

```
;; Producer/Consumer-Puffer mit Semaphoren

(define-record-type :buffer
  (really-make-buffer data waiting lock wait)
  buffer?
  (data buffer-data set-buffer-data!)
  (waiting buffer-waiting set-buffer-waiting!)
  (lock buffer-lock)
  (wait buffer-wait))

(define (make-buffer)
  (really-make-buffer #f
                     0
                     (make-semaphore 1)
                     (make-semaphore 0)))

(define (wait buffer)
  (set-buffer-waiting! buffer
                    (+ 1 (buffer-waiting buffer)))
  (semaphore-post (buffer-lock buffer))
  (semaphore-wait (buffer-wait buffer)))

(define (signal buffer)
  (let loop ()
    (if (> (buffer-waiting buffer) 0)
        (begin
          (set-buffer-waiting! buffer
                               (- (buffer-waiting buffer) 1))
          (semaphore-post (buffer-wait buffer))
          (loop))))))

(define (insert buffer value)
  (let loop ()
    (semaphore-wait (buffer-lock buffer))
    (cond
     ((buffer-data buffer)
      (wait buffer)
      (loop))
     (else
      (set-buffer-data! buffer value)
      (signal buffer)
      (semaphore-post (buffer-lock buffer))))))

(define (remove buffer)
  (let loop ()
    (semaphore-wait (buffer-lock buffer))
    (cond
     ((buffer-data buffer)
      => (lambda (value)
          (set-buffer-data! buffer #f)
          (signal buffer)
          (semaphore-post (buffer-lock buffer))
          value))
     (else
      (wait buffer)
      (loop))))))
```