Oh no! On the way to `add_comm` , a wild `succ_add` appears. `succ_add a b` is the proof that `(succ a) + b = succ (a + b)` for `a` and `b` numbers. This result is what's standing in the way of `x + y = y + x` . Again we have the problem that we are adding `b` to things, so we need to use induction to split into the cases where `b = 0` and `b` is a successor.

Note that `succ a + zb`

Next

Assumptions:

nb : `succ a + zb = succ (a + zb)`

Goal:

`succ (succ a + zb) = succ (a + succ zb)`

```
rw[nb]
```

Retry

Active Goal

Objects:

a zb : ℕ

Assumptions:

nb : `succ a + zb = succ (a + zb)`

Goal:

`succ (succ (a + zb)) = succ (a + succ zb)`

```
rw[add_succ a zb]
```

Retry

Active Goal

Objects:

a zb : ℕ

Assumptions:

nb : `succ a + zb = succ (a + zb)`

Goal:

`succ (succ (a + zb)) = succ (succ (a + zb))`

```
rfl
```

Retry

`level completed! 🎉`

add_succ

`(a d : ℕ) : a + MyNat.succ d = MyNat.succ (a + d)`

`add_succ a b` is the proof of `a + succ b = succ (a + b)` .