Introducing Booleans

Boole (1815-1864)

In Coq we define:

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

Inductive is similar to Haskell’s `data`:

```haskell
data Bool = True | False
```
Operations on Booleans

\[
\begin{align*}
\text{negb} & : \text{bool} \rightarrow \text{bool} \\
\text{negb} \; x & = \text{if } x \text{then false else true} \\
\text{andb} & : \text{bool} \rightarrow \text{bool} \rightarrow \text{bool} \\
\text{andb} \; x \; y & = \text{if } x \text{then } y \text{ else false} \\
\text{orb} & : \text{bool} \rightarrow \text{bool} \rightarrow \text{bool} \\
\text{orb} \; x \; y & = \text{if } x \text{ then true, else } y
\end{align*}
\]

<table>
<thead>
<tr>
<th>operation</th>
<th>infix</th>
</tr>
</thead>
<tbody>
<tr>
<td>andb</td>
<td></td>
</tr>
<tr>
<td>orb</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{andb'} \; x \; y & = \text{if } y \text{ then } x \text{ else false} \\
\text{orb'} \; x \; y & = \text{if } y \text{ then true else } x
\end{align*}
\]

Do \text{andb'} (\text{orb'}) define the same function as \text{andb} (\text{orb})?
Predicate logic

- Predicate logic extents propositional logic.
- We consider predicate logic over the Booleans for now.
- Predicate logic consists of:
  - **Sets** E.g. bool : Set.
  - **Terms** e.g. true, false : bool and if-then-else.
  - **Predicates and Relations** e.g. equality
    
    Given $t, u : A$ where $A : \text{Set}$ we obtain
    
    $t = u : \text{Prop}$

### Quantifiers

<table>
<thead>
<tr>
<th>Name</th>
<th>Math</th>
<th>Coq</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal quantifier</td>
<td>$\forall x : A, P$</td>
<td>forall x:A,P</td>
<td>for all</td>
</tr>
<tr>
<td>Existential quantifier</td>
<td>$\exists x : A, P$</td>
<td>exists x:A,P</td>
<td>exists</td>
</tr>
<tr>
<td></td>
<td>where $A : \text{Set}$.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We can define new functions, predicates and relations using **Definition**, see `104.v` for examples.
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 occurrence of $x$ in $P$ refers to $x : B$.

Quantifiers bind weaker than any other connective

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

is read as

$$\forall x : A, (P \rightarrow Q)$$
Rules for $\forall$

$\Gamma, x : D \vdash P \quad \text{x does not occur free in } \Gamma.$

\[ \Gamma \vdash \forall x : D, P \]

\text{intro} \quad \text{x}

$H : \forall x : D, P \in \Gamma$

$\Gamma \vdash d : D$

\[ \Gamma \vdash P[x := d] \] \quad \text{apply} \quad 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 \textit{free} occurrences of the variable $x$ are replaced by the term $d$. 
Rules for $\exists$

\[
\Gamma \vdash d : D \quad \Gamma \vdash P[x := d]
\]

\[
\Gamma \vdash \exists x : D, P
\]

- **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 $x : D$. 

\[
H : \exists x : D, P \in \Gamma \\
\Gamma \vdash \forall x : D, P \rightarrow R \\
\Gamma \vdash R
\]

$\text{elim}H$
Rules for $\equiv$

\[ \Gamma \vdash d : D \]
\[ \Gamma \vdash d = d \quad \text{reflexivity} \]

\[ H : d = e \in \Gamma \]
\[ \Gamma \vdash P[x := e] \]
\[ \Gamma \vdash P[x := d] \quad \text{rewrite}_H \]

- **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$\leftarrow$** which applies the equation in the other direction.
We can show:

\[ \forall b \ c : \text{bool}, \ b = \text{true} \land c = \text{true} \iff b \&\& c = \text{true} \]

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

→ completeness

← soundness
We can also reflect = (see ex03.v), there is a function \( \text{eqb} : \text{bool} \rightarrow \text{bool} \rightarrow \text{bool} \), s.t.

\[
\forall b \ c : \text{bool}, b = c \iff \text{eqb} \ b \ c = \text{true}
\]

We can even reflect quantifiers (see l03.v), there is a function \( \text{allb} : (\text{bool} \rightarrow \text{bool}) \rightarrow \text{bool} \), s.t.

\[
\forall f : \text{bool} \rightarrow \text{bool}, (\forall b : \text{bool}, fb = \text{true}) \iff \text{forallb} \ f = \text{true}
\]

This also works for \( \exists \).

As a consequence we can define a translation: given a \( P : \text{Prop} \) where \( P \) only uses \( \text{bool} \) then we have a translation \( P^* : \text{bool} \).

Hence, predicate logic over \( \text{bool} \) is \textbf{decidable}.
Coq tricks

To destruct an assumption $H : P \land Q$, 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 $H:A \to False$ 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.
<table>
<thead>
<tr>
<th>connective</th>
<th>Introduction</th>
<th>Elimination</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \rightarrow Q$</td>
<td>intro(s)</td>
<td>apply $Hyp$</td>
</tr>
<tr>
<td>$P \land Q$</td>
<td>split</td>
<td>elim $Hyp$</td>
</tr>
<tr>
<td>True</td>
<td>split</td>
<td></td>
</tr>
<tr>
<td>$P \lor Q$</td>
<td>left,right</td>
<td>case $Hyp$</td>
</tr>
<tr>
<td>False</td>
<td></td>
<td>case $Hyp$</td>
</tr>
<tr>
<td>forall $x : A, P$</td>
<td>intro(s)</td>
<td>apply $Hyp$</td>
</tr>
<tr>
<td>exists $x : A, P$</td>
<td>exists $wit$</td>
<td>elim $Hyp$</td>
</tr>
<tr>
<td>$a = b$</td>
<td>reflexivity</td>
<td>rewrite $Hyp$</td>
</tr>
</tbody>
</table>