

Table 1: Basic tactics

what do you want to do?	tactics
directly provide a proof term M	<code>exact M</code>
provide a proof term M with wildcards _ for subgoals	<code>refine M</code>
use an assumption	<code>exact, assumption, eassumption</code>
replace M in the goal by a variable and generalize over it	<code>generalize M</code>
add the type of M to the context	<code>pose proof M as H; generalize M and then intro H</code>
prove A and introduce it to the context	<code>assert (H: A)</code>
change the goal to G	<code>enough G</code>
unfold the definition of c (in the goal, in H, everywhere)	<code>unfold c; unfold c in H; unfold c in *</code>
fold back the definition of c	<code>fold c</code>
move hypothesis H back to the goal (“reverting” <code>intro H</code>)	<code>revert H</code>
remove hypothesis H	<code>clear H</code>
find a contradiction in the context	<code>contradiction</code>
simple Prolog-like automation	<code>auto, eauto</code>
automatically solve an “easy” goal	<code>easy, trivial</code>
automatically solve a linear arithmetic goal	<code>lia</code> (needs <code>Require Import Psatz</code>)
automatically prove a propositional formula	<code>tauto</code>
automatically prove a first-order formula	<code>firstorder</code>
combine <code>tauto</code> with <code>auto</code>	<code>intuition</code>
induction on t	<code>induction t as [...]</code>
reasoning by cases on an object t	<code>destruct t as [...]</code>
reasoning by cases on a proof H of an inductive predicate	<code>inversion H as [...]; inversion_clear H as [...]</code>
replace t with x and add x = t to the context	<code>remember t as x</code>
give up on the goal	<code>admit</code>