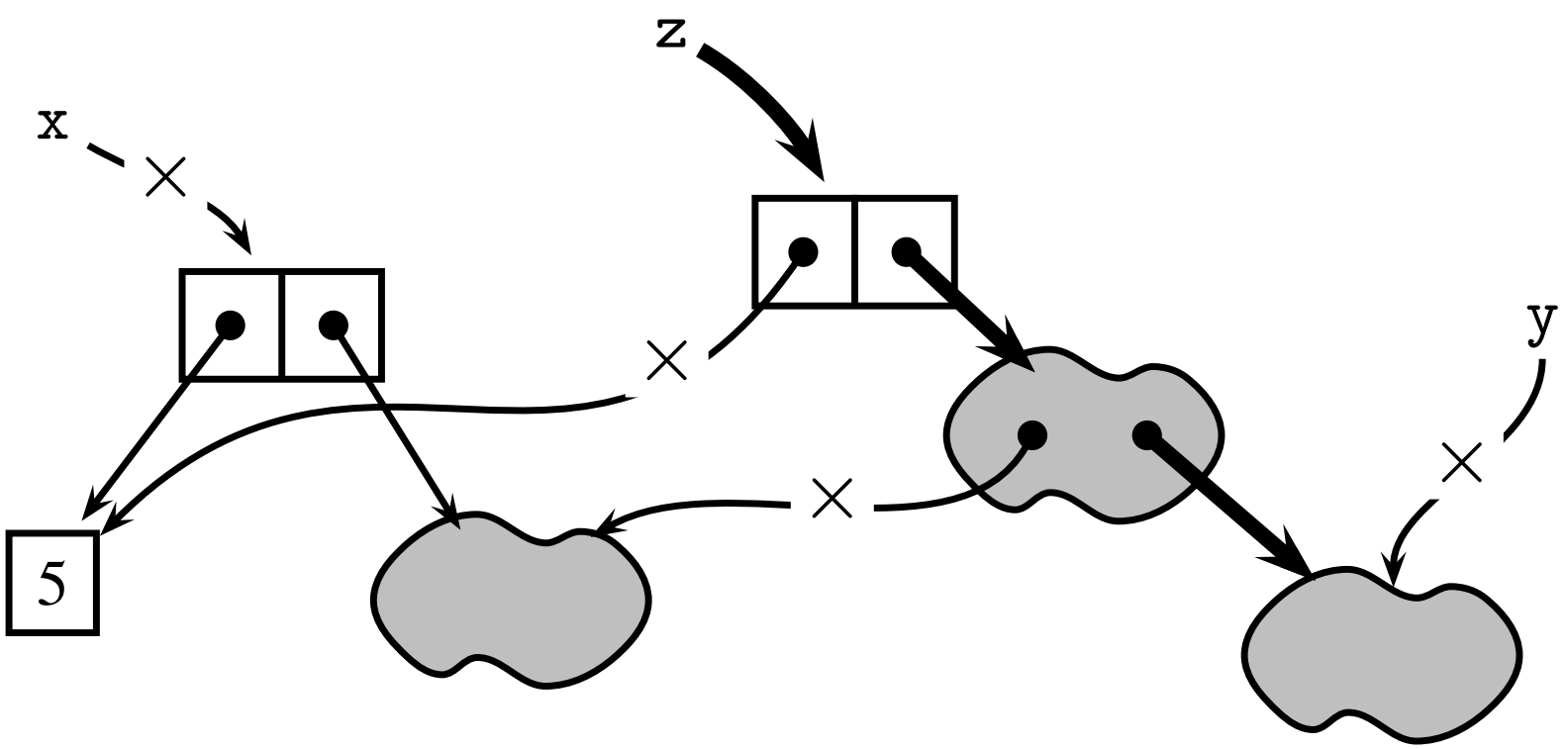
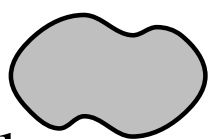
```
(define (length l)
  (if (null? l) 0 (+ 1 (length (cdr l)))))

(define (append l1 l2)
  (if (null? l1) l2
   (cons (car l1) (append (cdr l1) l2))))

(let x ← (cons 5 (cons (cons 6 nil) nil)) in
  (let y ← (cons 3 nil) in
    (let z ← (append x y) in
      (if (null? (car z)) 0 π: (length z)))))
```

(a) Example program.

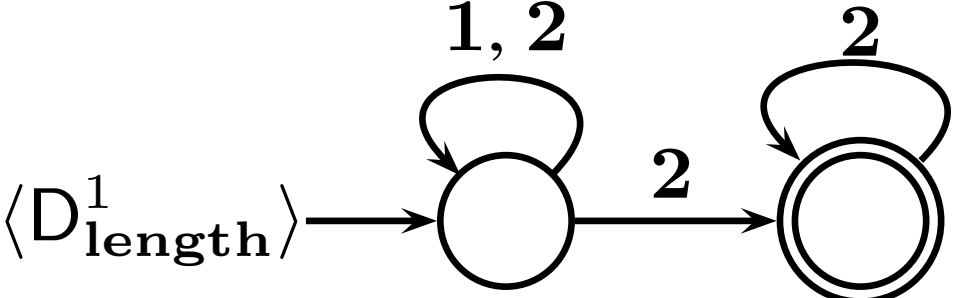


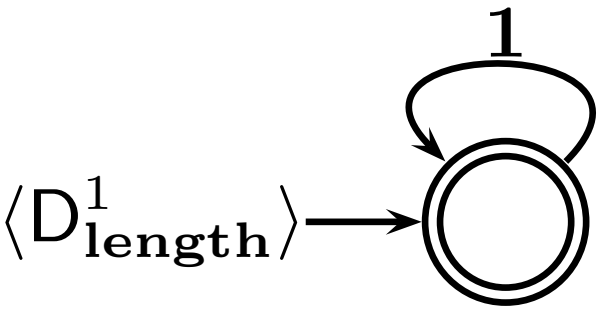
(b) Memory graph at π.  denotes a closure. Thick edges denote live links. Traversal stops at edges marked $\times$ during garbage collection for a liveness-based collector.

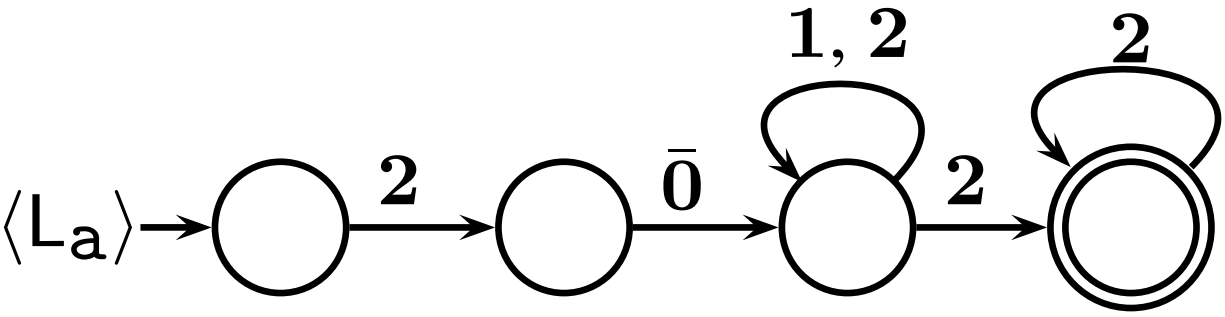
$$\begin{cases} \emptyset & \text{if } \sigma = \emptyset \\ \{\epsilon\} \cup \mathbf{0}\sigma & \text{otherwise} \end{cases}$$

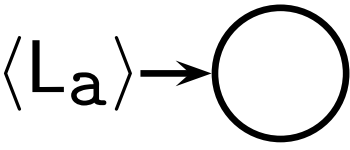
$$GC(\rho_1, S_1, H_1, (\textbf{let } x \leftarrow s \textbf{ in } e), \sigma) = (\rho, S, H)$$

a BIG closure


1, 2
2
2
⟨D¹length⟩


1
⟨D¹length⟩


⟨La⟩
2
0̄
1, 2
2
2


⟨La⟩


$$(\mathbf{2} \cup \mathbf{0})\sigma$$

$$GC(\rho_1, S_1, H_1, (\textbf{let } x \leftarrow s \textbf{ in } e), \sigma) = (\rho, S, H)$$