Structural Induction

To prove a property $\phi(l)$ for all lists l, it suffices to show:

- base case: $\phi([])$
- induction step: for any h and t, if $\phi(t)$, then $\phi(h :: t)$

$\phi([x_1, \ldots, x_n])$ can be proved in n steps.

Similarly for other recursive datatypes.

Some List Functions

```
fun app([], l)    = l
  | app(h::t, l) = h::app(t, l);
```

```
fun nrev []      = []
  | nrev (h::t) = (nrev t)@[h];
```

```
fun revto ([], l)    = l
  | revto (h::t, l) = revto (t,h::l);
```

We can prove properties such as:

$$\mathsf{app}(\mathsf{app}(l1, l2), l3) = \mathsf{app}(l1, \mathsf{app}(l2, l3))$$

$$\texttt{revto}(l1, l2) = \texttt{nrev}(l1)@l2$$

Append is Associative

For all $l1$, $l2$ and $l3$:

$$\mathsf{app}(\mathsf{app}(l1, l2), l3) = \mathsf{app}(l1, \mathsf{app}(l2, l3))$$

By structural induction on $l1$.

Base case:

$$\begin{aligned}
\mathsf{app}(\mathsf{app}([], l2), l3) &= \mathsf{app}(l2, l3) \\
&= \mathsf{app}([], \mathsf{app}(l2, l3))
\end{aligned}$$

Induction Step:

$$\begin{aligned}
\mathsf{app}(\mathsf{app}(h :: t, l2), l3) &= \mathsf{app}(h :: \mathsf{app}(t, l2), l3) \\
&= h :: \mathsf{app}(\mathsf{app}(t, l2), l3) \\
&= h :: \mathsf{app}(t, \mathsf{app}(l2, l3)) \\
&= \mathsf{app}(h :: t, \mathsf{app}(l2, l3))
\end{aligned}$$

Reverse

$$\forall\, l2\ \texttt{revto}(l1, l2) = \texttt{nrev}(l1)@l2$$

By structural induction on $l1$.

Base case:

$$\texttt{revto}([], l2) = l2 = []@l2 = \texttt{nrev}[]@l2$$

Induction Step:

$$\begin{aligned}
\texttt{revto}(h :: t, l2) &= \texttt{revto}(t, h :: l2) \\
&= \texttt{nrev}(t)@(h :: l2) \\
&= \texttt{nrev}(t)@[h]@l2 \\
&= \texttt{nrev}(h :: t)@l2
\end{aligned}$$

Other Examples

$$
\begin{aligned}
\texttt{nlength}(l1 @ l2) &= (\texttt{nlength}\, l1) @ (\texttt{nlength}\, l2) \\
\texttt{nrev}(l1 @ l2) &= (\texttt{nrev}\, l2) @ (\texttt{nrev}\, l1) \\
\texttt{nrev}(\texttt{nrev}\, l) &= l \\
l @ [] &= l \\
(\texttt{map}\, f) \circ (\texttt{map}\, g) &= \texttt{map}(f \circ g)
\end{aligned}
$$

Correctness preserving program transformations.

Structural Induction on Trees

```
datatype 'a tree =
         Lf
       | Br of 'a*'a tree*'a tree
```

To show $\phi(t)$ for all trees t, it suffices to show:

- $\phi(\texttt{Lf})$ Base case
- For any t_1, t_2 and x:
  if $\phi(t_1)$ and $\phi(t_2)$ then $\phi(\texttt{Br}(x, t_1, t_2))$.
  Induction step

$\phi(t)$ can be proved in $\texttt{size}(t)$ steps.

Two Preorder Functions

```
fun preorder Lf               = []
  | preorder (Br(x,t1,t2)) =
      x::preorder(t1)@preorder(t2);
```

```
fun preord (Lf, l)            = l
  | preord (Br(x,t1,t2),l) =
      x::preord(t1, preord (t2,l));
```

We can show that the two are equivalent:

$$\forall l\ \texttt{preord}(t, l) = \texttt{preorder}(t)@l$$

By induction on t

Preorder

Base Case:

$$\texttt{preord}(\texttt{Lf}, l) = l = []@l = \texttt{preorder}(\texttt{Lf})@l$$

Induction Step:

$$\begin{aligned}
&\texttt{preord}(\texttt{Br}(x, t_1, t_2), l) \\
&\qquad = x :: \texttt{preord}(t_1, \texttt{preord}(t_2, l)) \\
&\qquad = x :: \texttt{preorder}(t_1)@\texttt{preord}(t_2, l) \\
&\qquad = x :: \texttt{preorder}(t_1)@\texttt{preorder}(t_2) \\
&\qquad = \texttt{preorder}(x, t_1, t_2)
\end{aligned}$$

Induction Principles

Structural Induction works well on functions whose recursive definition follows the definition of the datatype.

Structural Recursion.

Some functions follow other recursive patterns.

> `mergesort` works by splitting its list in two.

We can show that $\phi(l)$ holds for all lists l by showing that:

$$\text{if } \forall\, l' \; \texttt{length}(l') < \texttt{length}(l) \rightarrow \phi(l')$$
$$\text{then } \phi(l)$$

More generally, *well-founded* induction works on any well-ordered set