Mathematics for Computer Scientists 2 (G52MC2)

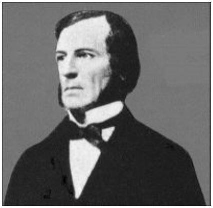
L05 : Bool and Predicate Logic

Thorsten Altenkirch

School of Computer Science
University of Nottingham

October 15, 2009

Introducing Booleans



Boole (1815-1864)

In Coq we define:

```
Inductive bool : Set :=
  | true : bool
  | false : bool.
```

`Inductive` is similar to Haskell's `data`:

```
data Bool = True | False
```

Operations on Booleans

$$
\begin{aligned}
\text{negb} &: \text{bool} \to \text{bool} \\
\text{negb}\ x &= \text{if } x \text{ then false else true} \\
\text{andb} &: \text{bool} \to \text{bool} \to \text{bool} \\
\text{andb}\ x\ y &= \text{if } x \text{ then } y \text{ else false} \\
\text{orb} &: \text{bool} \to \text{bool} \to \text{bool} \\
\text{orb}\ x\ y &= \text{if } x \text{ then true, else } y
\end{aligned}
$$

operation	infix
`andb`	`&&`
`orb`	`||`

$$
\begin{aligned}
\text{andb}'\ x\ y &= \text{if } y \text{ then } x \text{ else false} \\
\text{orb}'\ x\ y &= \text{if } y \text{ then true else } x
\end{aligned}
$$

- Do andb' (orb') define the same function as andb (orb)?

Predicate logic

- Predicate logic extents propositional logic.
- We consider predicate logic over the Booleans for now.
- Predicate logic consists of:

Sets E.g. $\mathrm{bool} : \mathbf{Set}$.

Terms e.g. $\mathrm{true}, \mathrm{false} : \mathrm{bool}$ and if-then-else.

Predicates and Relations e.g. equality
Given $t, u : A$ where $A : \mathbf{Set}$ we obtain
$t = u : \mathbf{Prop}$

Quantifiers

Name	Math	Coq	English
Universal quantifier	$\forall x : A, P$	`forall x:A,P`	for all
Existential quantifier	$\exists x : A, P$	`exists x:A,P`	exists

where $A : \mathbf{Set}$.

We can define new functions, predicates and relations using `Definition`, see `l04.v` for examples.

Syntax

- Quantifiers like $\forall x : A, P$ and $\exists x : A, P$ *bind* the variable x.
- The *scope* of the variable is only P.
- Variables can be *shadowed*, i.e. in the expression $\forall x : A, \forall x : B, P$ any occurence of x in P refers to $x : B$.
- Quantifiers bind weaker than any other connective

$$\forall x : A, P \to Q$$

is read as

$$\forall x : A, (P \to Q)$$

Rules for $\forall$

$$\frac{\Gamma, x : D \vdash P \qquad x \text{ does not occur free in } \Gamma.}{\Gamma \vdash \forall x : D, P}\ \texttt{intro}\ \mathrm{x}$$

$$\frac{\begin{array}{l} H : \forall x : D, P \in \Gamma \\ \Gamma \vdash d : D \end{array}}{\Gamma \vdash P[x := d]}\ \texttt{apply}\ \mathrm{H}$$

- `intro`: To prove $\forall x : D, P$ we assume $x : D$ and prove P.
- `apply`: To show $P[x := d]$ for $d : D$ it is enough, if we know $\forall x : D, P$.
- By $P[x := d]$ we mean that all *free* occurences of the variable x are replaced by the term d.

Rules for $\exists$

$$\frac{\Gamma \vdash d : D \quad \Gamma \vdash P[x := d]}{\Gamma \vdash \exists x : D, P}\ \mathtt{exists}\,\mathrm{d}$$

$$\frac{\begin{array}{c} H : \exists x : D, P \in \Gamma \\ \Gamma \vdash \forall x : D, P \to R \end{array}}{\Gamma \vdash R}\ \mathtt{elim}\,\mathrm{H}$$

- `exists`: To prove $\exists x : D, P$ it is enough to exhibit a term $d : D$ (the witness) and show $P[x := d]$.
- `elim`: To show R when we know $\exists x : D, P$ it is enough to show that P implies R for any $x : D$.

Rules for $=$

$$\frac{\Gamma \vdash d : D}{\Gamma \vdash d = d}\ \texttt{reflexivity}$$

$$\frac{\begin{array}{c} H : d = e \in \Gamma \\ \Gamma \vdash P[x := e] \end{array}}{\Gamma \vdash P[x := d]}\ \texttt{rewrite}\,\mathrm{H}$$

- `reflexivity`: For any $d : D$ we have $d = d$.
- `rewrite`: To show $P[x := d]$, if we know $d = e$ it is enough to show $P[x := e]$.
- There is also `rewrite<-` which applies the equation in the other direction.

bool vs **Prop**

Prop	$\wedge$	$\vee$	$\neg$	$\rightarrow$
bool	&&	||	`negb`	???

We can show:

$$\forall b\ c : \mathrm{bool}, b = \mathrm{true} \wedge c = \mathrm{true} \leftrightarrow b \&\& c = \mathrm{true}$$

and the same for $||$ and `negb`. See `l03.v`.

$\longrightarrow$ completeness

$\longleftarrow$ soundness

bool vs **Prop**

- We can also *reflect* $=$ (see ex03.v), there is a function $\texttt{eqb} : \mathrm{bool} \to \mathrm{bool} \to \mathrm{bool}$, s.t.

$$\forall b\ c : \mathrm{bool}, b = c \leftrightarrow \texttt{eqb}\ b\ c = \mathrm{true}$$

- We can even reflect quantifiers (see l03.v), there is a function $\texttt{allb} : (\mathrm{bool} \to \mathrm{bool}) \to \mathrm{bool}$, s.t.

$$\forall f : \mathrm{bool} \to \mathrm{bool}, (\forall b : \mathrm{bool}, f b = \mathrm{true}) \iff \texttt{forallb}\ f = \mathrm{true}$$

  This also works for $\exists$.
- As a consequence we can define a translation: given a P : **Prop** where P only uses bool then we have a translation $P^* : \mathrm{bool}$.
- Hence, predicate logic over bool is **decidable**.

Coq tricks

- To destruct an assumption `H : P ∧ Q`, use `destruct H as [HP HQ]`, which replaces the assumption `H` by `HP : P` and `HQ : Q`.
- To expand a definition `d` use `unfold d`, or `simpl` which expands and simplifies everything.
- If you have an assumption `H:A → False` and you want to prove any goal, you can just say `contradict H`.
- If you have an assumption like `H:true = false`, you can use `discriminate H` to 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*
forall $x : A, P$	`intro(s)`	`apply` *Hyp*
exists $x : A, P$	`exists` *wit*	`elim` *Hyp*
$a = b$	`reflexivity`	`rewrite` *Hyp*