



Theorembeweiserpraktikum

Even More Tactics

Jakob von Raumer, Sebastian Ullrich | SS 2021



THEOREM PROVER

focus vs. •

A correction: `focus` does not actually force the first goal to be closed

```
example (hp : p) (hq : q) : p ∧ q := by
  constructor
  focus -- `! p`
    skip
  focus
    -- still `! p`
  exact hq -- type mismatch
```

We introduced • (input as `\centerdot`) as a (prettier) way to do just that:

```
constructor
• skip -- unsolved goals: ... ! p
• exact hq -- is still checked
```

This is basically an unnamed case

While `focus` is not strictly structuring given this insight, we'll allow both this semester

What's Up with Those Dots Anyway

We've seen `·` before: in *terms*, it can be used as a nameless lambda

```
set_option pp.binder_types false
#check (· :: ·) -- fun a a_1 => a :: a_1 : ...
#check (1 + 2 * ·) -- fun a => 1 + 2 * a : Nat → Nat
#check [0, 1].map (·.succ) -- List.map (fun a => Nat.succ a) [0, 1] : List Nat
```

All occurrences of `·` are bound to the nearest surrounding parentheses, from left to right

refine

We already know we can arbitrarily nest terms and tactics:

```
example (hr : r) (hrp : r → p) (hq : q) : p ∧ q := by
  exact ⟨(by simp_all), hq⟩
```

We can use `refine` to move out nested tactic blocks

```
refine ⟨?p, hq⟩
case p => -- h p
  simp_all
```

`apply e` where `e : (h : p) → ... → q` can be thought of as a special case of `refine` : `refine e ?h ...`