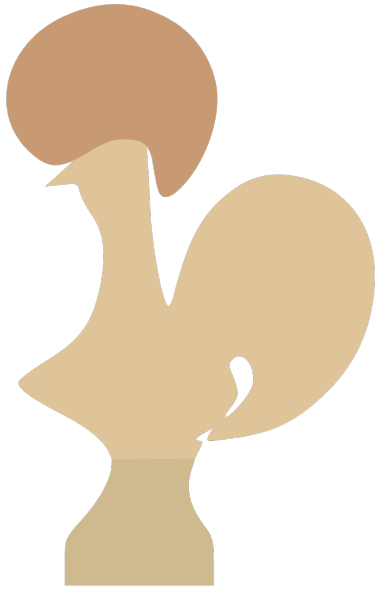
Coq Proof Assistant	LeanProver
Theorem	theorem
admit	sorry
reflexivity	rfl
rewrite H	rw [H]
rewrite <- H	rw [<- H]
simpl	simp
cdn	dsimp
auto	simp
discriminate	contradiction
destruct, case, elim	cases
;	<;>
-	\.
A|B	(frist|A|B)
subst	subst_vars
generalize dependent	revert
remember	have
split	apply And.intro
symmetry	apply Eq.symm

same
exact
apply
intros
assumption
unfold
contradiction
constructor
induction
repeat
try
refine
specialize

requires mathlib
left, right
ring
exists (use)
lia (linarith)



LEAN THEOREM PROVER

```
theorem plus_assoc:
  forall x y z: Nat,
  (x+y)+z = x+(y+z) := by
intros x y z
induction x with
| zero => simp
| succ x IH =>
  repeat rw [Nat.succ_add]
  rw [IH]
```

```
theorem add_comm:
  ∀ a b: Prop,
  a /\ b -> b /\ a := by
intros a b H
cases H with
| intro H1 H2 =>
  apply And.intro
  case left => exact H2
  case right => exact H1
```

```
theorem add_comm':
  ∀ a b: Prop,
  a /\ b -> b /\ a := by
intros a b H
have H1 := H.left
have H2: b := H.right
exact And.intro H2 H1
```

```
example (x y: Nat):
  succ x ≤ succ y
  <-> x ≤ y := by
apply Iff.intro
case mp =>
  apply ...
case mpr =>
  apply Nat.succ_le_succ
```

Authors: Paul Cadman, Gregor Feierabend, Walter Schulze