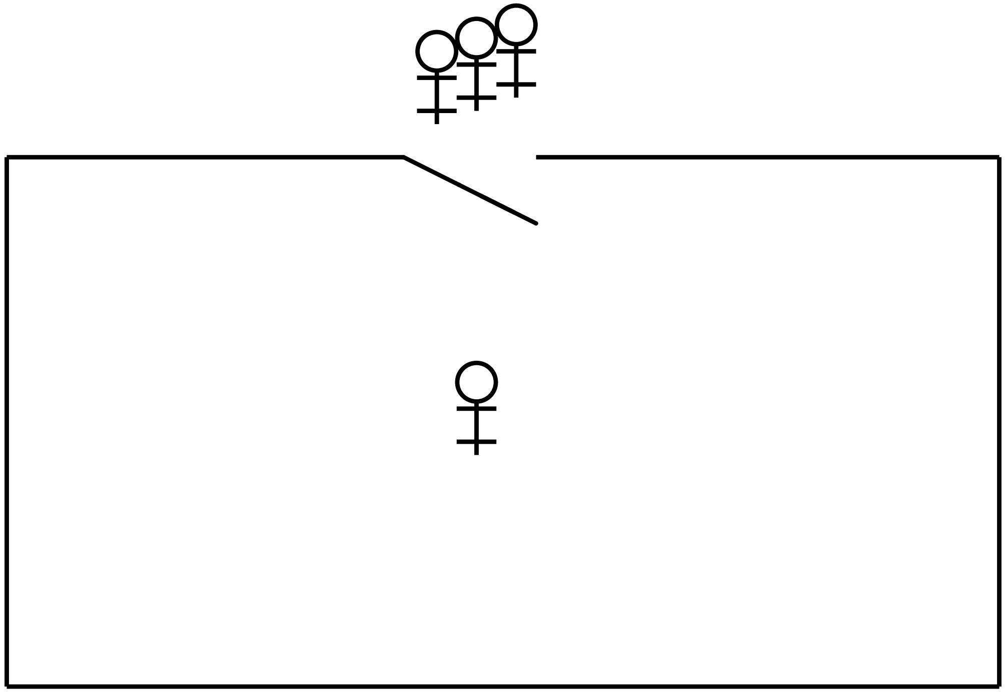
Algorithm 3.1: Critical section problem

global variables

p	q
local variables	local variables
loop forever	loop forever
non-critical section	non-critical section
preprotocol	preprotocol
critical section	critical section
postprotocol	postprotocol

Critical Section



Algorithm 3.2: First attempt

```
                      integer turn ← 1
                p                                    q
      loop forever                         loop forever
p1:       non-critical section       q1:       non-critical section
p2:       await turn = 1             q2:       await turn = 2
p3:       critical section           q3:       critical section
p4:       turn ← 2                   q4:       turn ← 1
```

Algorithm 3.3: History in a sequential algorithm

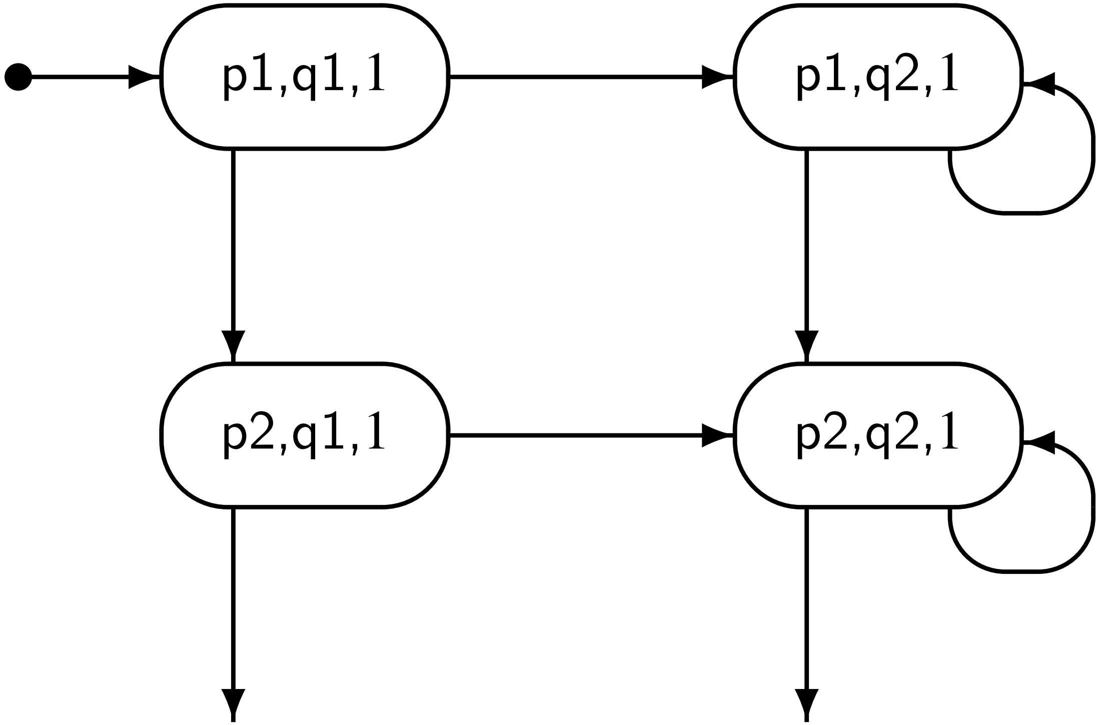
```
integer a ← 1, b ← 2
p1: Millions of statements
p2: a ← (a+b)*5
p3: . . .
```

Algorithm 3.4: History in a concurrent algorithm

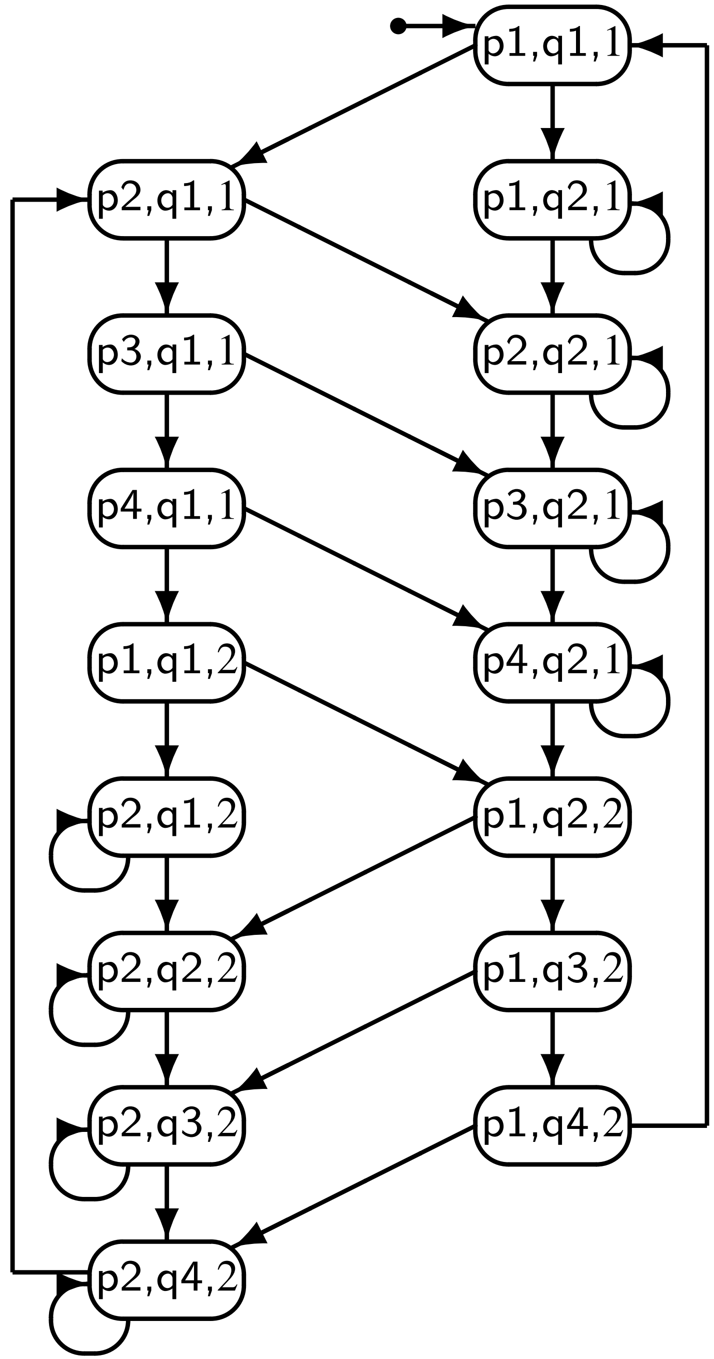
integer a ← 1, b ← 2

p	q
p1: Millions of statements	q1: Millions of statements
p2: a ← (a+b)*5	q2: b ← (a+b)*5
p3: . . .	q3: . . .

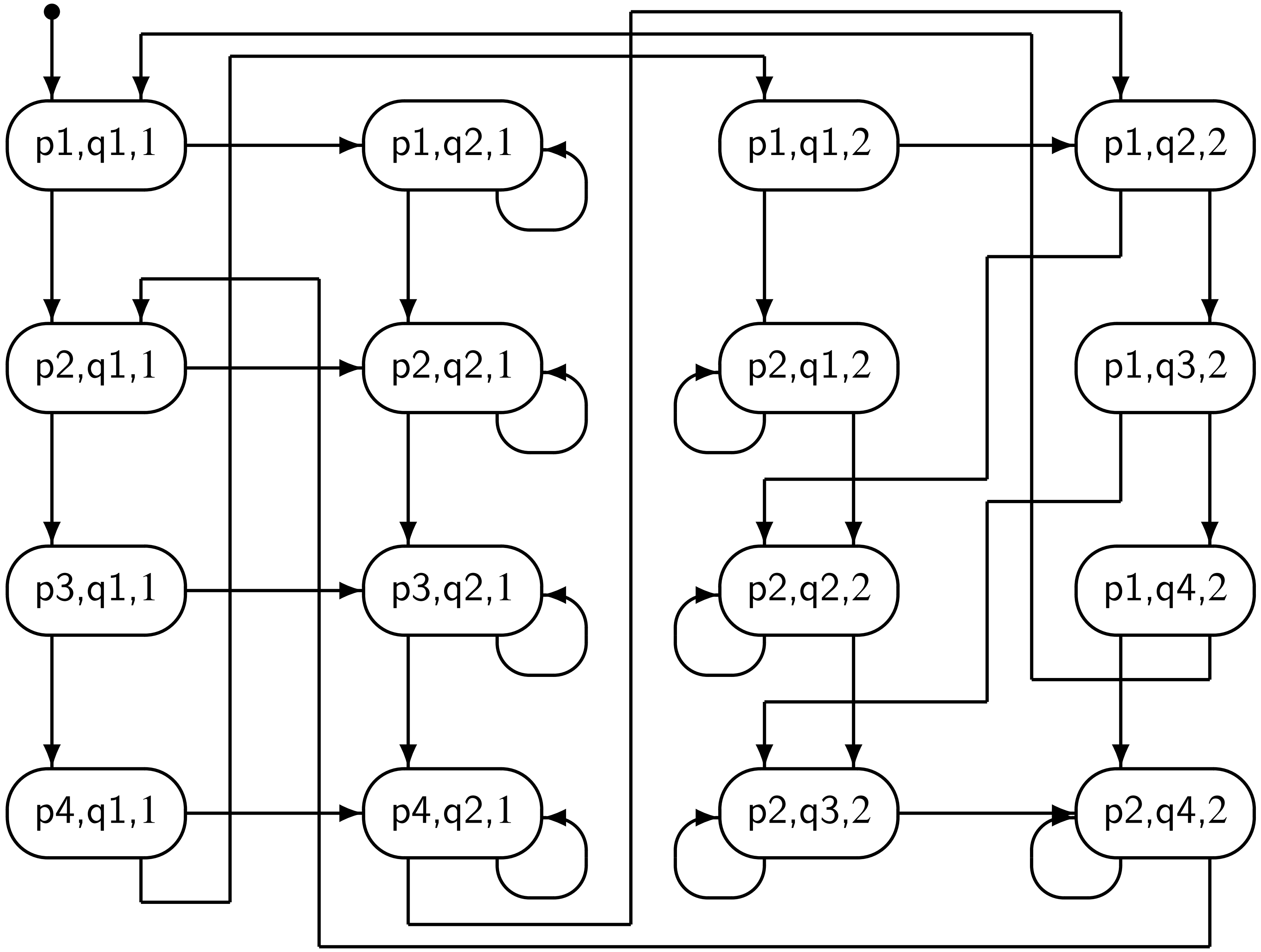
First States of the State Diagram



State Diagram for the First Attempt



Alternate Layout for the First Attempt (Not in the Book)

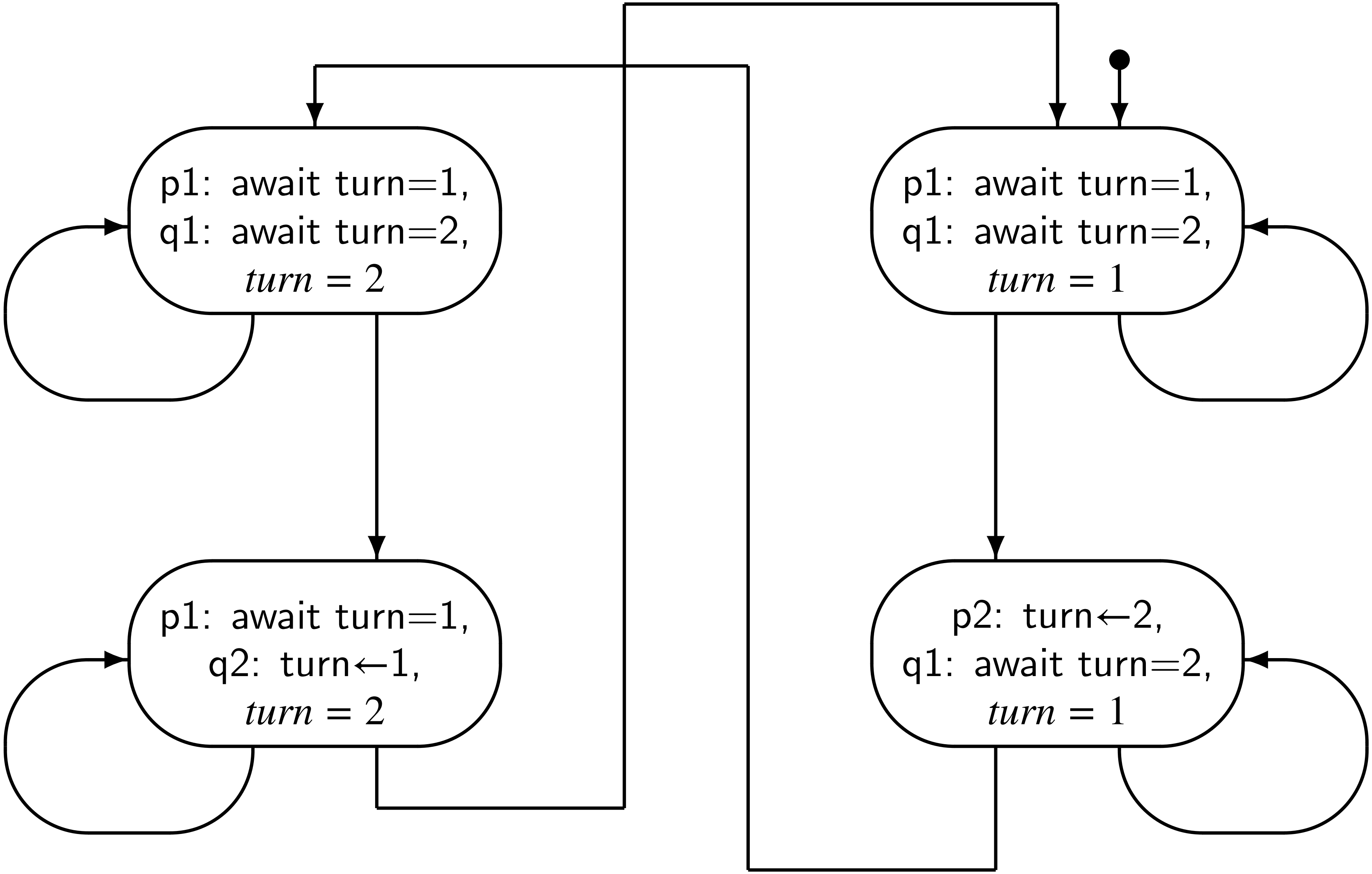


Algorithm 3.5: First attempt (abbreviated)

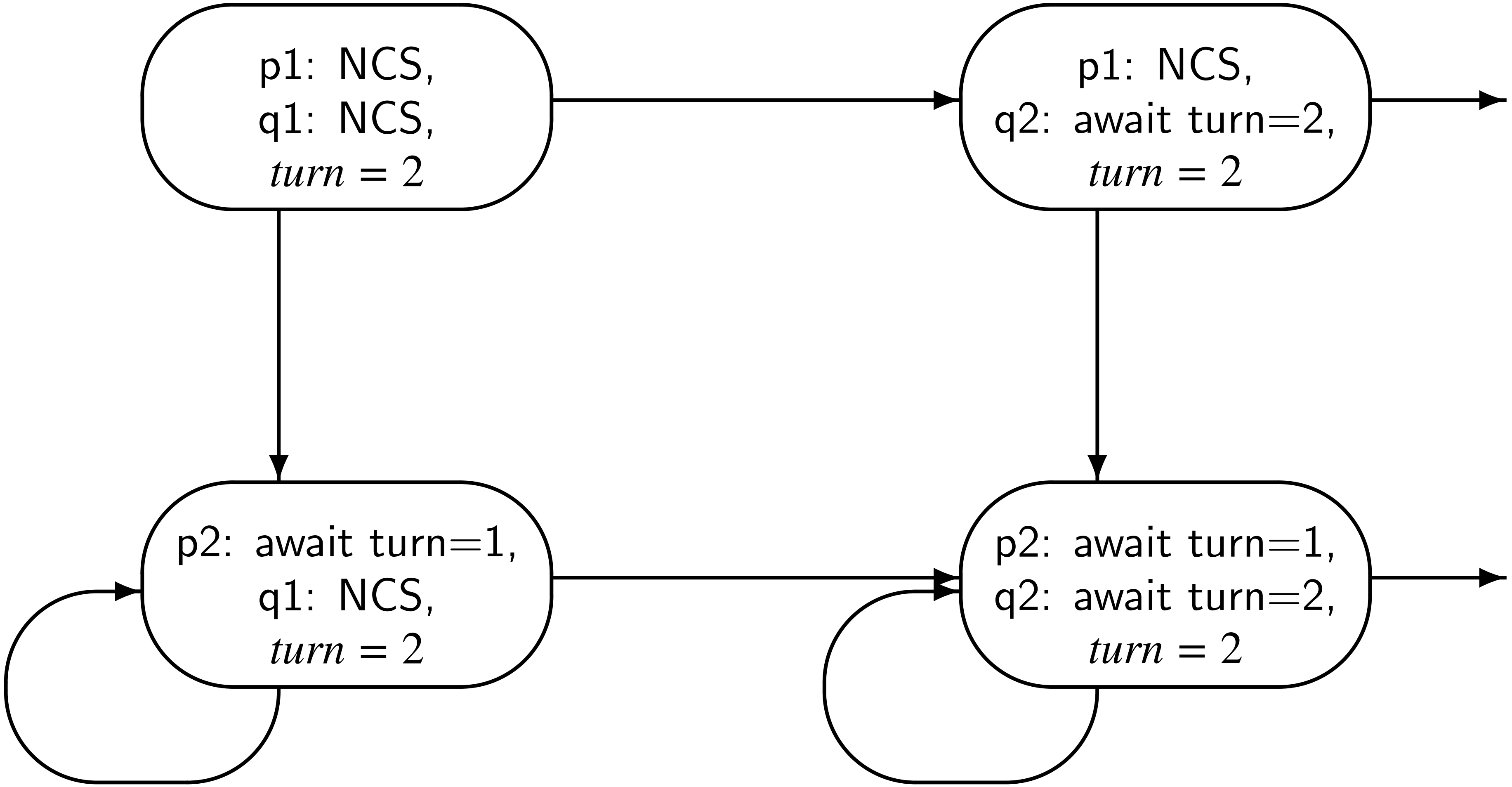
integer turn ← 1

p	**q**
loop forever	loop forever
p1: await turn $=$ 1	q1: await turn $=$ 2
p2: turn ← 2	q2: turn ← 1

State Diagram for the Abbreviated First Attempt



Fragment of the State Diagram for the First Attempt



p1: NCS,
q1: NCS,
turn = 2

p1: NCS,
q2: await turn=2,
turn = 2

p2: await turn=1,
q1: NCS,
turn = 2

p2: await turn=1,
q2: await turn=2,
turn = 2

Algorithm 3.6: Second attempt

boolean wantp ← false, wantq ← false

	p		q
	loop forever		loop forever
p1:	non-critical section	q1:	non-critical section
p2:	await wantq = false	q2:	await wantp = false
p3:	wantp ← true	q3:	wantq ← true
p4:	critical section	q4:	critical section
p5:	wantp ← false	q5:	wantq ← false

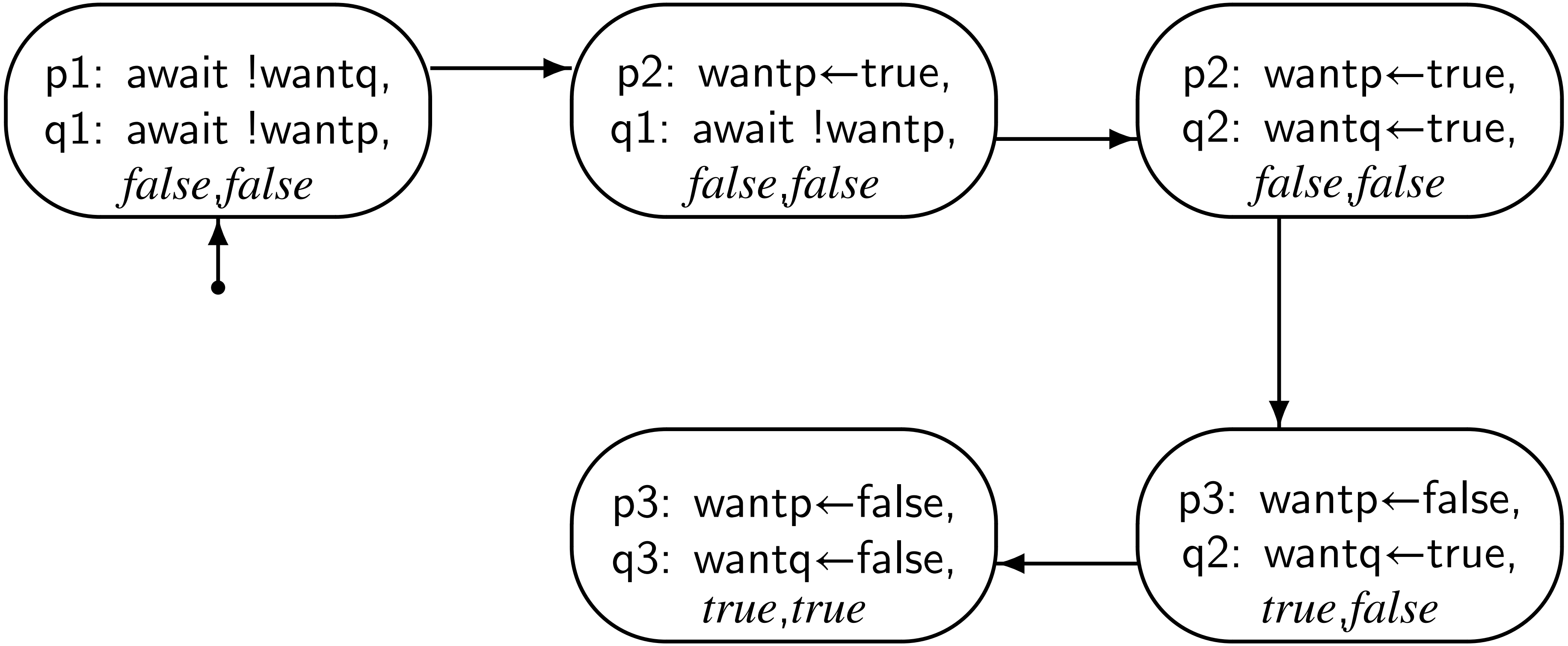
Algorithm 3.7: Second attempt (abbreviated)

```
                 boolean wantp ← false, wantq ← false

                 p                                      q

     loop forever                           loop forever
p1:     await wantq = false            q1:     await wantp = false
p2:     wantp ← true                   q2:     wantq ← true
p3:     wantp ← false                  q3:     wantq ← false
```

Fragment of the State Diagram for the Second Attempt



p1: await !wantq,
q1: await !wantp,
false,*false*

p2: wantp←true,
q1: await !wantp,
false,*false*

p2: wantp←true,
q2: wantq←true,
false,*false*

p3: wantp←false,
q2: wantq←true,
true,*false*

p3: wantp←false,
q3: wantq←false,
true,*true*

Scenario Showing that Mutual Exclusion Does Not Hold

Process p	Process q	wantp	wantq
p1: await wantq=false	q1: await wantp=false	*false*	*false*
p2: wantp←true	**q1: await wantp=false**	*false*	*false*
p2: wantp←true	q2: wantq←true	*false*	*false*
p3: wantp←false	**q3: wantq←true**	*true*	*false*
p3: wantp←false	q3: wantq←false	*true*	*true*

Algorithm 3.8: Third attempt

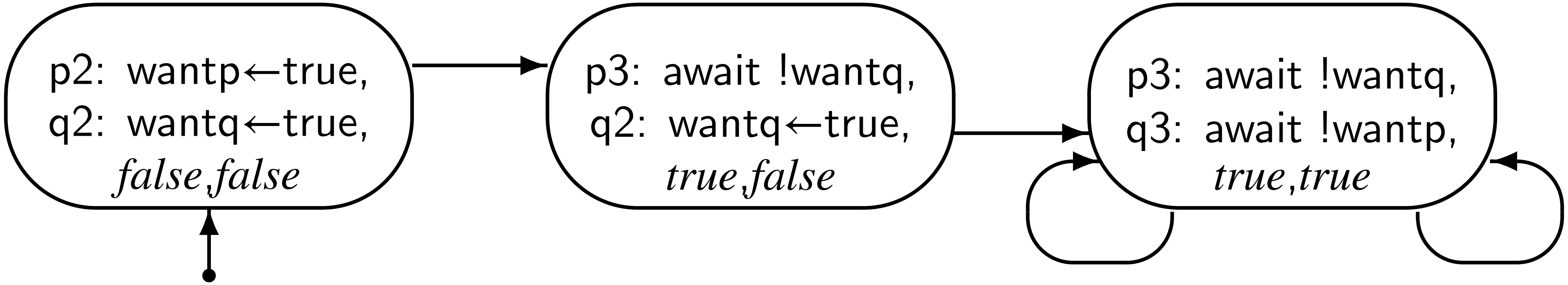
```
boolean wantp ← false, wantq ← false

              p                                  q
     loop forever                       loop forever
p1:     non-critical section       q1:     non-critical section
p2:     wantp ← true               q2:     wantq ← true
p3:     await wantq = false        q3:     await wantp = false
p4:     critical section           q4:     critical section
p5:     wantp ← false              q5:     wantq ← false
```

Scenario Showing Deadlock in the Third Attempt

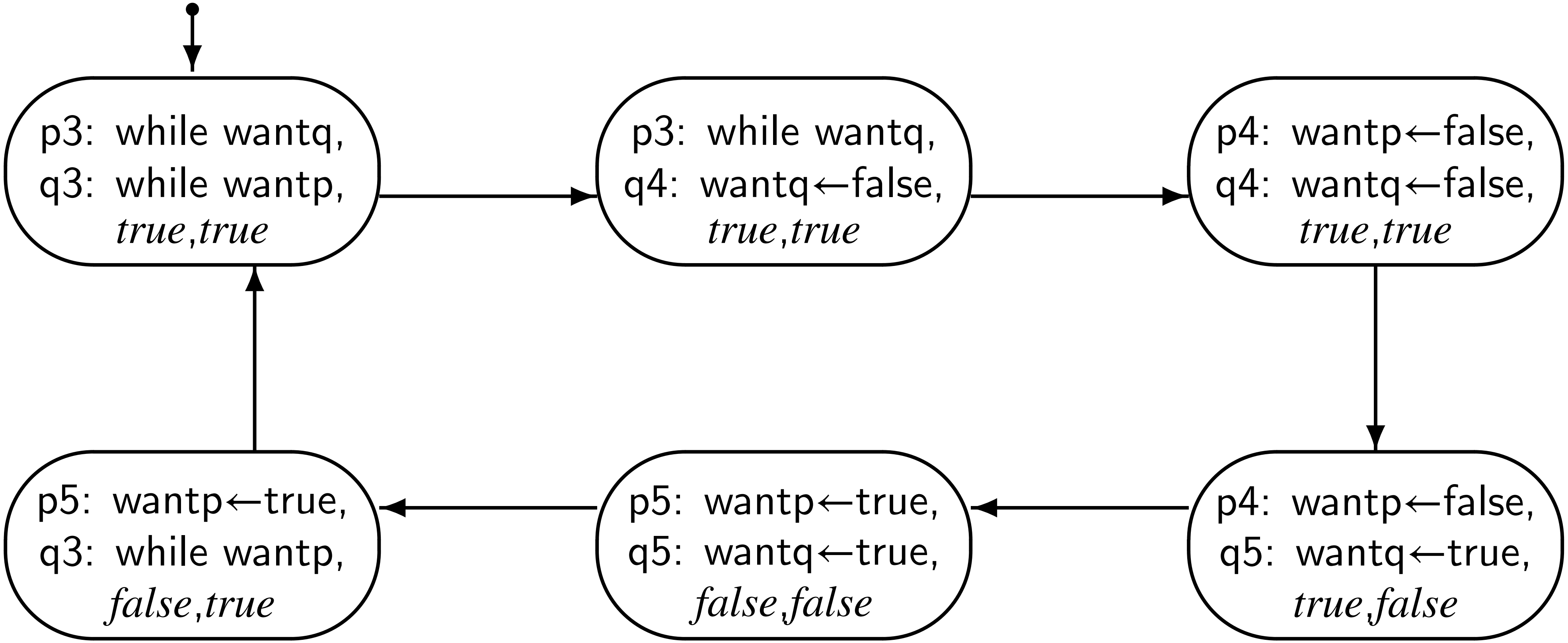
Process p	Process q	wantp	wantq
p1: non-critical section	q1: non-critical section	*false*	*false*
p2: wantp←true	**q1: non-critical section**	*false*	*false*
p2: wantp←true	q2: wantq←true	*false*	*false*
p3: await wantq=false	**q2: wantq←true**	*true*	*false*
p3: await wantq=false	q3: await wantp=false	*true*	*true*

Fragment of the State Diagram Showing Deadlock



Algorithm 3.9: Fourth attempt	
boolean wantp ← false, wantq ← false	
p	**q**
loop forever	loop forever
p1: non-critical section	q1: non-critical section
p2: wantp ← true	q2: wantq ← true
p3: while wantq	q3: while wantp
p4: wantp ← false	q4: wantq ← false
p5: wantp ← true	q5: wantq ← true
p6: critical section	q6: critical section
p7: wantp ← false	q7: wantq ← false

Cycle in the State Diagram for the Fourth Attempt



p3: while wantq,
q3: while wantp,
true,*true*

p3: while wantq,
q4: wantq←false,
true,*true*

p4: wantp←false,
q4: wantq←false,
true,*true*

p4: wantp←false,
q5: wantq←true,
true,*false*

p5: wantp←true,
q5: wantq←true,
false,*false*

p5: wantp←true,
q3: while wantp,
false,*true*

Algorithm 3.10: Dekker's algorithm

```
boolean wantp ← false, wantq ← false
integer turn ← 1
```

p

```
        loop forever
p1:         non-critical section
p2:         wantp ← true
p3:         while wantq
p4:             if turn = 2
p5:                 wantp ← false
p6:                 await turn = 1
p7:                 wantp ← true
p8:         critical section
p9:         turn ← 2
p10:        wantp ← false
```

q

```
        loop forever
q1:         non-critical section
q2:         wantq ← true
q3:         while wantp
q4:             if turn = 1
q5:                 wantq ← false
q6:                 await turn = 2
q7:                 wantq ← true
q8:         critical section
q9:         turn ← 1
q10:        wantq ← false
```

Algorithm 3.11: Critical section problem with test-and-set

```
integer common ← 0
```

p

```
      integer local1
      loop forever
p1:      non-critical section
         repeat
p2:         test-and-set(
               common, local1)
p3:      until local1 = 0
p4:      critical section
p5:      common ← 0
```

q

```
      integer local2
      loop forever
q1:      non-critical section
         repeat
q2:         test-and-set(
               common, local2)
q3:      until local2 = 0
q4:      critical section
q5:      common ← 0
```

Algorithm 3.12: Critical section problem with exchange

```
integer common ← 1
```

p	q
integer local1 ← 0	integer local2 ← 0
loop forever	loop forever
p1: non-critical section	q1: non-critical section
repeat	repeat
p2: exchange(common, local1)	q2: exchange(common, local2)
p3: until local1 = 1	q3: until local2 = 1
p4: critical section	q4: critical section
p5: exchange(common, local1)	q5: exchange(common, local2)

p

```
     integer local1 ← 0
     loop forever
p1:     non-critical section
        repeat
p2:        exchange(common, local1)
p3:     until local1 = 1
p4:     critical section
p5:     exchange(common, local1)
```

q

```
     integer local2 ← 0
     loop forever
q1:     non-critical section
        repeat
q2:        exchange(common, local2)
q3:     until local2 = 1
q4:     critical section
q5:     exchange(common, local2)
```

Algorithm 3.13: Peterson's algorithm

```
                boolean wantp ← false, wantq ← false
                integer last ← 1

                 p                                      q

      loop forever                            loop forever
p1:      non-critical section           q1:      non-critical section
p2:      wantp ← true                   q2:      wantq ← true
p3:      last ← 1                       q3:      last ← 2
p4:      await wantq = false or         q4:      await wantp = false or
                 last = 2                                 last = 1
p5:      critical section               q5:      critical section
p6:      wantp ← false                  q6:      wantq ← false
```

Algorithm 3.14: Manna-Pnueli algorithm

```
integer wantp $\leftarrow 0$, wantq $\leftarrow 0$
```

p

```
      loop forever
p1:      non-critical section
p2:      if wantq $= -1$
             wantp $\leftarrow -1$
          else wantp $\leftarrow 1$
p3:      await wantq $\neq$ wantp
p4:      critical section
p5:      wantp $\leftarrow 0$
```

q

```
      loop forever
q1:      non-critical section
q2:      if wantp $= -1$
             wantq $\leftarrow 1$
          else wantq $\leftarrow -1$
q3:      await wantp $\neq -$ wantq
q4:      critical section
q5:      wantq $\leftarrow 0$
```

Algorithm 3.15: Doran-Thomas algorithm

```
boolean wantp ← false, wantq ← false
integer turn ← 1
```

p

```
     loop forever
p1:     non-critical section
p2:     wantp ← true
p3:     if wantq
p4:        if turn = 2
p5:           wantp ← false
p6:           await turn = 1
p7:           wantp ← true
p8:        await wantq = false
p9:     critical section
p10:    wantp ← false
p11:    turn ← 2
```

q

```
     loop forever
q1:     non-critical section
q2:     wantq ← true
q3:     if wantp
q4:        if turn = 1
q5:           wantq ← false
q6:           await turn = 2
q7:           wantq ← true
q8:        await wantp = false
q9:     critical section
q10:    wantq ← false
q11:    turn ← 1
```