Induction

Pattern

- constant
- variable name
- wildcard
- tuple of patterns
- constructors applied to patterns

```
fun length([]: int list): int = 0
|   length(_:: L) = 1 + length L

fun fact(n: int): int =
    case n of
         0 => 1
      |  x => x * fact(x-1)

fun f(x: int, y: int) =
    case x+y of
         x => 0   // new binding
     |   _ => 1   // never reaches this case
(* REQ: n>=0
 * ENS: fact n = n!
 * )
fun FACT(0, acc) = acc
 |  FACT(n, acc) = FACT(n-1, acc*n)
```

THM: acc * fact(n) = FACT(n, acc) n: int

BC: $n = 0$.

$$\begin{aligned} acc * fact(0) &= acc * 1 && [\text{fact clause 1}] \\ &= acc && [\text{math}] \\ &= FACT(0, acc) && [\text{reverse step FACT clause 1}] \end{aligned}$$

IS: $n = k + 1$ for some k

IH: acc * fact(k) = FACT(k, acc)

$$
\begin{aligned}
acc * fact(n) &= \text{acc} * \text{fact}(k+1) && [\text{IS}] \\
&= \text{acc} * (k * \text{fact}(k+1-1)) && [\text{fact clause 2, k+1 valuable}] \\
&= \text{acc} * \text{fact}(k) * k && [\text{math}] \\
&= \text{FACT}(k, acc * (k+1)) && [\text{IH}] \\
&= \text{FACT}(k+1-1, acc * (k+1)) && [\text{reverse step FACT clause 2}]
\end{aligned}
$$

Note:

- some of the internal $=$s cannot be changed to $\Rightarrow^*$, i.e. $\Rightarrow^*$ is stronger than $=$ and the former is not stated in IH.
- “valuable” means evaluates to a value

Proving Techniques

- step
- reverse step
- IH
- Lemma given