We introduce a generalised reduction over sequences. The function takes as parameters an associative reduction operator, an initial accumulator value, and a sequence of elements. The accumulator need not have the same type as the elements of the sequence. For instance, we can use a reduction to count the number of occurrences of a particular value in a sequence.

$$
\begin{array}{l}
[S, T] \\
\hline
\mathit{reduce} : (S \times T \to S) \times S \times \mathrm{seq}\, T \to S \\
\hline
\forall f : S \times T \to S \bullet \forall s : S \bullet \forall ts : \mathrm{seq}\, T \bullet \\
\quad \mathit{reduce}(f, s, ts) = \\
\qquad \textbf{if } \# ts = 0 \\
\qquad \textbf{then } s \\
\qquad \textbf{else } \mathit{reduce}(f, f(s, \mathit{head}\ ts), \mathit{tail}\ ts)
\end{array}
$$

This definition hints at a linear reduction, but an efficient implementation could perform a tree-like reduction on vector hardware.