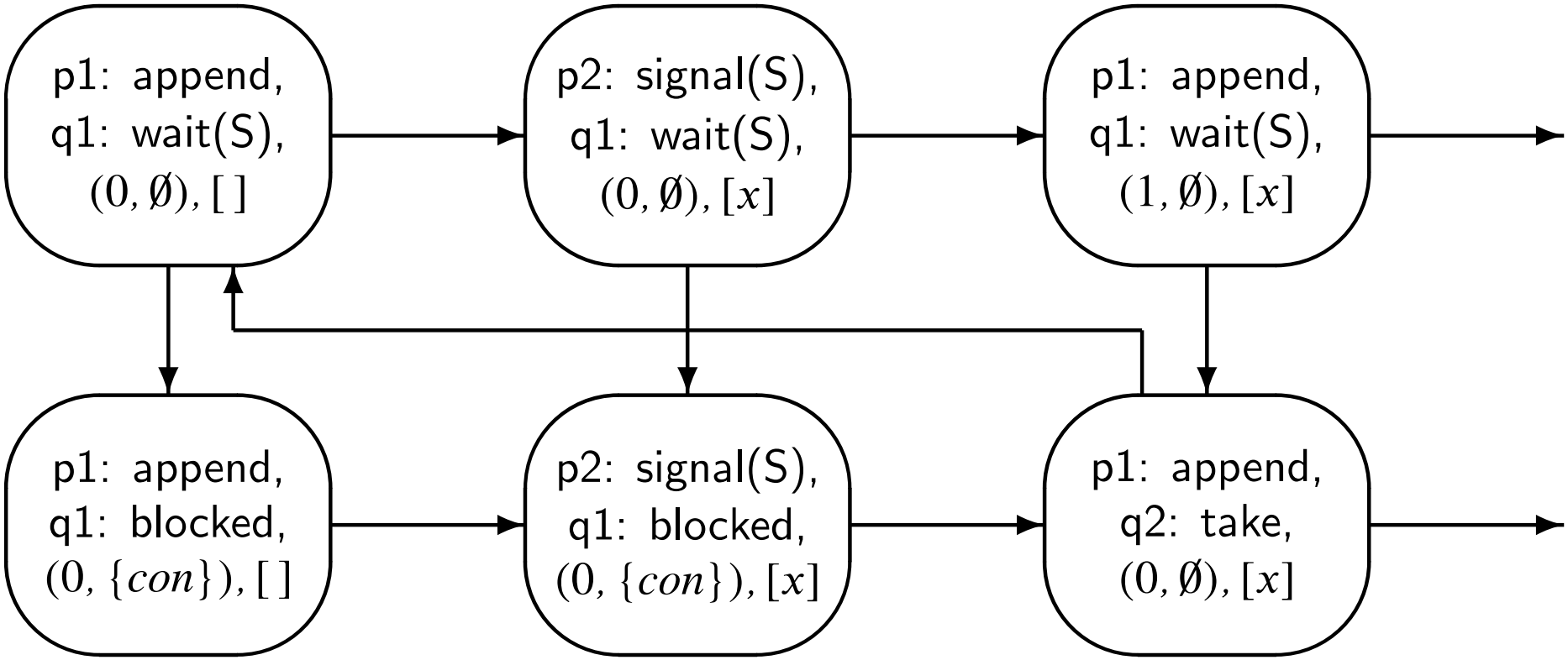
**Algorithm 6.6: Producer-consumer (infinite buffer)**

```
infinite queue of dataType buffer ← empty queue
semaphore notEmpty ← (0, ∅)
```

| **producer** | **consumer** |
|---|---|
| dataType d | dataType d |
| loop forever | loop forever |
| p1: d ← produce | q1: wait(notEmpty) |
| p2: append(d, buffer) | q2: d ← take(buffer) |
| p3: signal(notEmpty) | q3: consume(d) |

# Partial State Diagram for Producer-Consumer with Infinite Buffer



**Algorithm 6.7: Producer-consumer (infinite buffer, abbreviated)**

```
                infinite queue of dataType buffer ← empty queue
                semaphore notEmpty ← (0, ∅)
```

| **producer** | **consumer** |
|---|---|
| dataType d | dataType d |
| loop forever | loop forever |
| p1: append(d, buffer) | q1: wait(notEmpty) |
| p2: signal(notEmpty) | q2: d ← take(buffer) |

**Algorithm 6.8: Producer-consumer (finite buffer, semaphores)**

```
finite queue of dataType buffer ← empty queue
semaphore notEmpty ← (0, ∅)
semaphore notFull ← (N, ∅)
```

| **producer** | **consumer** |
|---|---|
| dataType d | dataType d |
| loop forever | loop forever |
| p1: d ← produce | q1: wait(notEmpty) |
| p2: wait(notFull) | q2: d ← take(buffer) |
| p3: append(d, buffer) | q3: signal(notFull) |
| p4: signal(notEmpty) | q4: consume(d) |

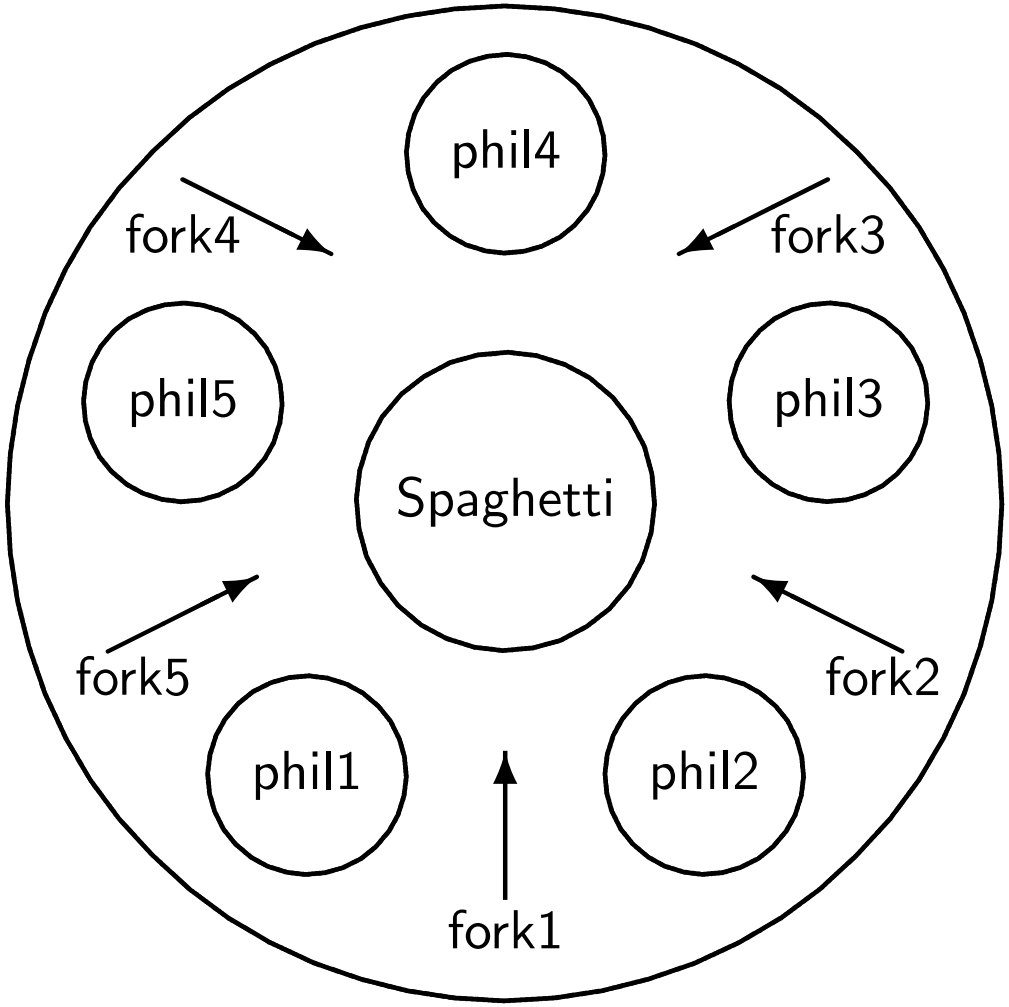
# Scenario with Busy Waiting

| n | Process p | Process q | S |
|---|---|---|---|
| 1 | **p1: wait(S)** | q1: wait(S) | 1 |
| 2 | p2: signal(S) | **q1: wait(S)** | 0 |
| 3 | **p2: signal(S)** | q1: wait(S) | 0 |
| 4 | **p1: wait(S)** | q1: wait(S) | 1 |

**Algorithm 6.9: Dining philosophers (outline)**

```
      loop forever
p1:       think
p2:       preprotocol
p3:       eat
p4:       postprotocol
```

# The Dining Philosophers



**Algorithm 6.10: Dining philosophers (first attempt)**

```
                semaphore array [0..4] fork ← [1,1,1,1,1]
        loop forever
p1:         think
p2:         wait(fork[i])
p3:         wait(fork[i+1])
p4:         eat
p5:         signal(fork[i])
p6:         signal(fork[i+1])
```

**Algorithm 6.11: Dining philosophers (second attempt)**

```
                    semaphore array [0..4] fork ← [1,1,1,1,1]
                    semaphore room ← 4
    loop forever
p1:     think
p2:     wait(room)
p3:     wait(fork[i])
p4:     wait(fork[i+1])
p5:     eat
p6:     signal(fork[i])
p7:     signal(fork[i+1])
p8:     signal(room)
```

| Algorithm 6.12: Dining philosophers (third attempt) | |
|---|---|
| semaphore array [0..4] fork ← [1,1,1,1,1] | |
| | **philosopher 4** |
| | loop forever |
| p1: | think |
| p2: | wait(fork[0]) |
| p3: | wait(fork[4]) |
| p4: | eat |
| p5: | signal(fork[0]) |
| p6: | signal(fork[4]) |