Coq Proof Rules

A table with text and numbers

Description automatically generated

A black and white text

Description automatically generated

A white grid with black text

Description automatically generated

# Exact or Assumption

* Use an assumption to cancel a goal.

# Assert

* Adding new facts in the goal to be proven.
* Implements backwards reasoning.

# Intros

* Hypotheses move above the line. The names of variables and hypotheses appear before a colon, followed by their type.
* Does multiple introductions and infer names when none are given.

# Apply

* Takes as an argument a proof h of a proposition ∀x1 . . . xn, A1 → · · · Ap → B.
* It tries to find terms ti such that the current goal is equivalent to B[xi ← ti]i=1...n and generates subgoals corresponding to Aj [xi ← ti]i=1...n. If some of the xi are not inferred by the system, it is always possible to use the variant with (xi:= ti).

# Split

* Assume left side is true as a goal and the right side is true as a goal.

# Left

* Assume left hand side of the disjunction is true.

# Right

* Assume right hand side of the disjunction is true.

# Destruct

* The destruct t tactic generates a new subgoal for each constructor and introduces new variables and hypothesis corresponding to the arguments of the constructor.

# Reflexivity

* The introduction rule of equality.

# Rewrite

* Replace all the occurrences of u in P(u).

# t1 ; t2

* Applies tactic t1 then tactic t2 on generated subgoals.

# t1 || t2

* Applies tactic t1, when it fails, applies t2.

# Try t

* Applies tactic t, does nothing when t fails.

Repeat t

* Repeats tactic t until it fails.

# Contradiction

* Solves the goal when False, or A and ¬A appear in the hypotheses.

# Tauto

* Solves propositional tautologies.

Trivial

* Tries very simple lemmas to solve the goal.

Auto

* Searches in a database of lemmas to solve the goal.

# Intuition

* Removes the propositional structure of the goal then auto.

Omega

* Solves goals in linear arithmetic.

# Induction

* 2 goals: 0 and Inductive Case.

# Simpl

* Replace calls of a recursive function on a complex argument by the corresponding value, as stated by the definition of the function.

# Discriminate

* Usable when one hypothesis asserts that 0 = S e or true = false.
* Hypothesis is self-contradictory and goals containing this kind of hypothesis are automatically discarded by this tactic.

# Injection

* Usable when one hypothesis has the form Sx = Sy to deduce x = y

# Ring

* Does associative-commutative rewriting

# Inversion

* Analyses all the constructors of the inductive predicate.
* Discards the ones that could not have been applied.
* When some constructors could have been applied it creates a new goal where the premises of this constructor are added in the context.

# Notes:

* Can replace definitions using ‘unfold’.
* Open scope z-scope for using integers.