CS 13

Mathematical Foundations of Computer Science

Structural Induction

We define the type **List** as follows:

- `[]` is a `List`.
- If $x \in \mathbb{N}$ and $L \in$ **List**, then $x :: L$ is a **List**.

Thus, every function on **List**s will have two cases:

- one for the empty case, and
- one for the "one more element" case.

Example

$$\begin{aligned} \texttt{len}(\texttt{[]}) &= 0 \\ \texttt{len}(x :: L) &= 1 + \texttt{len}(L) \end{aligned}$$

Example

$$\begin{aligned} \texttt{sum}(\texttt{[]}) &= 0 \\ \texttt{sum}(x :: L) &= x + \texttt{sum}(L) \end{aligned}$$

$$\begin{aligned}\texttt{len}(\texttt{[]}) &= 0 \\ \texttt{len}(x::L) &= 1+\texttt{len}(L)\end{aligned}$$

$$\begin{aligned}\texttt{sum}(\texttt{[]}) &= 0 \\ \texttt{sum}(x::L) &= x+\texttt{sum}(L)\end{aligned}$$

Example

Claim: For all $L \in$ **List**, where the list does not contain 0,

$$\texttt{sum}(L) \geq \texttt{len}(L)$$

We go by structural induction.

- **Case** `[]`**:** $\texttt{sum}(\texttt{[]}) = 0 = \texttt{len}(\texttt{[]})$.
- **Case** $x::L$**:** Note that

$$\begin{aligned}\texttt{sum}(x::L) &= x+\texttt{sum}(L) && \text{[By Definition of sum]} \\ &\geq x+\texttt{len}(L) && \text{[By IH]} \\ &\geq 1+\texttt{len}(L) && [\text{Since } x \in \mathbb{N} \setminus \{0\}] \\ &\geq \texttt{len}(x::L) && \text{[By Definition of len]}\end{aligned}$$

$$\begin{aligned} \texttt{sum}(\texttt{[]}) &= 0 \\ \texttt{sum}(x::L) &= x + \texttt{sum}(L) \end{aligned}$$

$$\begin{aligned} \texttt{sum2}(\texttt{acc},\ \texttt{[]}) &= \texttt{acc} \\ \texttt{sum2}(\texttt{acc},\ x::L) &= \texttt{sum2}(\texttt{acc} + x,\ L) \end{aligned}$$

Example

Claim: For all $L \in$ **List** and $\texttt{acc} \in \mathbb{N}$,

$$\texttt{sum(L)} + \texttt{acc} = \texttt{sum2}(\texttt{acc},\ L)$$

We go by structural induction.

- **Case** `[]`**:** $\texttt{sum}(\texttt{[]}) + \texttt{acc} = \texttt{acc} = \texttt{sum2}(\texttt{acc},\ \texttt{[]})$.
- **Case** $x::L$**:** Note that

$$\begin{aligned} \texttt{sum}(x::L) + \texttt{acc} &= x + \texttt{sum}(L) + \texttt{acc} && \text{[By Definition of } \texttt{sum}\text{]} \\ &= \texttt{sum2}(x + \texttt{acc},\ L) && \text{[By IH]} \\ &= \texttt{sum2}(\texttt{acc},\ x::L) && \text{[By Definition of } \texttt{sum2}\text{]} \end{aligned}$$

$$
\begin{aligned}
\texttt{sum}(\,\texttt{[]}\,) &= 0\\
\texttt{sum}(x::L) &= x + \texttt{sum}(L)
\end{aligned}
$$

$$
\begin{aligned}
\texttt{sum2}(\texttt{acc},\ \texttt{[]}\,) &= \texttt{acc}\\
\texttt{sum2}(\texttt{acc},\ x::L) &= \texttt{sum2}(\texttt{acc} + x,\ L)
\end{aligned}
$$

Example

Claim: For all $L \in$ **List** and $\texttt{acc} \in \mathbb{N}$,

$$\texttt{sum}(L) + \texttt{acc} = \texttt{sum2}(\texttt{acc},\ L)$$

We go by structural induction.

- **Case** `[]`**:** $\texttt{sum}(\,\texttt{[]}\,) + \texttt{acc} = \texttt{acc} = \texttt{sum2}(\texttt{acc},\ \texttt{[]}\,)$.
- **Case** $x::L$**:** Note that

$$
\begin{aligned}
\texttt{sum}(x::L) + \texttt{acc} &= x + \texttt{sum}(L) + \texttt{acc} && \text{[By Definition of } \texttt{sum}\text{]}\\
&= \texttt{sum2}(x + \texttt{acc},\ L) && \text{[By IH]}\\
&= \texttt{sum2}(\texttt{acc},\ x::L) && \text{[By Definition of } \texttt{sum2}\text{]}
\end{aligned}
$$

$$\begin{aligned} \texttt{append}(\texttt{[]}, a) &= a :: \texttt{[]} \\ \texttt{append}(x :: L, a) &= x :: \texttt{append}(L, a) \end{aligned}$$

$$\begin{aligned} \texttt{removeLast}(\texttt{[]}) &= \texttt{[]} \\ \texttt{removeLast}(x :: L) &= \textbf{if } L \texttt{ == [] } \textbf{then } \texttt{[] } \textbf{else } x :: \texttt{removeLast}(L) \end{aligned}$$

Example

Claim: For all $L \in$ **List** and $a \in \mathbb{N}$,

$$\texttt{removeLast}(\texttt{append}(L, a)) = L$$

We go by structural induction.

- **Case** **[]:** $\texttt{removeLast}(\texttt{append}(\texttt{[]}, a)) = \texttt{removeLast}(a :: \texttt{[]}) = \texttt{[]}$.
- **Case** $x :: L$**:** Note that

$$\begin{aligned} \texttt{rL}(\texttt{ap}(x :: L, a)) &= \texttt{rL}(x :: \texttt{ap}(L, a)) \\ &= \textbf{if } \texttt{ap}(L, a) \texttt{ == [] } \textbf{then } \texttt{[] } \textbf{else } x :: \texttt{rL}(\texttt{ap}(L, a)) \\ &= \textbf{if } \texttt{ap}(L, a) \texttt{ == [] } \textbf{then } \texttt{[] } \textbf{else } x :: L \\ &= x :: L \end{aligned}$$