
5th Asian-Pacific Summer School on Formal Methods

August 5-10, 2013, Tsinghua University, Beijing, China

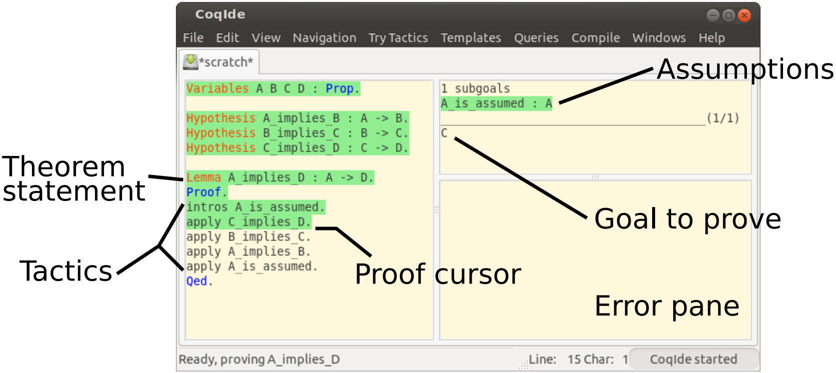
# Interactive Development of Proofs

Guillaume Melquiond

August 5th, 2013

http://sts.thss.tsinghua.edu.cn/Coqschool2013

# Coq Graphical Interface



# Coq Propositional Logic in a Nutshell

Type of formulas: `Prop`

Logic connectives:

- Implication: `A -> B`
  If $A$ is a provable formula, then $B$ is provable too.
- Conjunction: `A /\ B`
  Both $A$ and $B$ are provable formulas.
- Disjunction: `A \/ B`
  Either $A$ is a provable formula, or $B$ is. (Or both are.)
- Negation: `not A`
  If $A$ is provable, then any formula is provable.

Constants:

- `True`: trivially provable.
- `False`: if provable, anything is.

# A Few Words About Syntax

| Operators | Associativity | Prefix form |
|---|---|---|
| `not A` | | |
| `A /\ B` | right | `and A B` |
| `A \/ B` | right | `or A B` |
| `A -> B` | right | |

Example: `A /\ B /\ C -> (A /\B) /\ C`
represents $(A \wedge (B \wedge C)) \to ((A \wedge B) \wedge C)$.

# Some Top-level Commands

- `Variable name : type` defines a new symbol of given type.
  Synonyms: `Hypothesis`, `Axiom`, `Parameter`.

- `Theorem name : formula` starts the proof of a formula.
  It is given a name for later reuse, once the proof is complete.
  Synonyms: `Lemma`, `Corollary`, `Example`.

- `Qed` checks that a proof is complete and saves it.

- `Goal formula` starts the proof of a formula.
  Same as `Theorem`, except that it cannot be reused later.

- `Definition name : type := value` defines a new constant (or function) with the given type and value.

# Forward and Backward Reasoning

If the formulas $A \to B$, $B \to C$, $C \to D$ have been proved beforehand, how does one prove $A \to D$ ?

- Forward reasoning: assuming $A$ is provable, prove $D$.
  1. Deduce $B$ from $A$ and the lemma proving $A \to B$.
  2. Deduce $C$ from $B$ and the lemma proving $B \to C$.
  3. Deduce $D$ from $C$ and the lemma proving $C \to D$.
  4. We are done, since we wanted to prove $D$.
- Backward reasoning: assuming $A$ is provable, prove $D$.
  1. Prove $C$ instead, by applying the lemma proving $C \to D$.
  2. Prove $B$ instead, by applying the lemma proving $B \to C$.
  3. Prove $A$ instead, by applying the lemma proving $A \to B$.
  4. We are done, since we have assumed $A$ and have to prove $A$.

Notes:

- Forward reasoning and backward reasoning are not exclusive.
- Backward reasoning is generally easier in Coq.

# Coq Script

```
Variables A B C D : Prop.

Hypothesis A_implies_B : A -> B.
Hypothesis B_implies_C : B -> C.
Hypothesis C_implies_D : C -> D.

Lemma A_implies_D : A -> D.
Proof.
intros A_is_assumed.
apply C_implies_D.
apply B_implies_C.
apply A_implies_B.
apply A_is_assumed.
Qed.
```

# Handling Implications

Remember: $A \rightarrow B$ means that $B$ is provable if $A$ is.

If the goal is $F_1 \rightarrow F_2 \rightarrow \ldots \rightarrow F_n \rightarrow G$,
tactic `intros h1 h2 ...  hn` performs the following steps:

1. It assumes that $F_1, \ldots, F_n$ are provable, and puts the corresponding lemmas named `h1`, ..., `hn` in the context.
2. It replaces the goal by $G$.

# Applying Lemmas and Hypotheses

If the goal is $G$ and
if lemma `L` proves $F_1 \to F_2 \to \ldots \to F_n \to G$,
tactic `apply L` performs the following steps:

1. It replaces the current goal by $F_1$.
2. It creates $n - 1$ new goals that require to prove $F_2, \ldots, F_n$.

If lemma `L` simply proves $G$, the current goal is removed
and the next one takes its place.

# Proving Conjunctions in Goal

Remember: $A \wedge B$ means that both $A$ and $B$ are provable.

If the goal is $G_1 \wedge G_2$,
tactic `split` replaces the current goal by $G_1$
and adds a new goal $G_2$.

Variant: lemma `conj` proves $A \to B \to A \wedge B$ for any formulas $A$ and $B$, so `apply conj` has the same effect.

# Using Conjunctions in Context

If an assumption `h` proves a conjunction $F_1 \wedge F_2$,
tactic `destruct h as [h1 h2]` performs the following steps:

1. It removes the assumption `h` from the context.
2. It introduces two new assumptions `h1` and `h2` that prove $F_1$ and $F_2$ respectively.

Variant: the two tactics `intros h ; destruct h as [h1 h2]` can be written `intros [h1 h2]` for short.

# Coq Script

```
Variables A B : Prop.

Lemma and_comm : A /\ B -> B /\ A.
Proof.
intros hAB.
destruct hAB as [hA hB].
split.
- apply hB.
- apply hA.
Qed.
```

# Proving Disjunctions in Goal

Remember: $A \vee B$ means that $A$ or $B$ is provable.

If the goal is $G_1 \vee G_2$, tactic `left` replaces it with $G_1$.
Similarly, tactic `right` replaces the goal with $G_2$.

Variants: lemma `or_introl` (resp. `or_intror`) proves $A \to A \vee B$ (resp. $B \to A \vee B$) for any formulas $A$ and $B$, so tactic `apply or_introl` (resp. `apply or_intror`) has the same effect.

# Using Disjunctions in Context

If an assumption `h` proves a disjunction $F_1 \vee F_2$,
tactic `destruct h as [h1|h2]` performs the following steps:

1. It removes the assumption `h` from the context.
2. It introduces an assumption `h1` that proves $F_1$.
3. It creates a second goal with an assumption `h2` that proves $F_2$.

Variant: the two tactics `intros h ; destruct h as [h1|h2]` can be written `intros [h1|h2]` for short.

# Handling Negations and Constants

- `not A` is syntactic sugar for $A \to False$.
  It can thus be handled like any implication.
- Tactic `apply I` proves the constant goal *True*.
- Since *False* implies any formula,
  the current goal can be replaced by *False* with tactic `exfalso`.
  Variant: `apply False_ind`.

# Coq Script

```
Variables A B : Prop.

Lemma not_not : A -> not (not A).
Proof.
intros hA hnotA.
apply hnotA.
apply hA.
Qed.

Lemma excluded_middle : A /\ not A -> B.
Proof.
intros [hA hnotA].
exfalso.
apply hnotA.
apply hA.
Qed.
```

# Forward Reasoning

Assuming the current goal is formula $G$,
tactic `assert (h : F)` performs the following steps:

1. It replaces the current goal with formula $F$.
2. It creates a new goal $G$ and adds to its context the assumption that there is a proof of $F$ called `h`.

# Forward Reasoning

If hypothesis `h` proves formula $F$ and
if lemma `L` proves $F \rightarrow G$,
tactic `apply L in h` changes `h` so that it proves $G$.

# If and Only If

Formula `A <-> B` is syntactic sugar for $(A \to B) \wedge (B \to A)$.
As a goal, it can thus be handled like any conjunction.

If lemma `L` proves $A \leftrightarrow B$, tactic `apply -> L` behaves as

```
assert (h : A <-> B).
apply L.
...
destruct h as [AtoB BtoA].
apply AtoB.
```

that is, it proves a goal $A \to B$.

Tactic `apply <- L` proves $B \to A$.

# Some Other Low-Level Tactics

- Tactic `clear h` removes an assumption named `h` from the context.
- Tactic `revert h` performs the opposite of `intros h`:
  1. Assumption `h` of a proof of formula $F$ is removed from the context.
  2. The current goal is changed from $G$ to $F \to G$.
- Tactic `generalize h` is the same as `revert h`, except that it does not remove the assumption from the context.

# Quantifying Over Formulas

Formula `forall X:Prop, F` means that formula $F$ is provable whichever formula is substituted to the free occurrences of $X$ in $F$.

Example: lemma `conj` (cf tactic `split`) is actually

$$\forall A\ B : Prop,\ A \to B \to A \wedge B.$$

If the current goal is $\forall X : Prop, F$,
tactic `intros A` performs the following steps:

1. It introduces an arbitrary formula named $A$ in the context.
2. It replaces the current goal with $F$, in which all the free occurrences of $X$ have been substituted by $A$.

# Coq Script

```
Lemma and_comm :
  forall A B : Prop, A /\ B -> B /\ A.
Proof.
intros A B [hA hB].
split.
- apply hB.
- apply hA.
Qed.

Goal True /\ False.
Proof.
apply and_comm.
```

# First-Order Logic: Types, Values, and Quantified Formulas

We now introduce types (e.g. `bool`, `nat`) and typed values (e.g. `true`, `0`, `1`).

If $P$ is a predicate of type $T \rightarrow Prop$ and $x$ is a value of type $T$, then $P\ x$ is a formula. Also valid for higher arity.

Formula `forall x:T, F` means that formula $F$ is provable whichever value of type $T$ is substituted to the free occurrences of $x$ in $F$.

# Handling Universally-Quantified Formulas

If the goal is a formula $\forall x : T,\ P$,
tactic `intros a` performs the following steps:

1. It adds a new value $a$ of type $T$ in the context.
2. It replaces the goal with the formula $P$ in which all the free occurrences of $x$ have been replaced by $a$.

If lemma `L` proves a formula $\forall x : T,\ F_1 \rightarrow \ldots \rightarrow F_n \rightarrow P$,
tactic `apply L` performs the following steps:

1. It searches a value $v$ such that the current goal is $P$ with all the occurrences of $x$ replaced by $v$.
2. It creates new goals for all the hypotheses $F_i$ after replacing all their occurrences of $x$ with $v$.

# Proving Equalities in Goal

Relation `eq` has type $T \to T \to \mathit{Prop}$ for any type $T$.
$x = y$ is syntactic sugar for `eq x y`.

Tactic `reflexivity` proves a goal $v = v$.

Variant: lemma `eq_refl` proves $\forall x : T,\ x = x$,
so tactic `apply eq_refl` has the same effect.

# Using Equalities in Context

Given a lemma `L` proving the formula $F_1 \to \ldots \to F_n \to x = y$, tactic `rewrite L` performs the following steps:

1. It substitutes all the occurrences of expression $x$ in the current goal with expression $y$.
2. It creates $n$ additional goals $F_1, \ldots, F_n$.

Variants:

- Tactic `rewrite <- L` replaces all the occurrences of $y$ in the current goal by $x$.
- Tactic `rewrite L at 1 3 4` replaces some specific occurrences of $x$.
- Tactic `rewrite L in h` replaces all the occurrences of $x$ in assumption `h`.

# Coq Script

```
Variable T : Type.

Lemma eq_sym :
  forall x y : T, x = y -> y = x.
Proof.
intros x y heq.
rewrite heq.
apply eq_refl.
Qed.
```

# Existential Quantifiers

Formula `exists x:T, F` means that there exists a value $v$ of type $T$ such that $F$ is provable when all the free occurrences of $x$ are substituted by $v$.

If the goal is $\exists x : T,\ F$, tactic `exists v` replaces it with formula $F$ in which all the occurrences of $x$ are substituted by $v$.

If an assumption `h` proves $\exists x : T,\ F$,
tactic `destruct h as [v hv]` performs the following steps:

1. It removes `h` from the context.
2. It introduces a value of type $T$ named `v` in the context.
3. It introduces a proof named `hv` of formula $F$ in which all the free occurrences of $x$ are substituted by `v`.

# Coq Script

```
Variable T : Type.
Variables P Q : T -> Prop.

Lemma exists_and :
  (exists z:T, (P z /\ Q z)) ->
  (exists x:T, P x) /\ (exists y:T, P y).
Proof.
intros h.
destruct h as [z hz].
destruct hz as [Pz Qz].
split.
- exists z.
  apply Pz.
- exists z.
  apply Pz.
Qed.
```

# Some Vernacular Commands

- `Check L` displays the type of `L`.
  If `L` is a theorem, it displays its statement.
- `Print t` displays the value of `t`.
- `SearchAbout n` displays all the theorems that mention `n`.
- `SearchPattern F` displays all the theorems that prove `F`.
  Note: placeholders _ are allowed in `F`.
- `SearchRewrite t` displays all the theorems that prove either `t = _` or `_ = t`.