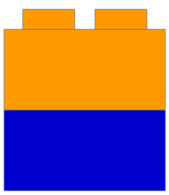
Visualizing Path Semantics Using LEGO Bricks

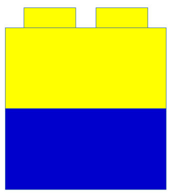
by Sven Nilsen, 2020

In this paper I show how to visualize path semantics using LEGO bricks.

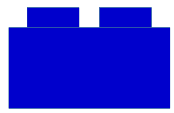
Path semantics is kind of like building with LEGO bricks, but instead of bricks, there are functions.



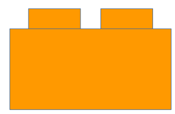
false : bool



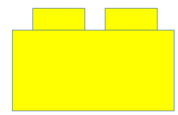
true : bool



bool



false

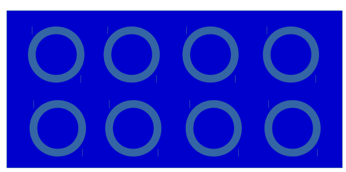


true

Imagine that every symbol is assigned a lego brick and there are rules for how to use them.

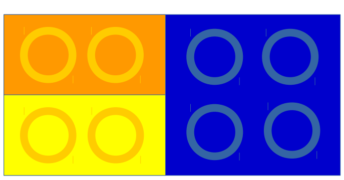
A function of type `bool → bool` is can be thought of as a larger brick of same color as `bool`:

bool → bool
(side view)

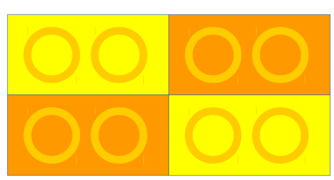


bool → bool
(top view)

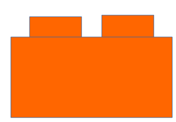
A function of type `bool → bool` must be defined for `false` and `true`.



undefined function of
type `bool → bool`



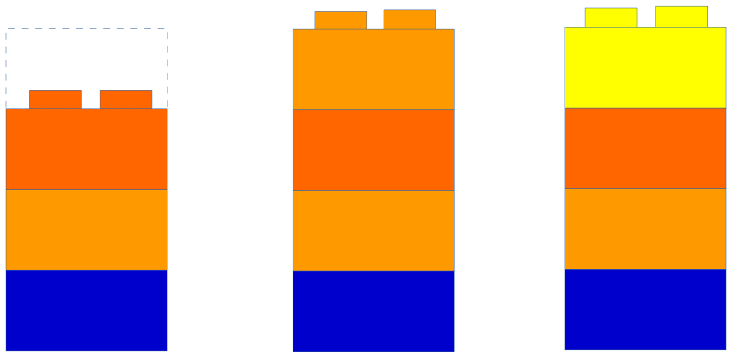
not : bool → bool



not

When you define a function, e.g. `not` you can refer to it using a symbolic "brick".
This brick can only be used according to specific rules.

Since `not` takes a `bool` as input, it can be put on top of `false` or `true`.
In turn, a brick can be placed on top of `not`, but only if `not` returns it for the value underneath.

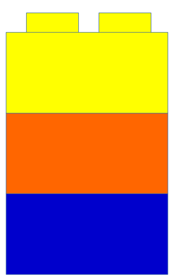


incorrect correct

In path semantics, this is written:

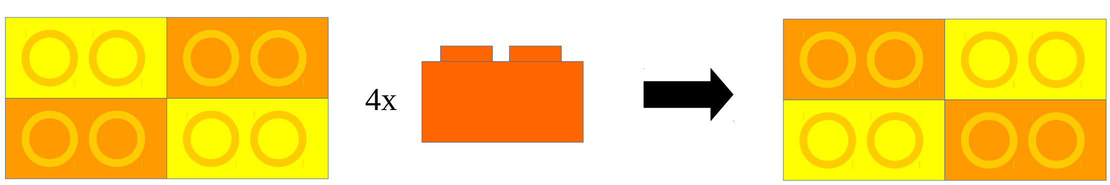
    false : [not] true

You can also leave out the input, `[not] true`:



The missing brick is implicitly understood from the context.
In this case, there is only one solution, which is `false : [not] true`.

Now, you can try use the `not` brick 4 times on top of the definition of the `not` function:



Notice that this pattern corresponds to the same definition of `not`, but reversed.
The order of the arguments does not matter, so this definition is equal to `not`!

Congratulations! You have proved the following statement in path semantics:

    not[not] <=> not