Answers to the mid-term examination on XML

Christian Rinderknecht

16 October 2006

1 Queues

A *queue* is a stack where the items are popped at the **bottom**, not at the top. Pushing an item in a queue is called *enqueuing*, whereas the special pop is called *dequeuing*.

$$\textsc{Enqueue} \rightarrow \begin{array}{c|c|c|c|c|c|c}\hline & a & b & c & d & e & \\ \hline\end{array} \rightarrow \textsc{Dequeue}$$

$$\textsc{Push}, \textsc{Pop} \leftrightarrow \begin{array}{c|c|c|c|c|c|}\hline & a & b & c & d & e \\ \hline\end{array}$$

We want to implement queues with two stacks:

$$\textsc{Enqueue} \rightarrow \begin{array}{c|c|c|c|}\hline & a & b & c \\ \hline\end{array} \quad \begin{array}{|c|c|c}\hline d & e & \\ \hline\end{array} \rightarrow \textsc{Dequeue}$$

So Enqueue is Push on the first stack and Dequeue is Pop on the second. If the second stack is empty, we swap the stacks and reverse the (new) second:

$$\textsc{Enqueue} \rightarrow \begin{array}{c|c|c|c|}\hline & a & b & c \\ \hline\end{array} \quad \begin{array}{|c}\hline \\ \hline\end{array} \rightarrow \textsc{Dequeue}\,\mathbf{???}$$

$$\textsc{Enqueue} \rightarrow \begin{array}{c|}\hline \\ \hline\end{array} \quad \begin{array}{|c|c|c|c}\hline a & b & c & \\ \hline\end{array} \rightarrow \textsc{Dequeue}$$

Data. Let $\textsc{Queue}(S, T)$ be the queue made of the stacks S and T; let $\textsc{Push}(e, S)$ be the stack S with item e on top; let $\textsc{Empty}$ be the empty stack.

Functions. Let $\textsc{Enqueue}(e, Q)$ rewrite to the queue Q where item e is added; let $\textsc{Dequeue}(Q)$ rewrite to the pair (e, Q') where e is the item on the head of queue Q and Q' is the remaining queue.

Question. Define $\textsc{Enqueue}(e, Q)$ and $\textsc{Dequeue}(Q)$, where Q is a queue.

Answer.

$$
\begin{aligned}
\textsc{Dequeue}(\textsc{Queue}(S, \textsc{Push}(e, T))) &\xrightarrow{1} (e, \textsc{Queue}(S, T)) \\
\textsc{Dequeue}(\textsc{Queue}(S, \textsc{Empty})) &\xrightarrow{2} \textsc{Dequeue}(\textsc{Queue}(\textsc{Empty}, \textsc{Rev}(S))) \\
\textsc{Enqueue}(e, \textsc{Queue}(S, T)) &\xrightarrow{3} \textsc{Queue}(\textsc{Push}(e, S), T)
\end{aligned}
$$

But there is a problem with $\textsc{Dequeue}$ if the queue is empty. Indeed, in this case we have

$$
\begin{aligned}
&\textsc{Dequeue}(\textsc{Queue}(\textsc{Empty}, \textsc{Empty})) \\
&\qquad \xrightarrow{2} \textsc{Dequeue}(\textsc{Queue}(\textsc{Empty}, \textsc{Rev}(\textsc{Empty}))) \\
&\qquad\qquad \rightarrow \textsc{Dequeue}(\textsc{Queue}(\textsc{Empty}, \textsc{Empty}))
\end{aligned}
$$

which is a never-ending loop! The fix consists in forbidding the use of rule 2 when the queue is empty. One easy way is simply to say

$$
\textsc{Dequeue}(\textsc{Queue}(S, \textsc{Empty})) \xrightarrow{2} \textsc{Dequeue}(\textsc{Queue}(\textsc{Empty}, \textsc{Rev}(S))) \quad \text{if } S \neq \textsc{Empty}
$$

Another way is to add an error-case rule like

$$
\textsc{Dequeue}(\textsc{Queue}(\textsc{Empty}, \textsc{Empty})) \xrightarrow{4} \textsc{Error}
$$

and prefer it over rule 3.

2 Stacks

Question. Define $\textsc{Bottom}(S)$ be the pair (e, T) where e is the item at the bottom of stack S and T is S without e.

Answer.

$$
\begin{aligned}
\textsc{Bottom}(S) &\rightarrow \textsc{Bottom}'(\textsc{Rev}(S)) \\
\textsc{Bottom}'(\textsc{Push}(e, T)) &\rightarrow (e, \textsc{Rev}(T))
\end{aligned}
$$

Note. The two questions are related. Indeed, $\textsc{Bottom}$ provides a means to use a stack as a queue: $\textsc{Enqueue}$ is simply $\textsc{Push}$ and $\textsc{Dequeue}$ is $\textsc{Bottom}$! But $\textsc{Bottom}$ is expensive: in order to get the element at the bottom, i.e. dequeue, we have to reverse the whole stack two times (see S and T), except one element. But in the first question, $\textsc{Dequeue}$ is cheap: if the second stack is not empty, it is simply equivalent to a $\textsc{Pop}$ (one rewrite). If it is empty, then the cost is to reverse the first stack, i.e. to traverse *only once* all the elements — instead of two times, every time, with $\textsc{Bottom}$. There is a direct way to make queues (that is, without using stacks). We need here an empty queue, noted Q_0, but **no** $\textsc{Queue}$.

Example. The queue containing only item e is $\textsc{Enqueue}(e, \mathrm{Q}_0)$. The queue containing item e and then f is $\textsc{Enqueue}(f, \textsc{Enqueue}(e, \mathrm{Q}_0))$. We have

$$\textsc{Dequeue}(\textsc{Enqueue}(f, \textsc{Enqueue}(e, \mathrm{Q}_0))) \to \cdots \to (\textsc{Enqueue}(f, \mathrm{Q}_0), e)$$

Question. Give the rewrite rules defining $\textsc{Dequeue}$.

Answer.

$$\begin{aligned}
\textsc{Dequeue}(\textsc{Enqueue}(e, Q_0)) &\to (Q_0, e) \\
\textsc{Dequeue}(\textsc{Enqueue}(e, Q)) &\to (\textsc{Enqueue}(e, Q'), f) \\
&\quad \text{where } q \neq Q_0 \text{ and } \textsc{Dequeue}(Q) \to (Q', f)
\end{aligned}$$