Queues/Signature

There is another common and useful linear data structure call **queue**.

As the stack, it is fairly intuitive, since we experience the concept when we are waiting at some place to get some goods or service.

Let us call $\textsc{Queue}$(item) the specification of a queue over elements of the item type.

- **Parameter types**
  - The type item of the elements in the queue.
- **Defined types**
  - The type of the queue is t.

Queues/Constructors and other functions

- **Constructors**
  - $\textsc{Empty} : \mathsf{t}$
    Expression $\textsc{Empty}$ represents the empty queue.
  - $\textsc{Enqueue} : \mathsf{item} \times \mathsf{t} \to \mathsf{t}$
    Expression $\textsc{Enqueue}(e, q)$ denotes the queue q with element e added at the end.
- **Other functions**
  - $\textsc{Dequeue} : \mathsf{t} \to \mathsf{t} \times \mathsf{item}$
    Expression $\textsc{Dequeue}(q)$ denotes the pair made of the *first* element of q and the remaining queue. The queue q must not be empty.

Queues/Equations

$$
\begin{aligned}
\textsc{Dequeue}(\textsc{Enqueue}(e, \textsc{Empty})) &= (\textsc{Empty}, e)\\
\textsc{Dequeue}(\textsc{Enqueue}\ \boldsymbol{(}e\boldsymbol{,}\ q\boldsymbol{)})) &= (\textsc{Enqueue}(e, q_1), e')\\
\text{where}\quad (q_1, e') &= \textsc{Dequeue}(q)\\
\text{and}\quad q &\neq \textsc{Empty}
\end{aligned}
$$

They are easy to orient since q is a proper subterm of $\textsc{Enqueue}(e, q)$:

$$
\begin{aligned}
\textsc{Dequeue}(\textsc{Enqueue}(e, \textsc{Empty})) &\to (\textsc{Empty}, e)\\
\textsc{Dequeue}(\textsc{Enqueue}(e, q)) &\to (\textsc{Enqueue}(e, q_1), e')
\end{aligned}
$$

where $q \neq \textsc{Empty}$ and where $\textsc{Dequeue}(q) \to (q_1, e')$.
Note that we can remove the condition by replacing q by $\textsc{Enqueue}(\dots)$.