Cheatsheet for tactics

`intro h`

- ‘Assume h is a proof of P’
- ‘Assume P is true’
- ’Let a be an arbitrary element of `Xin a`∀ a ∈ X‘ term.

If the goal is `P → Q`, creates a hypothesis h : P and changes the goal to `Q`.

If the goal is `∀ a∈ X, p x` then `intro a` creates a term a of type `X` and changes the goal to `p x`.

`intros h1 h2`

Does several `intro` commands at once.

`exact h`

- Our hypothesis h establishes the result

Applies the hypothesis `h` to prove the theorem. `h` must be exactly the desired goal. .

`assumption`

- Now we see that our hypotheses yield the result.

If one of the hypotheses exactly proves the theorem, apply that hypothesis.

`apply`

- Given h, to show `Q` it suffices to show P.

If a hypothesis says `h : P → Q` and our goal is `Q` then `apply h` replaces `Q` by P.

`rw` or `rewrite`

- Given `h : a=b` we can replace a by `b` (or in the goal or in a hypothesis.
- By hypothesis h, we can replace *left side of h* with *right side of h*. Or, with the ← version, we can replace the *right side of h* with the *left side of h* in the goal.

Given a hypothesis that asserts the equality of two things (= or ↔), replace one thing by the other.

To replace in the *goal*:

By default, given `h : a=b`, the command `rw h` replaces `a` by `b` The command `rw ← h` replaces `b` by `a`.

To replace in a *hypothesis*, use `at`.

Given hypotheses `hab: a=b` and `hbc: b=c` then `rw hbc at hab` changes `hab` to `a=c`. Note that you are using `hbc` to rewrite `hab`.

`by_contra`

- We will prove the contrapositive, so assume the conclusion is false.

The tactic `by_contra` makes the negation of the goal a hypothesis and changes the goal to `false`.

`cases`

- Given `P ∨ Q`, we consider separately the cases when `P` is true and when `Q` is true. (`P ∨ Q` is a hypothesis)
- To prove `P ∧ Q` we prove `P` and `Q` separately.
- If we know there exists an `x` of type T satisfying a property `p x`, we can assume separately that `x` is of type T and that `p x` is a hypothesis. In other words, decompose a `∃ x, p x` hypothesis into `∃ x` hypothesis and a `p x` hypothesis.

If `x` is a *hypothesis* about an inductive type then `cases` breaks up that hypothesis into its component parts. For example, if `h` is a proof of an and term then `cases h` is are the separate proofs of the terms. If 'h' is a proof about a product, then `cases h` are the separate proofs of the terms.

`left` and `right`

- It suffices to prove the proposition on the left/right.

When a goal is made up of two parts, either of which suffices, `left` and `right` replace the goal with the corresponding part. For example, if the goal is `P ∨ Q` then `left` is like `apply` for the implication `P → P ∨ Q` and right is `apply` for `Q → P ∨ Q`.

`use`

- Then `x` satisfies the desired conditions.

`use x` says to instantiate the `∃ y, p y` clause of a goal with `x`, turning the goal into `p x`.

`split`

- We consider the component propositions to our conclusion in turn.

`split` breaks up a compound goal (like `P ∧ Q` or `P ↔ Q`) into subgoals.

`refl`

- True by definition.