples which follow we must suppose that the following preamble has been used.

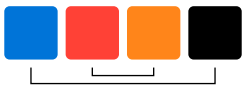
```
#import "wick.typ": wick

// Only used to make the examples shorter
#set box(width: 20pt, height: 20pt, radius: 2pt)
```

The hole package functionality is contained within the `wick` function. By calling `wick(..)` on some content you place a *contraction point*. Every second contraction point you place, gets contracted with the previous one.

```
$
#wick(box(fill: red))#h(3pt)
#wick(box(fill: orange))
#wick(box(fill: blue))#h(3pt)
#wick(box(fill: black))
$
```

By default every contraction point is given an `id` equal to the integer `0`. Since only contraction points with the same `id` get contracted together, by specifying different `ids` is possible to get multiple overlapping contractions.

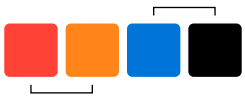
```
$
#wick(id: 1, box(fill: blue))#h(3pt)
#wick(box(fill: red))#h(3pt)
#wick(box(fill: orange))#h(3pt)
#wick(id: 1, box(fill: black))
$
```



The `id` is not required to be an integer; for example `strings` are another reasonable choice. That being said, integer `ids` have a special behaviour. Did you notice how the outer contraction in the last example automatically moved further away? When `id` is of type `int`, the distance of the contraction line, which is otherwise specified by the parameter `dy`, is instead computed as `dy * (1 + id / 2.0)`.

The `pos` parameter of the `wick` function allows the user to specify whether the contraction should be drawn above `pos: top` or below `pos: bottom` the equation. The default is to contract below.

```
$
#wick(pos: bottom, box(fill: red))#h(3pt)
#wick(pos: bottom, box(fill: orange))#h(3pt)
#wick(pos: top, box(fill: blue))#h(3pt)
#wick(pos: top, box(fill: black))
$
```



Only contraction points with the same value of `pos` can be contracted together, just like for `id`. Other then these two, there are no other constraints of which contraction points can be actually contracted.

Point specific styling

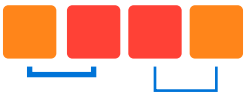
At the moment, the only point specific option is `dx`, which shifts horizontally the place where the line starts.

```
$
#wick(box(fill: orange), stroke: blue + 2pt)
#wick(box(fill: red))#h(3pt)
#wick(box(fill: red), dy: 1em, stroke: 2pt)#h(3pt)
#wick(box(fill: orange), stroke: blue)
$
```

Shared styling options

There are some styling options that regard the contraction as a hole. In this scenarios both the first and the second contraction point can specify the styling option, and we say that the option is *shared*. The default value of every shared styling option is `auto`. If the two contraction points set different values for the same shared option, the one specified by the second contraction point wins. If no value is specified a default is used.

```
$
#wick(box(fill: orange), stroke: blue + 2pt)
#wick(box(fill: red))#h(3pt)
#wick(box(fill: red), dy: 1em, stroke: 2pt)#h(3pt)
#wick(box(fill: orange), stroke: blue)
$
```



```
$
#wick(box(fill: red))#h(3pt)
```

```
#wick(box(fill: orange), offset: 0pt)
$
```

A boolean option called `flat` allows to specify whether the contraction line should be kept horizontal or not, when contracting objects of different heights.

```
#wick(pos:top, box(width: 30pt, height: 30pt, fill: red)) times
#wick(pos: top, box(fill: orange)) = 0
#h(30pt)
#wick(pos:top, box(width: 30pt, height: 30pt, fill: red)) times
#wick(pos: top, flat: false, box(fill: orange)) = 0
```

The `flat` parameter has no effect if `pos` is set to `bottom`.