Coq Proof Assistant: Propositions and Proofs

Mirco Kocher

Logic and Theory Group
Institute of Computer Science and Applied Mathematics
Universität Bern

2012

Overview

Overview

Overview

Overview

Overview

Overview

Overview

Overview

Truth table

$(P \rightarrow Q)$

- Classical logic
- Assign to every variable a denotation true or false
- Formula is valid iff true in all cases
- Question "is the proposition P true?"

Truth table

$(P \to Q)$

- Classical logic
- Assign to every variable a denotation true or false
- Formula is valid iff true in all cases
- Question "is the proposition P true?"

Truth table

$(P \rightarrow Q)$

- Classical logic
- Assign to every variable a denotation true or false
- Formula is valid iff true in all cases
- Question "is the proposition P true?"

Truth table

$(P \to Q)$

- Classical logic
- Assign to every variable a denotation true or false
- Formula is valid iff true in all cases
- Question "is the proposition P true?"

Coq system

$(P \to Q)$

- Intuitionistic logic
- Obtain a proof of Q from a proof of P
- Arbitrary proof of P constructs a proof of Q
- Question "what are the proof of P (if any)?"

Coq system

$(P \to Q)$

- Intuitionistic logic
- Obtain a proof of Q from a proof of P
- Arbitrary proof of P constructs a proof of Q
- Question "what are the proof of P (if any)?"

Coq system

$(P \to Q)$

- Intuitionistic logic
- Obtain a proof of Q from a proof of P
- Arbitrary proof of P constructs a proof of Q
- Question "what are the proof of P (if any)?"

Coq system

$(P \to Q)$

- Intuitionistic logic
- Obtain a proof of Q from a proof of P
- Arbitrary proof of P constructs a proof of Q
- Question "what are the proof of P (if any)?"

Hypothesis

`Hypothesis h:P`

- Local declaration
- *h* is the name of the hypothesis
- P is its statement
- Synonymous to
  `Variable h:P`
- Use
  `Hypotheses`
  or
  `Variables`
  to declare several at a time

Hypothesis

`Hypothesis h:P`

- Local declaration
- *h* is the name of the hypothesis
- P is its statement
- Synonymous to
  `Variable h:P`
- Use
  `Hypotheses`
  or
  `Variables`
  to declare several at a time

Hypothesis

`Hypothesis h:P`

- Local declaration
- *h* is the name of the hypothesis
- P is its statement
- Synonymous to
  `Variable h:P`
- Use
  `Hypotheses`
  or
  `Variables`
  to declare several at a time

Hypothesis

`Hypothesis h:P`

- Local declaration
- *h* is the name of the hypothesis
- P is its statement
- Synonymous to

  `Variable h:P`
- Use

  `Hypotheses`

  or

  `Variables`

  to declare several at a time

Hypothesis

`Hypothesis h:P`

- Local declaration
- *h* is the name of the hypothesis
- P is its statement
- Synonymous to

  `Variable h:P`
- Use

  `Hypotheses`

  or

  `Variables`

  to declare several at a time

Section

The section contains all Hypoteses / Variables from the Context

Start section sec1 with

```
Section sec1
```

End section sec1 with

```
End sec1
```

Section

The section contains all Hypoteses / Variables from the Context

Start section sec1 with

```
Section sec1
```

End section sec1 with

```
End sec1
```

Section

The section contains all Hypoteses / Variables from the Context

Start section sec1 with

```
Section sec1
```

End section sec1 with

```
End sec1
```

Axiom

`Axiom x:P`

- Global declaration
- Synonymous to
  `Parameter x:P`

Environment contains axioms

Context contains hypotheses

$E, \Gamma \vdash \pi : P$

Axiom

`Axiom x:P`

- Global declaration
- Synonymous to
  `Parameter x:P`

Environment contains axioms

Context contains hypotheses

$E, \Gamma \vdash \pi : P$

Axiom

`Axiom x:P`

- Global declaration
- Synonymous to
  `Parameter x:P`

Environment contains axioms

Context contains hypotheses

$E, \Gamma \vdash \pi : P$

Axiom

```
Axiom x:P
```

- Global declaration
- Synonymous to
  ```
  Parameter x:P
  ```

Environment contains axioms

Context contains hypotheses

$E, \Gamma \vdash \pi : P$

Axiom

`Axiom x:P`

- Global declaration
- Synonymous to
  `Parameter x:P`

Environment contains axioms

Context contains hypotheses

$E, \Gamma \vdash \pi : P$

Goals and Tactics

Goals: what needs to be proven

Goal: $E, \Gamma \vdash P$
Construct a proof of P. Should be a well-formed term t in the environment E and context Γ
Term t is called a *solution*

Tactics: commands to decompose this goal into simpler goals

g ist input goal and g_1, g_2, ..., g_k are output goals
Possible to construct a solution of g from the solutions of goals g_i

Goals and Tactics

Goals: what needs to be proven

Goal: $E, \Gamma \vdash P$
Construct a proof of P. Should be a well-formed term t in the environment E and context Γ
Term t is called a *solution*

Tactics: commands to decompose this goal into simpler goals

g ist input goal and g_1, g_2, ..., g_k are output goals
Possible to construct a solution of g from the solutions of goals g_i

Goals and Tactics

Goals: what needs to be proven

Goal: $E, \Gamma \vdash P$
Construct a proof of P. Should be a well-formed term t in the environment E and context Γ
Term t is called a *solution*

Tactics: commands to decompose this goal into simpler goals

g ist input goal and g_1, g_2, ..., g_k are output goals
Possible to construct a solution of g from the solutions of goals g_i

Goals and Tactics

Goals: what needs to be proven

Goal: $E, \Gamma \vdash P$
Construct a proof of P. Should be a well-formed term t in the environment E and context Γ
Term t is called a *solution*

Tactics: commands to decompose this goal into simpler goals

g ist input goal and g_1, g_2, ..., g_k are output goals
Possible to construct a solution of g from the solutions of goals g_i

Goals and Tactics

Goals: what needs to be proven

Goal: $E, \Gamma \vdash P$
Construct a proof of P. Should be a well-formed term t in the environment E and context Γ
Term t is called a *solution*

Tactics: commands to decompose this goal into simpler goals

g ist input goal and g_1, g_2, ..., g_k are output goals
Possible to construct a solution of g from the solutions of goals g_i

Goals and Tactics

Goals: what needs to be proven

Goal: $E, \Gamma \vdash P$
Construct a proof of P. Should be a well-formed term t in the environment E and context Γ
Term t is called a *solution*

Tactics: commands to decompose this goal into simpler goals

g ist input goal and g_1, g_2, ..., g_k are output goals
Possible to construct a solution of g from the solutions of goals g_i

Goals and Tactics

Goals: what needs to be proven

Goal: $E, \Gamma \vdash P$
Construct a proof of P. Should be a well-formed term t in the environment E and context Γ
Term t is called a *solution*

Tactics: commands to decompose this goal into simpler goals

g ist input goal and g_1, g_2, ..., g_k are output goals
Possible to construct a solution of g from the solutions of goals g_i

intros

Goal: $(P \to Q) \to (Q \to R) \to (P \to R)$

`intros H H' p`

- Transform the task of construction a proof into proving R with those hypotheses added
- $H : P \to Q$
- $H' : Q \to R$ and
- $p : P$
- New subgoal: R

Simplifies the statement to prove and increases the resources available

intros

Goal: $(P \to Q) \to (Q \to R) \to (P \to R)$

```
intros H H' p
```

- Transform the task of construction a proof into proving R with those hypotheses added
- $H : P \to Q$
- $H' : Q \to R$ and
- $p : P$
- New subgoal: R

Simplifies the statement to prove and increases the resources available

intros

Goal: $(P \to Q) \to (Q \to R) \to (P \to R)$

```
intros H H' p
```

- Transform the task of construction a proof into proving R with those hypotheses added
- $H : P \to Q$
- $H' : Q \to R$ and
- $p : P$
- New subgoal: R

Simplifies the statement to prove and increases the resources available

intros

Goal: $(P \to Q) \to (Q \to R) \to (P \to R)$

```
intros H H' p
```

- Transform the task of construction a proof into proving R with those hypotheses added
- $H : P \to Q$
- $H' : Q \to R$ and
- $p : P$
- New subgoal: R

Simplifies the statement to prove and increases the resources available

intros

Goal: $(P \to Q) \to (Q \to R) \to (P \to R)$

```
intros H H' p
```

- Transform the task of construction a proof into proving R with those hypotheses added
- $H : P \to Q$
- $H' : Q \to R$ and
- $p : P$
- New subgoal: R

Simplifies the statement to prove and increases the resources available

intros

Goal: $(P \to Q) \to (Q \to R) \to (P \to R)$

```
intros H H' p
```

- Transform the task of construction a proof into proving R with those hypotheses added
- $H : P \to Q$
- $H' : Q \to R$ and
- $p : P$
- New subgoal: R

Simplifies the statement to prove and increases the resources available

intros

Goal: $(P \to Q) \to (Q \to R) \to (P \to R)$

```
intros H H' p
```

- Transform the task of construction a proof into proving R with those hypotheses added
- $H : P \to Q$
- $H' : Q \to R$ and
- $p : P$
- New subgoal: R

Simplifies the statement to prove and increases the resources available

intros

Goal: $(P \to Q) \to (Q \to R) \to (P \to R)$

```
intros H H' p
```

- Transform the task of construction a proof into proving R with those hypotheses added
- $H : P \to Q$
- $H' : Q \to R$ and
- $p : P$
- New subgoal: R

Simplifies the statement to prove and increases the resources available

apply

Subgoal: R

Hypothesis $H' : Q \to R$ and $H : P \to Q$

```
apply H'
```

- Use the hypothesis H' to advance our proof
- Argument has to be a premise and a conclusion
- Creates new goal for the premise
- New subgoal: Q

Applying hypothesis H gives the new subgoal P

apply

Subgoal: R

Hypothesis $H' : Q \rightarrow R$ and $H : P \rightarrow Q$

`apply H'`

- Use the hypothesis H' to advance our proof
- Argument has to be a premise and a conclusion
- Creates new goal for the premise
- New subgoal: Q

Applying hypothesis H gives the new subgoal P

apply

Subgoal: R

Hypothesis $H' : Q \to R$ and $H : P \to Q$

```
apply H'
```

- Use the hypothesis H' to advance our proof
- Argument has to be a premise and a conclusion
- Creates new goal for the premise
- New subgoal: Q

Applying hypothesis H gives the new subgoal P

apply

Subgoal: R

Hypothesis $H' : Q \to R$ and $H : P \to Q$

`apply H’`

- Use the hypothesis H' to advance our proof
- Argument has to be a premise and a conclusion
- Creates new goal for the premise
- New subgoal: Q

Applying hypothesis H gives the new subgoal P

apply

Subgoal: R

Hypothesis $H' : Q \to R$ and $H : P \to Q$

`apply H'`

- Use the hypothesis H' to advance our proof
- Argument has to be a premise and a conclusion
- Creates new goal for the premise
- New subgoal: Q

Applying hypothesis H gives the new subgoal P

apply

Subgoal: R

Hypothesis $H' : Q \to R$ and $H : P \to Q$

`apply H'`

- Use the hypothesis H' to advance our proof
- Argument has to be a premise and a conclusion
- Creates new goal for the premise
- New subgoal: Q

Applying hypothesis H gives the new subgoal P

assumption

Subgoal: P

Hypothesis $p : P$

`assumption`

- Statement to proof is exactly statement of hypothesis p
- Succeeds without generating any new goal

`No more subgoals`

Proof complete

assumption

Subgoal: P

Hypothesis $p : P$

`assumption`

- Statement to proof is exactly statement of hypothesis p
- Succeeds without generating any new goal

`No more subgoals`

Proof complete

assumption

Subgoal: P

Hypothesis $p : P$

`assumption`

- Statement to proof is exactly statement of hypothesis p
- Succeeds without generating any new goal

`No more subgoals`

Proof complete

assumption

Subgoal: P

Hypothesis $p : P$

`assumption`

- Statement to proof is exactly statement of hypothesis p
- Succeeds without generating any new goal

`No more subgoals`

Proof complete

assumption

Subgoal: P

Hypothesis $p : P$

`assumption`

- Statement to proof is exactly statement of hypothesis p
- Succeeds without generating any new goal

`No more subgoals`

Proof complete

Finish

```
Qed
```

- Saves the theorem's name, statement and proof term
- Displays the sequence of tactics.

```
intros H H' p.
apply H'.
apply H.
assumption.
```

```
Print theorem-name
```

- Shows the proof like any *Gallina* definition

```
theorem-name = fun (H:P -> Q)(H':Q -> R)(p:P) => H' (H p)
 : (P -> Q) -> (Q -> R) -> P -> R
```

Finish

`Qed`

- Saves the theorem's name, statement and proof term
- Displays the sequence of tactics.

```
intros H H' p.
apply H'.
apply H.
assumption.
```

`Print theorem-name`

- Shows the proof like any *Gallina* definition

```
theorem-name = fun (H:P -> Q)(H':Q -> R)(p:P) => H' (H p)
 : (P -> Q) -> (Q -> R) -> P -> R
```

Finish

```
Qed
```

- Saves the theorem's name, statement and proof term
- Displays the sequence of tactics.

```
intros H H' p.
apply H'.
apply H.
assumption.
```

```
Print theorem-name
```

- Shows the proof like any *Gallina* definition

```
theorem-name = fun (H:P -> Q)(H':Q -> R)(p:P) => H' (H p)
  : (P -> Q) -> (Q -> R) -> P -> R
```

Finish

```
Qed
```

- Saves the theorem's name, statement and proof term
- Displays the sequence of tactics.

```
intros H H' p.
apply H'.
apply H.
assumption.
```

```
Print theorem-name
```

- Shows the proof like any *Gallina* definition

```
theorem-name = fun (H:P -> Q)(H':Q -> R)(p:P) => H' (H p)
 : (P -> Q) -> (Q -> R) -> P -> R
```

Finish

```
Qed
```

- Saves the theorem's name, statement and proof term
- Displays the sequence of tactics.

```
intros H H' p.
apply H'.
apply H.
assumption.
```

```
Print theorem-name
```

- Shows the proof like any *Gallina* definition

```
theorem-name = fun (H:P -> Q)(H':Q -> R)(p:P) => H' (H p)
 : (P -> Q) -> (Q -> R) -> P -> R
```

Transitivity

$$(P \to Q) \to (Q \to R) \to (P \to R)$$

One Shot

Not all the details for the proof is needed

A few automatic tactics are able to solve this goal

```
Theorem transitivity : (P->Q) -> (Q->R) -> P -> R.
Proof.

auto.
Qed.
```

Good usage requires the command Proof after Theorem or Lemma

Makes the proof documents more readable

One Shot

Not all the details for the proof is needed

A few automatic tactics are able to solve this goal

```
Theorem transitivity : (P->Q) -> (Q->R) -> P -> R.
Proof.

auto.
Qed.
```

Good usage requires the command Proof after Theorem or Lemma

Makes the proof documents more readable

One Shot

Not all the details for the proof is needed

A few automatic tactics are able to solve this goal

```
Theorem transitivity : (P->Q) -> (Q->R) -> P -> R.
Proof.
```

```
auto.
Qed.
```

Good usage requires the command Proof after Theorem or Lemma

Makes the proof documents more readable

One Shot

Not all the details for the proof is needed

A few automatic tactics are able to solve this goal

```
Theorem transitivity : (P->Q) -> (Q->R) -> P -> R.
Proof.

auto.
Qed.
```

Good usage requires the command Proof after Theorem or Lemma

Makes the proof documents more readable

One Shot

Not all the details for the proof is needed

A few automatic tactics are able to solve this goal

```
Theorem transitivity : (P->Q) -> (Q->R) -> P -> R.
Proof.

auto.
Qed.
```

Good usage requires the command Proof after Theorem or Lemma

Makes the proof documents more readable

exact

The exact tactic takes any term (with right type) as argument

The whole proof could be given as an argument

```
Theorem delta : (P->P->Q) -> P -> Q.
Proof.
exact (fun (H:P->P->Q)(p:P) => H p p).
Qed.
```

Or even shorter:

```
Theorem delta : (P->P->Q) -> P -> Q.
Proof (fun (H:P->P->Q)(p:P) => H p p).
```

exact

The exact tactic takes any term (with right type) as argument

The whole proof could be given as an argument

```
Theorem delta : (P->P->Q) -> P -> Q.
Proof.
exact (fun (H:P->P->Q)(p:P) => H p p).
Qed.
```

Or even shorter:

```
Theorem delta : (P->P->Q) -> P -> Q.
Proof (fun (H:P->P->Q)(p:P) => H p p).
```

exact

The exact tactic takes any term (with right type) as argument

The whole proof could be given as an argument

```
Theorem delta : (P->P->Q) -> P -> Q.
Proof.
```

```
exact (fun (H:P->P->Q)(p:P) => H p p).
Qed.
```

Or even shorter:

```
Theorem delta : (P->P->Q) -> P -> Q.
Proof (fun (H:P->P->Q)(p:P) => H p p).
```

exact

The exact tactic takes any term (with right type) as argument

The whole proof could be given as an argument

```
Theorem delta : (P->P->Q) -> P -> Q.
Proof.
exact (fun (H:P->P->Q)(p:P) => H p p).
Qed.
```

Or even shorter:

```
Theorem delta : (P->P->Q) -> P -> Q.
Proof (fun (H:P->P->Q)(p:P) => H p p).
```

exact

The exact tactic takes any term (with right type) as argument

The whole proof could be given as an argument

```
Theorem delta : (P->P->Q) -> P -> Q.
Proof.
exact (fun (H:P->P->Q)(p:P) => H p p).
Qed.
```

Or even shorter:

```
Theorem delta : (P->P->Q) -> P -> Q.
Proof (fun (H:P->P->Q)(p:P) => H p p).
```

Modus Ponens

apply tactic uses the Modus Ponens

$$\textbf{App}\ \frac{E, \Gamma \vdash t : P \rightarrow Q \qquad E, \Gamma \vdash t' : P}{E, \Gamma \vdash tt' : Q}$$

Term t : $P_1 \rightarrow P_2 \rightarrow ... \rightarrow P_n \rightarrow Q$
Goal : $P_k \rightarrow P_{k+1} \rightarrow ... \rightarrow P_n \rightarrow Q$

`apply t`

generates k-1 goals with statements $P_1, ..., P_{k-1}$

if new goals have solution $t_1, t_2, ..., t_{k-1}$
solution is $t\ t_1\ t_2\ ...\ t_{k-1}$ for initial goal

Modus Ponens

apply tactic uses the Modus Ponens

$$\textbf{App}\ \frac{E,\Gamma \vdash t : P \rightarrow Q \qquad E,\Gamma \vdash t' : P}{E,\Gamma \vdash tt' : Q}$$

Term t : $P_1 \rightarrow P_2 \rightarrow ... \rightarrow P_n \rightarrow Q$
Goal : $P_k \rightarrow P_{k+1} \rightarrow ... \rightarrow P_n \rightarrow Q$

`apply t`

generates k-1 goals with statements $P_1, ..., P_{k-1}$

if new goals have solution $t_1, t_2, ..., t_{k-1}$
solution is $t\ \ t_1\ \ t_2\ \ ...\ \ t_{k-1}$ for initial goal

Modus Ponens

apply tactic uses the Modus Ponens

$$\textbf{App}\ \frac{E,\Gamma \vdash t : P \to Q \qquad E,\Gamma \vdash t' : P}{E,\Gamma \vdash tt' : Q}$$

Term t : $P_1 \to P_2 \to \ldots \to P_n \to Q$
Goal : $P_k \to P_{k+1} \to \ldots \to P_n \to Q$

`apply t`

generates k-1 goals with statements $P_1, \ldots, P_{k-1}$

if new goals have solution $t_1, t_2, \ldots, t_{k-1}$
solution is $t\ t_1\ t_2\ \ldots\ t_{k-1}$ for initial goal

Modus Ponens

apply tactic uses the Modus Ponens

$$\textbf{App}\ \frac{E, \Gamma \vdash t : P \to Q \qquad E, \Gamma \vdash t' : P}{E, \Gamma \vdash tt' : Q}$$

Term t : $P_1 \to P_2 \to ... \to P_n \to Q$
Goal : $P_k \to P_{k+1} \to ... \to P_n \to Q$

```
apply t
```

generates k-1 goals with statements $P_1, ..., P_{k-1}$

if new goals have solution $t_1, t_2, ..., t_{k-1}$
solution is $t\ \ t_1\ \ t_2\ \ ...\ \ t_{k-1}$ for initial goal

Modus Ponens

apply tactic uses the Modus Ponens

$$\textbf{App}\ \frac{E, \Gamma \vdash t : P \rightarrow Q \qquad E, \Gamma \vdash t' : P}{E, \Gamma \vdash tt' : Q}$$

Term t : $P_1 \rightarrow P_2 \rightarrow ... \rightarrow P_n \rightarrow Q$
Goal : $P_k \rightarrow P_{k+1} \rightarrow ... \rightarrow P_n \rightarrow Q$

`apply t`

generates k-1 goals with statements $P_1, ..., P_{k-1}$

if new goals have solution $t_1, t_2, ..., t_{k-1}$
solution is $t\ \ t_1\ \ t_2\ \ ...\ \ t_{k-1}$ for initial goal

intros

```
intros v_1, v_2, ..., v_n
```

is the same as

```
intro v_1, intro v_2, ..., intro v_n
```

intro v_1 takes the first implication as a hypothesis called v_1

Proof Theorem K : $P \to Q \to P$
Goal is $P \to Q \to P$

```
intro p.
```

Hypothesis $p : P$ and new Goal $Q \to P$

intros

```
intros v_1, v_2, ..., v_n
```

is the same as

```
intro v_1, intro v_2, ..., intro v_n
```

intro v_1 takes the first implication as a hypothesis called v_1

Proof Theorem K : $P \to Q \to P$
Goal is $P \to Q \to P$

```
intro p.
```

Hypothesis $p : P$ and new Goal $Q \to P$

intros

`intros v_1, v_2, ..., v_n`

is the same as

`intro v_1, intro v_2, ..., intro v_n`

intro v_1 takes the first implication as a hypothesis called v_1

Proof Theorem K : $P \to Q \to P$
Goal is $P \to Q \to P$

`intro p.`

Hypothesis $p : P$ and new Goal $Q \to P$

intros

```
intros v_1, v_2, ..., v_n
```

is the same as

```
intro v_1, intro v_2, ..., intro v_n
```

intro v_1 takes the first implication as a hypothesis called v_1

Proof Theorem K : $P \to Q \to P$
Goal is $P \to Q \to P$

```
intro p.
```

Hypothesis $p : P$ and new Goal $Q \to P$

intros

```
intros v_1, v_2, ..., v_n
```

is the same as

```
intro v_1, intro v_2, ..., intro v_n
```

intro v_1 takes the first implication as a hypothesis called v_1

Proof Theorem K : $P \to Q \to P$
Goal is $P \to Q \to P$

```
intro p.
```

Hypothesis $p : P$ and new Goal $Q \to P$

Handling

`Show i`

Display goal *i* with complete context
Coq displays the goals after each proof step

`Undo n`

Go back *n* steps and try an alternative if goal can not be solved

`Restart`

Go back to the beginning of the proof

`Abort`

Abandon the proof

Handling

`Show i`

Display goal *i* with complete context
Coq displays the goals after each proof step

`Undo n`

Go back *n* steps and try an alternative if goal can not be solved

`Restart`

Go back to the beginning of the proof

`Abort`

Abandon the proof

Handling

`Show i`

Display goal *i* with complete context
Coq displays the goals after each proof step

`Undo n`

Go back *n* steps and try an alternative if goal can not be solved

`Restart`

Go back to the beginning of the proof

`Abort`

Abandon the proof

Handling

`Show i`

Display goal *i* with complete context
Coq displays the goals after each proof step

`Undo n`

Go back *n* steps and try an alternative if goal can not be solved

`Restart`

Go back to the beginning of the proof

`Abort`

Abandon the proof