

**M. Ben-Ari**

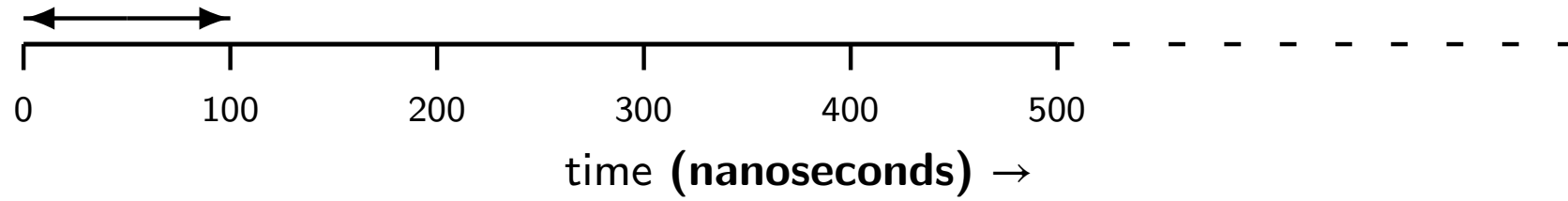
# **Principles of Concurrent and Distributed Programming**

**Second Edition**

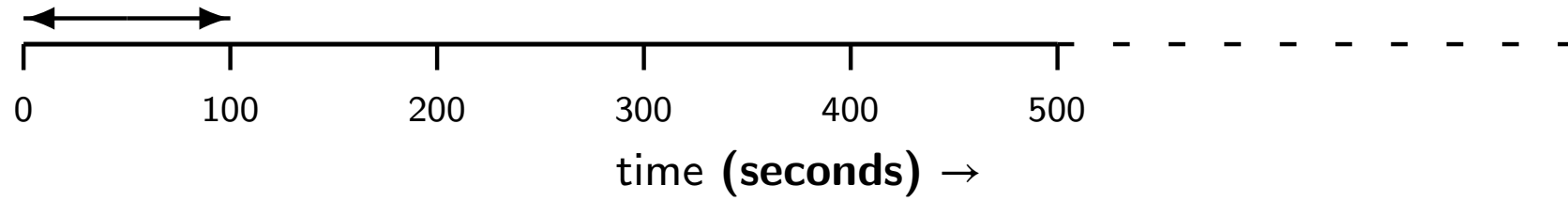
**Addison-Wesley, 2006**

**© Mordechai Ben-Ari 2006**

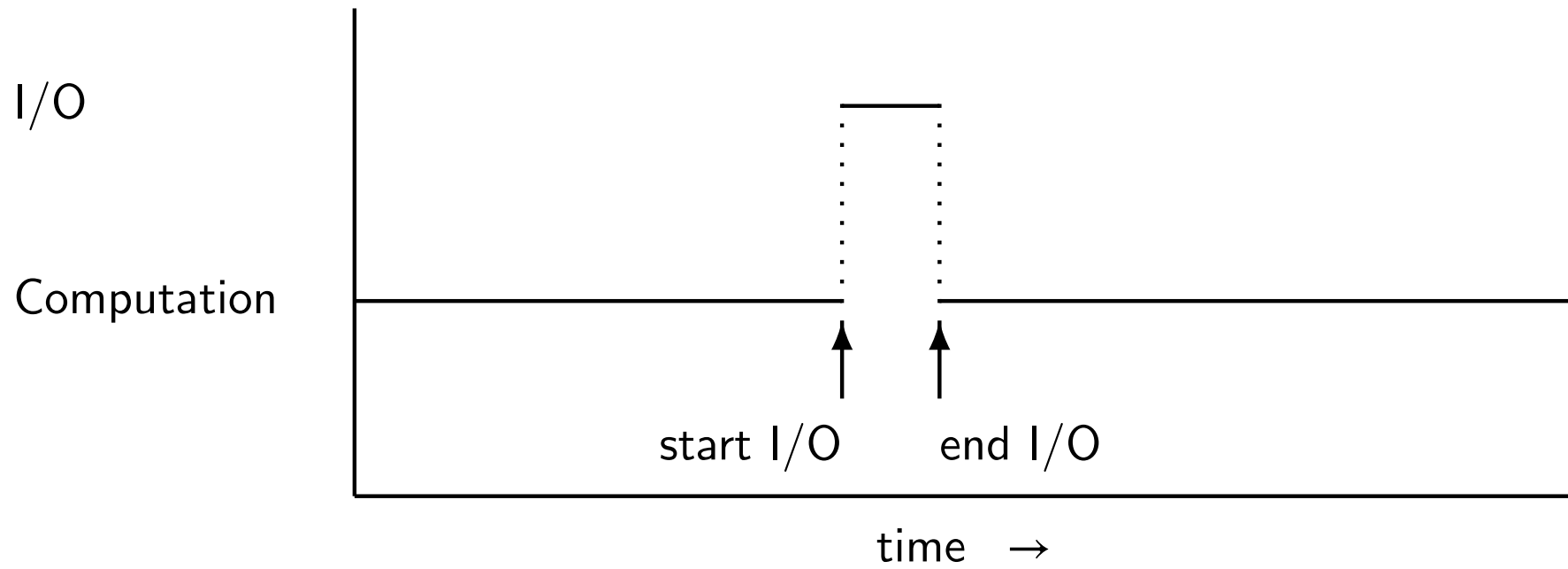
# Computer Time



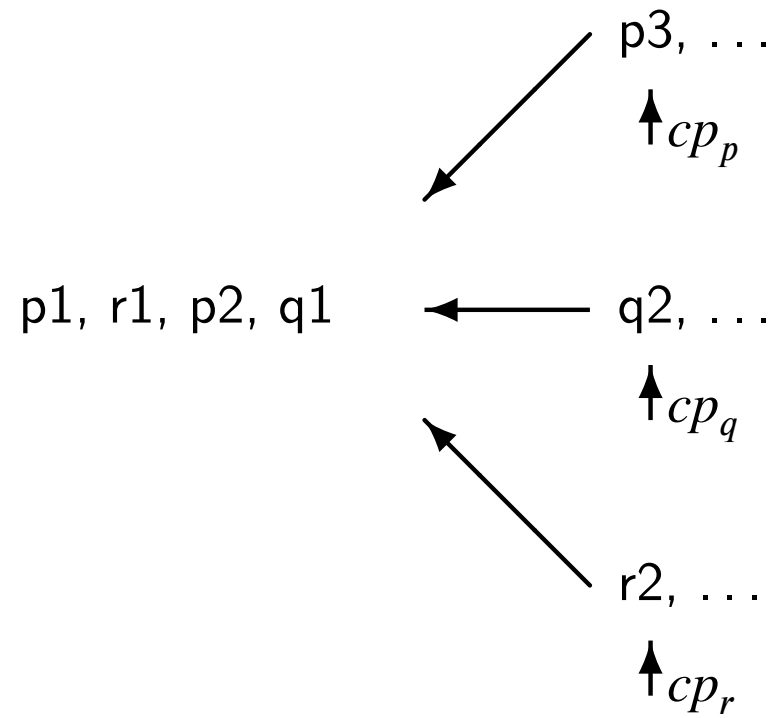
# Human Time



# Concurrency in an Operating System



# Interleaving as Choosing Among Processes



# Possible Interleavings

$p1 \rightarrow q1 \rightarrow p2 \rightarrow q2,$

$p1 \rightarrow q1 \rightarrow q2 \rightarrow p2,$

$p1 \rightarrow p2 \rightarrow q1 \rightarrow q2,$

$q1 \rightarrow p1 \rightarrow q2 \rightarrow p2,$

$q1 \rightarrow p1 \rightarrow p2 \rightarrow q2,$

$q1 \rightarrow q2 \rightarrow p1 \rightarrow p2.$

<b>Algorithm 2.1: Trivial concurrent program</b>	
integer $n \leftarrow 0$	
<b>p</b>	<b>q</b>
integer $k1 \leftarrow 1$ p1: $n \leftarrow k1$	integer $k2 \leftarrow 2$ q1: $n \leftarrow k2$

### Algorithm 2.2: Trivial sequential program

integer  $n \leftarrow 0$

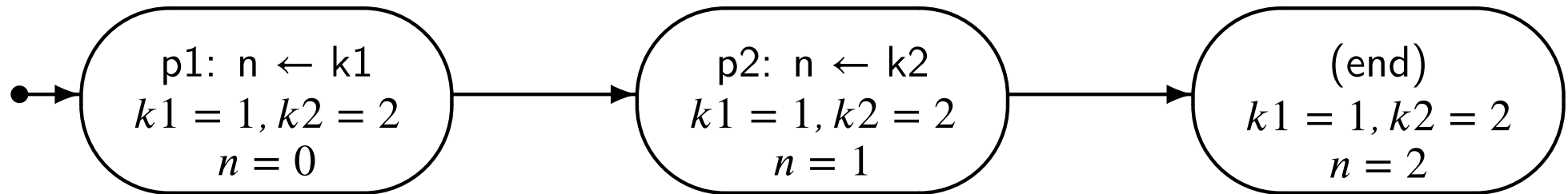
integer  $k1 \leftarrow 1$

integer  $k2 \leftarrow 2$

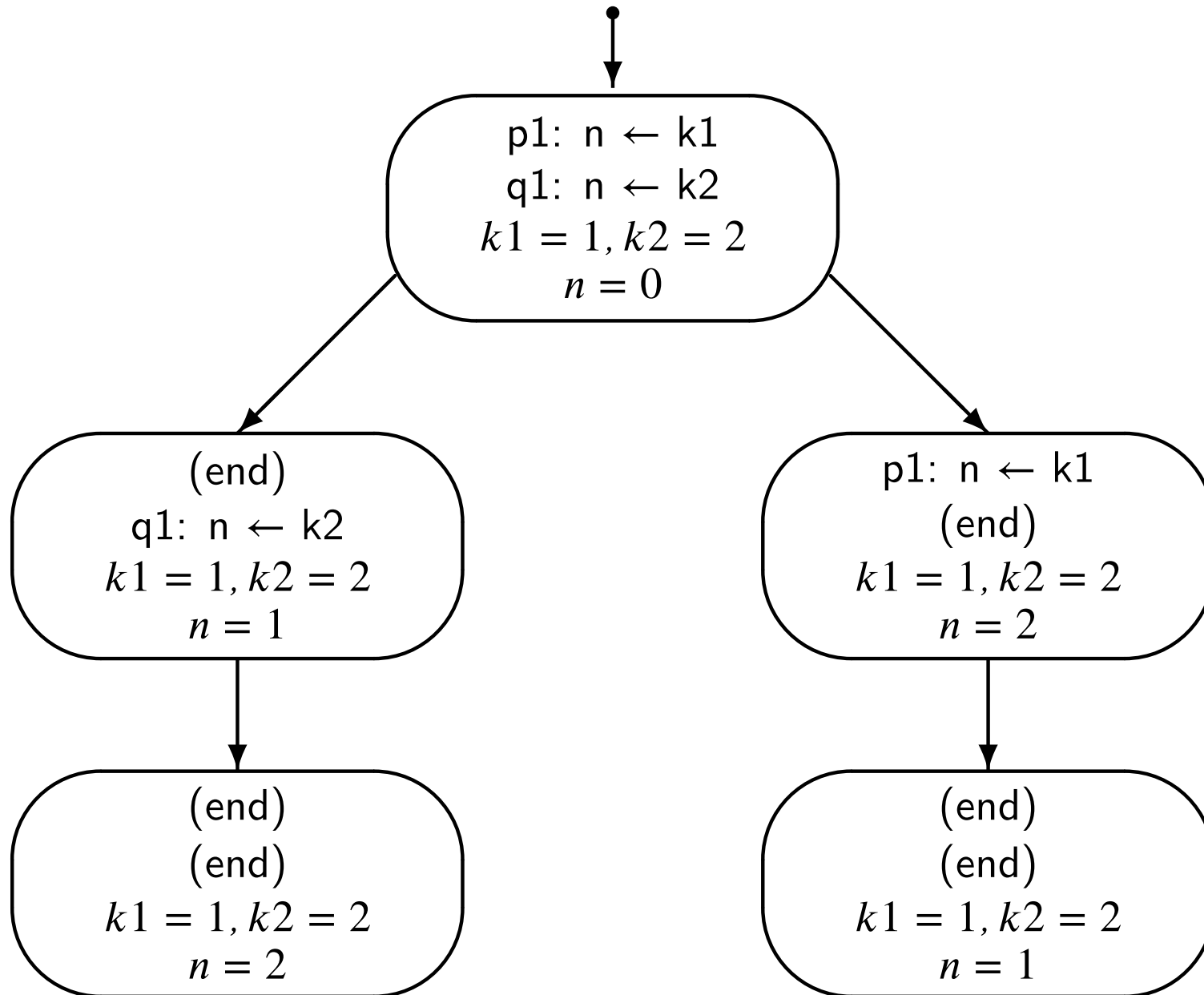
p1:  $n \leftarrow k1$

p2:  $n \leftarrow k2$

# State Diagram for a Sequential Program



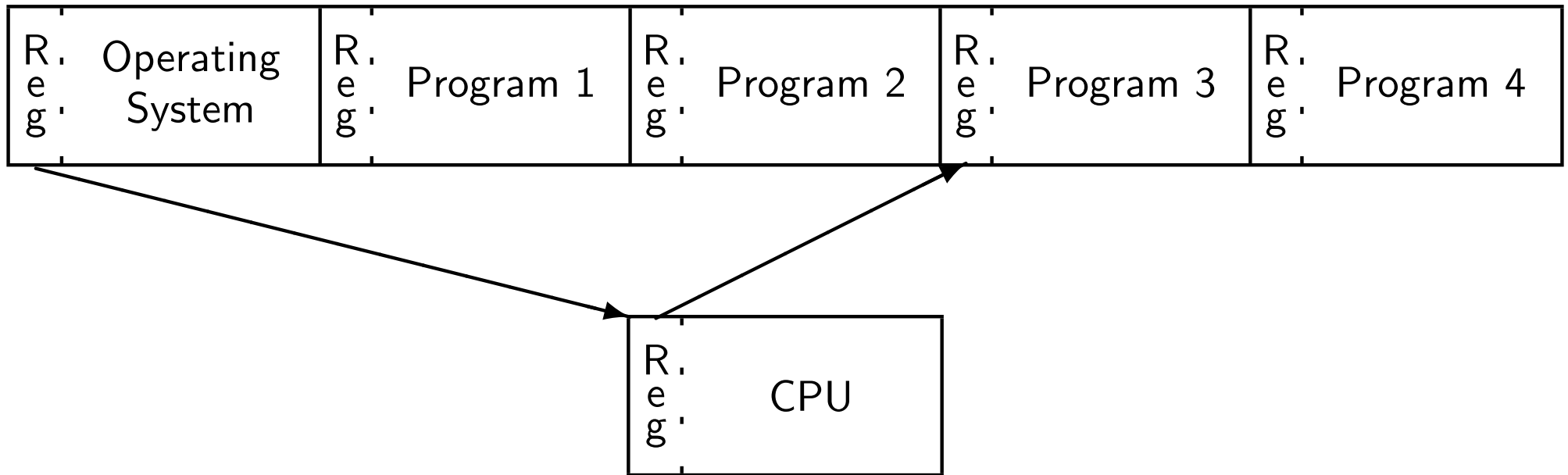
# State Diagram for a Concurrent Program



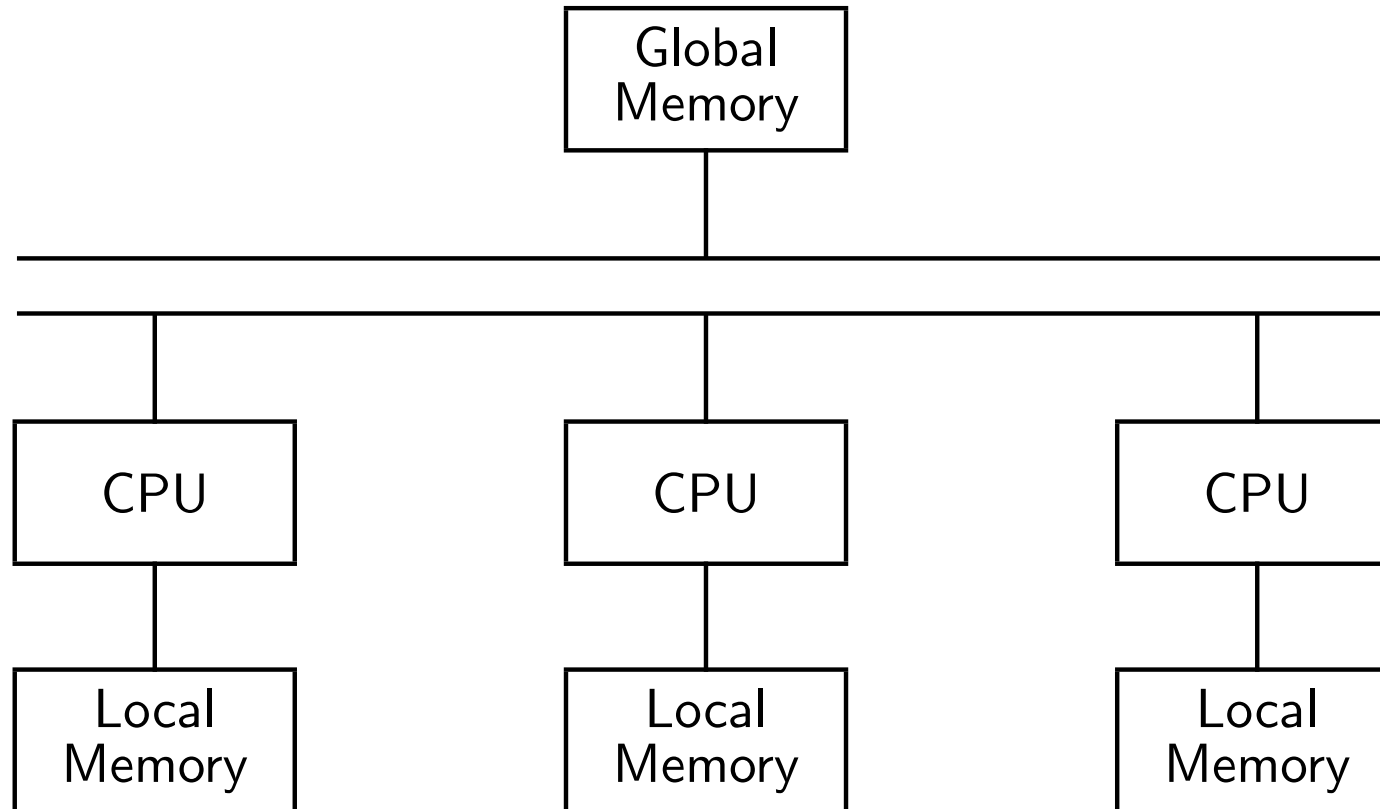
# Scenario for a Concurrent Program

Process p	Process q	n	k1	k2
<b>p1: n ← k1</b>	q1: n ← k2	0	1	2
(end)	<b>q1: n ← k2</b>	1	1	2
(end)	(end)	2	1	2

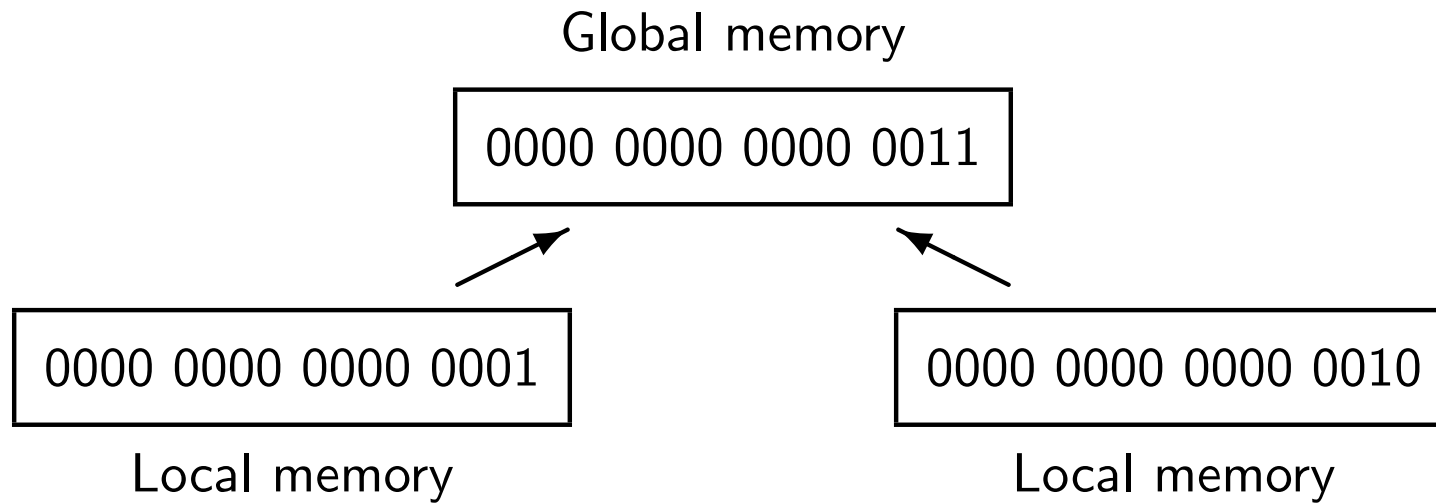
# Multitasking System



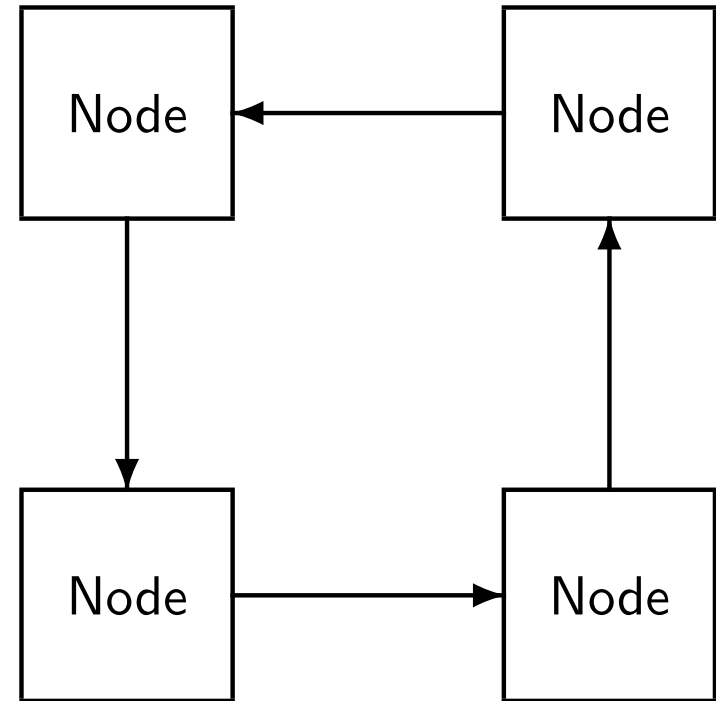
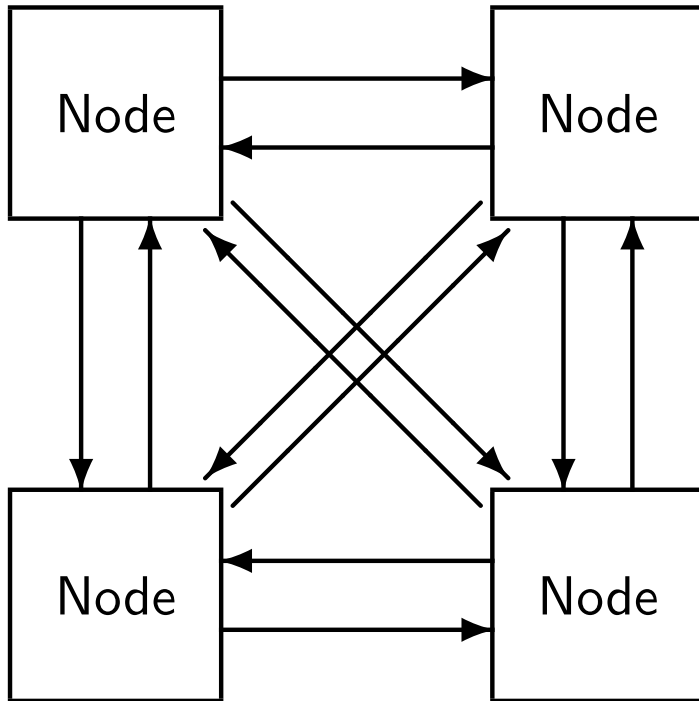
# Multiprocessor Computer



# Inconsistency Caused by Overlapped Execution



# Distributed Systems Architecture



<b>Algorithm 2.3: Atomic assignment statements</b>	
integer $n \leftarrow 0$	
<b>p</b>	<b>q</b>
$p1: n \leftarrow n + 1$	$q1: n \leftarrow n + 1$

# Scenario for Atomic Assignment Statements

Process p	Process q	n
<b>p1: n←n+1</b>	q1: n←n+1	0
(end)	<b>q1: n←n+1</b>	1
(end)	(end)	2

Process p	Process q	n
p1: n←n+1	<b>q1: n←n+1</b>	0
<b>p1: n←n+1</b>	(end)	1
(end)	(end)	2

### Algorithm 2.4: Assignment statements with one global reference

integer  $n \leftarrow 0$

**p**

**q**

integer temp

integer temp

p1: temp  $\leftarrow$  n

q1: temp  $\leftarrow$  n

p2: n  $\leftarrow$  temp + 1

q2: n  $\leftarrow$  temp + 1

# Correct Scenario for Assignment Statements

Process p	Process q	n	p.temp	q.temp
<b>p1: temp←n</b>	q1: temp←n	0	?	?
<b>p2: n←temp+1</b>	q1: temp←n	0	0	?
(end)	<b>q1: temp←n</b>	1	0	?
(end)	<b>q2: n←temp+1</b>	1	0	1
(end)	(end)	2	0	1

# Incorrect Scenario for Assignment Statements

Process p	Process q	n	p.temp	q.temp
<b>p1: temp←n</b>	q1: temp←n	0	?	?
p2: n←temp+1	<b>q1: temp←n</b>	0	0	?
<b>p2: n←temp+1</b>	q2: n←temp+1	0	0	0
(end)	<b>q2: n←temp+1</b>	1	0	0
(end)	(end)	1	0	0

### Algorithm 2.5: Stop the loop A

integer  $n \leftarrow 0$

boolean  $\text{flag} \leftarrow \text{false}$

**p**

p1: while  $\text{flag} = \text{false}$

p2:  $n \leftarrow 1 - n$

**q**

q1:  $\text{flag} \leftarrow \text{true}$

q2:

### Algorithm 2.6: Assignment statement for a register machine

integer  $n \leftarrow 0$

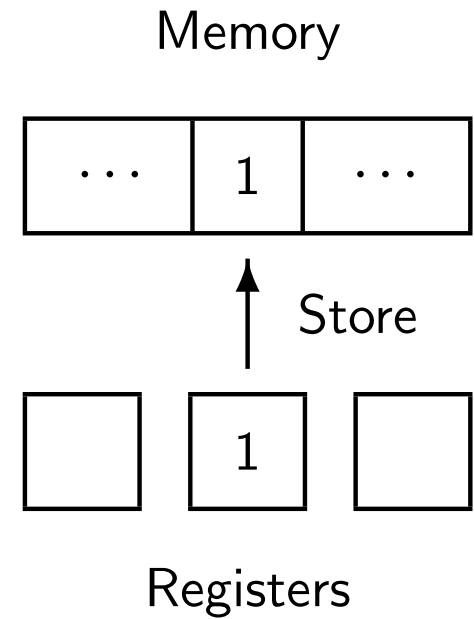
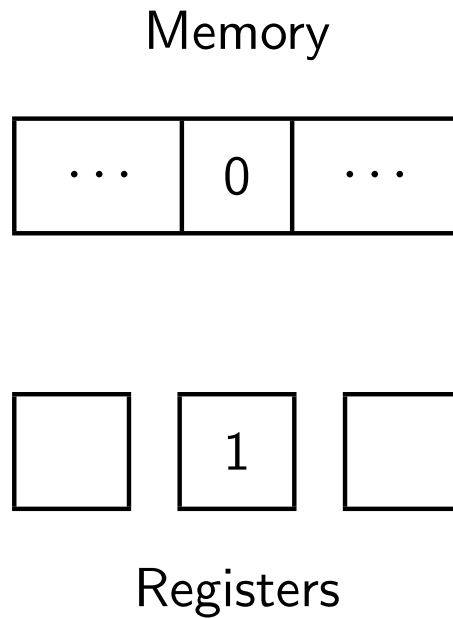
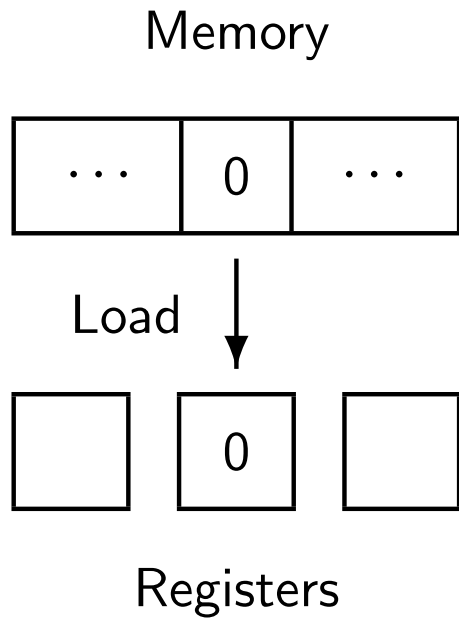
**p**

p1: load R1,n  
p2: add R1,#1  
p3: store R1,n

**q**

q1: load R1,n  
q2: add R1,#1  
q3: store R1,n

# Register Machine



# Scenario for a Register Machine

Process p	Process q	n	p.R1	q.R1
<b>p1: load R1,n</b>	q1: load R1,n	0	?	?
p2: add R1,#1	<b>q1: load R1,n</b>	0	0	?
<b>p2: add R1,#1</b>	q2: add R1,#1	0	0	0
p3: store R1,n	<b>q2: add R1,#1</b>	0	1	0
<b>p3: store R1,n</b>	q3: store R1,n	0	1	1
(end)	<b>q3: store R1,n</b>	1	1	1
(end)	(end)	1	1	1

### Algorithm 2.7: Assignment statement for a stack machine

integer  $n \leftarrow 0$

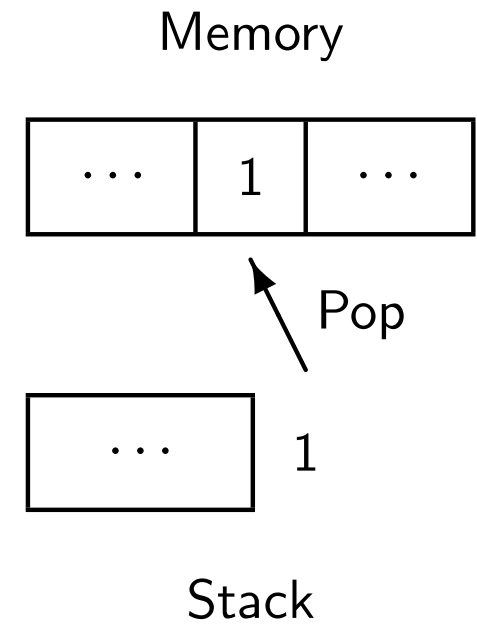
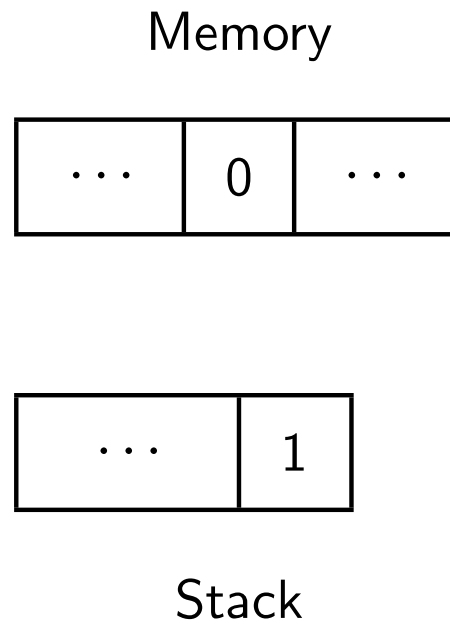
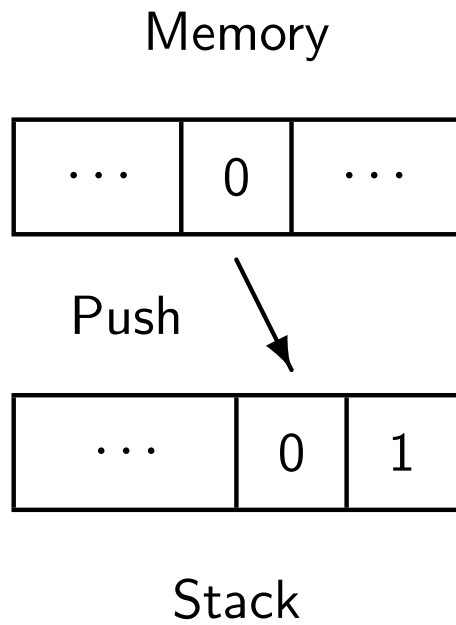
**p**

p1: push n  
p2: push #1  
p3: add  
p4: pop n

**q**

q1: push n  
q2: push #1  
q3: add  
q4: pop n

# Stack Machine



### Algorithm 2.8: Volatile variables

integer  $n \leftarrow 0$

**p**

**q**

integer local1, local2

integer local

p1:  $n \leftarrow \text{some expression}$

q1:  $\text{local} \leftarrow n + 6$

p2: *computation not using  $n$*

q2:

p3:  $\text{local1} \leftarrow (n + 5) * 7$

q3:

p4:  $\text{local2} \leftarrow n + 5$

q4:

p5:  $n \leftarrow \text{local1} * \text{local2}$

q5:

### Algorithm 2.9: Concurrent counting algorithm

integer  $n \leftarrow 0$

**p**

**q**

integer temp

integer temp

p1: do 10 times

q1: do 10 times

p2: temp  $\leftarrow$  n

q2: temp  $\leftarrow$  n

p3: n  $\leftarrow$  temp + 1

q3: n  $\leftarrow$  temp + 1

# Concurrent Program in Pascal

```
1  program count;  
2  var n: integer := 0;  
3  
4  procedure p;  
5  var temp, i: integer;  
6  begin  
7    for i := 1 to 10 do  
8      begin  
9        temp := n;  
10       n := temp + 1  
11      end  
12 end;  
13  
14  
15
```

# Concurrent Program in Pascal

```
16  procedure q;  
17  var temp, i: integer;  
18  begin  
19    for i := 1 to 10 do  
20      begin  
21        temp := n;  
22        n := temp + 1  
23      end  
24  end;  
25  
26  begin  
27    cobegin p; q coend;  
28    writeln(' The value of n is ', n)  
29  end.
```

# Concurrent Program in C

```
1  int n = 0;  
2  
3  void p() {  
4      int temp, i;  
5      for (i = 0; i < 10; i++) {  
6          temp = n;  
7          n = temp + 1;  
8      }  
9  }  
10  
11  
12  
13  
14  
15
```

# Concurrent Program in C

```
16  void q() {
17      int temp, i;
18      for (i = 0; i < 10; i++) {
19          temp = n;
20          n = temp + 1;
21      }
22  }
23
24  void main() {
25      cobegin { p(); q(); }
26      cout << "The value of n is " << n << "\n";
27  }
```

# Concurrent Program in Ada

```
1  with Ada.Text_IO; use Ada.Text_IO;
2  procedure Count is
3      N: Integer := 0;
4      pragma Volatile(N);
5
6      task type Count_Task;
7      task body Count_Task is
8          Temp: Integer;
9      begin
10         for I in 1..10 loop
11             Temp := N;
12             N := Temp + 1;
13         end loop;
14     end Count_Task;
15
```

# Concurrent Program in Ada

```
16 begin
17   declare
18     P, Q: Count_Task;
19   begin
20     null;
21   end;
22   Put_Line("The value of N is " & Integer'Image(N));
23 end Count;
```

# Concurrent Program in Java

```
1  class Count extends Thread {  
2      static volatile int n = 0;  
3  
4      public void run() {  
5          int temp;  
6          for (int i = 0; i < 10; i++) {  
7              temp = n;  
8              n = temp + 1;  
9          }  
10     }  
11  
12  
13  
14  
15
```

# Concurrent Program in Java

```
16    public static void main(String[] args) {  
17        Count p = new Count();  
18        Count q = new Count();  
19        p.start ();  
20        q.start ();  
21        try {  
22            p.join ();  
23            q.join ();  
24        }  
25        catch (InterruptedException e) { }  
26        System.out.println ("The value of n is " + n);  
27    }  
28 }
```

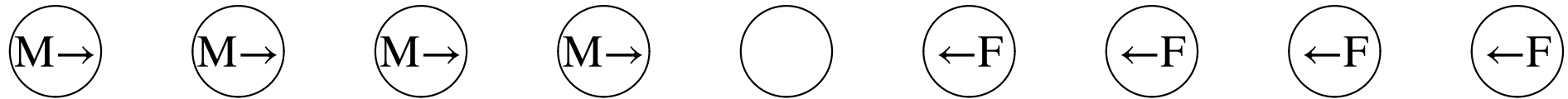
# Concurrent Program in Promela

```
1  #include "for.h"  
2  #define TIMES 10  
3  byte    n = 0;  
4  
5  proctype P() {  
6      byte temp;  
7      for (i,1, TIMES)  
8          temp = n;  
9          n = temp + 1  
10     rof (i)  
11 }  
12  
13  
14  
15
```

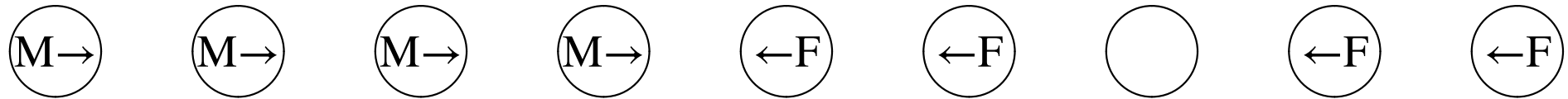
# Concurrent Program in Promela

```
16  init {  
17      atomic {  
18          run P();  
19          run P()  
20      }  
21      (_nr_pr == 1);  
22      printf ("MSC: The value is %d\n", n)  
23  }
```

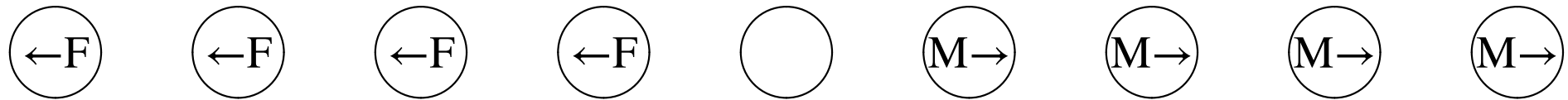
# Frog Puzzle



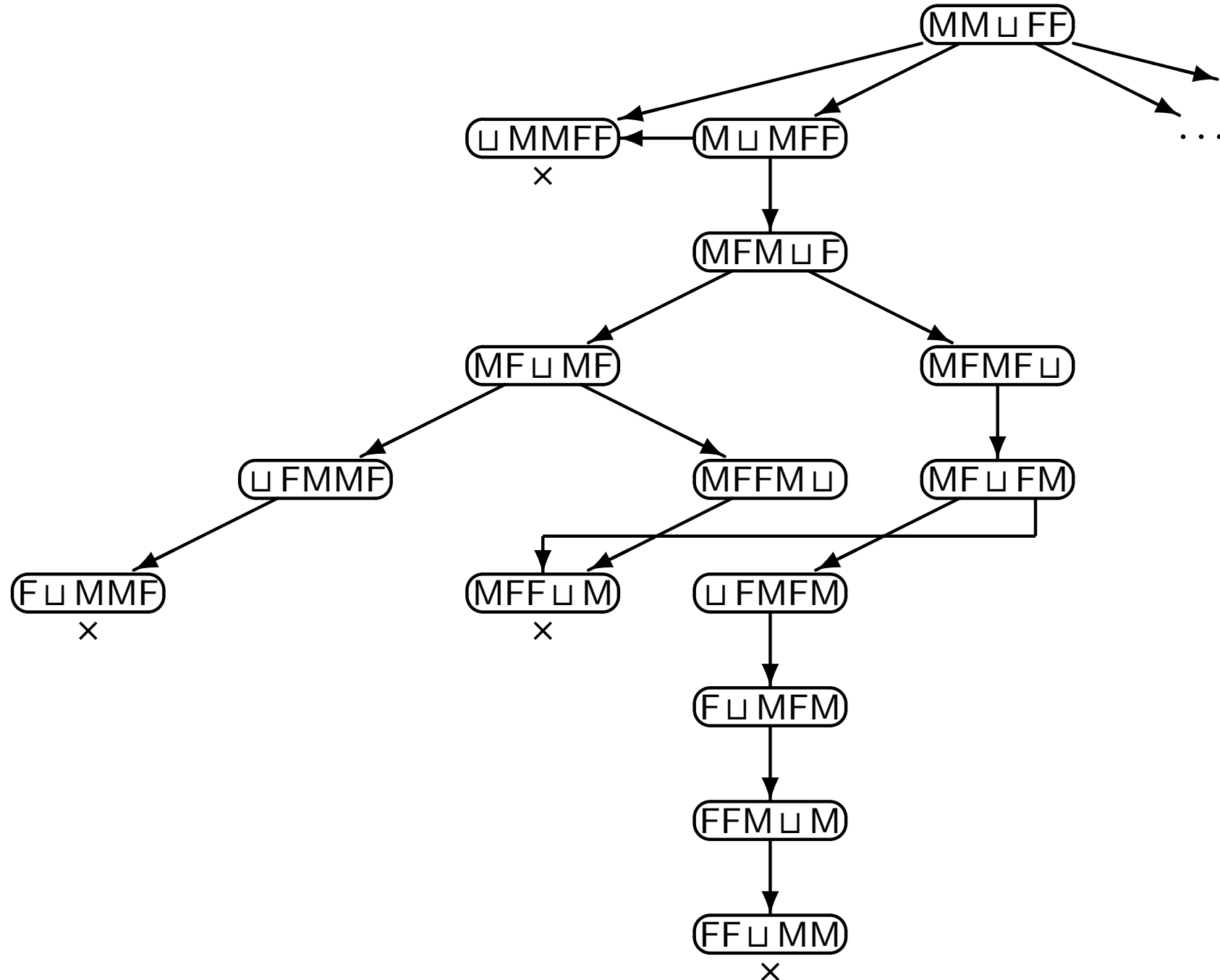
# One Step of the Frog Puzzle



# Final State of the Frog Puzzle



# (Partial) State Diagram for the Frog Puzzle



### Algorithm 2.10: Incrementing and decrementing

integer  $n \leftarrow 0$

**p**

**q**

integer temp

integer temp

p1: do K times

q1: do K times

p2: temp  $\leftarrow$  n

q2: temp  $\leftarrow$  n

p3: n  $\leftarrow$  temp + 1

q3: n  $\leftarrow$  temp - 1

### Algorithm 2.11: Zero A

boolean found

**p**

**q**

integer  $i \leftarrow 0$

integer  $j \leftarrow 1$

p1: found  $\leftarrow$  false

q1: found  $\leftarrow$  false

p2: while not found

q2: while not found

p3:  $i \leftarrow i + 1$

q3:  $j \leftarrow j - 1$

p4: found  $\leftarrow f(i) = 0$

q4: found  $\leftarrow f(j) = 0$

### Algorithm 2.12: Zero B

boolean found  $\leftarrow$  false

**p**

**q**

integer  $i \leftarrow 0$

integer  $j \leftarrow 1$

p1: while not found

q1: while not found

p2:  $i \leftarrow i + 1$

q2:  $j \leftarrow j - 1$

p3: found  $\leftarrow f(i) = 0$

q3: found  $\leftarrow f(j) = 0$

### Algorithm 2.13: Zero C

boolean found  $\leftarrow$  false

**p**

**q**

integer  $i \leftarrow 0$

integer  $j \leftarrow 1$

p1: while not found

q1: while not found

p2:      $i \leftarrow i + 1$

q2:      $j \leftarrow j - 1$

p3:     if  $f(i) = 0$

q3:     if  $f(j) = 0$

p4:         found  $\leftarrow$  true

q4:         found  $\leftarrow$  true

### Algorithm 2.14: Zero D

boolean found  $\leftarrow$  false

integer turn  $\leftarrow$  1

**p**

**q**

integer  $i \leftarrow 0$

integer  $j \leftarrow 1$

p1: while not found

q1: while not found

p2:     await turn = 1

q2:     await turn = 2

        turn  $\leftarrow$  2

        turn  $\leftarrow$  1

p3:      $i \leftarrow i + 1$

q3:      $j \leftarrow j - 1$

p4:     if  $f(i) = 0$

q4:     if  $f(j) = 0$

p5:         found  $\leftarrow$  true

q5:         found  $\leftarrow$  true

### Algorithm 2.15: Zero E

boolean found  $\leftarrow$  false

integer turn  $\leftarrow$  1

**p**

integer  $i \leftarrow 0$

p1: while not found

p2:     await turn = 1

        turn  $\leftarrow$  2

p3:      $i \leftarrow i + 1$

p4:     if  $f(i) = 0$

p5:         found  $\leftarrow$  true

p6:     turn  $\leftarrow$  2

**q**

integer  $j \leftarrow 1$

q1: while not found

q2:     await turn = 2

        turn  $\leftarrow$  1

q3:      $j \leftarrow j - 1$

q4:     if  $f(j) = 0$

q5:         found  $\leftarrow$  true

q6:     turn  $\leftarrow$  1

### Algorithm 2.16: Concurrent algorithm A

integer array [1..10] C ← ten *distinct* initial values

integer array [1..10] D

integer myNumber, count

p1: myNumber ← C[i]

p2: count ← number of elements of C less than myNumber

p3: D[count + 1] ← myNumber

### Algorithm 2.17: Concurrent algorithm B

integer  $n \leftarrow 0$

**p**

**q**

p1: while  $n < 2$

p2:     write( $n$ )

q1:  $n \leftarrow n + 1$

q2:  $n \leftarrow n + 1$

### Algorithm 2.18: Concurrent algorithm C

integer  $n \leftarrow 1$

**p**

**q**

p1: while  $n < 1$

p2:  $n \leftarrow n + 1$

q1: while  $n \geq 0$

q2:  $n \leftarrow n - 1$

### Algorithm 2.19: Stop the loop B

integer  $n \leftarrow 0$   
boolean  $\text{flag} \leftarrow \text{false}$

**p**

p1: while  $\text{flag} = \text{false}$   
p2:      $n \leftarrow 1 - n$   
p3:

**q**

q1: while  $\text{flag} = \text{false}$   
q2:     if  $n = 0$   
q3:          $\text{flag} \leftarrow \text{true}$

### Algorithm 2.20: Stop the loop C

integer  $n \leftarrow 0$   
boolean  $\text{flag} \leftarrow \text{false}$

**p**

p1: while  $\text{flag} = \text{false}$   
p2:      $n \leftarrow 1 - n$

**q**

q1: while  $n = 0$  // Do nothing  
q2:      $\text{flag} \leftarrow \text{true}$

### Algorithm 2.21: Welfare crook problem

integer array[0..N] a, b, c  $\leftarrow$  ... (as required)

integer i  $\leftarrow$  0, j  $\leftarrow$  0, k  $\leftarrow$  0

loop

p1: if condition-1

p2: i  $\leftarrow$  i + 1

p3: else if condition-2

p4: j  $\leftarrow$  j + 1

p5: else if condition-3

p6: k  $\leftarrow$  k + 1

else exit loop