Lean Cheatsheet

In the following table, *name* always refers to a name already known to Lean while *new_name* refers to a new name provided by the user. When one of these words appears twice in the same line, the appearances do not designate the same name. *expr* designates an expression, for example the name of an object in the context, an arithmetic expression that is a function of such objects, a hypothesis in the context, or a lemma applied to any of these.

Logical symbol	Appears in goal	Appears in hypothesis
∀ (for all)	`intro` *new_name*	`apply` *expr* or `specialize` *name expr*
∃ (there exists)	`use` *expr*	`cases` *expr* `with` *new_name new_name*
→ (implies)	`intro` *new_name*	`apply` *expr* or `specialize` *name expr*
↔ (if and only if)	`split`	`rw` *expr* or `rw ←` *expr*
∧ (and)	`split`	`cases` *expr* `with` *new_name new_name*
∨ (or)	`left` or `right`	`cases` *expr* `with` *new_name new_name*
¬ (not)	`intro` *new_name*	`apply` *expr* or `specialize` *name expr*

Note: Traditional paper-based practice uses ⇒ for implication, uses ⟺ for equivalence, and does not use a notation for “and”, “or” and “not”.

In the left-hand column of the following table, the parts in brackets are optional. The effect of these parts is also in brackets in the right-hand column. It is almost always a matter of specifying that a manipulation, which acts by default on the goal, must be performed rather on a certain hypothesis named *hyp*.

Tactic	Effect
`exact` *expr*	asserts that the goal can be satisfied by *expr*
`have` *new_name* : *fact*	introduces a name *new_name* asserting that *fact* is provable
`unfold` *name* (`at` *hyp*)	unfold the definition of *name* in the goal (or in the hypothesis *hyp*)
`change` *expr* (`at` *hyp*)	transform the goal (or the hypothesis *hyp*) into the expression *expr* to which it is equivalent by definition
`rw` (←) *expr* (`at` *hyp*)	in the goal (or in the hypothesis *hyp*), replace the left-hand side (or the right-hand side, if ← is present) of the equality or equivalence *expr* by the other side. The expression to be replaced must appear explicitly, one may use `unfold` or `change` to ensure this.
`linarith`	prove the goal by a linear combination of hypotheses
`ring`	prove the goal by combining the axioms of a commutative (semi)ring
`library_search`	search for a single existing lemma which closes the goal, also using local hypotheses.
`choose` *new_name new_name* `using` *expr*	given *expr* : $\forall x, \exists y, P(x,y)$, use the axiom of choice to produce a function $x \mapsto y(x)$ satisfying $\forall x, P(x, y(x))$
`exfalso`	apply the rule *ex falso quod libet*
`by_contradiction` *new_name*	start a proof by contradiction, using *new_name* as name for the hypothesis that is the negation of the goal
`by_cases` *new_name* : *expr*	split the proof into two cases depending on whether *expr* is true or false, using *new_name* as name for this hypothesis
`contrapose`	transform a goal of the form *expr* → *expr* into its contrapositive
`push_neg` (`at` *hyp*)	push negations in the goal (or in the hypothesis *hyp*)