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

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

$(P \to Q)$

- 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

$(P \to Q)$

- 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?"

Coq system

$(P \rightarrow Q)$

- 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

$(P \rightarrow Q)$

- 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 $H' : Q \to R$ 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

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

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

Emphasis is everything

The following word is emphasized is a way that's clearly visible on a beamer. In case you want a stronger emphasis, it's **possible too**. Commands used for that are defined in preamble.tex, you can tweak the

visual style from one place.

Columns and paragraphs

Arranging it in columns is also a possibility.

Note that column width can be custom.

Inference trees

You can use bussproofs to display inference rules and derivations:

$$\dfrac{\top_1 \qquad \dfrac{\top_2}{\bot \vee \top_2}\,(\vee_2)}{\top_1 \wedge (\bot \vee \top_2)}\,(\wedge)$$

Note: it works like a stack.