

A question on the correctness of a nonblocking algorithm

The $swap(m,v)$ atomic operation receives two parameters: m - a shared memory variable, and v - some value. The $swap$ operation atomically sets m 's value to v and returns its previous value to the calling process. The $fetch-and-inc(c)$ operation receives a single parameter c - a shared memory integer variable. It atomically increments its value and returns the previous value to the calling process.

Consider the following proposed nonblocking implementation of a FIFO queue from $fetch-and-inc$ and $swap$ operations. The algorithm uses a shared counter c , supporting the $fetch-and-inc$ and read operations and initialized to 0, and a shared infinite array $vals$, each element of which initialized to null and supporting the $swap$ operation.

fetch-and-inc c initially 0, swap vals[] initially null

Enqueue(val)

$i := fetch-and-inc(c)$

$vals[i] := val$

Dequeue()

$i := c$

for ($k := 0$ to $i-1$) {
 $v := swap(vals[k], null)$
 if ($v \neq null$)

 return v

}

return $null$

- Describe a detailed execution showing that the algorithm is not linearizable.
- Present a small change to the algorithm to make it linearizable (and still lock-free).