

Computer-certified Proofs

Module: MATH5004M

Student ID: 200684195

Supervisor: Dr Nicola Gambino

Ismail Suleman

Overview of Coq

What is Coq

- ▶ Functional programming language
- ▶ Dependent types
- ▶ Construct certified programs, prove theorems

Type hierarchy

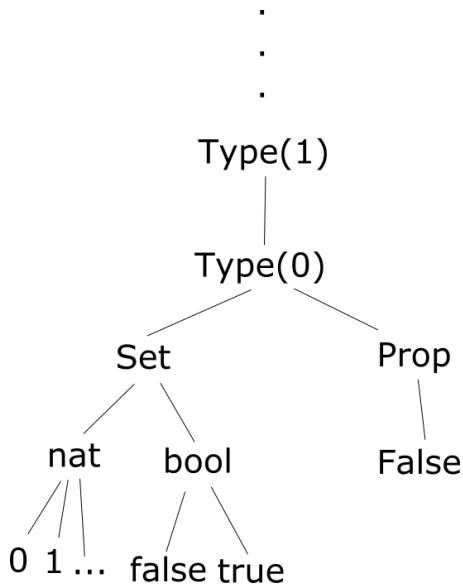


Figure 1: Hierarchy of types in Coq

Function types

- ▶ $\text{nat} \rightarrow \text{bool} \rightarrow \text{nat} \rightarrow \text{nat}$
- ▶ $\text{nat} \rightarrow (\text{bool} \rightarrow (\text{nat} \rightarrow \text{nat}))$

Dependent product

$$\frac{A: \mathbf{s} \quad a: A \vdash B: \mathbf{s}'}{\text{forall } a: A, B: \mathbf{s}''}$$

$$\frac{A: \mathbf{Set} \quad a: A \vdash B: \mathbf{Prop}}{\text{forall } a: A, B: \mathbf{Prop}}$$

Example code

```
(* Some example code *)
2 Section example_code.
  Variables (A: Set) (P Q: A → Prop) (R: A → A → Prop).
4
  Theorem all_perm: (forall a b: A, R a b) → forall a b: A, R b a.
6 Proof (fun H a b ⇒ H b a).

8 Theorem all_perm': (forall a b: A, R a b) → forall a b: A, R b a.
  Proof.
10   intros H a b.
    apply H.
12 Qed.
```

2

```
Inductive Z_inf_branch_tree: Set :=  
  Z_inf_leaf: Z_inf_branch_tree  
| Z_inf_node: Z → (nat → Z_inf_branch_tree) →  
  Z_inf_branch_tree.
```

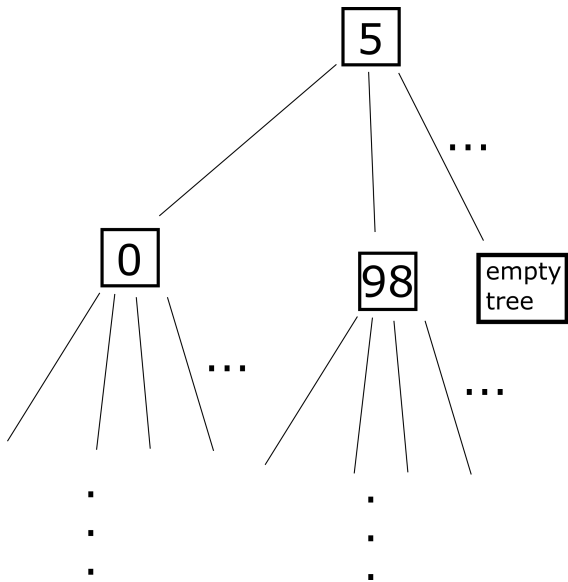



Figure 2: Graphical representation of $Z_{\text{inf_branch_tree}}$

Rewriting a subset of the standard library

Existing state of the standard library

- ▶ >40 folders, >400 files
- ▶ Code spread across files
- ▶ Large dependency tree
- ▶ Comments

2

```
Inductive between k : nat → Prop :=  
  | bet_emp : between k k  
  | bet_S : forall l, between k l → P l → between k (S l).
```

Rewriting a subset of the standard library

```
1 Theorem plus_comm : forall n m : nat, n + m = m + n. Admitted.
Theorem plus_assoc : forall n m p : nat, n + (m + p) = (n + m) +
  p. Admitted.
3
Theorem plus_swap : forall n m p : nat, n + (m + p) = m + (n +
  p).
5 Proof.
  intros n m p.
7  assert (H: n + (m + p) = n + m + p).
  {rewrite → plus_assoc. reflexivity. }
9  assert (H1: n + m = m + n).
  {rewrite → plus_comm. reflexivity. }
11 rewrite → H. rewrite → H1. rewrite → plus_assoc.
    reflexivity.
Qed.
```

```
1 subgoal
2 n, m, p : nat
  H : n + (m + p) = n + m + p
4 H1 : n + m = m + n
-----(1/1)
6 n + (m + p) = m + (n + p)
```

`rewrite` \rightarrow H.

```
1 subgoal
2 n, m, p : nat
  H : n + (m + p) = n + m + p
4 H1 : n + m = m + n
----- (1/1)
6 n + m + p = m + (n + p)
```

```
rewrite → H1.
```

```
1 subgoal
2 n, m, p : nat
  H : n + (m + p) = n + m + p
4 H1 : n + m = m + n
----- (1/1)
6 m + n + p = m + (n + p)
```



```
rewrite → plus_assoc.
```

```
1 subgoal
2 n, m, p : nat
  H : n + (m + p) = n + m + p
4 H1 : n + m = m + n
----- (1/1)
6 m + n + p = m + n + p
```

```
reflexivity. Qed.
```

Conclusion

- ▶ Coq is complicated, but powerful
- ▶ More rigorous proofs, but often less-readable