departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Monitoring Concurrency Errors: Detection of Deadlocks, Atomicity Violations, and Data Races (1)

Concurrency and Parallelism — 2017-18

Master in Computer Science

(Mestrado Integrado em Eng. Informática)

Joao Lourenço <joao.lourenco@fct.unl.pt>

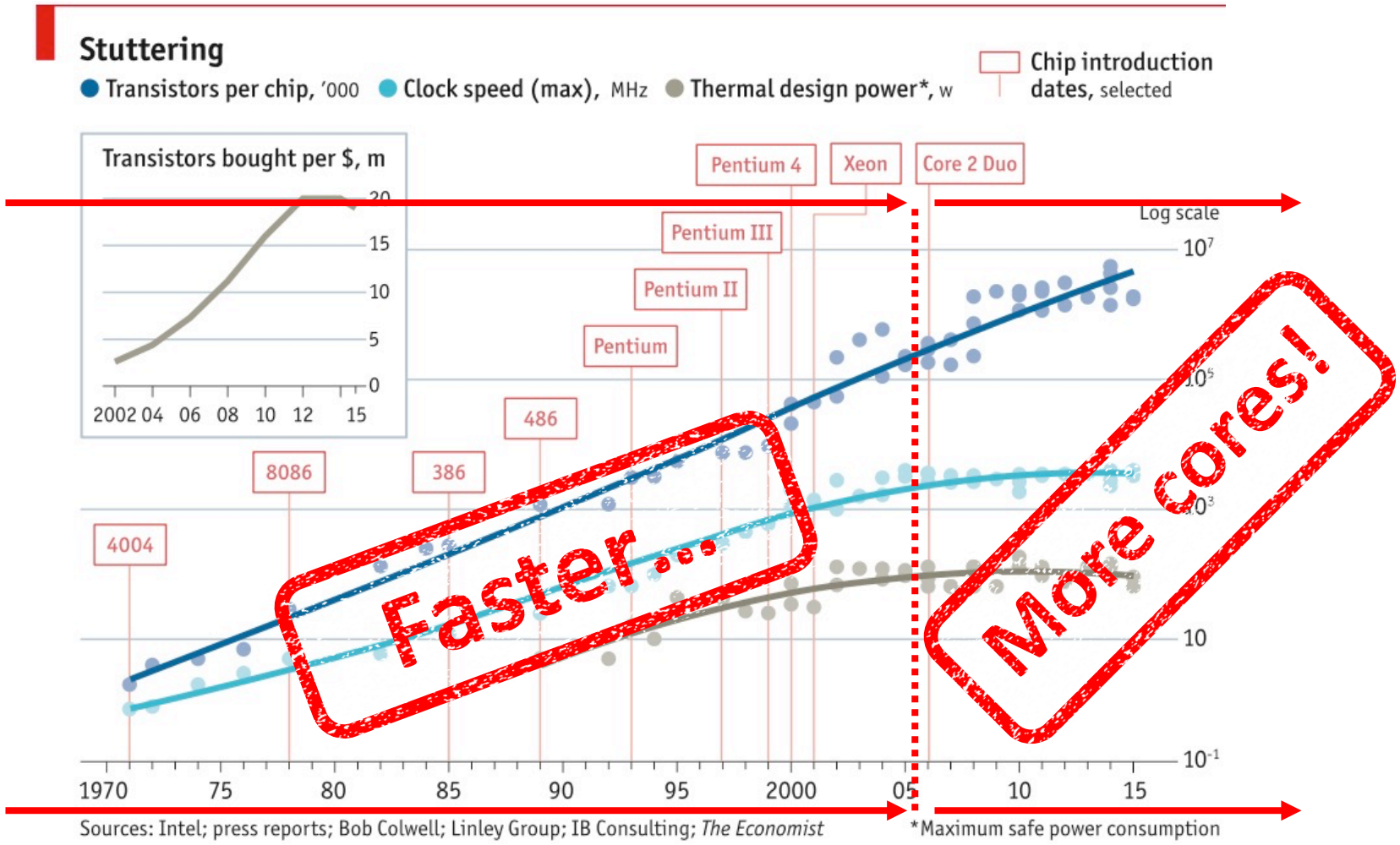
Agenda

- Why are we here?
- Concurrency Anomalies
- Assigning Semantics to Concurrent Programs
- Concurrency Errors
  - Detection of data races
  - Detection of high-level data races and stale value errors
  - Detection of deadlocks

Why are we here?

It is Moore's fault! ;)

Moore's Law


Stuttering
Transistors per chip, '000
Clock speed (max), MHz
Thermal design power*, w
Chip introduction dates, selected
Transistors bought per $, m
Pentium 4
Xeon
Core 2 Duo
Pentium III
Pentium II
Pentium
486
8086
386
4004
Log scale
Faster...
More cores!
1970
75
80
85
90
95
2000
05
10
15
Sources: Intel; press reports; Bob Colwell; Linley Group; IB Consulting; The Economist
*Maximum safe power consumption

Is there a problem?

To use all the processor cores…

…we need multiple cooperating processes!

But… Is there a problem?

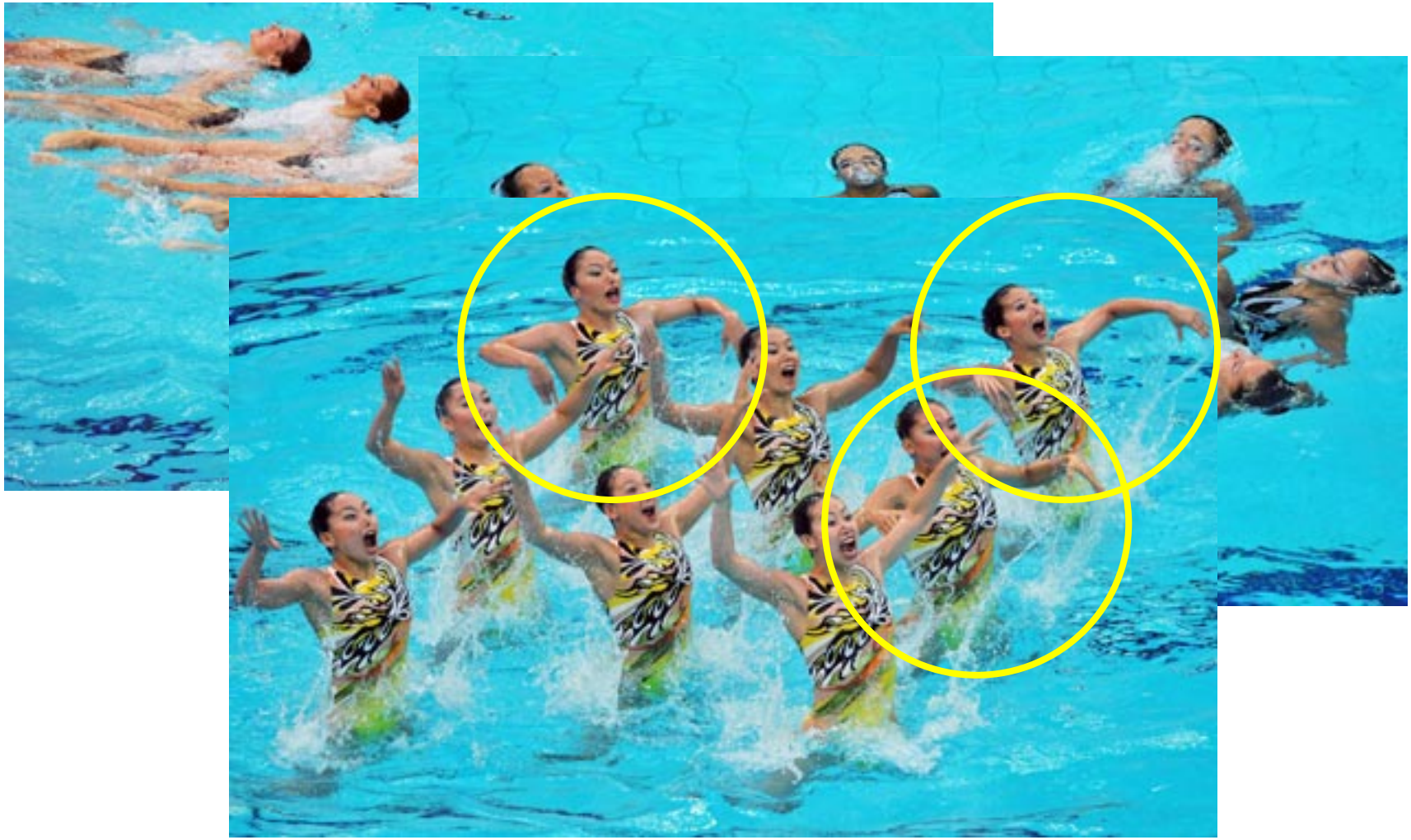
But… Is there a problem?

Concurrency Anomalies

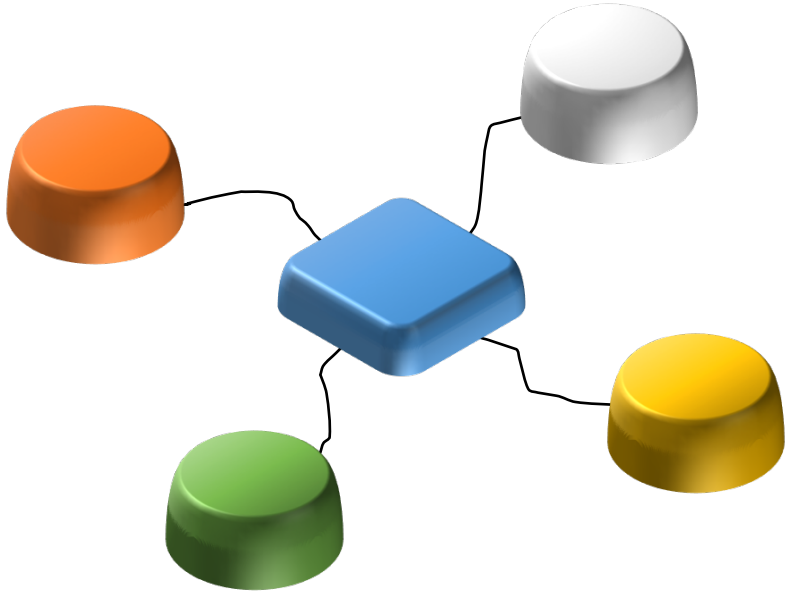
What and why?

Single Process

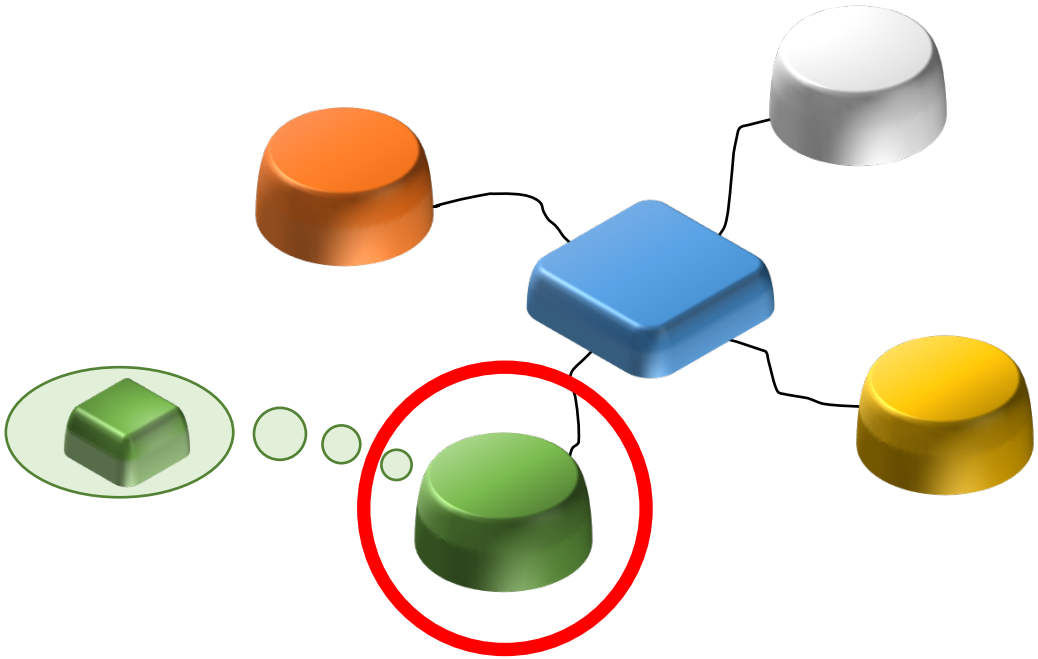
Multiple Processes



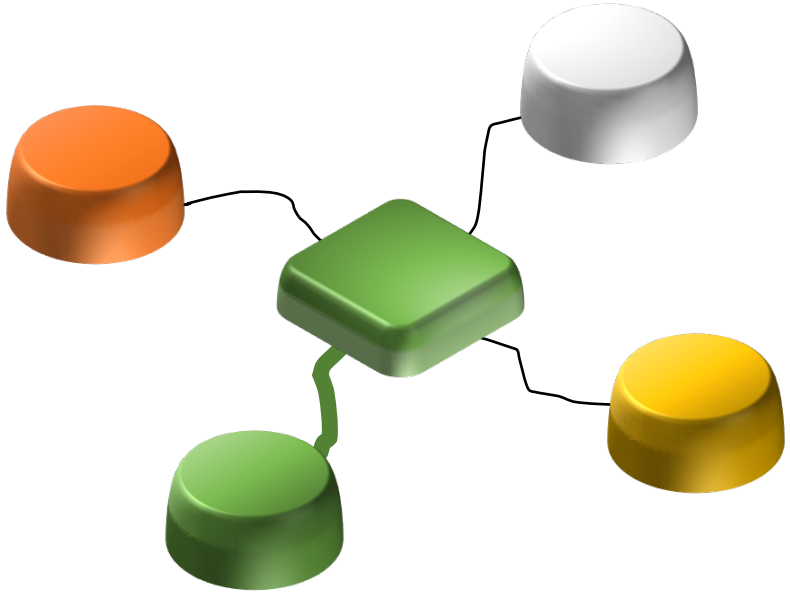
Sharing Resources (unsync.)



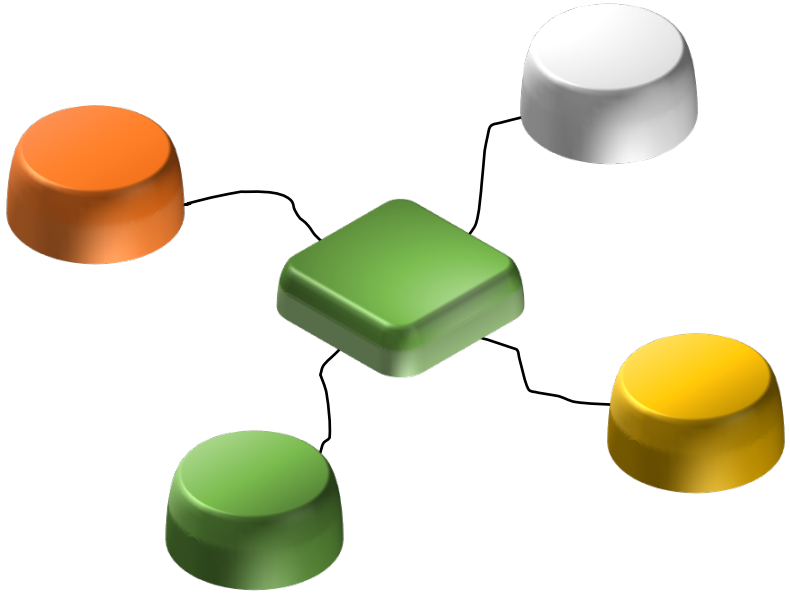
Sharing Resources (unsync.)



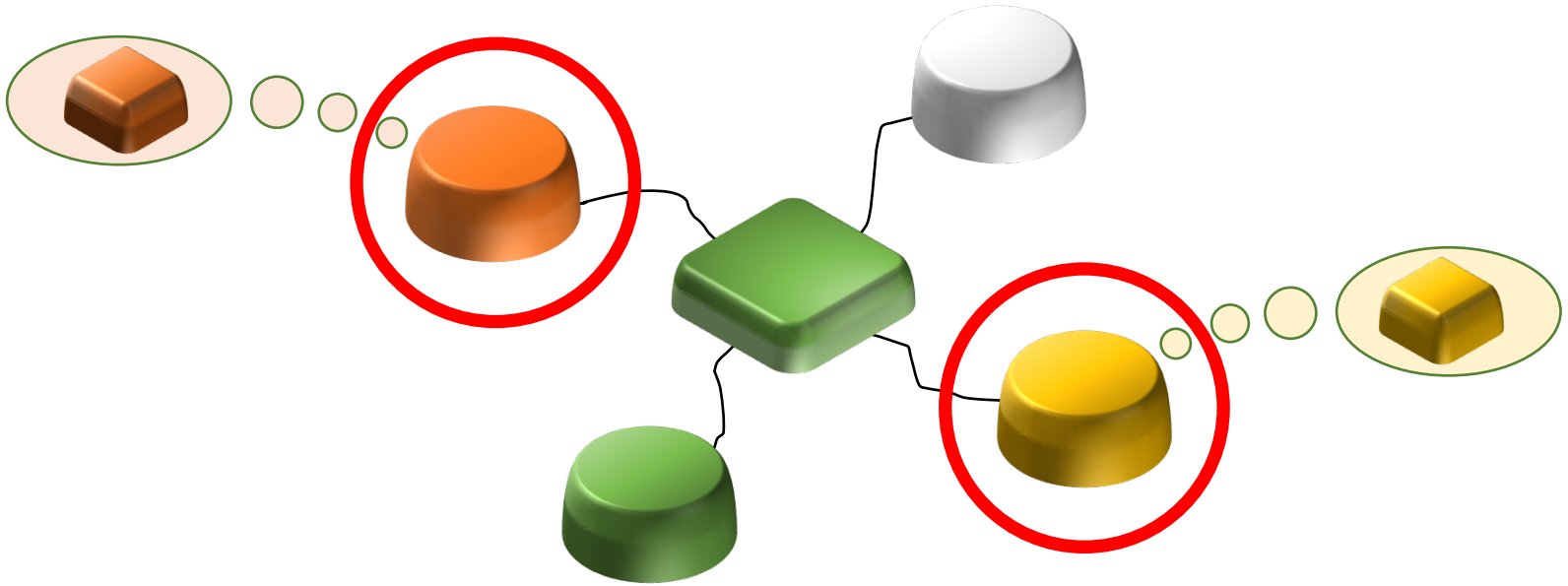
Sharing Resources (unsync.)



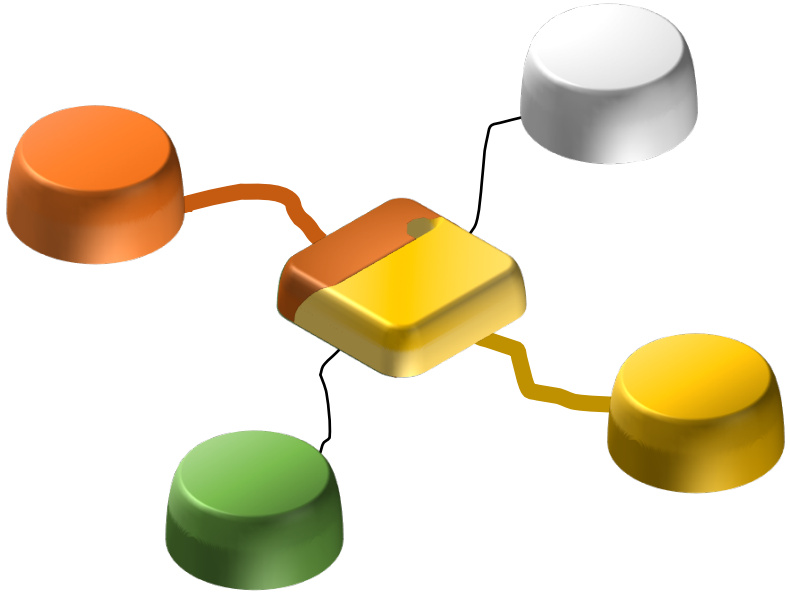
Sharing Resources (unsync.)



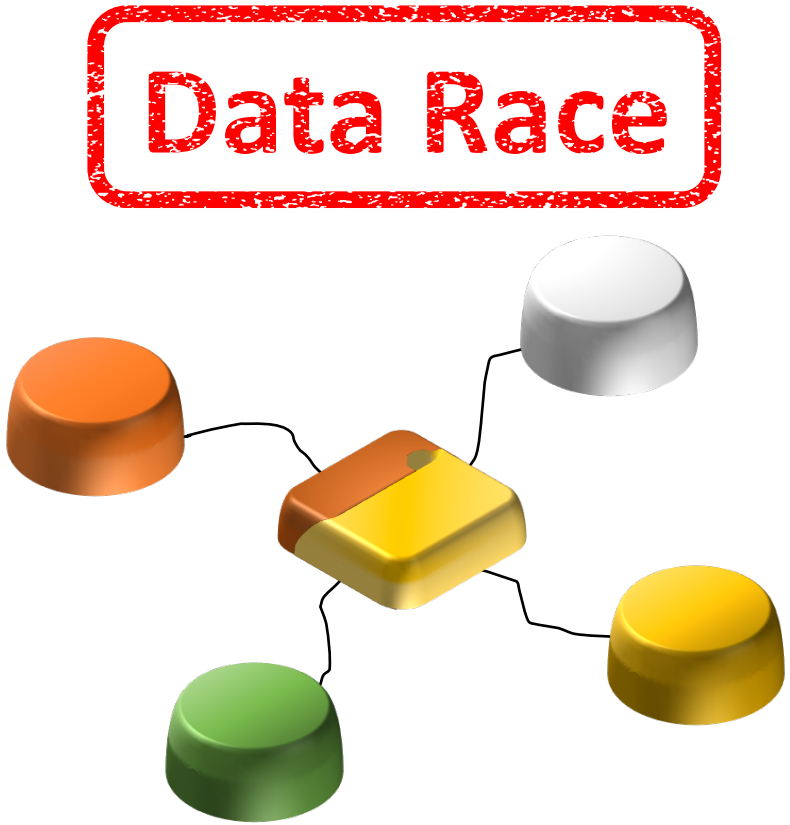
Sharing Resources (unsync.)



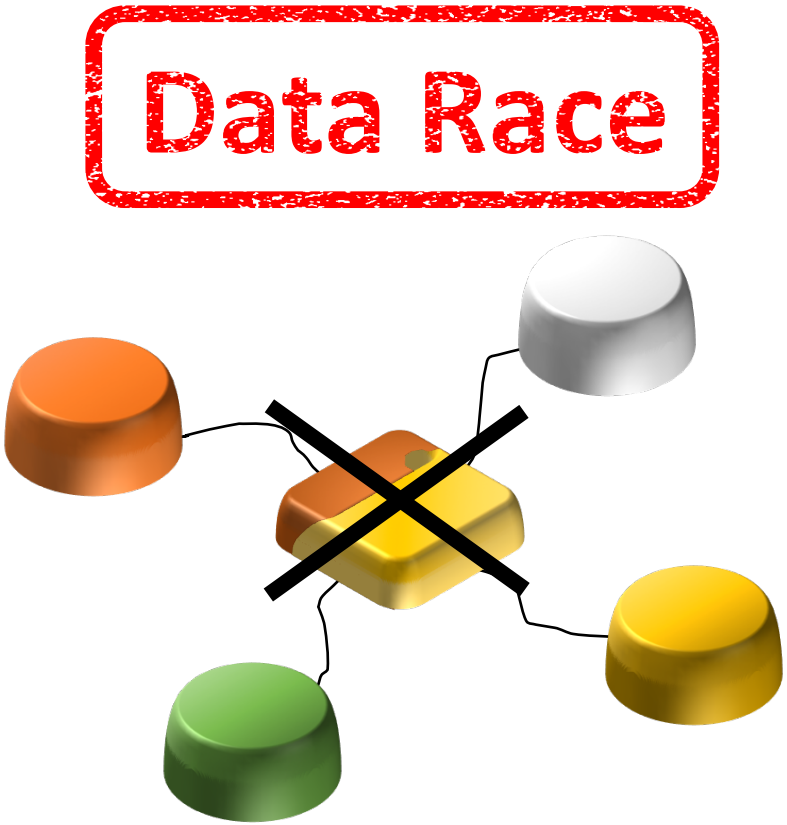
Sharing Resources (unsync.)



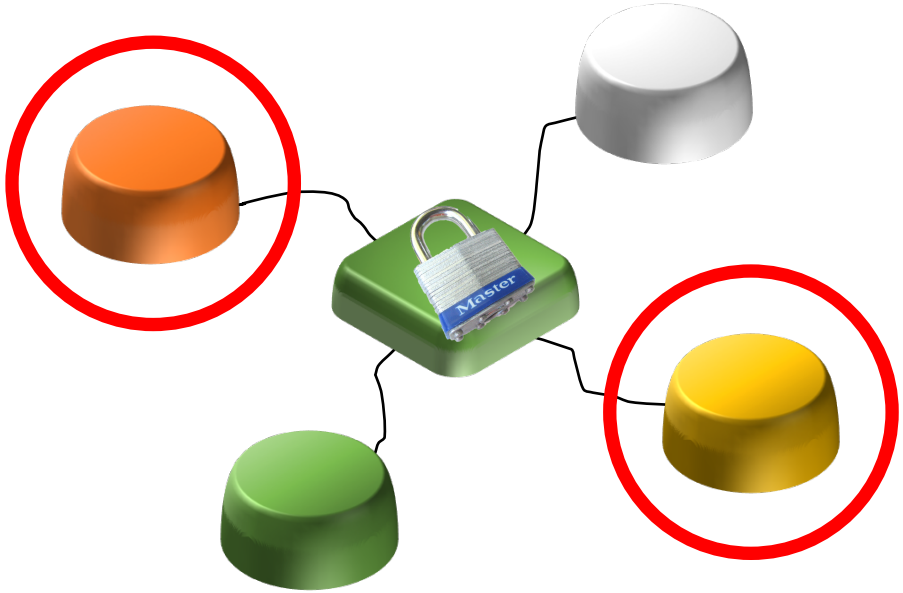
Sharing Resources (unsync.)



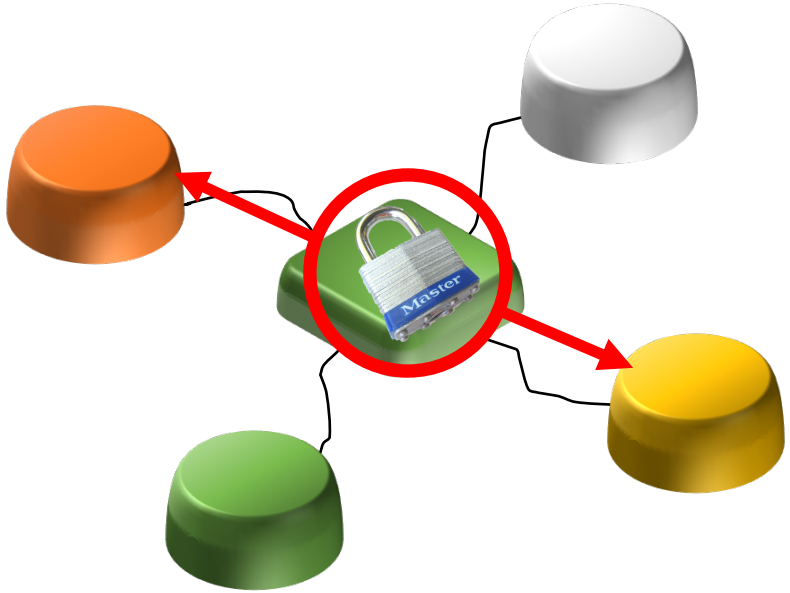
How to avoid Data Races?



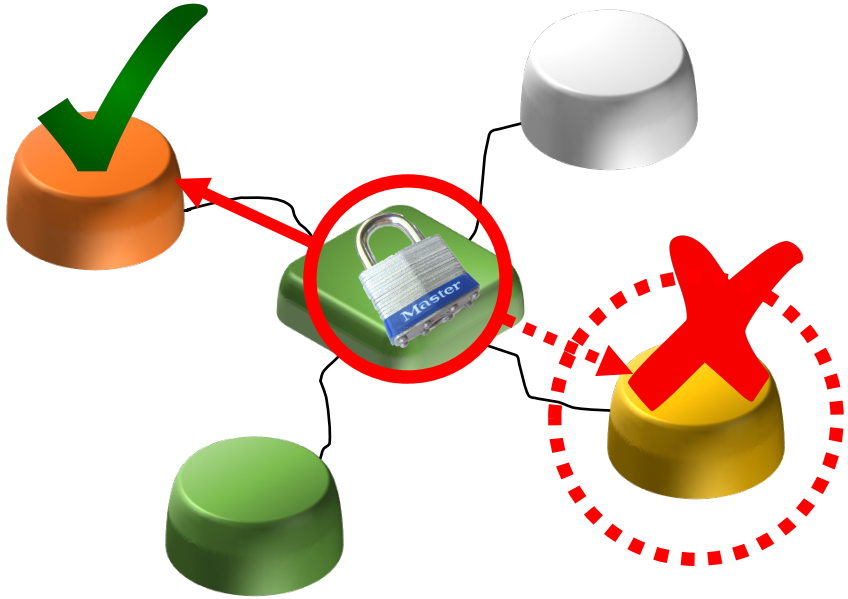
Sharing Resources (sync.)



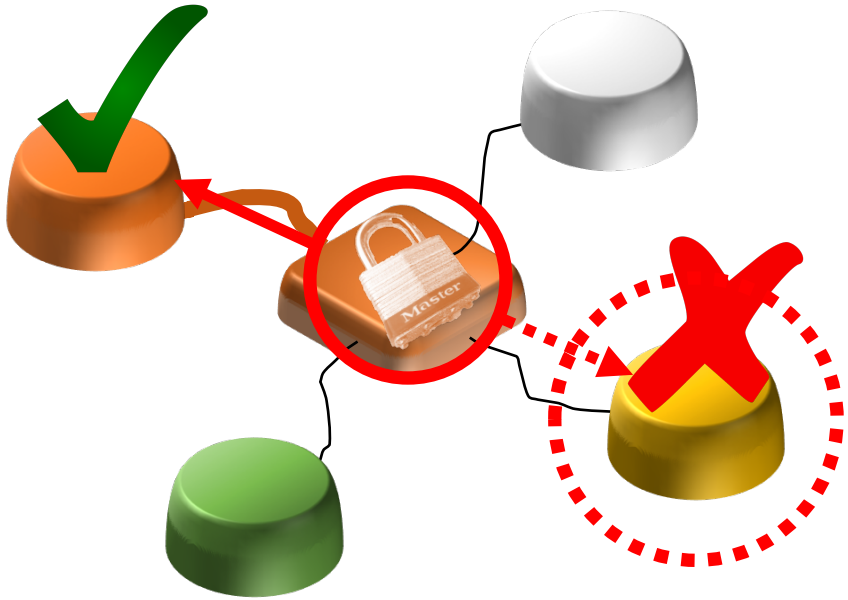
Sharing Resources (sync.)



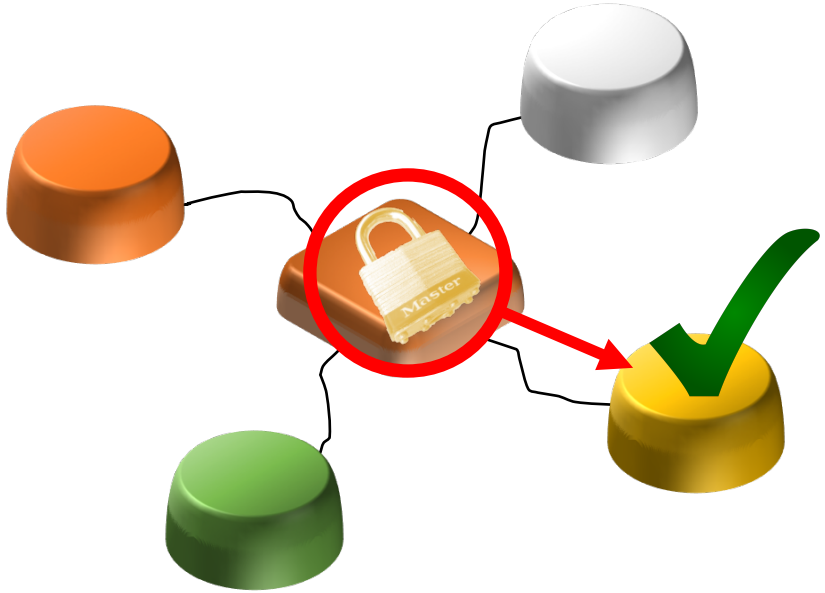
Sharing Resources (sync.)



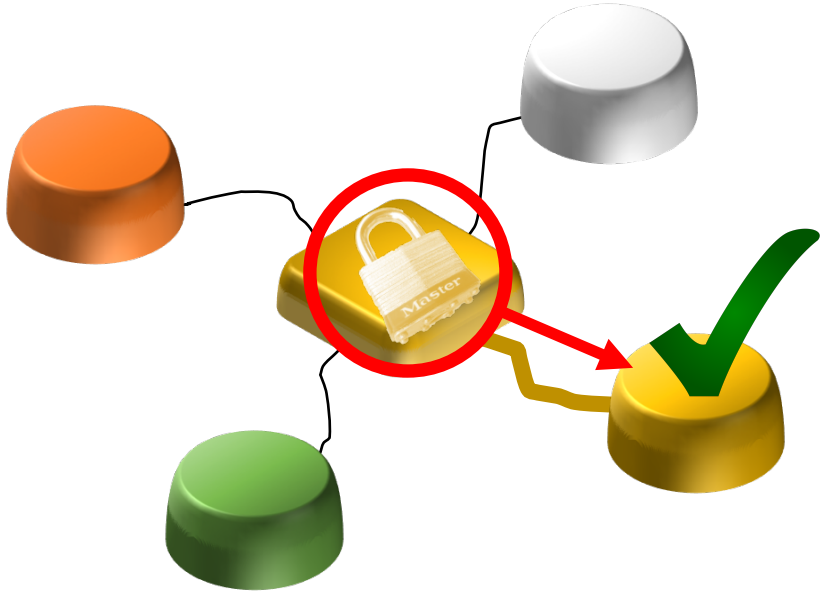
Sharing Resources (sync.)



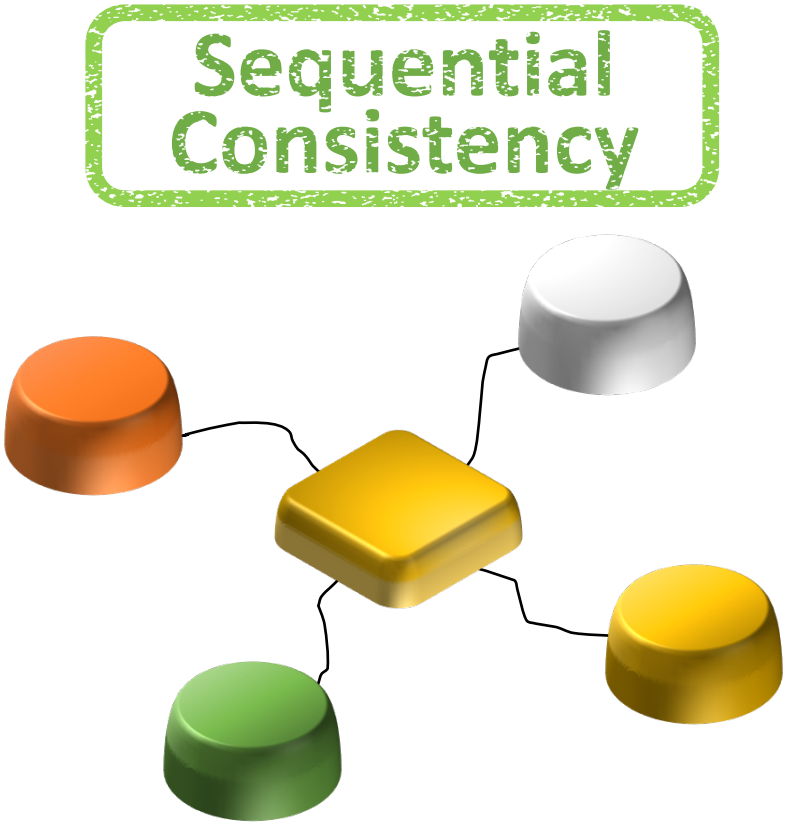
Sharing Resources (sync.)



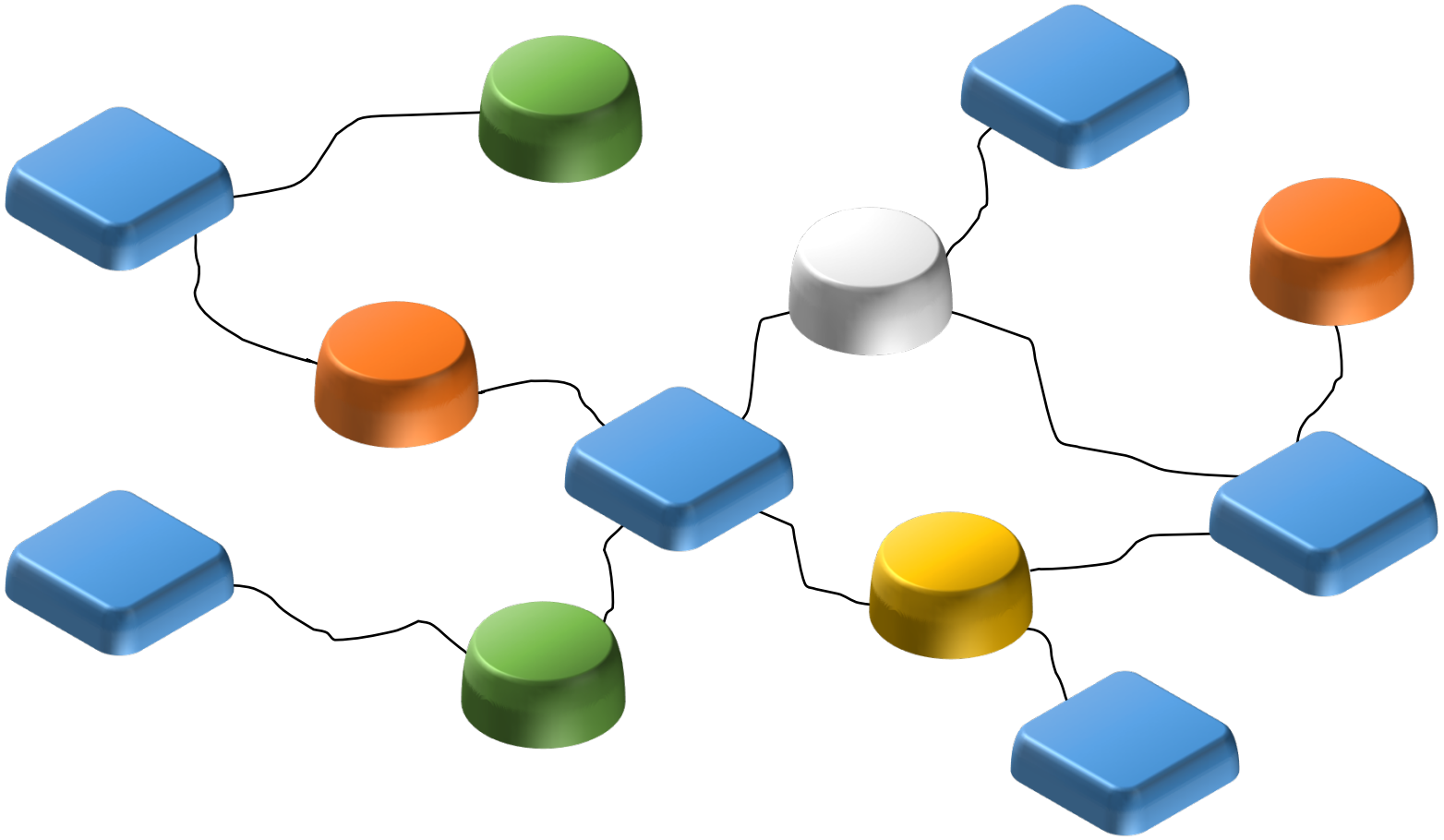
Sharing Resources (sync.)



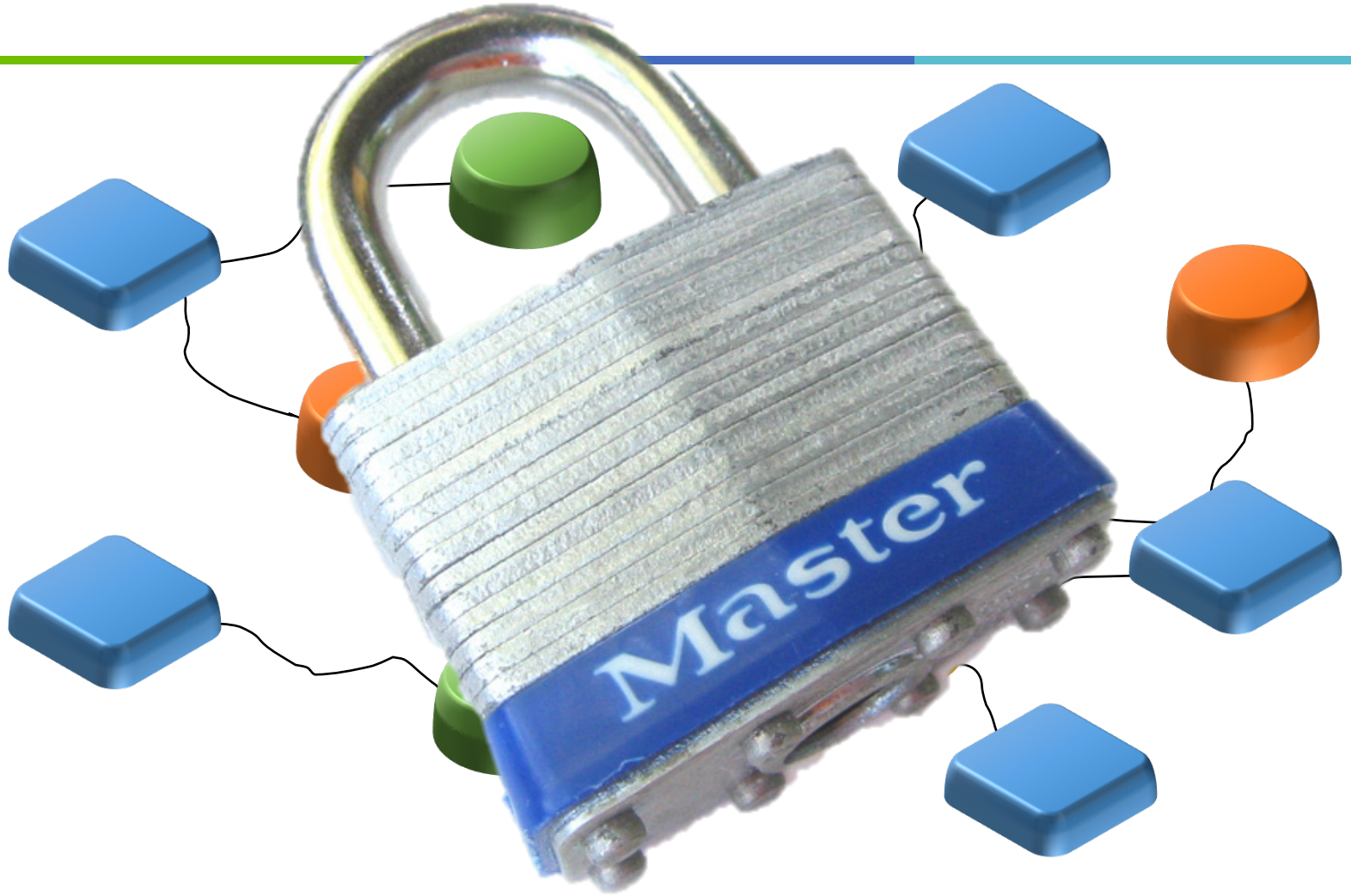
Sharing Resources (sync.)



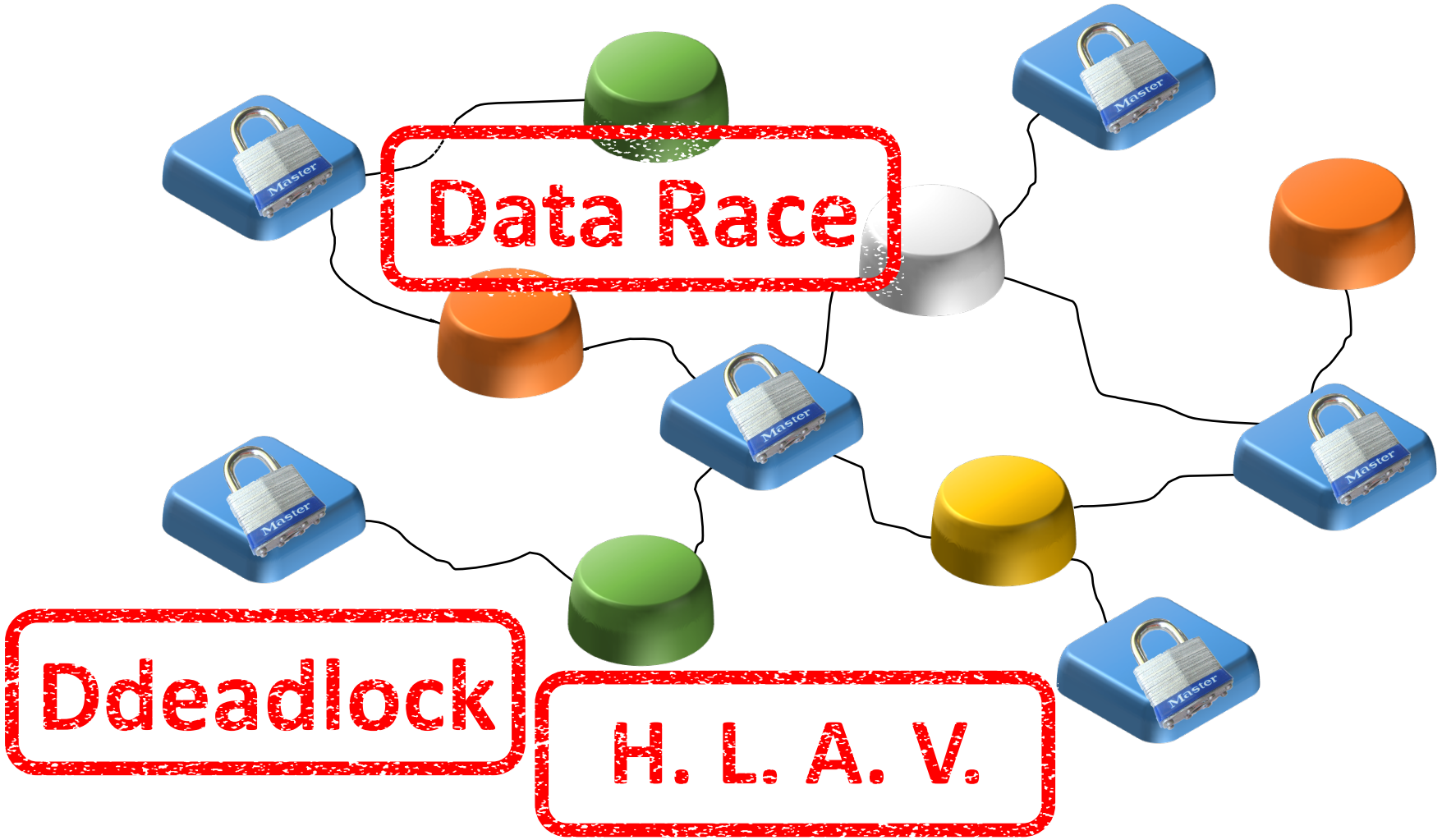
Sharing Multiple Resources



Sharing Multiple Resources



Sharing Multiple Resources



Assigning Semantics to Concurrent Programs

Assigning Semantics to Concurrent Programs

X = Y = 0

X,Y => Global Vars
a, b => Local Vars

X = 1 Y = 2	a = Y b = X

Assigning Semantics to Concurrent Programs

X = Y = 0

X,Y => Global Vars
a, b => Local Vars

X = 1 Y = 2	a = Y b = X

- What are the final values for 'X', 'Y', 'a' and 'b'?
  - X = 1, Y = 2

Assigning Semantics to Concurrent Programs

X = Y = 0

X,Y => Global Vars
a, b => Local Vars

X = 1 Y = 2	a = Y b = X

- What are the final values for 'X', 'Y', 'a' and 'b'?
  - X = 1, Y = 2, a = ?, b = ?

Assigning Semantics to Concurrent Programs

X = Y = 0

X,Y => Global Vars
a, b => Local Vars

X = 1 Y = 2	a = Y b = X

- What are the final values for 'X', 'Y', 'a' and 'b'?
  - X = 1, Y = 2, a = ?, b = ?
- Depends on the interleavings of the assignments
  - Sequential Consistency [Lamport'79]
  - Program behavior = set of interleavings

Assigning Semantics to Concurrent Programs

X = Y = 0

X,Y => Global Vars
a, b => Local Vars

X = 1
Y = 2

a = Y
b = X

X = 1

Y = 2

a = Y

b = X

Assigning Semantics to Concurrent Programs

X = Y = 0

X,Y => Global Vars
a, b => Local Vars

X = 1
Y = 2

a = Y
b = X

X = 1

Y = 2

a = Y

b = X

a=2, b=1

Assigning Semantics to Concurrent Programs

X = Y = 0

X,Y => Global Vars
a, b => Local Vars

X = 1 Y = 2	a = Y b = X

X = 1	X = 1
Y = 2	a = Y
a = Y	b = X
b = X	Y = 2

a=2, b=1

Assigning Semantics to Concurrent Programs

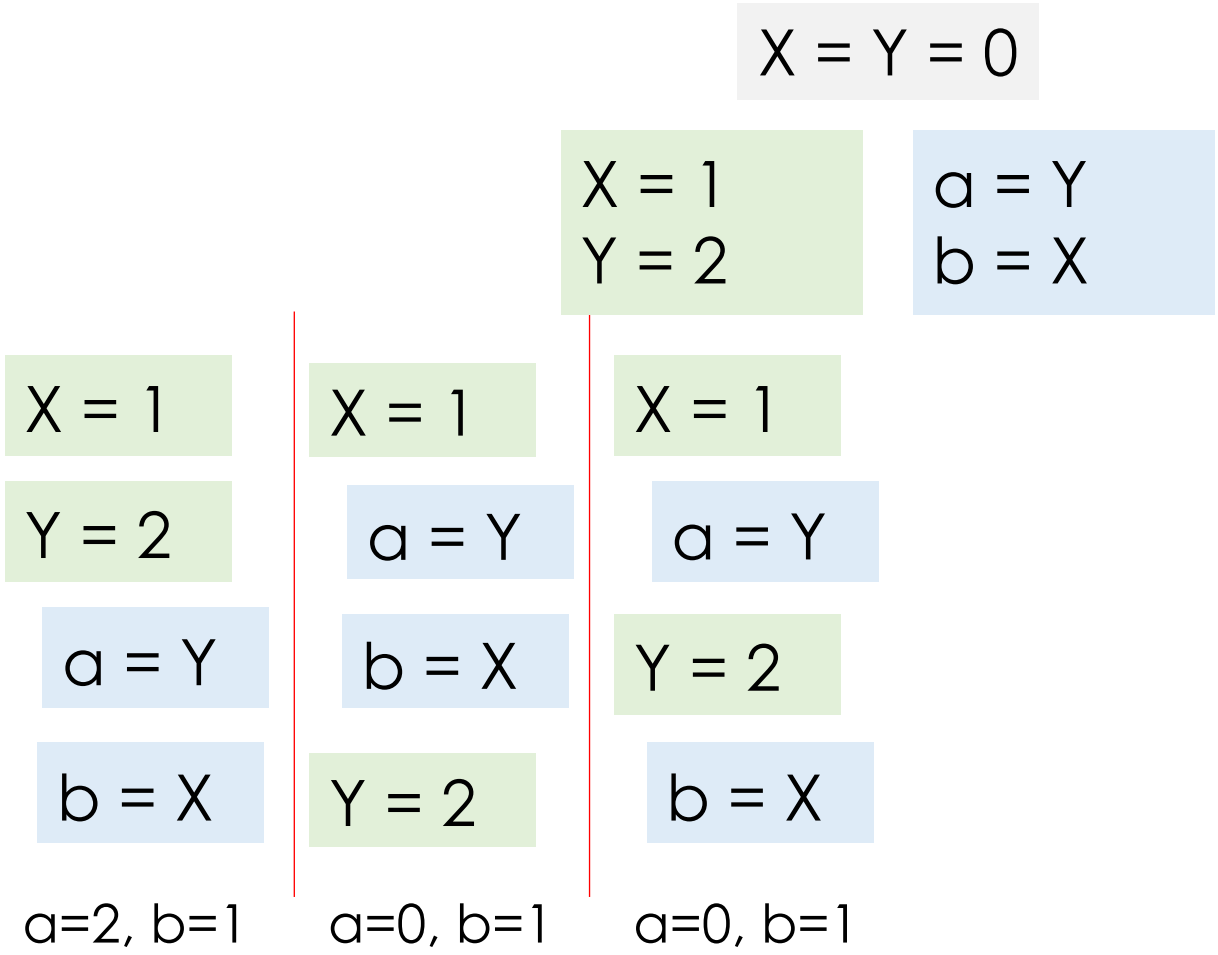
X = Y = 0

X,Y => Global Vars
a, b => Local Vars

X = 1 Y = 2	a = Y b = X

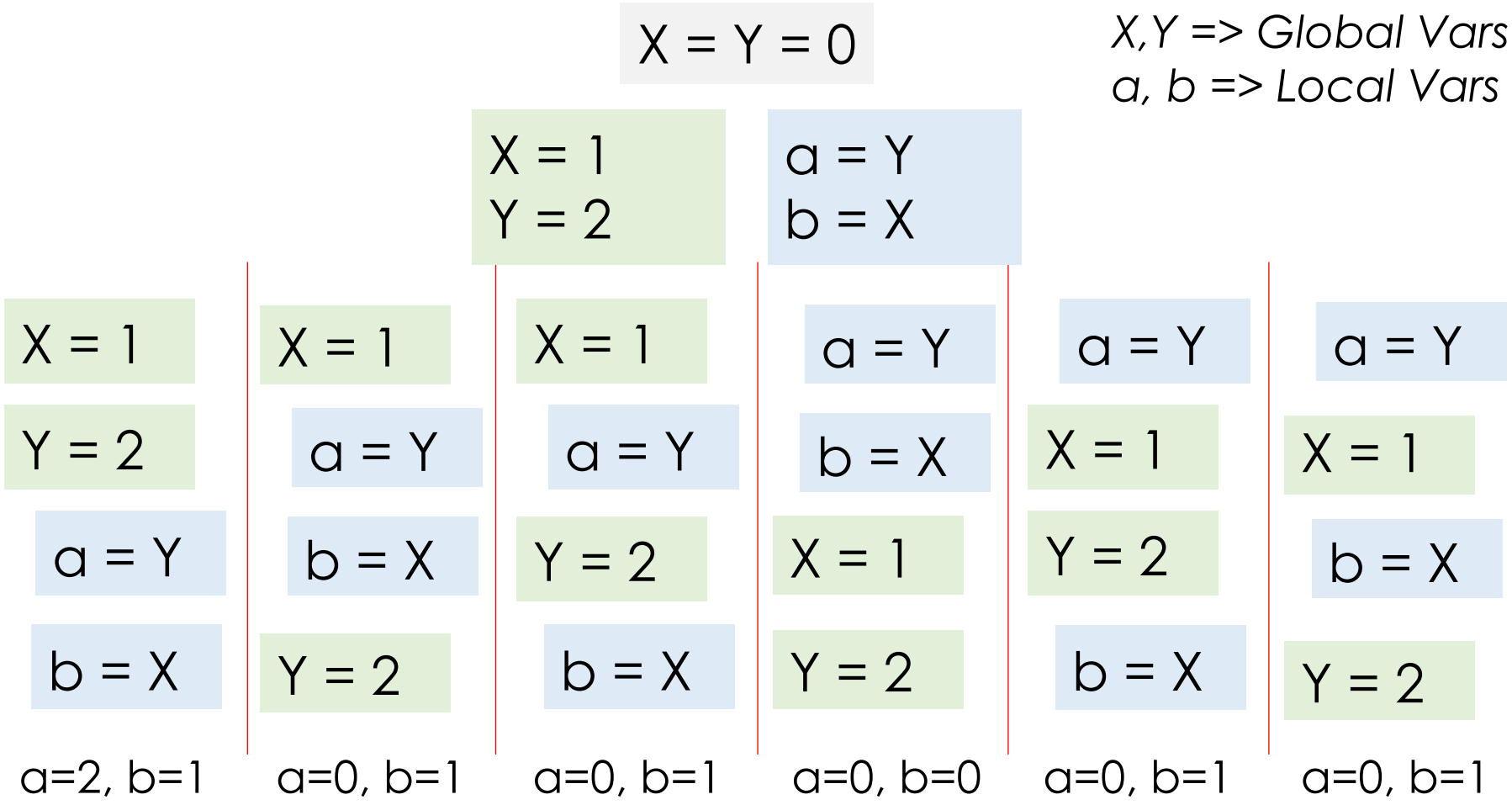
X = 1	X = 1
Y = 2	a = Y
a = Y	b = X
b = X	Y = 2
a=2, b=1	a=0, b=1

Assigning Semantics to Concurrent Programs



X,Y => Global Vars
a, b => Local Vars

Assigning Semantics to Concurrent Programs

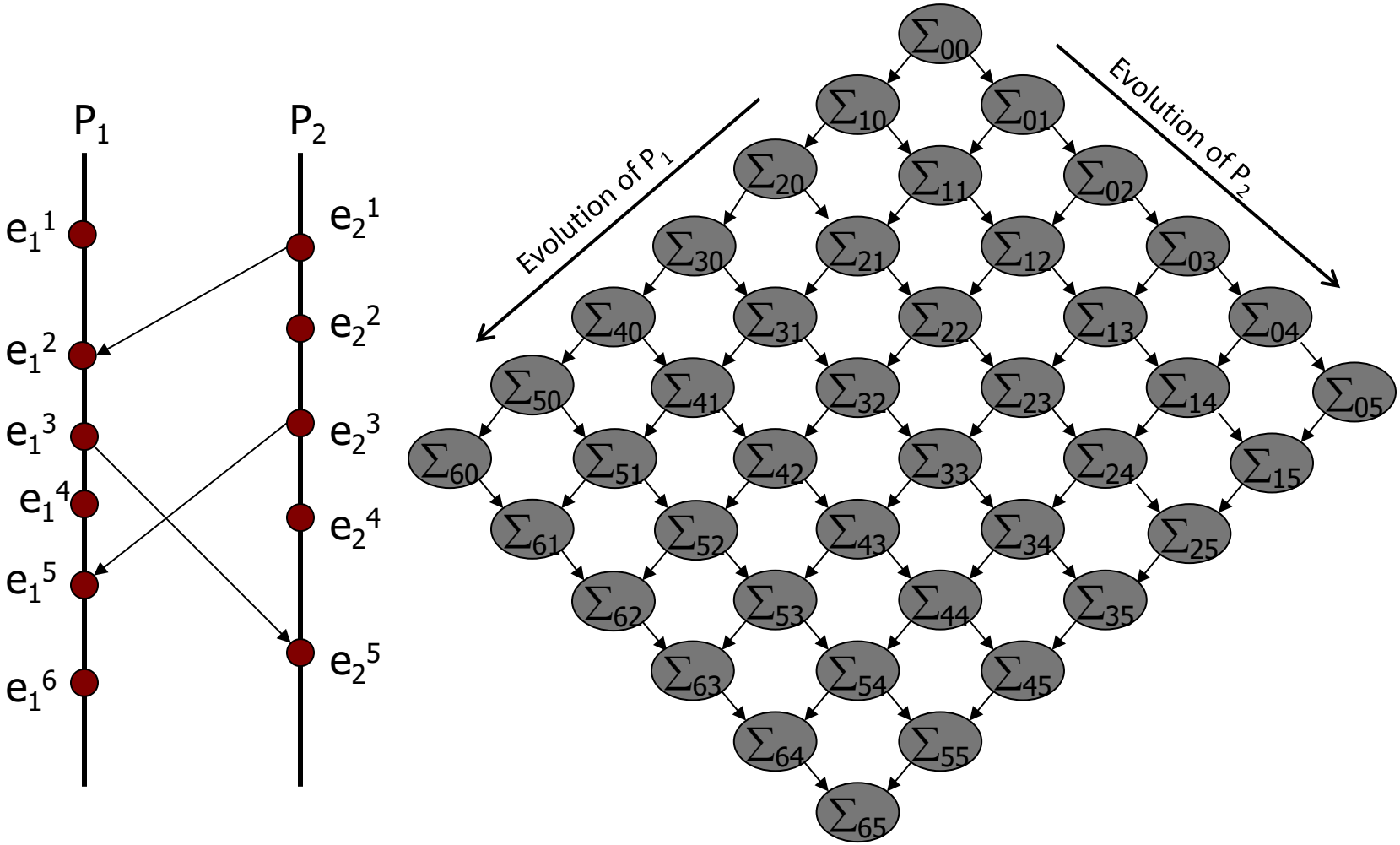


Sequential Consistency

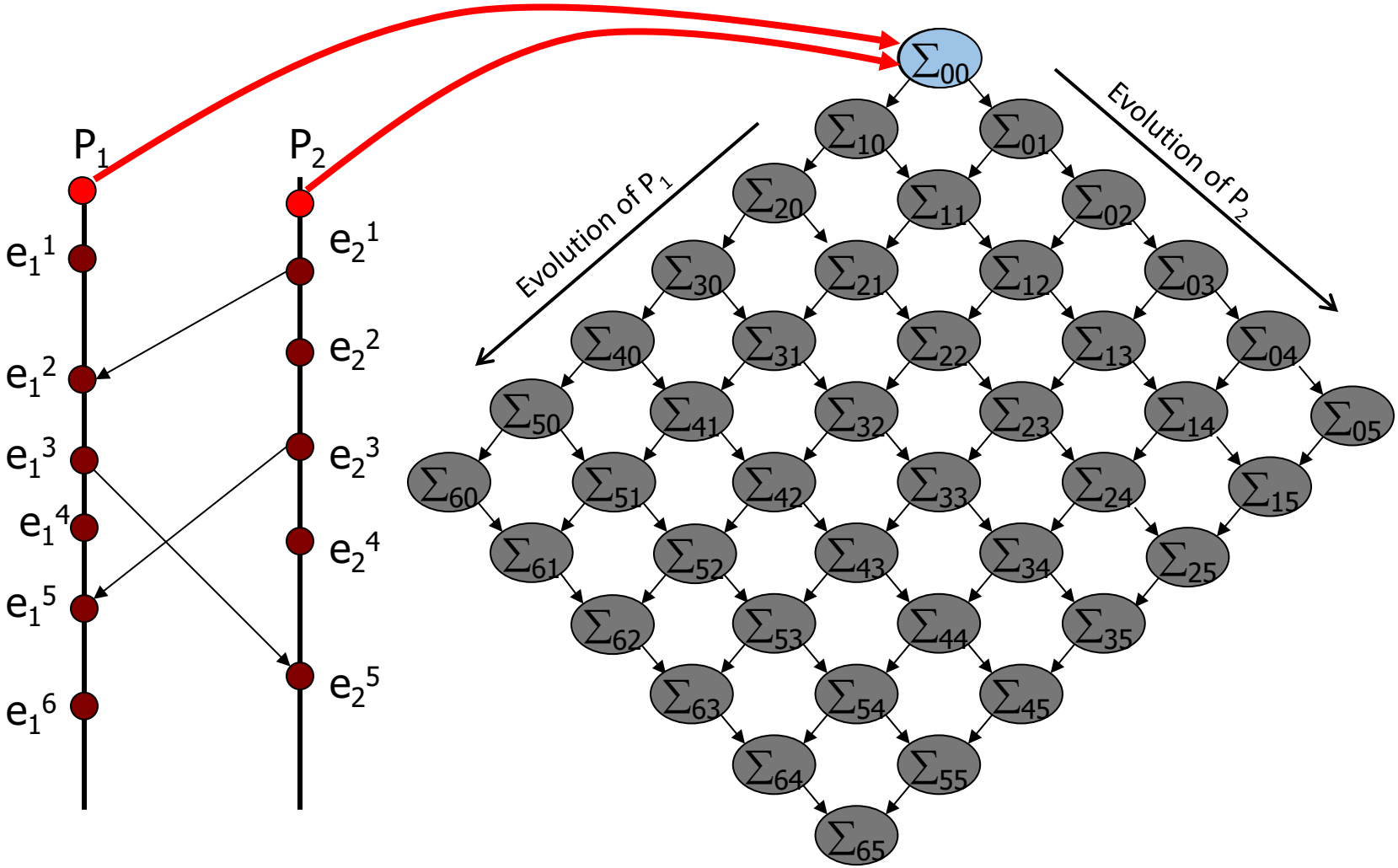
- Instructions are executed by the order they appear in the program
- Memory behaves as a shared array
  - Reads and writes are effective immediately

- This is naturally true for sequential programs…

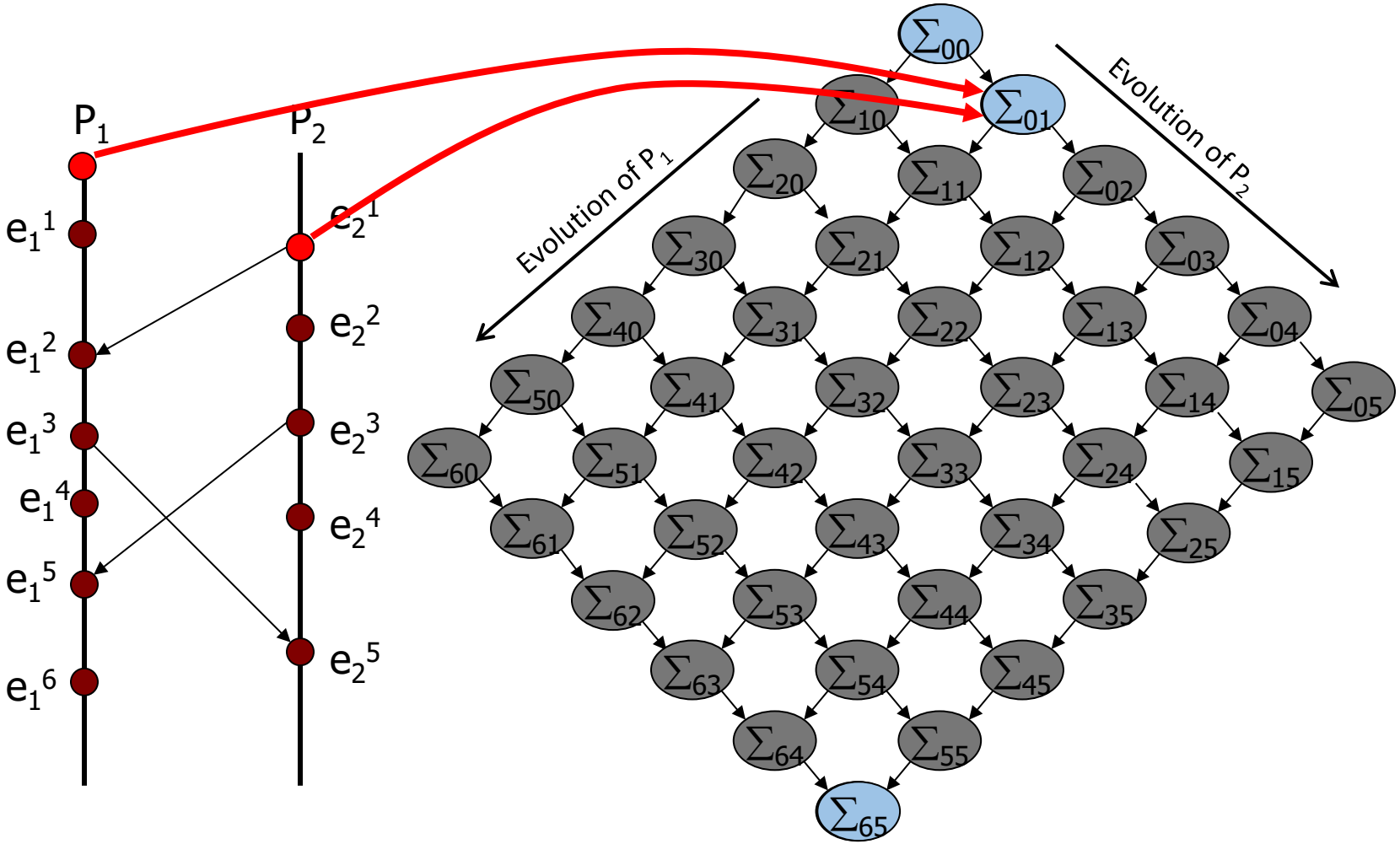
State explosion in concurrent programs



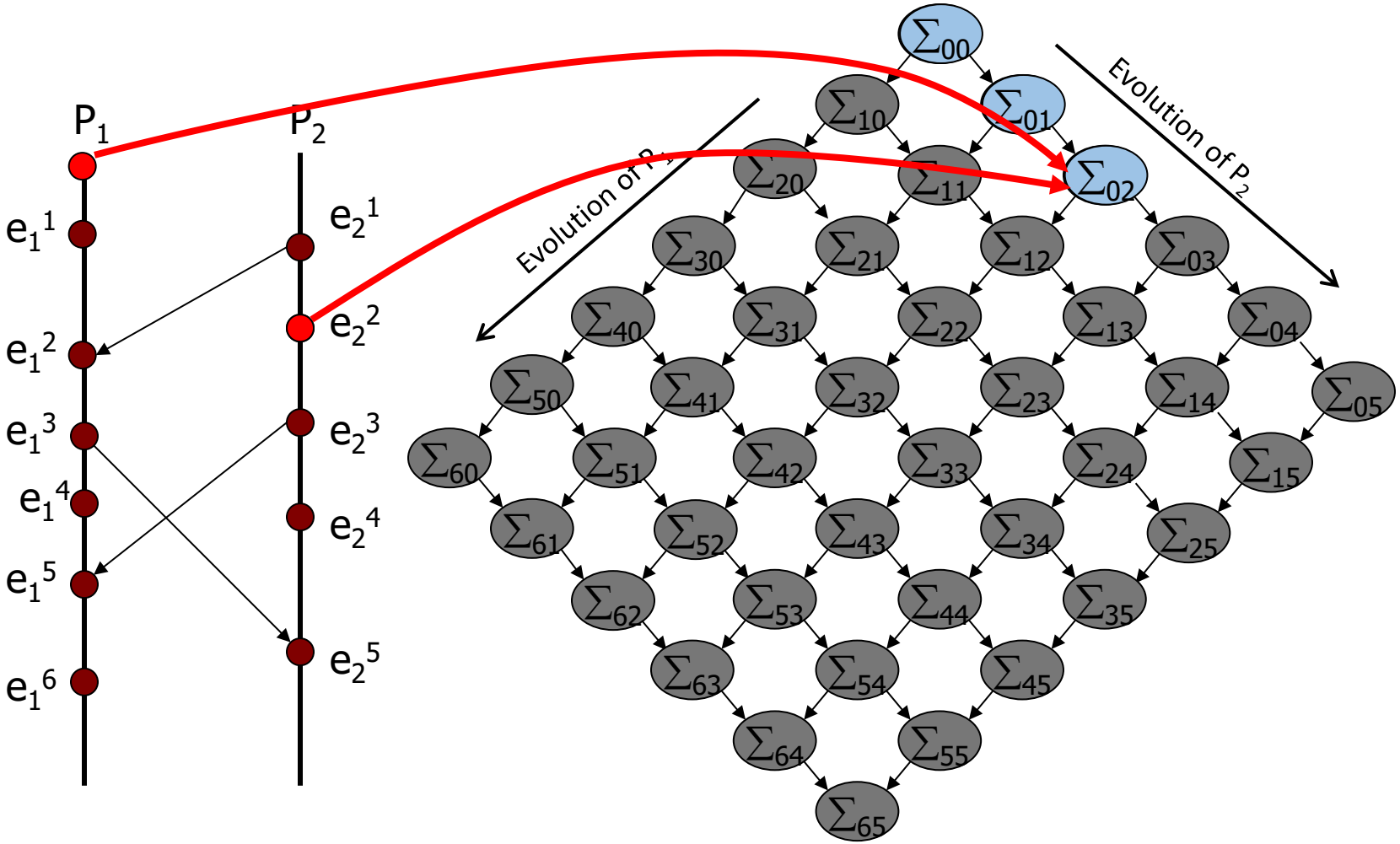
State explosion in concurrent programs



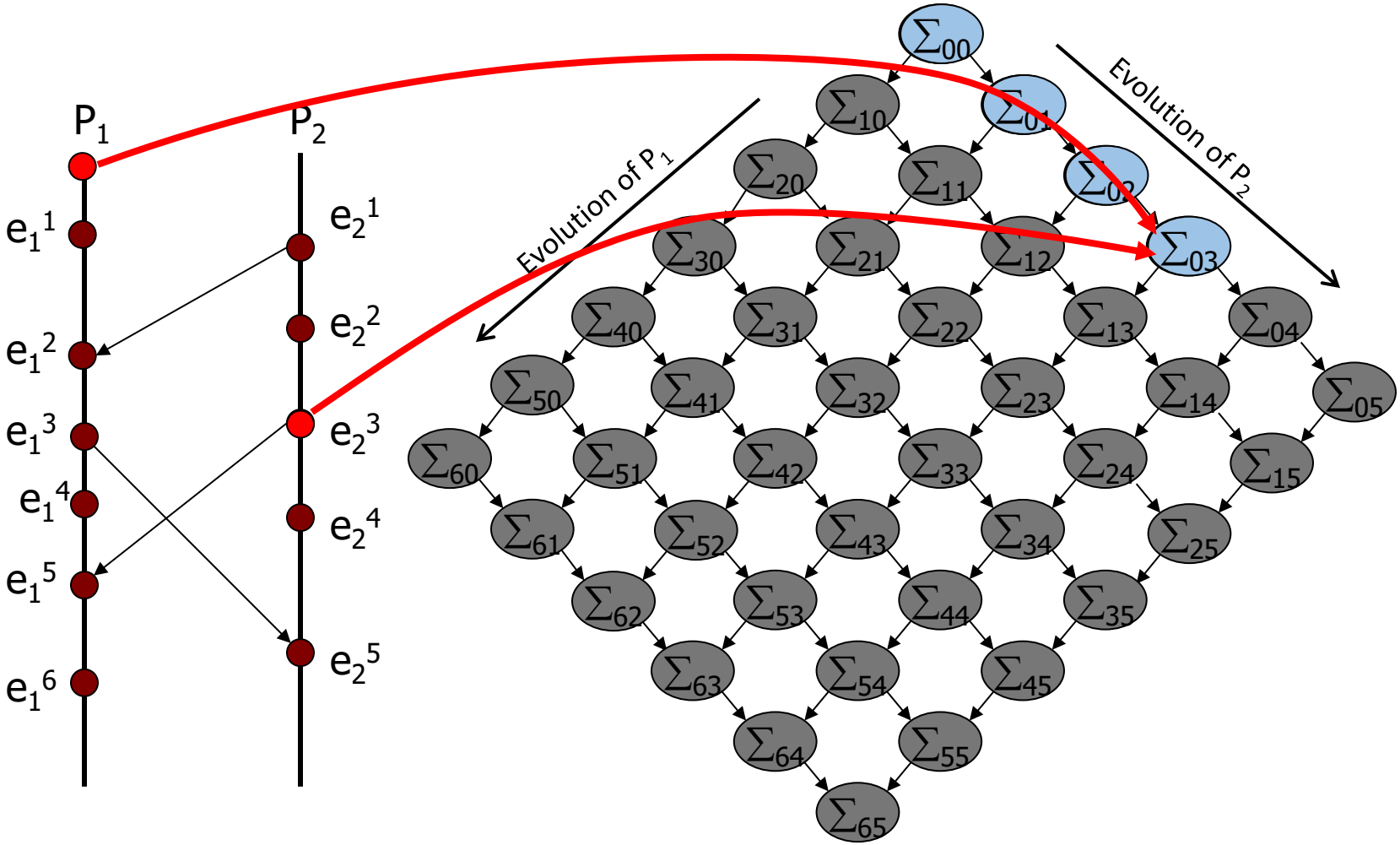
State explosion in concurrent programs



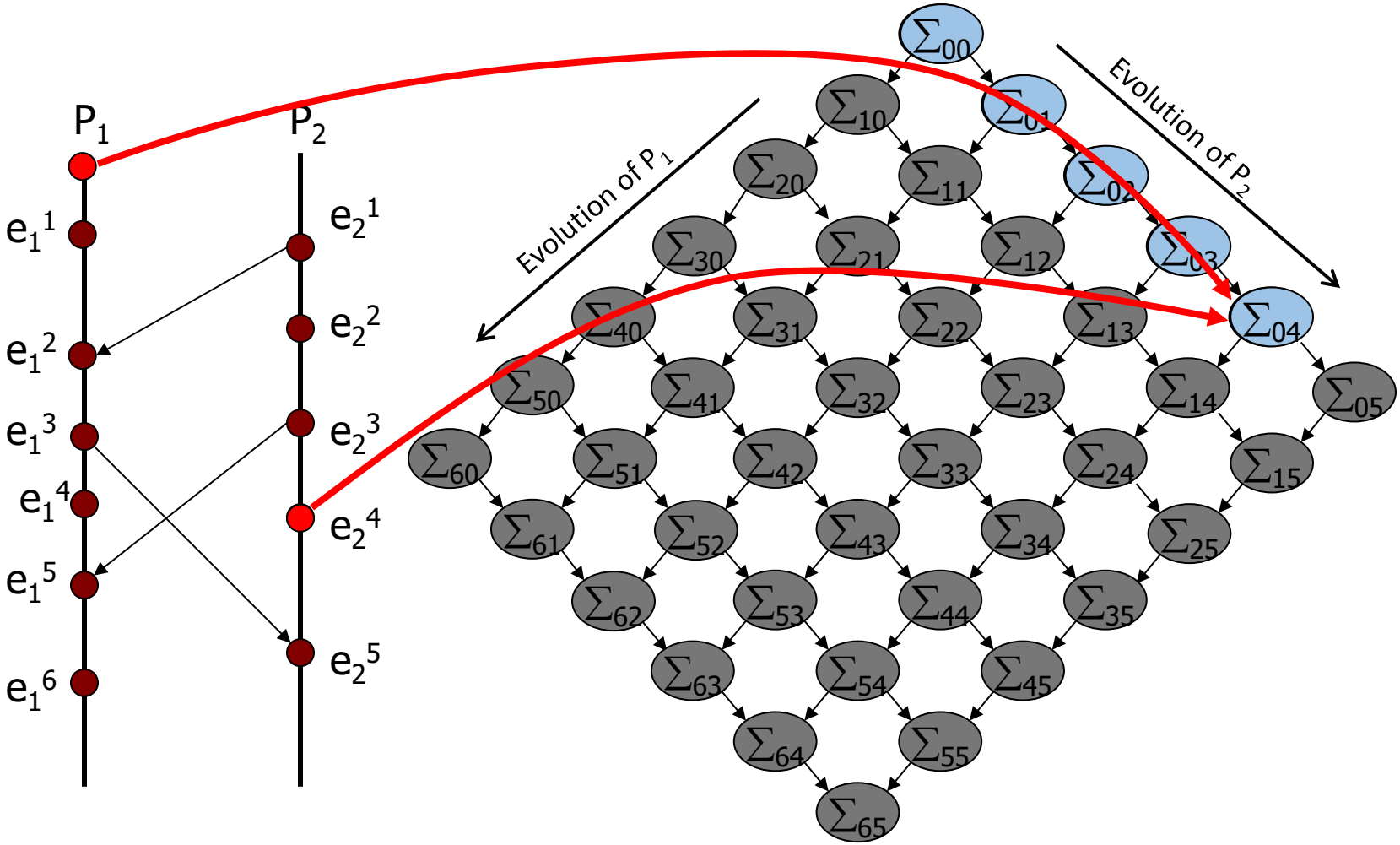
State explosion in concurrent programs



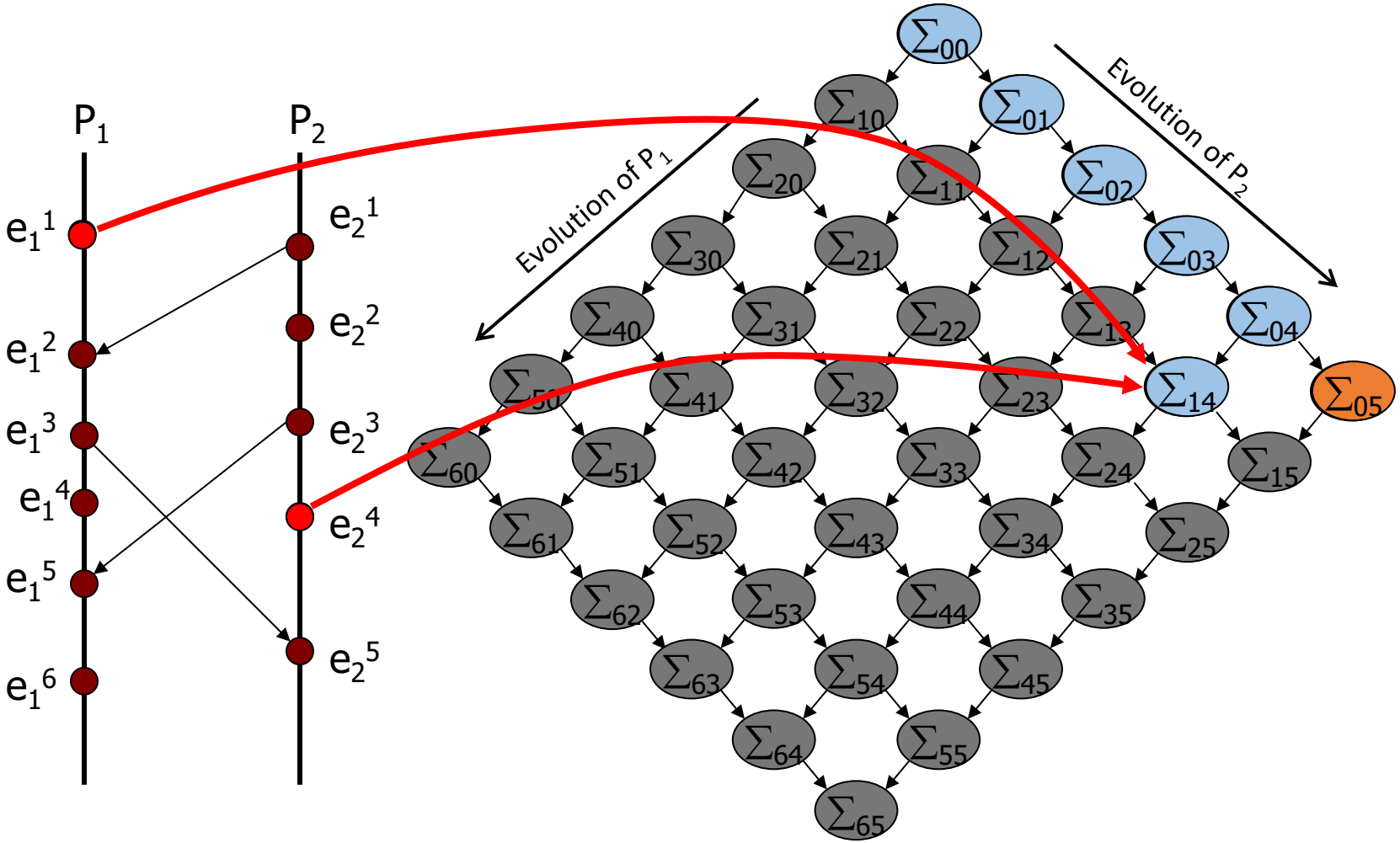
State explosion in concurrent programs



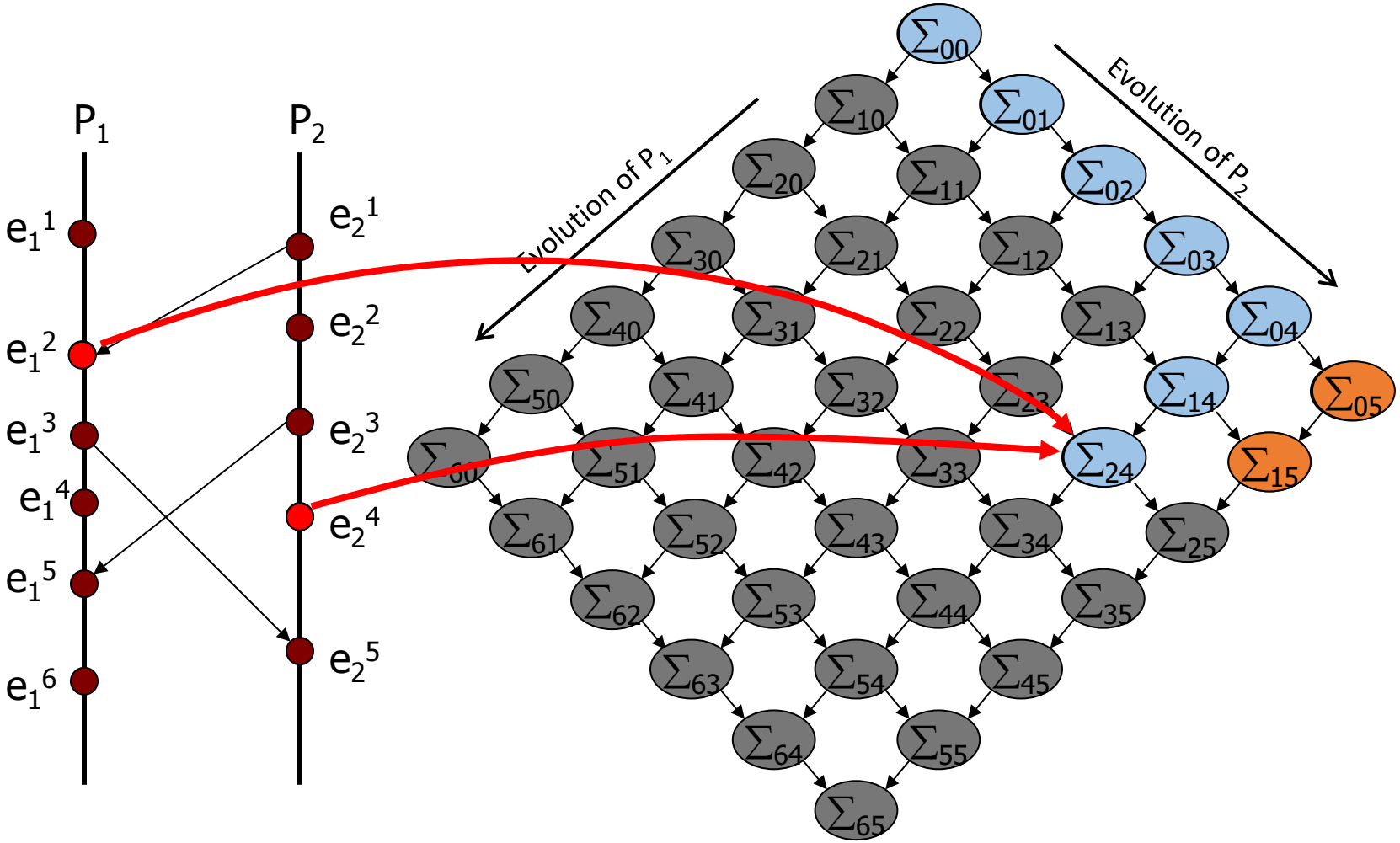
State explosion in concurrent programs



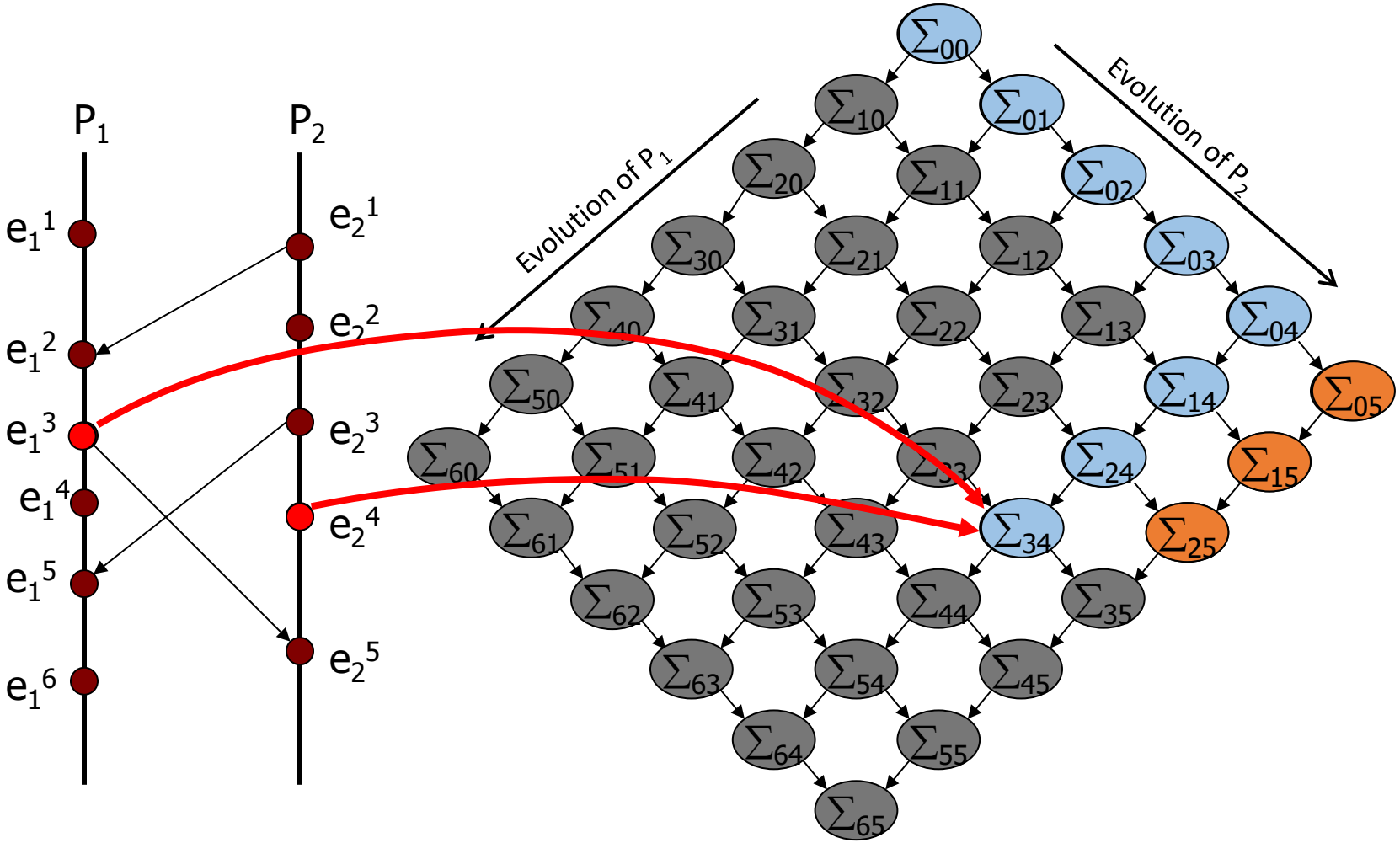
State explosion in concurrent programs



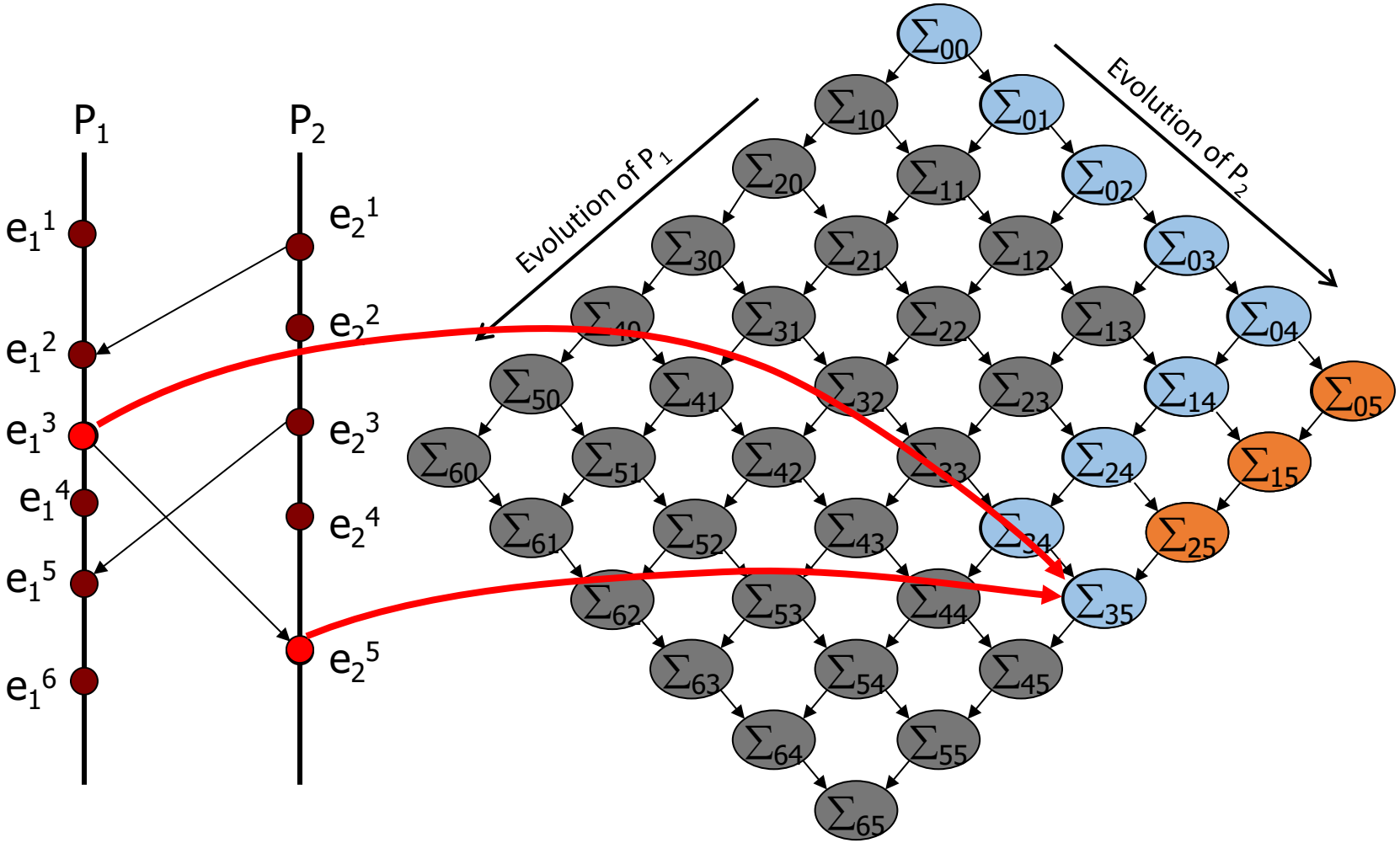
State explosion in concurrent programs



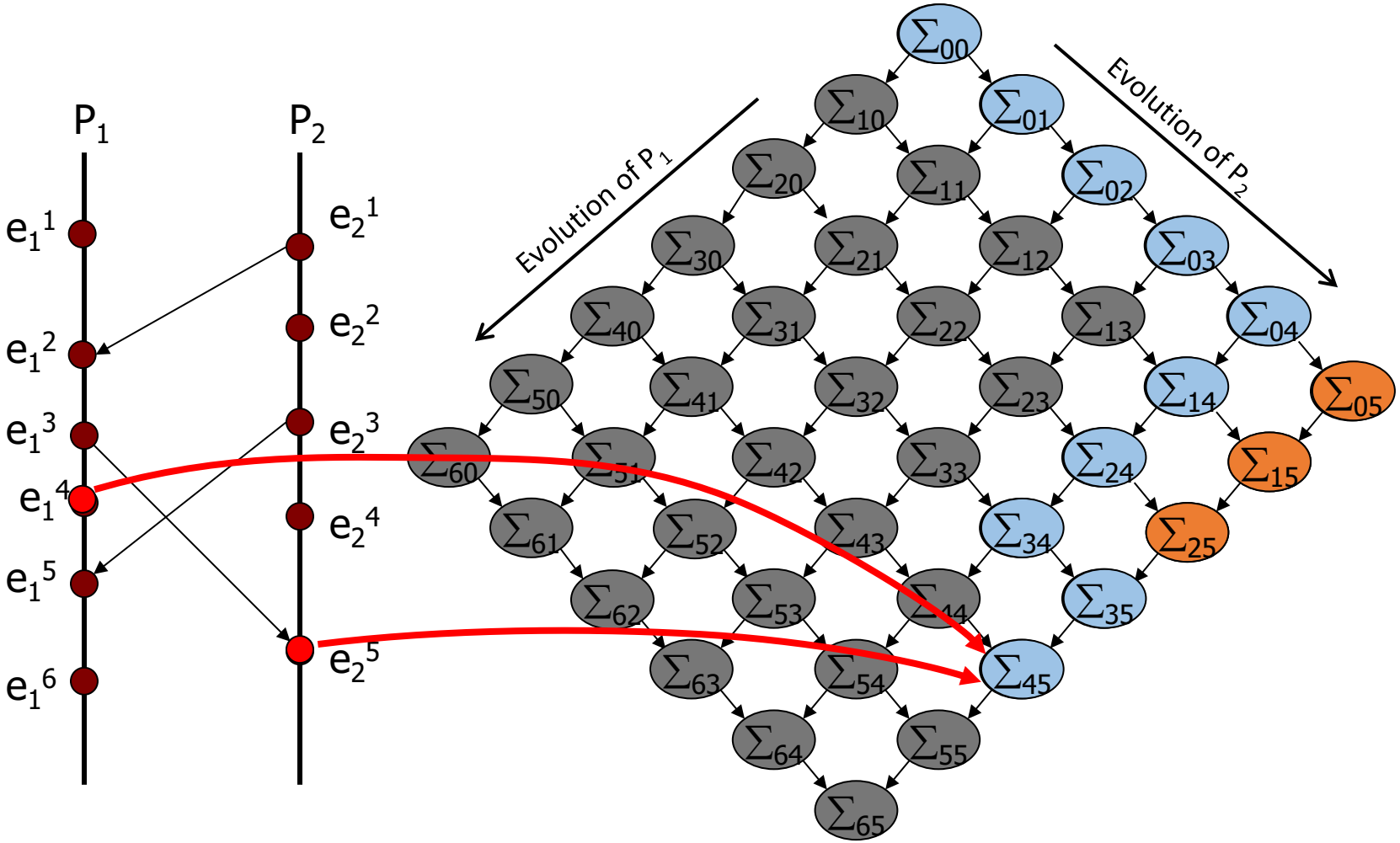
State explosion in concurrent programs



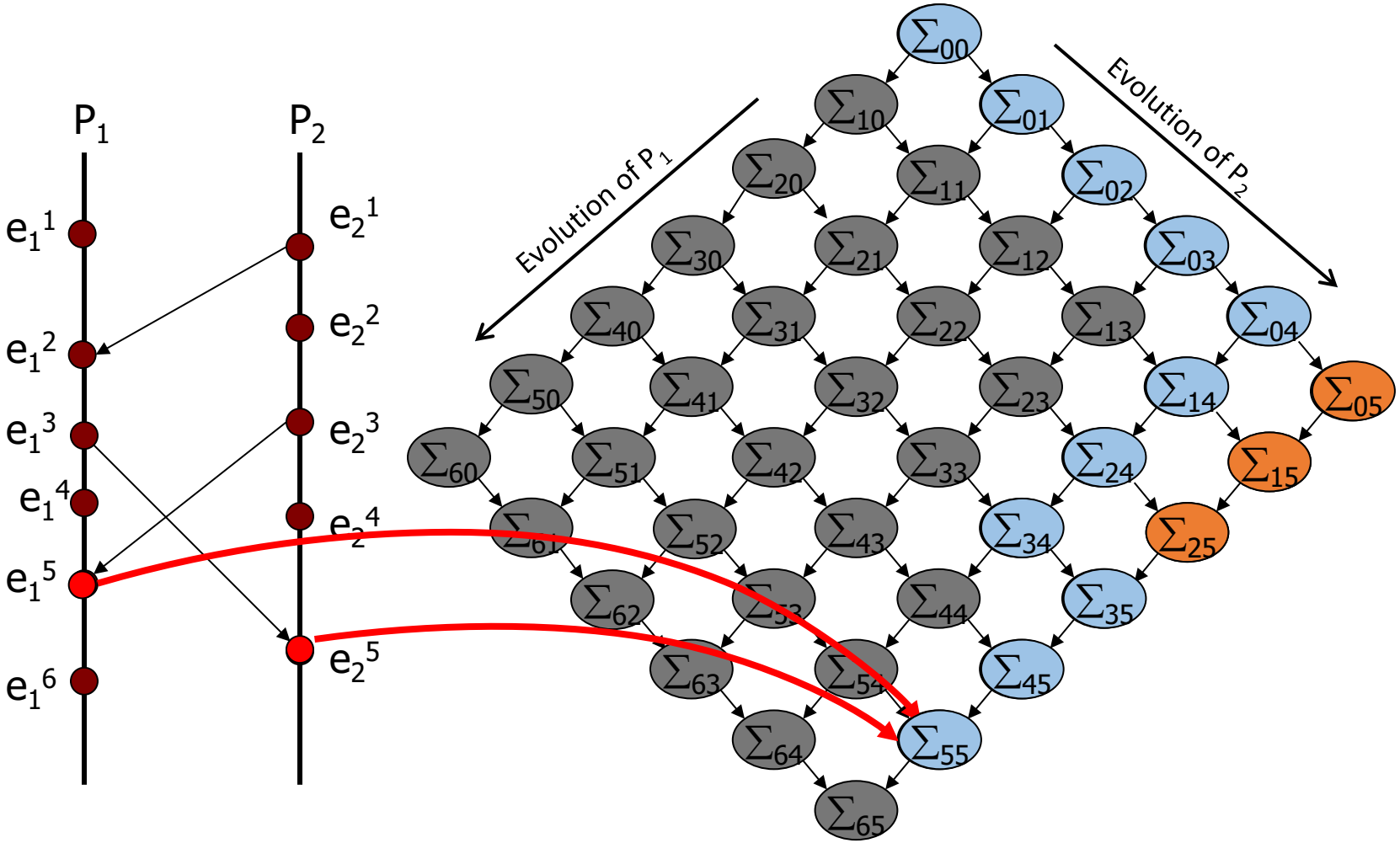
State explosion in concurrent programs



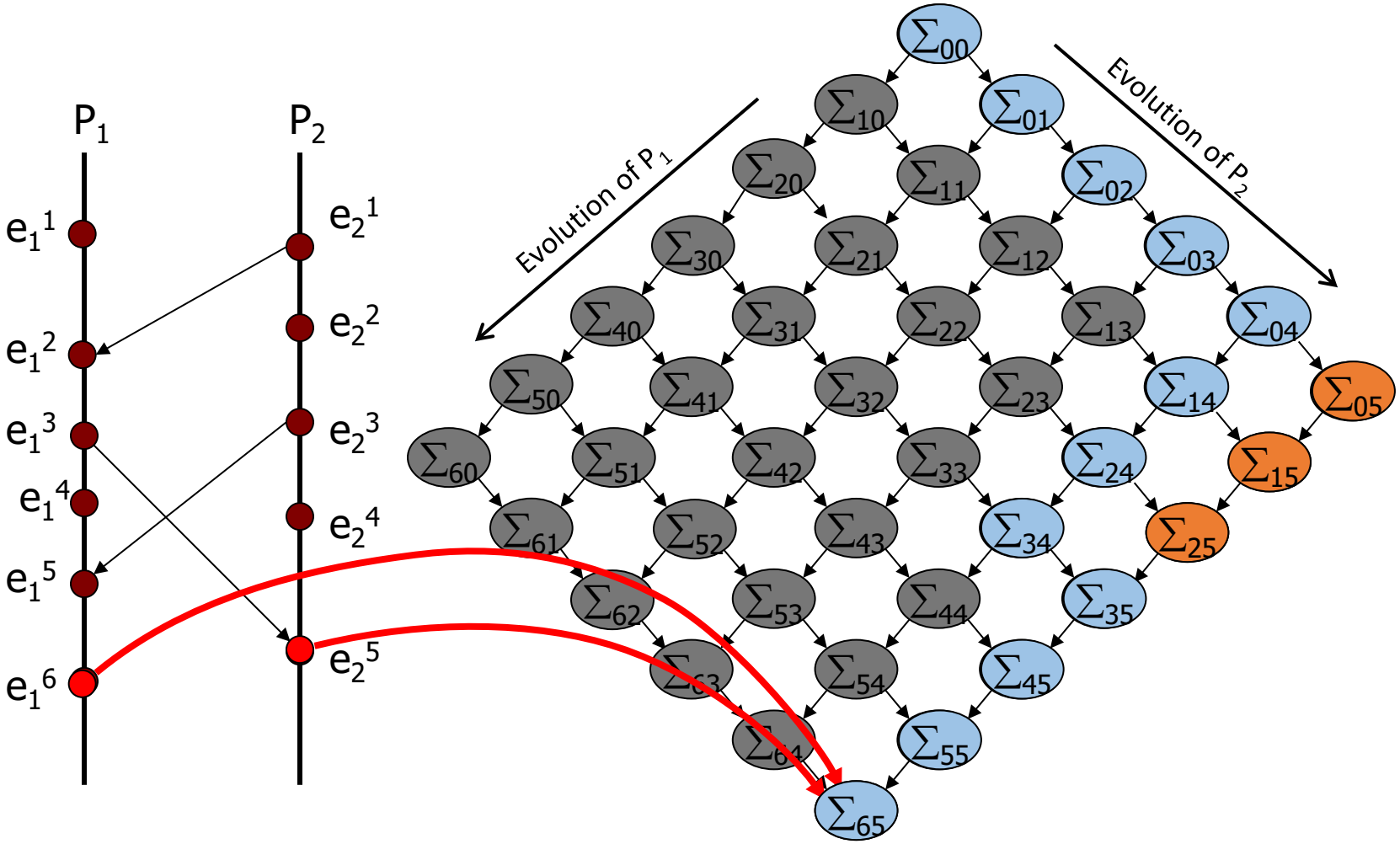
State explosion in concurrent programs



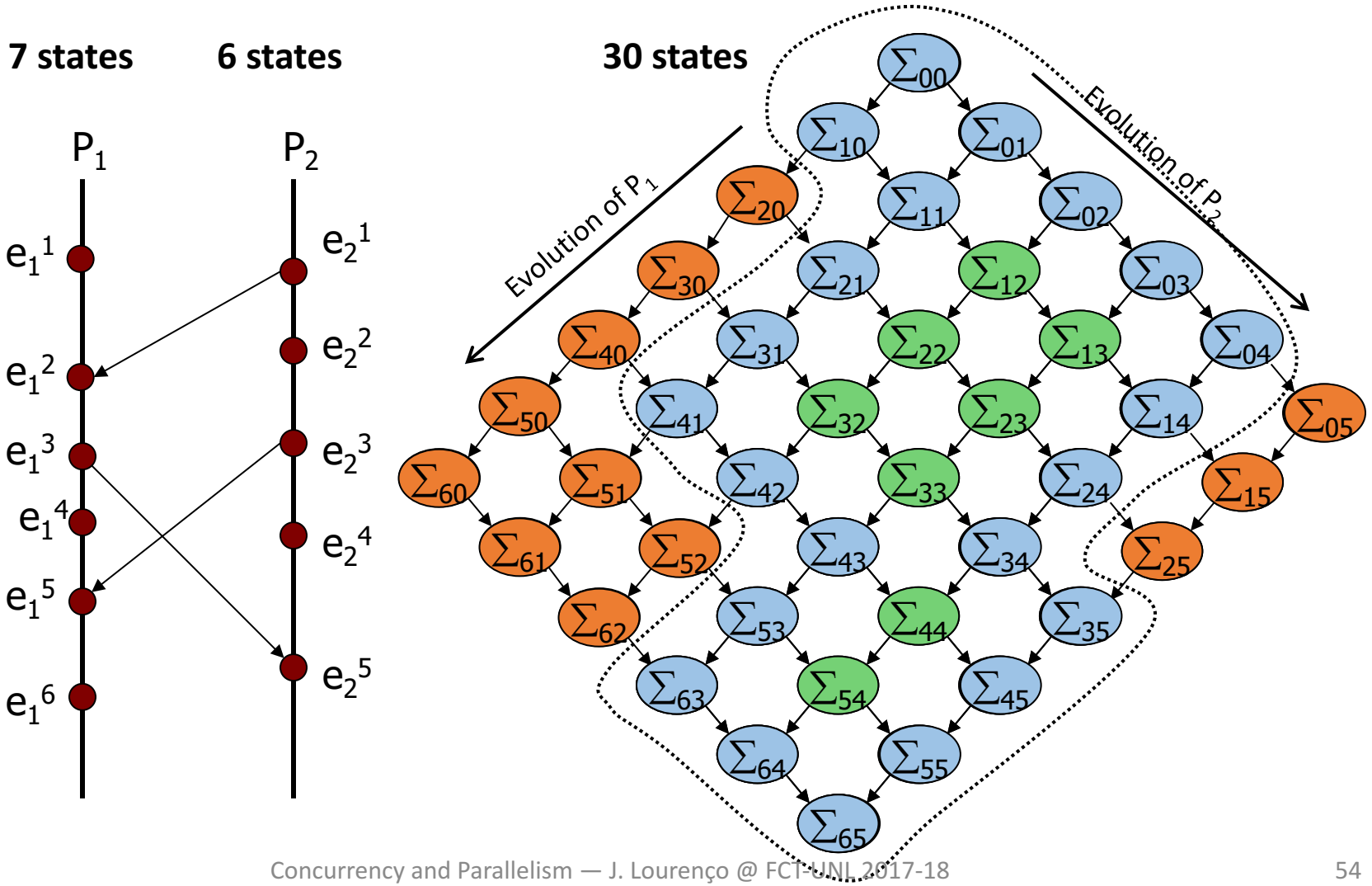
State explosion in concurrent programs



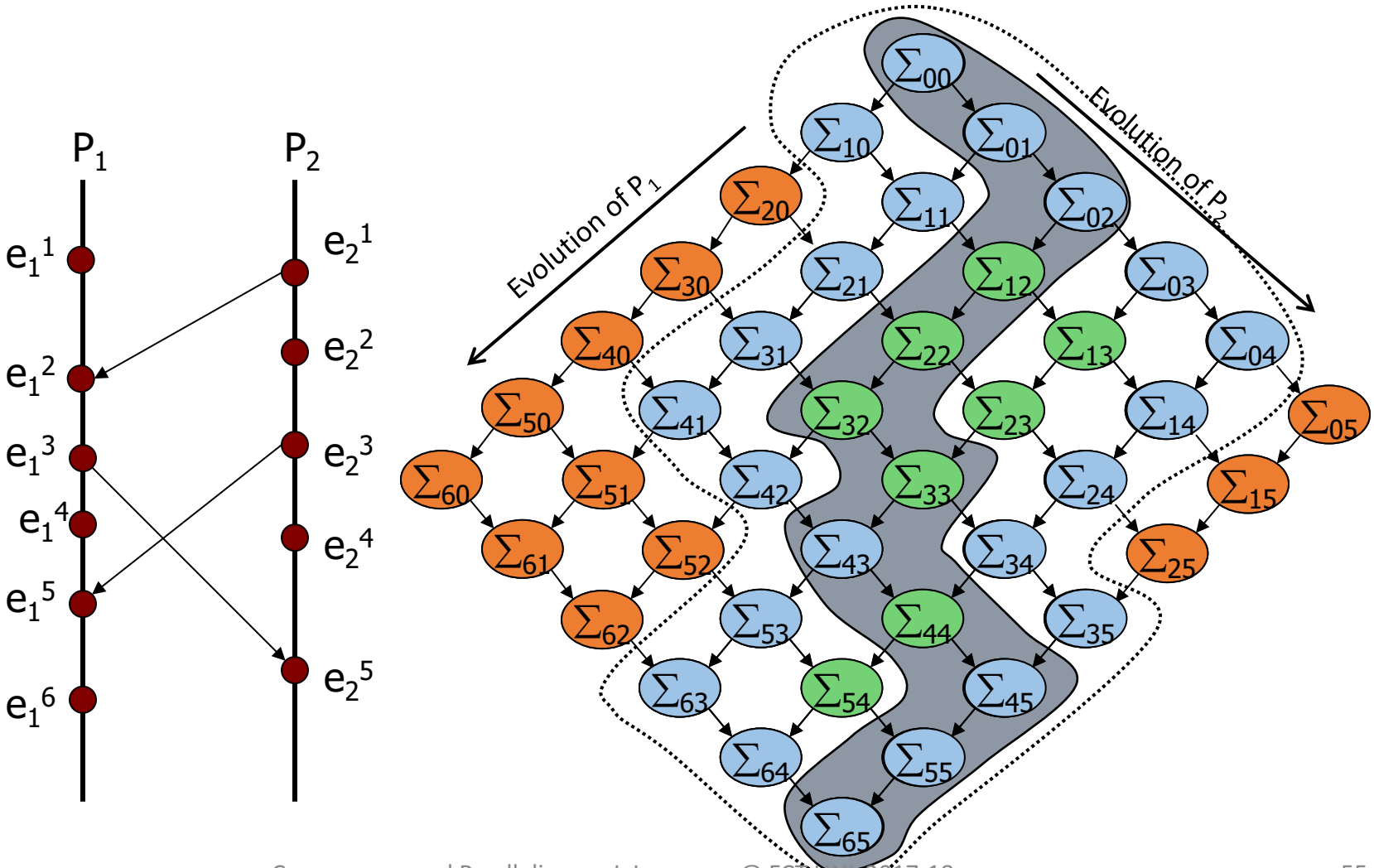
State explosion in concurrent programs



State explosion in concurrent programs



Consistent run



Concurrency Errors

Common Concurrency Errors

- Data races (atomicity violations)
- Ordering violations
- Unintended sharing
- High-level atomicity violations
- Deadlocks and livelocks

Data Race

- Code is supposed to execute atomically
  - Multiple dependent instructions to manipulate some data
- Interleaving with instructions of another thread that access the same data

Thread 1

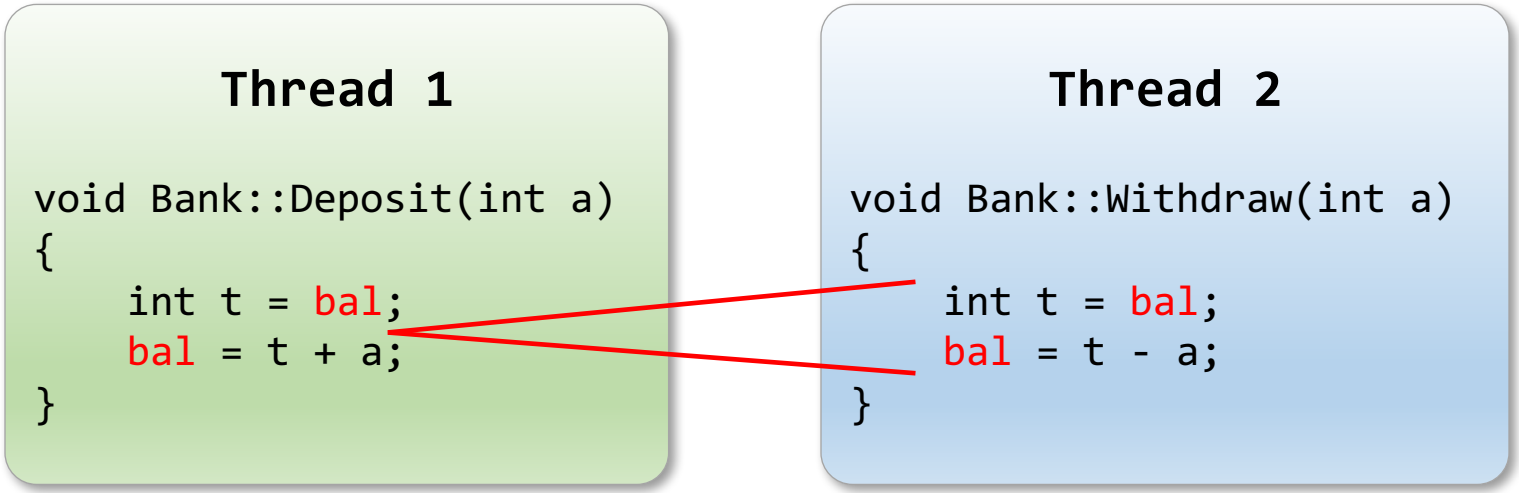
```
void Bank::Deposit(int a)
{
    int t = bal;
    bal = t + a;
}
```

Thread 2

```
void Bank::Withdraw(int a)
{
    int t = bal;
    bal = t - a;
}
```

Data Race

- Code is supposed to execute atomically
  - Multiple dependent instructions to manipulate some data
- Interleaving with instructions of another thread that access the same data



The widthdraw is not reflected in the final balance!

Ordering Violation

- Missing or incorrect synchronization between two processes
  (e.g., a producer and a consumer)

Thread 1

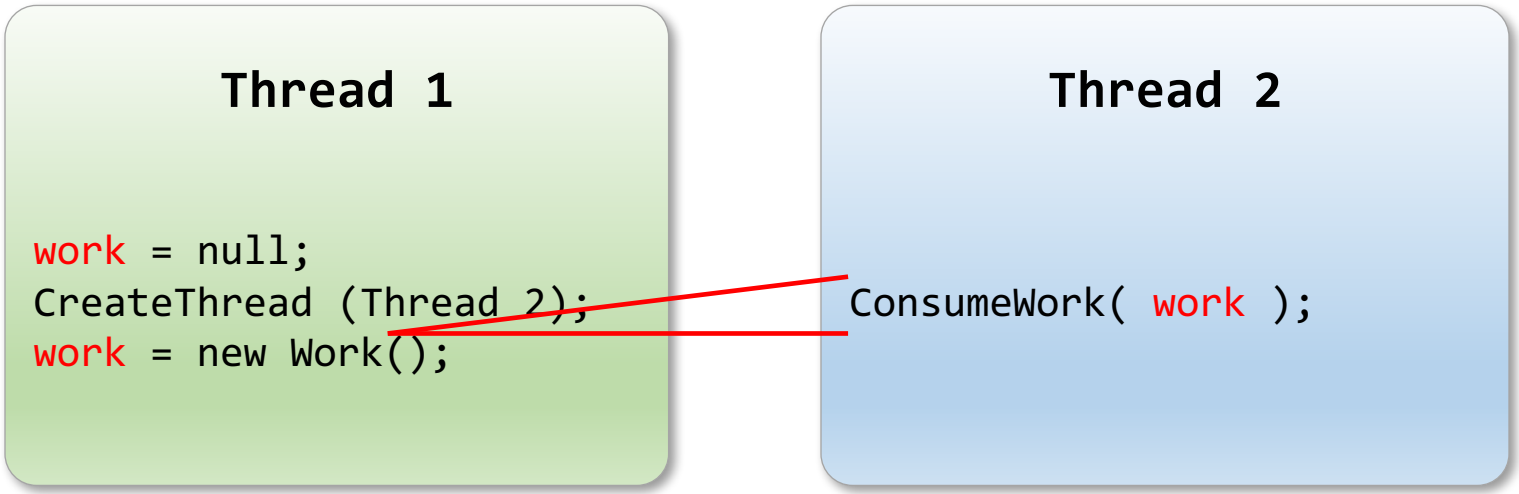
```
work = null;
CreateThread (Thread 2);
work = new Work();
```

Thread 2

```
ConsumeWork( work );
```

Ordering Violation

- Missing or incorrect synchronization between two processes
  (e.g., a producer and a consumer)



'work' is not initialized yet!

Unintended Sharing

- Processes accidentally share data
  - 'work()' is executed by both threads concurrently

```
void work() {
   static int local = 0;
   …
   local += …
   …
}
```

Thread 1

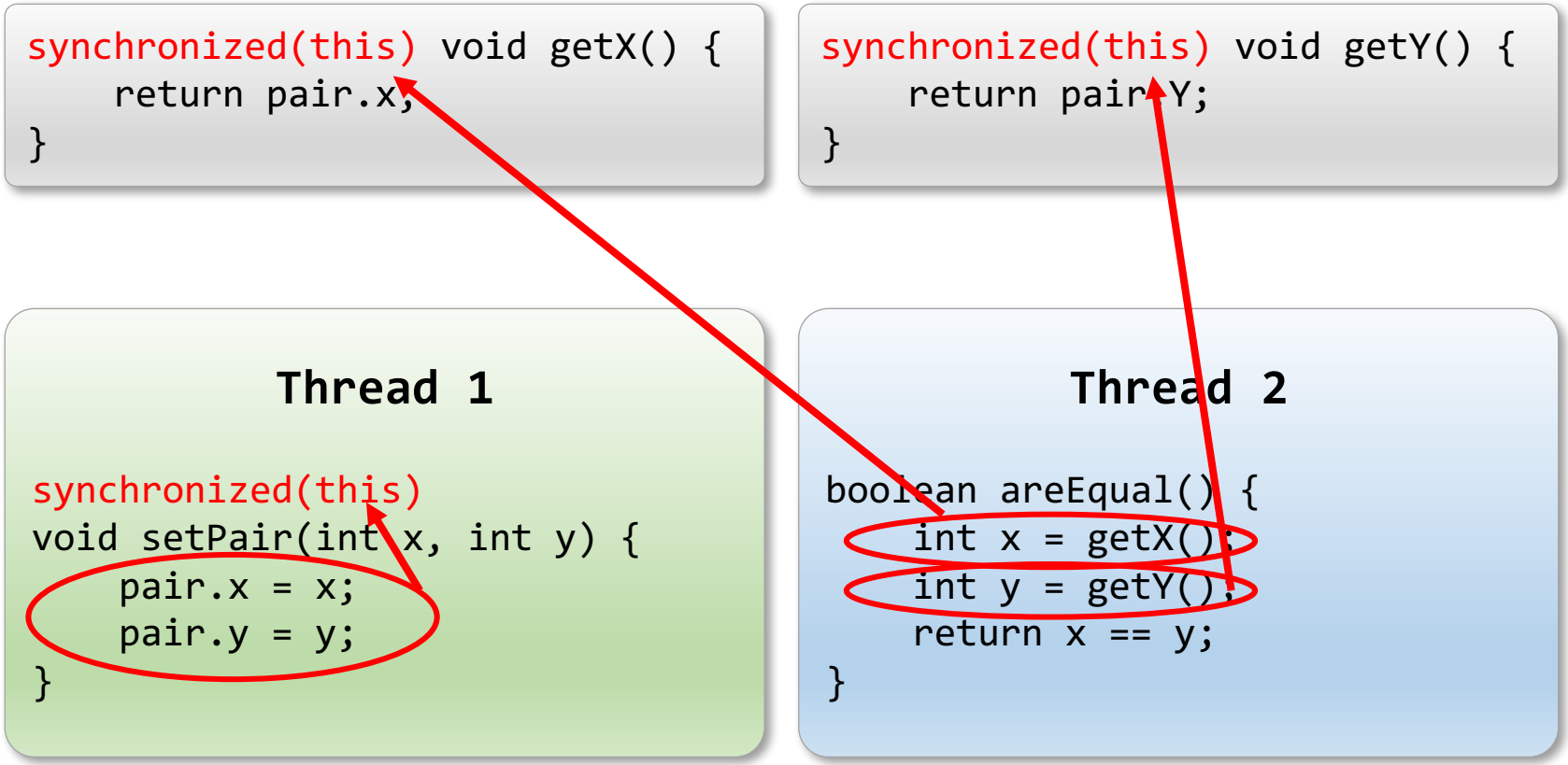
```
…
work()
…
```

Thread 2

```
…
work()
…
```

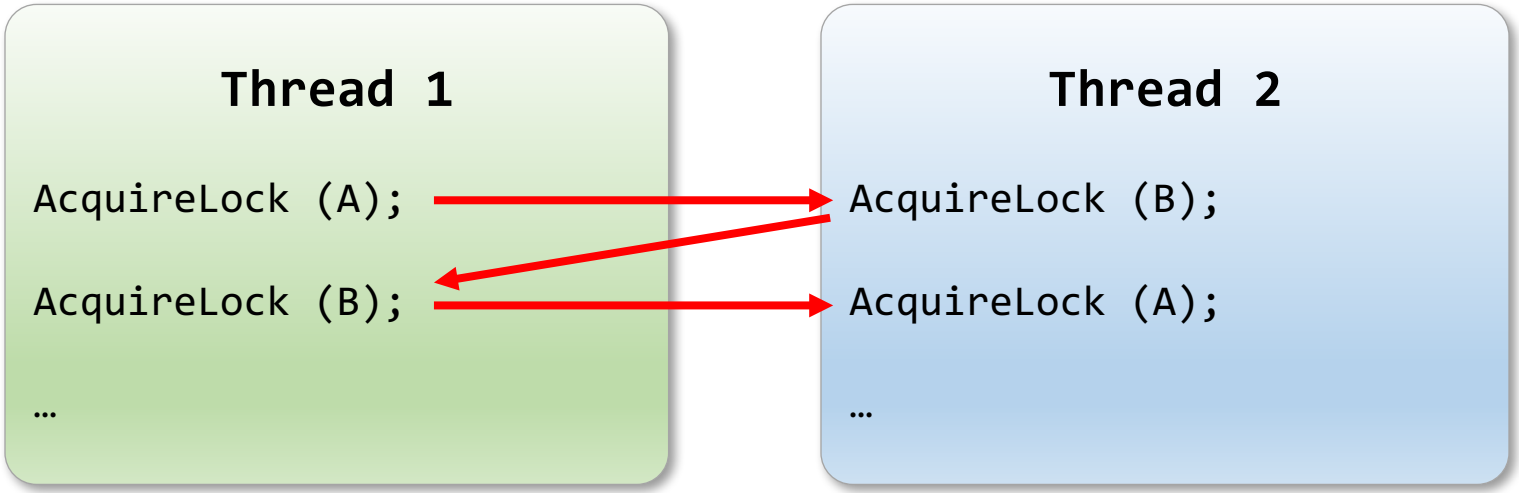
High-Level Data Race

- Wrongly defined atomic blocks



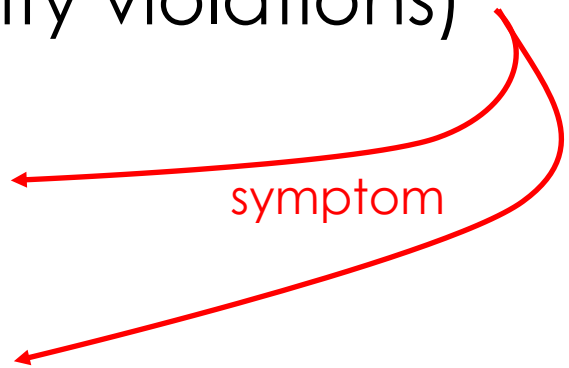
Deadlock

- Processes are waiting forever for each other



Common Concurrency Errors

- Data races (atomicity violations)
- Ordering violations
- Unintended sharing
- High-level atomicity violations
- Deadlocks and livelocks



Common Concurrency Errors

- Data races (atomicity violations)
- Ordering violations
- Unintended sharing
- High-level atomicity violations
- Deadlocks and livelocks

Concurrency Errors

Data Races

What is a Data Race?

- Two conflicting memory accesses happening concurrently
- Which means:
  - They access the same memory location
  - At least one is an update (write)

What is a Data Race?

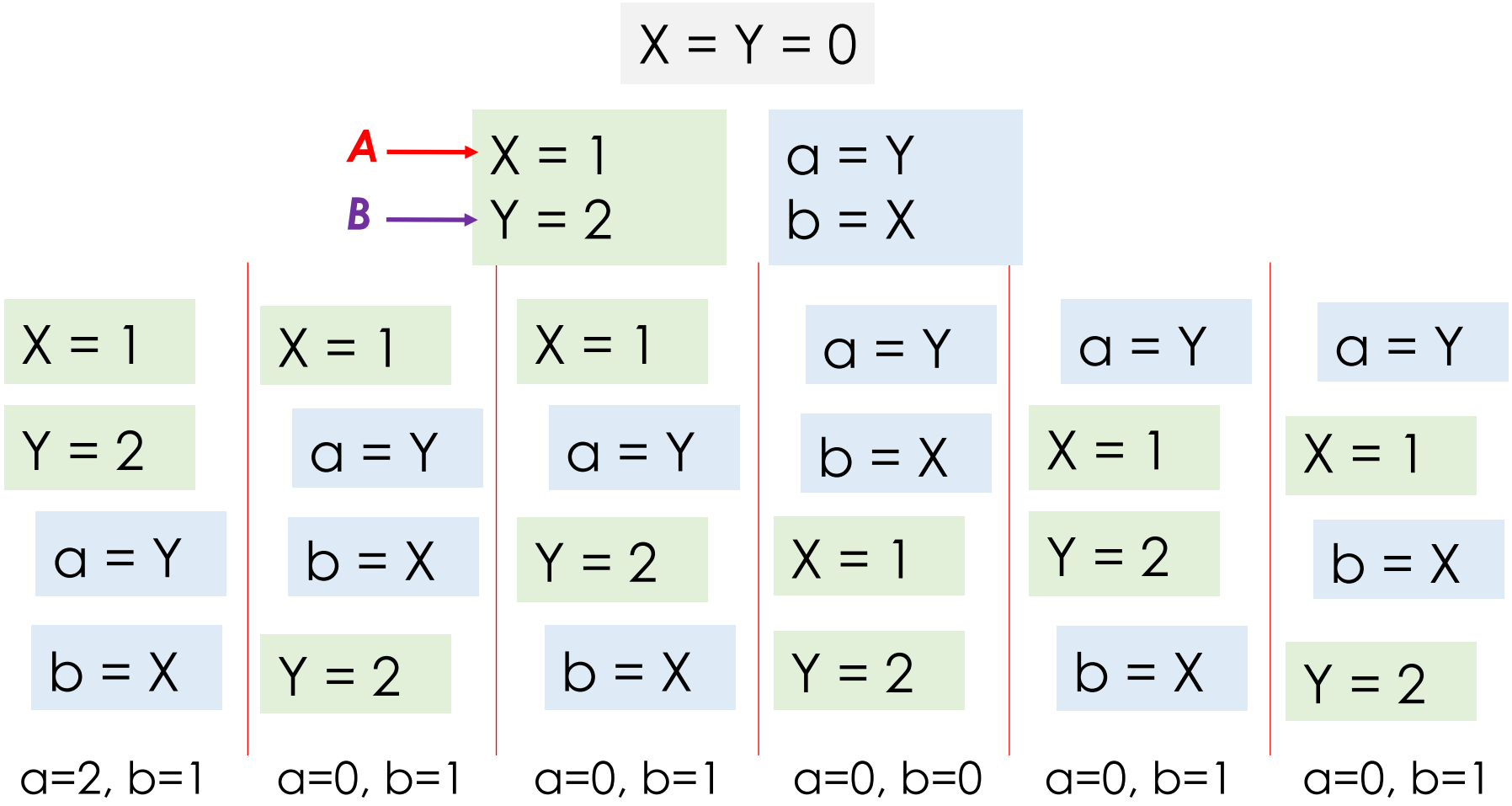
- Two conflicting memory accesses happening concurrently
- Which means:
  - They access the same memory location
  - At least one is an update (write)

- Write — Write
- Write — Read
- Read — Write
- Read — Read

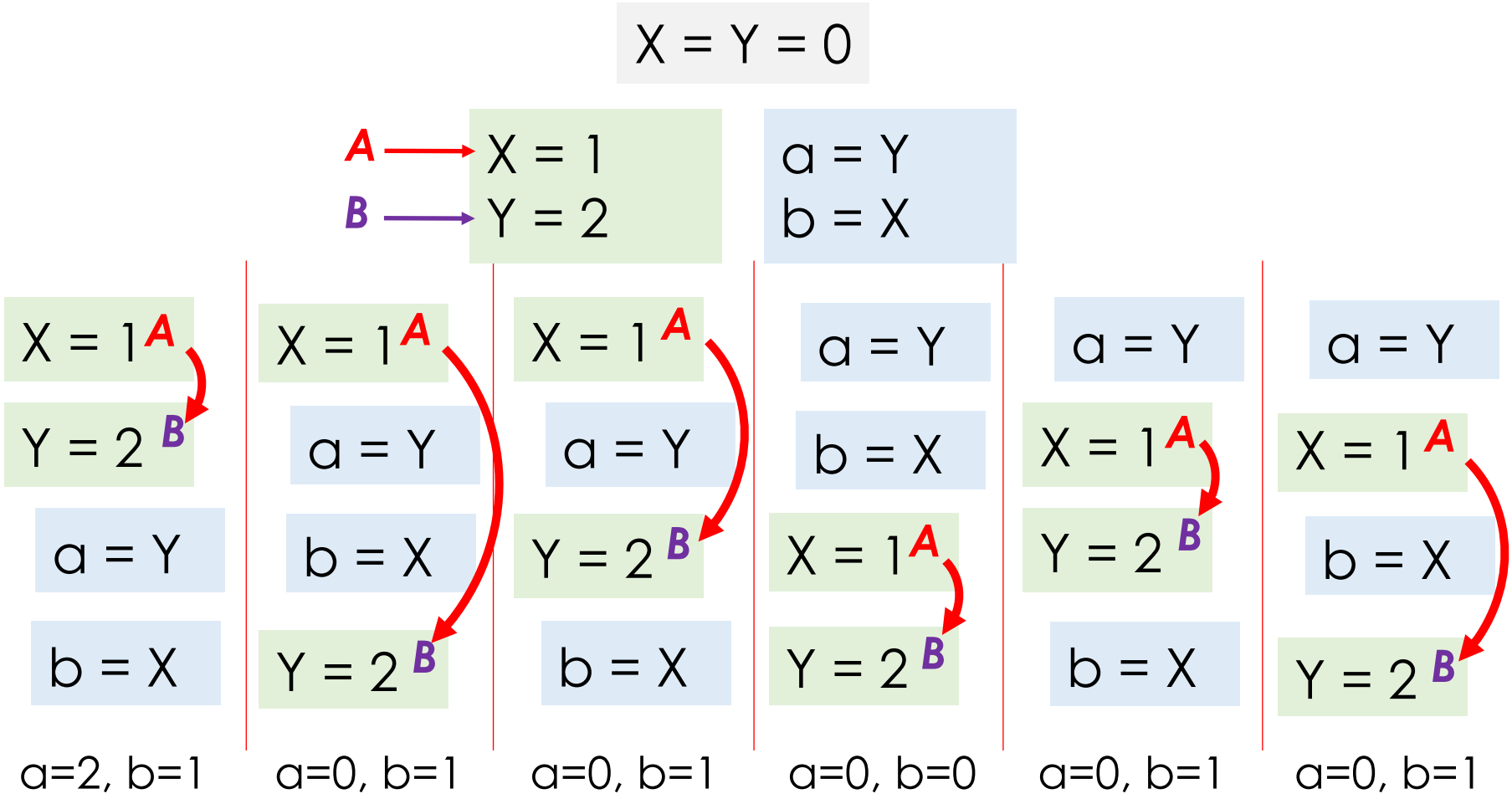
What means "Happens Concurrently"?

- Two events A and B happen concurrently if both
  A, B
  and
  B, A
  are possible sequentially consistent executions of those events

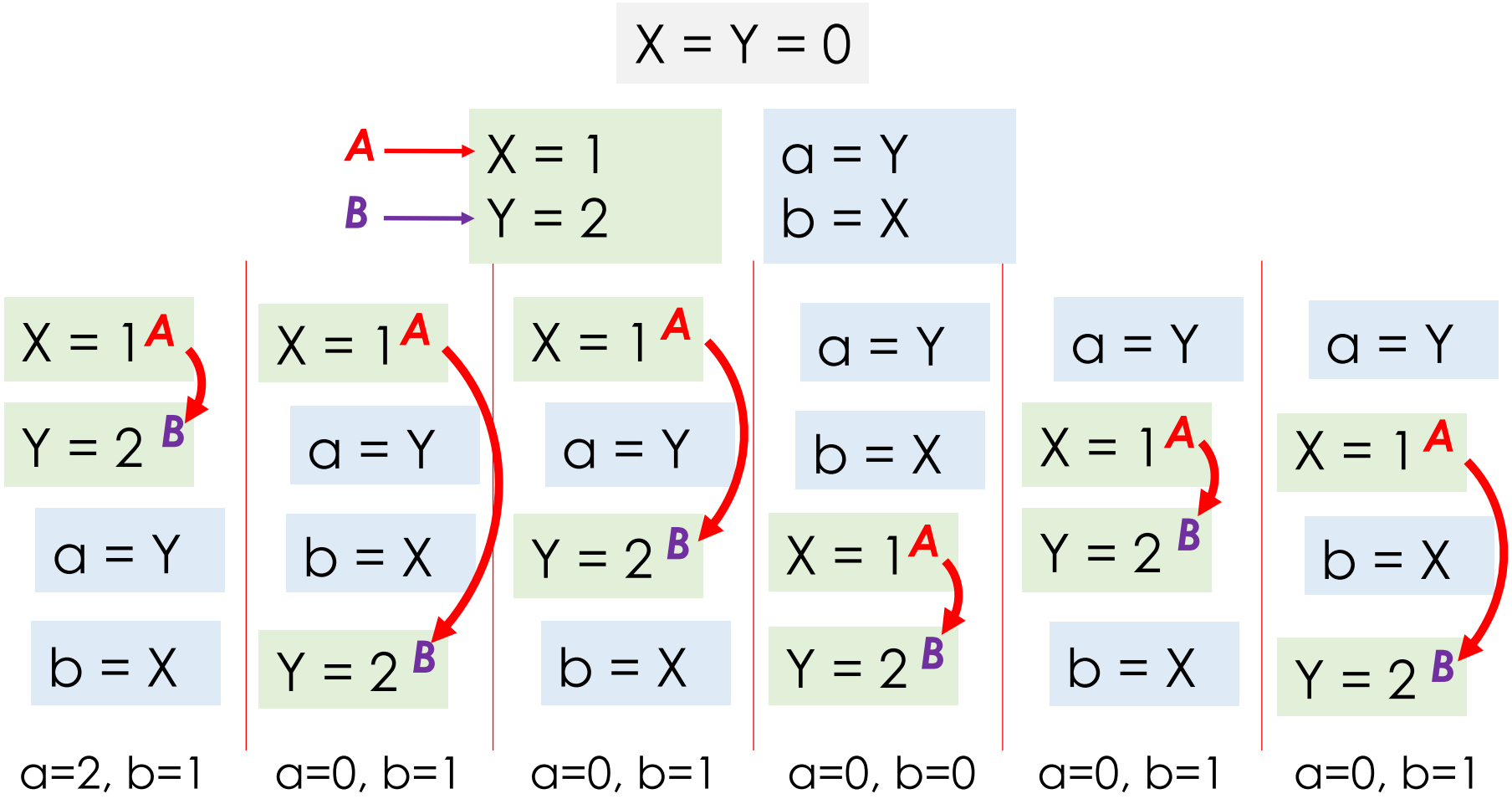
Assigning Semantics to Concurrent Programs



Assigning Semantics to Concurrent Programs

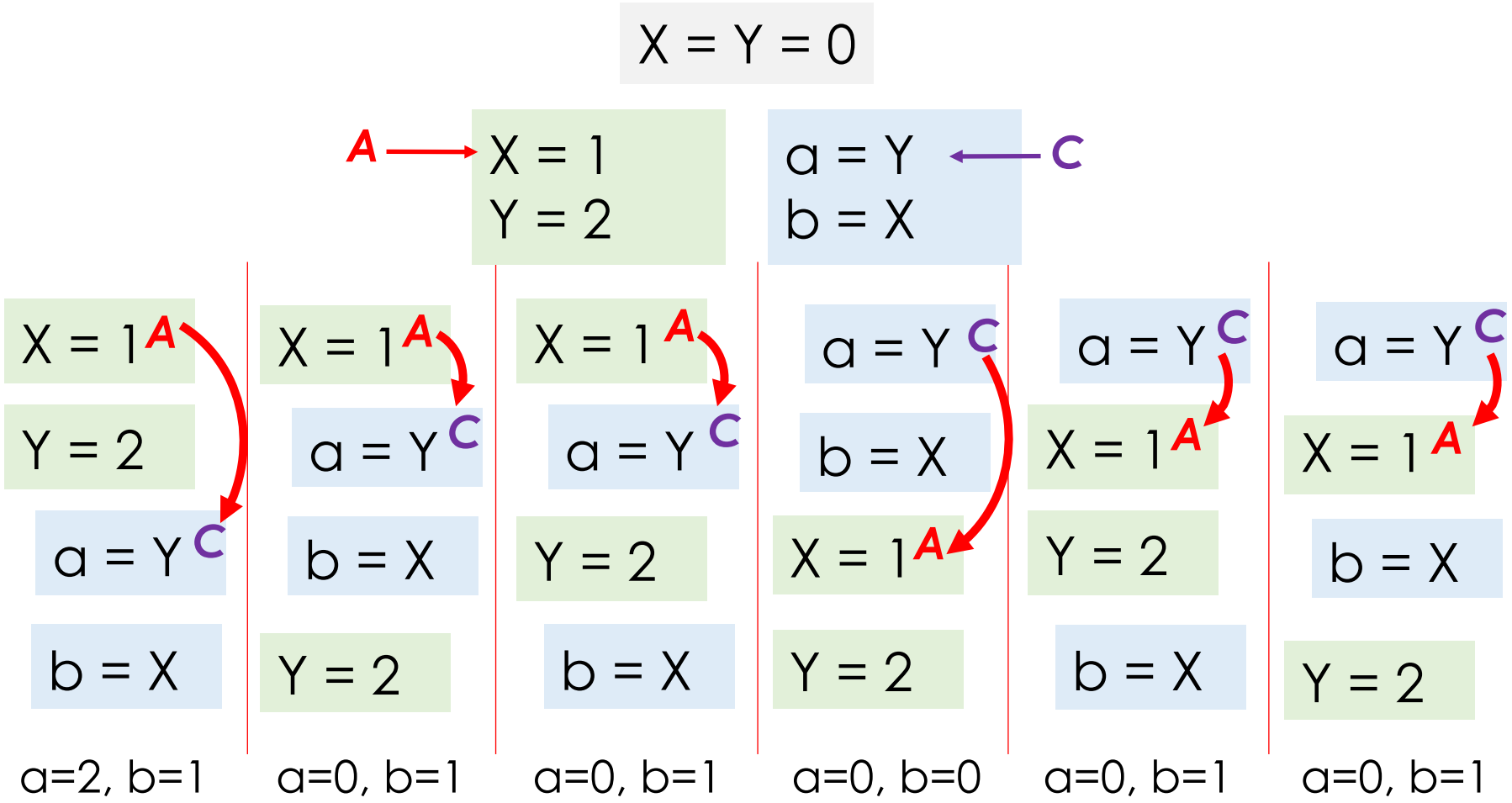


Assigning Semantics to Concurrent Programs



Always "A, B". Events 'A' and 'B' ***are not*** *concurrent!*

Assigning Semantics to Concurrent Programs



Both "A, C" and "C, A". Events 'A' and 'C' ***are*** *concurrent!*

Question

'**x**' is a shared variable, initially 0

Q: Knowing that processes A and B execute concurrently, what are the possible values for 'x' after both processes terminate?

Any value in the range 5 to 10! ***Wrong!!!***

Process A

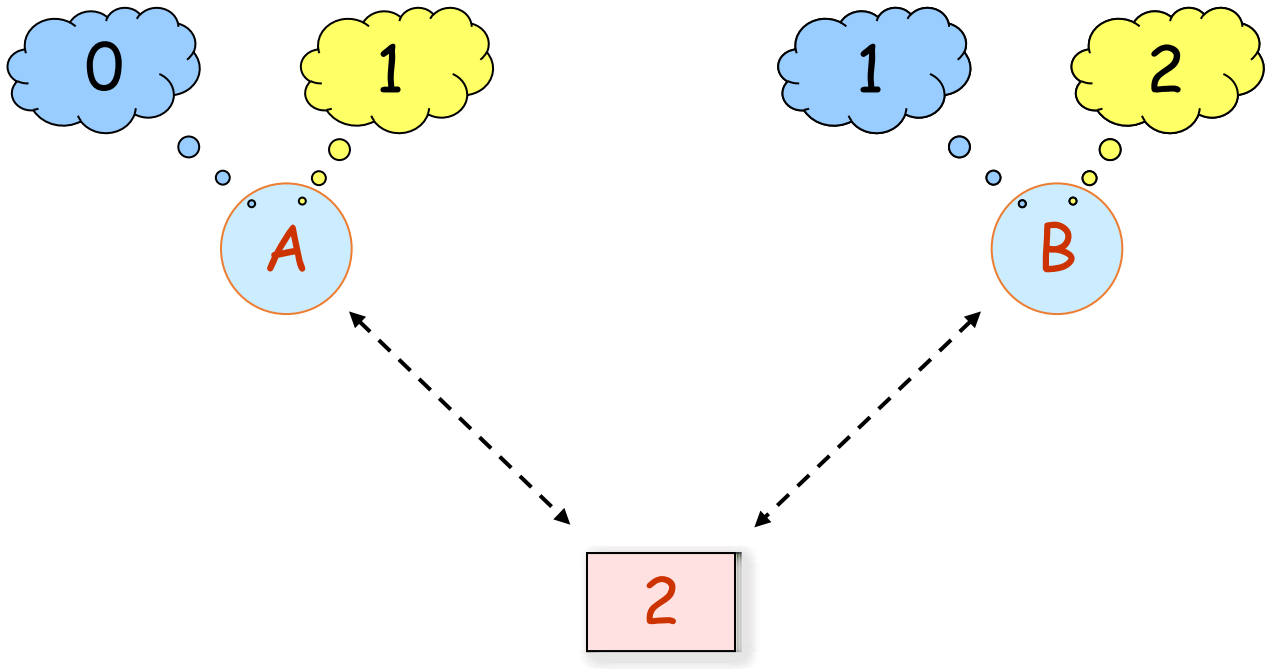
```
for (i = 0; i < 5;  i++) {
    x = x + 1
}
```

Process B

```
for (j = 0; j < 5;  j++) {
    x = x + 1
}
```

Any value in the range 2 to 10! ***How???***

The smallest value is 2



How to Detect a Data Race?

- Two concurrent accesses to a shared memory location
- At least one of them is a write

- How to monitor memory accesses?
- How to detect if two accesses are (or may be) concurrent?

Acknowledgments

- Some parts of this presentation was based in publicly available slides and PDFs
  - www.cs.cornell.edu/courses/cs4410/2011su/slides/lecture10.pdf
  - www.microsoft.com/en-us/research/people/madanm/
  - williamstallings.com/OperatingSystems/
  - codex.cs.yale.edu/avi/os-book/OS9/slide-dir/

The END