Mathematics for Computer Scientists 2 (G52MC2)

L02 : Coq basics, propositional logic

Thorsten Altenkirch

School of Computer Science
University of Nottingham

October 1, 2009

Propositional logic

- **Proposition**: A statement which can be true or false.
- Coq: `P : Prop` means `P` is a proposition.
- Propositional variables: stand for any proposition; e.g. in Coq:

```
Variables P Q R : Prop.
```

- **Tautology**: A proposition containing propositional variables which is always true.
- Propositional constants `True`, `False`.
- Basic propositional **connectives**:

Name	Math	Coq	English
Implication	$P \to Q$	`P -> Q`	If P then Q
Conjunction	$P \land Q$	`P /\ Q`	P and Q
Disjunction	$P \lor Q$	`P \/ Q`	P or Q
Equivalence	$P \leftrightarrow Q$	`P <-> Q`	P if and only if Q
			P iff Q
Negation	$\neg P$	`~ P`	not P

Syntactic conventions

- $\to$ is right-associative, i.e.

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

- $\lor$, $\land$ bind stronger than $\to$ (and $\leftrightarrow$), i.e.

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

- $\land$ binds stronger than $\lor$:

$$P \lor Q \land R = P \lor (Q \land R)$$

- $\neg$ binds stronger that $\land$:

$$\neg P \land Q = (\neg P) \land Q$$

Coq basics

- Start a proof with `Lemma` or `Theorem`. Give it a name! E.g.
  `Lemma andCom :  P /\ Q -> Q /\ P`
- Coq displays the *proof state*. The user issues *tactics* until Coq says `Proof completed.`
- Finish with `Qed.`
  Leaves the proof state and saves the proof under the given name.
- Read `p : P` as `p` is a proof of `P`, e.g.
  `andCom : P /\ Q -> Q /\ P`

Coq proof state (example)

```
2 subgoals
  H : P /\ Q
  H1 : P
  H2 : Q
  ============================
   Q

subgoal 2 is:
 P
```

- Two subgoals: currently proving `Q`, when we are finished we prove `P`.
- Assumptions above `===...===`.
- Assumptions have names,
  e.g. `H`, `H1`, `H2`
- Current goal (e.g. `Q`) is below the line.
- If current Goal = one of the assumptions, use `exact`, e.g. `exact H2.`

Proof rules / basic tactics

- General pattern

$$\frac{\text{premise (what we need to show)}}{\text{conclusion (what we want to show)}}\ \text{name of the rule}$$

- Read proof rules from bottom to top!
- We write $\Gamma \vdash P$ for
  From the set of assumptions Γ *(Gamma), we can prove* P.
- The symbol $\vdash$ (turnstyle) replaces Coq's `===...===`
- Example:

$$\frac{H : P \in \Gamma}{\Gamma \vdash P}\ \texttt{exact H}$$

- Read $H : P \in \Gamma$ as $H : P$ occurs in Γ.

Rules for implication

$$\frac{\Gamma, H : P \vdash Q}{\Gamma \vdash P \to Q}\ \texttt{intro}\,\mathrm{H} \qquad \frac{\begin{array}{l} H : P \to Q \in \Gamma \\ \Gamma \vdash P \end{array}}{\Gamma \vdash Q}\ \texttt{apply}\,\mathrm{H}$$

- `intro`: to prove $P \to Q$, assume P and prove Q.
- `apply`: If we know $P \to Q$ then to prove Q it is enough to prove P.
- The actual behaviour of `apply` is more subtle!
- See examples in l01.v

Rules for conjunction

$$\frac{\Gamma \vdash P \qquad \Gamma \vdash Q}{\Gamma \vdash P \wedge Q}\,\texttt{split} \qquad \frac{H : P \wedge Q \in \Gamma \quad \Gamma \vdash P \to Q \to R}{\Gamma \vdash R}\,\texttt{elim}\,\mathrm{H}$$

- `split`: to prove $P \wedge Q$ prove P and then Q.
- `elim`: If we know $P \wedge Q$ then to prove R it is enough to prove $P \to Q \to R$.
- See examples in l02.v

Rules for disjunction

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q}\,\texttt{left} \qquad \frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q}\,\texttt{right} \qquad \frac{\begin{array}{c} H : P \vee Q \in \Gamma \\ \Gamma \vdash P \to R \quad \Gamma \vdash Q \to R \end{array}}{\Gamma \vdash R}\,\texttt{case}\,\mathrm{H}$$

- `left`: to prove $P \vee Q$ prove P.
- `right`: to prove $P \vee Q$ prove Q.
- `case`: If we know $P \vee Q$ then to prove R it is enough to prove $P \to R$ and $Q \to R$.
- See examples in l02.v

Rules for True and False

$$\frac{}{\Gamma \vdash \mathrm{True}}\ \texttt{split} \qquad \frac{H : \mathrm{False} \in \Gamma}{\Gamma \vdash R}\ \texttt{case}\ \mathbf{H}$$

- `split`: to prove True you need to prove nothing.
- `case`: if you know False you can prove anything.

Summary

connective	Introduction	Elimination
$P \to Q$	`intro(s)`	`apply` *Hyp*
$P \land Q$	`split`	`elim` *Hyp*
True	`split`	
$P \lor Q$	`left,right`	`case` *Hyp*
False		`case` *Hyp*

Defined connectives

negation

$$\neg P = P \to \text{False}$$

iff

$$P \leftrightarrow Q = (P \to Q) \wedge (Q \to P)$$