NOVA SCHOOL OF SCIENCE & TECHNOLOGY
COMPUTER SCIENCE DEPARTMENT

Assigning Semantics to Concurrent Programs

lecture 15 (2021-05-10)

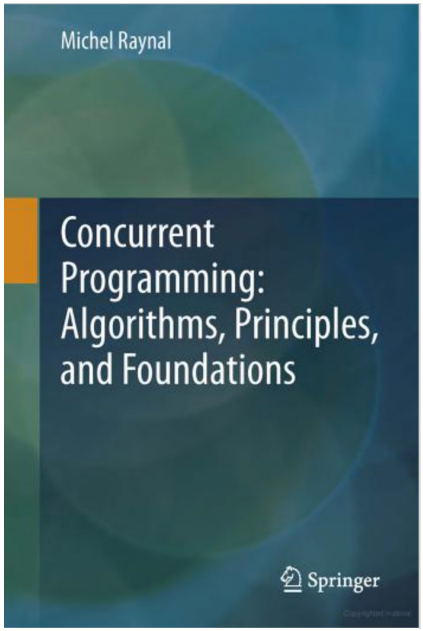
Master in Computer Science and Engineering

— Concurrency and Parallelism / 2020-21 —

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

Summary

- Assigning Semantics to Concurrent Programs
  - Atomicity/Linearizability; Sequential consistency Serializability
- Runs
  - Program state as a N-dimensional lattice
  - Consistent and Inconsistent runs
- **Reading list:**
  - **Chapter 4** of the book Raynal M.; **Concurrent Programming: Algorithms, Principles, and Foundations;** Springer-Verlag Berlin Heidelberg (2013); ISBN: 978-3-642-32026-2



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'?

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 statements
  - 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

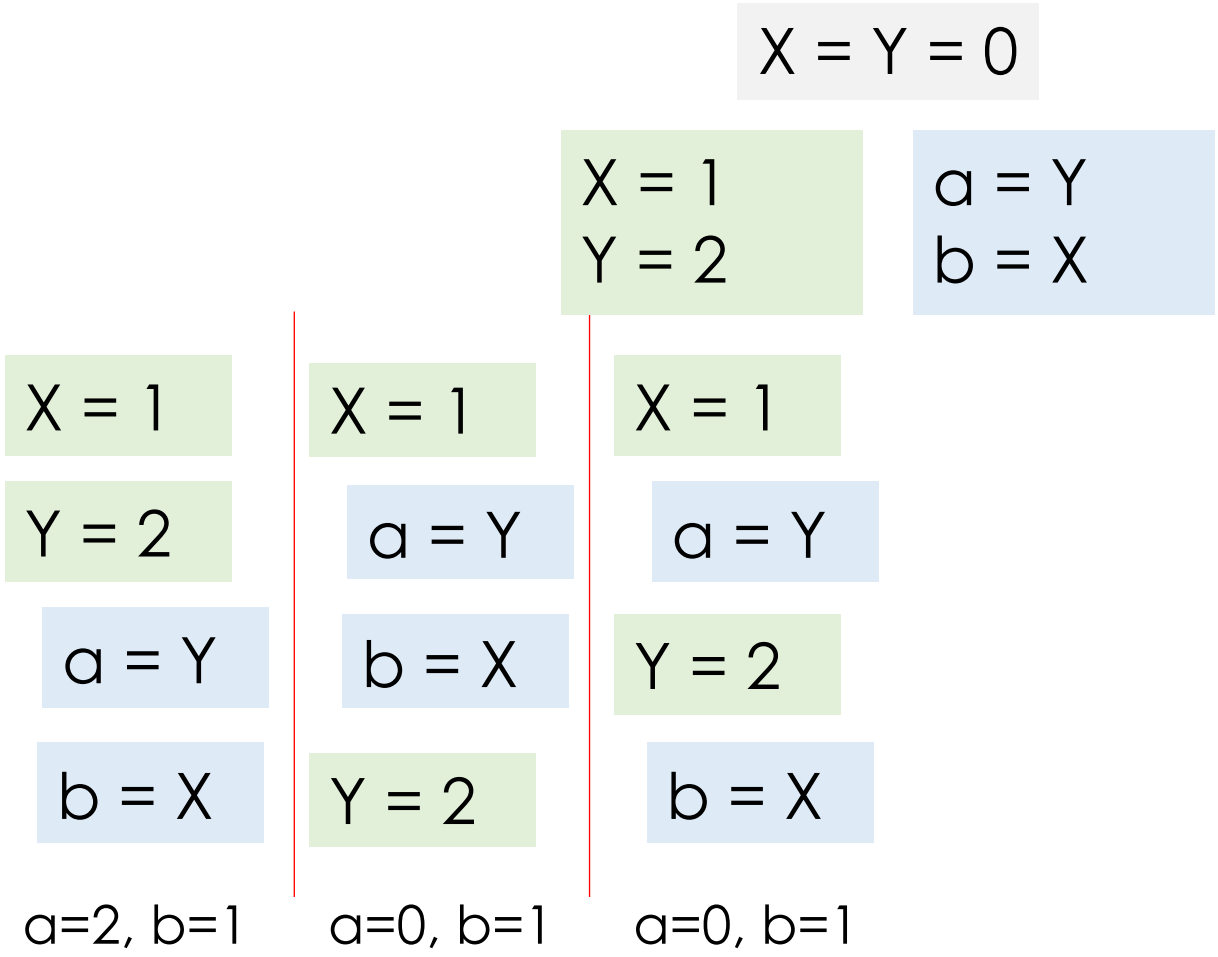
X = Y = 0

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

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

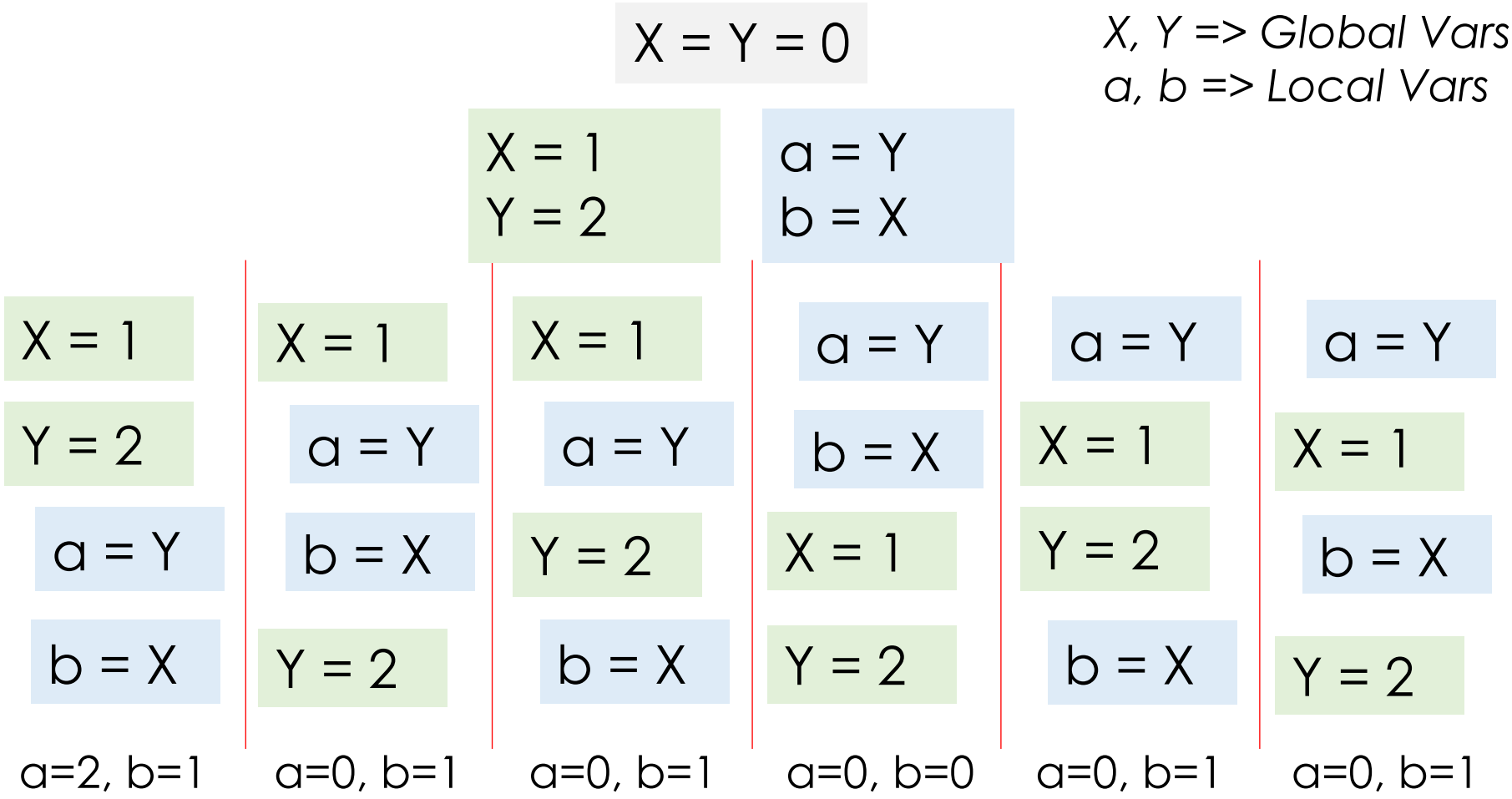
X = 1	X = 1
Y = 2	a = Y
a = Y	b = X
b = X	Y = 2
a=2, b=1	a=0, b=1

Assigning Semantics to Concurrent Programs



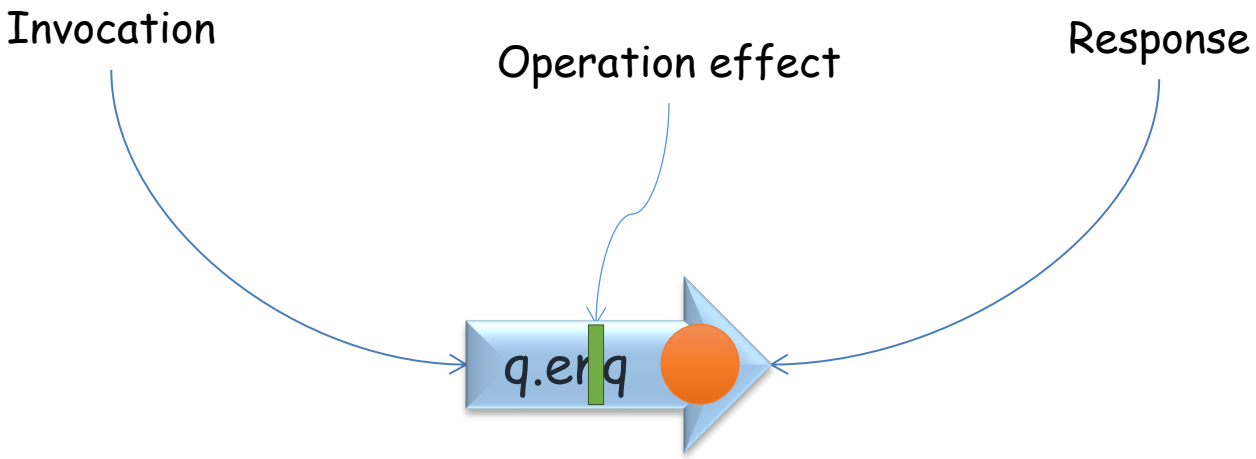
X, Y => Global Vars
a, b => Local Vars

Assigning Semantics to Concurrent Programs



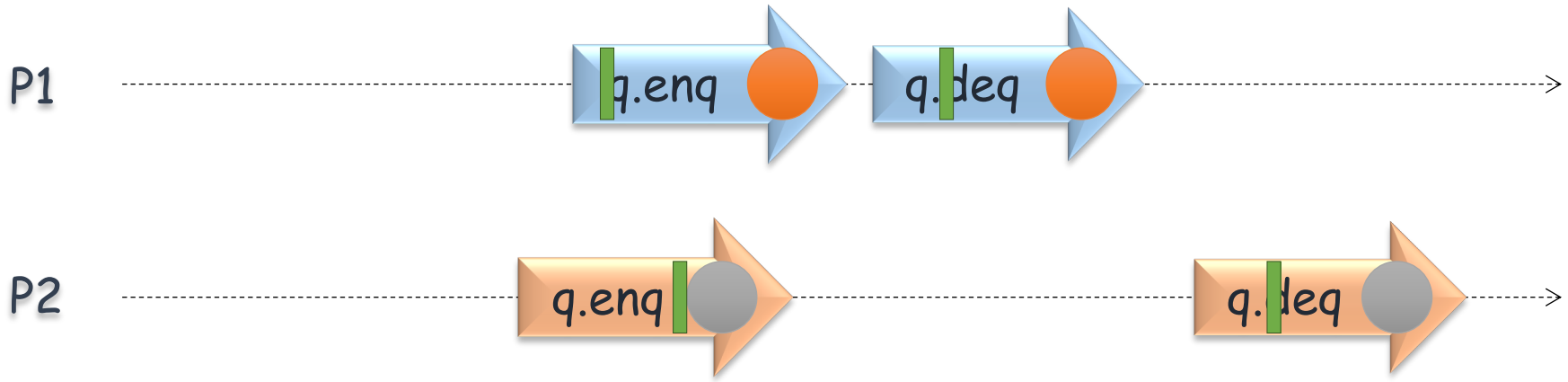
Sequential Consistency and Linearizability

- Each operation should appear to "take effect" instantaneously at some moment between its invocation and response



Sequential Consistency and Linearizability

- Each operation should appear to "take effect" instantaneously at some moment between its invocation and response
- Object is correct if its behaviour can be explained by a sequential execution of the (concurrent) operations



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

- Be aware that:
  - This is naturally true for sequential programs…
  - But it is not true for concurrent programs!

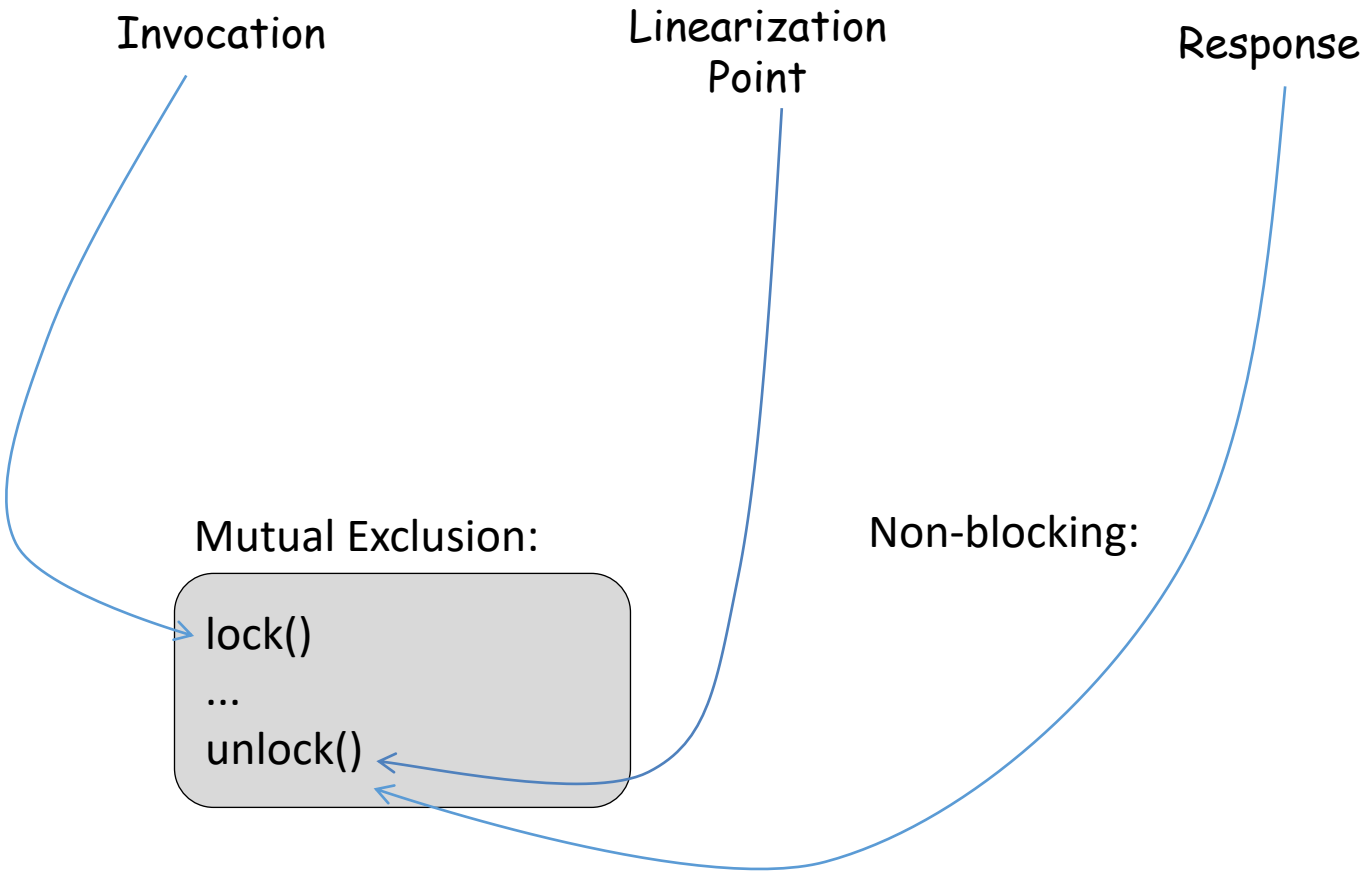
Linearizability

- Instructions are executed by the order they appear in the program
- Memory behaves as a shared array
  - Reads and writes are effective immediately
- Need to preserve real time order
- Be aware that:
  - This is naturally true for sequential programs…
  - But it is not true for concurrent programs!

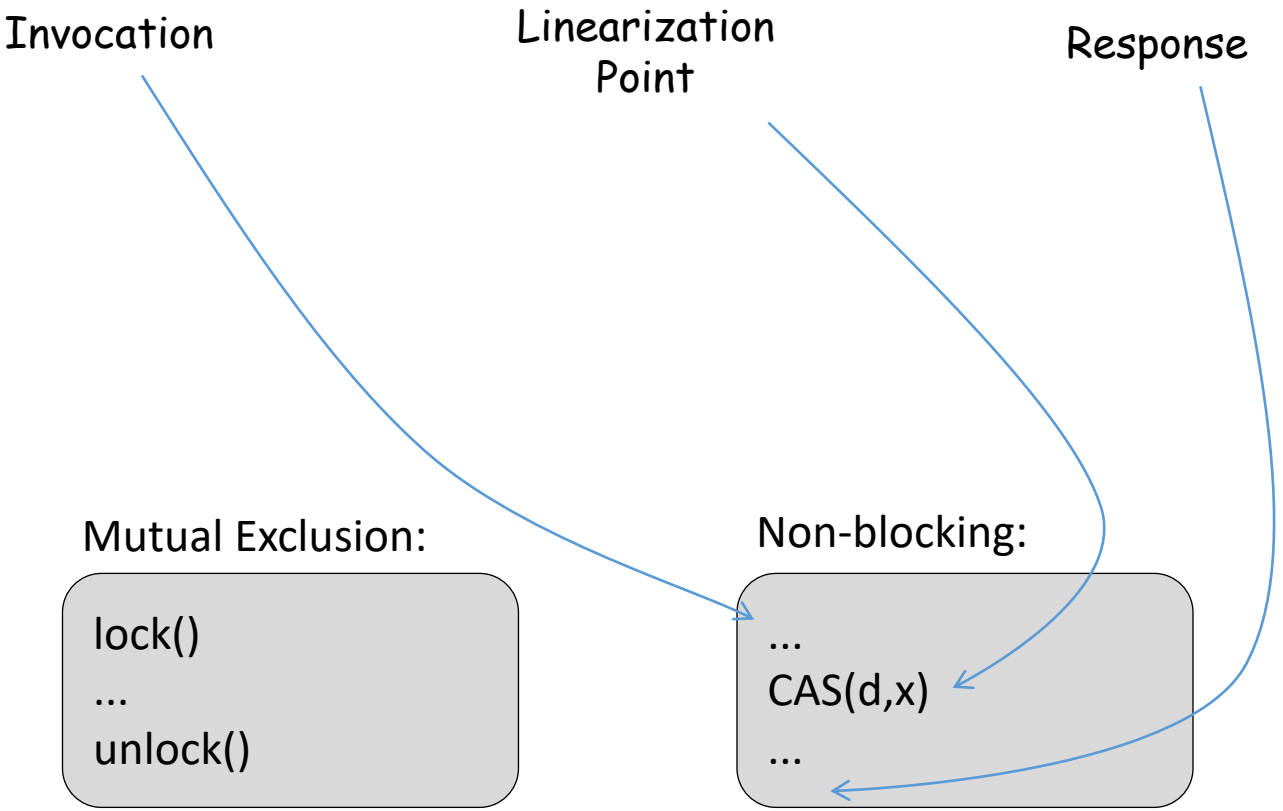
Linearizability

- Permits programmers to specify and reason about concurrent objects using known techniques from the sequential domain
- Provides the illusion that each operation applied by concurrent processes takes effect instantaneously between its invocation and its response

Terminology



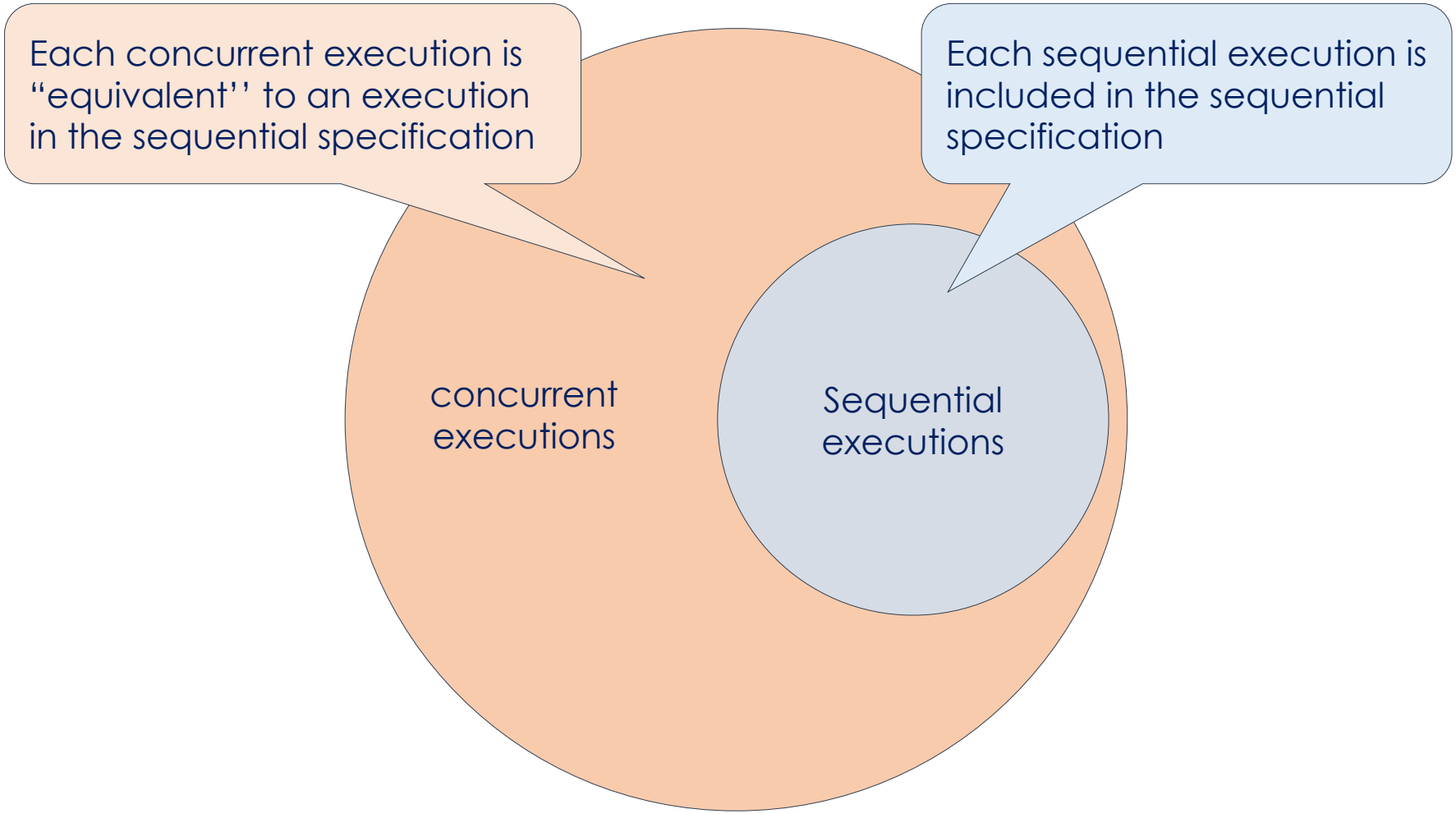
Terminology



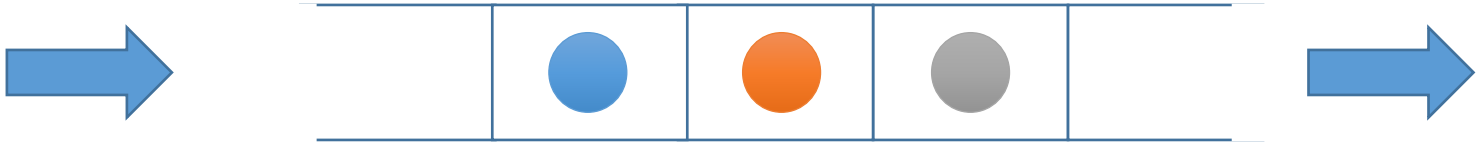
Sequential Specification of an object

- **Queue:** In sequential executions, the enqueue operation inserts a value to the queue and the dequeue operation returns and deletes the oldest value in the queue

Linearizability for an object



Example: Concurrent Queue

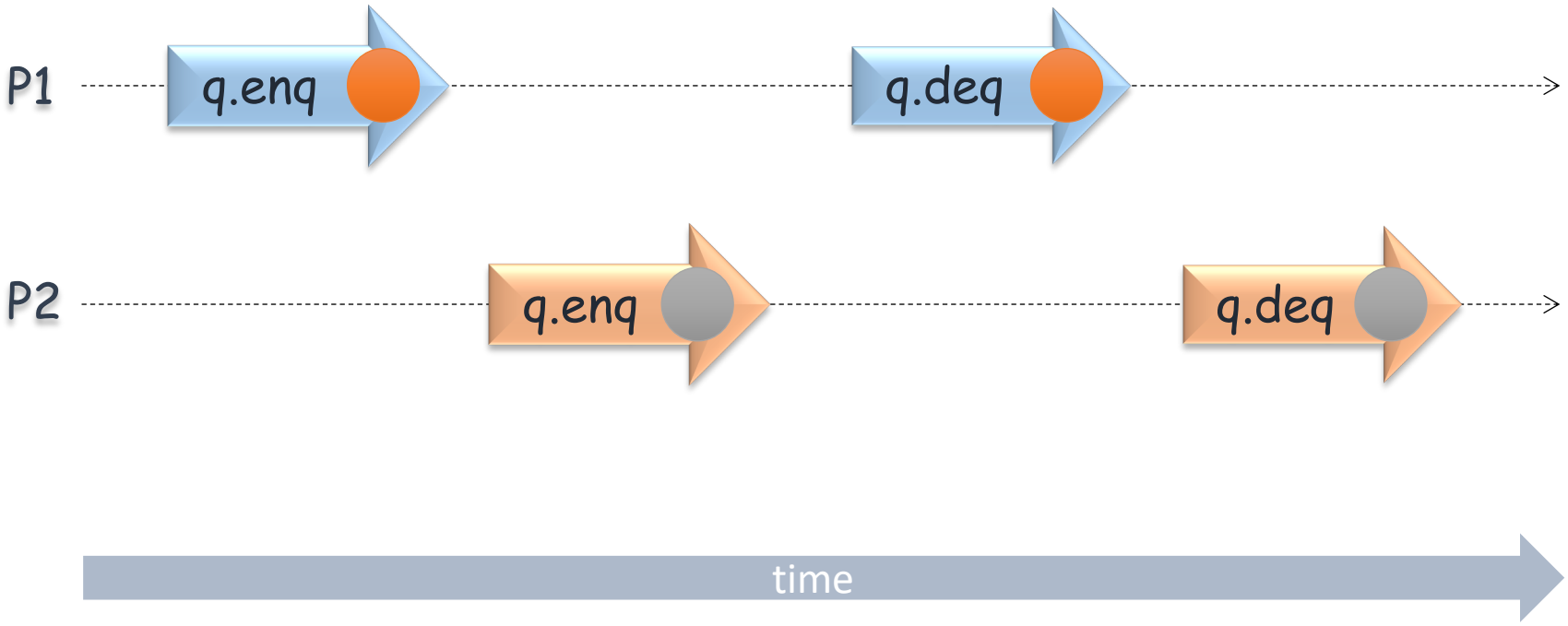


q.enq( )

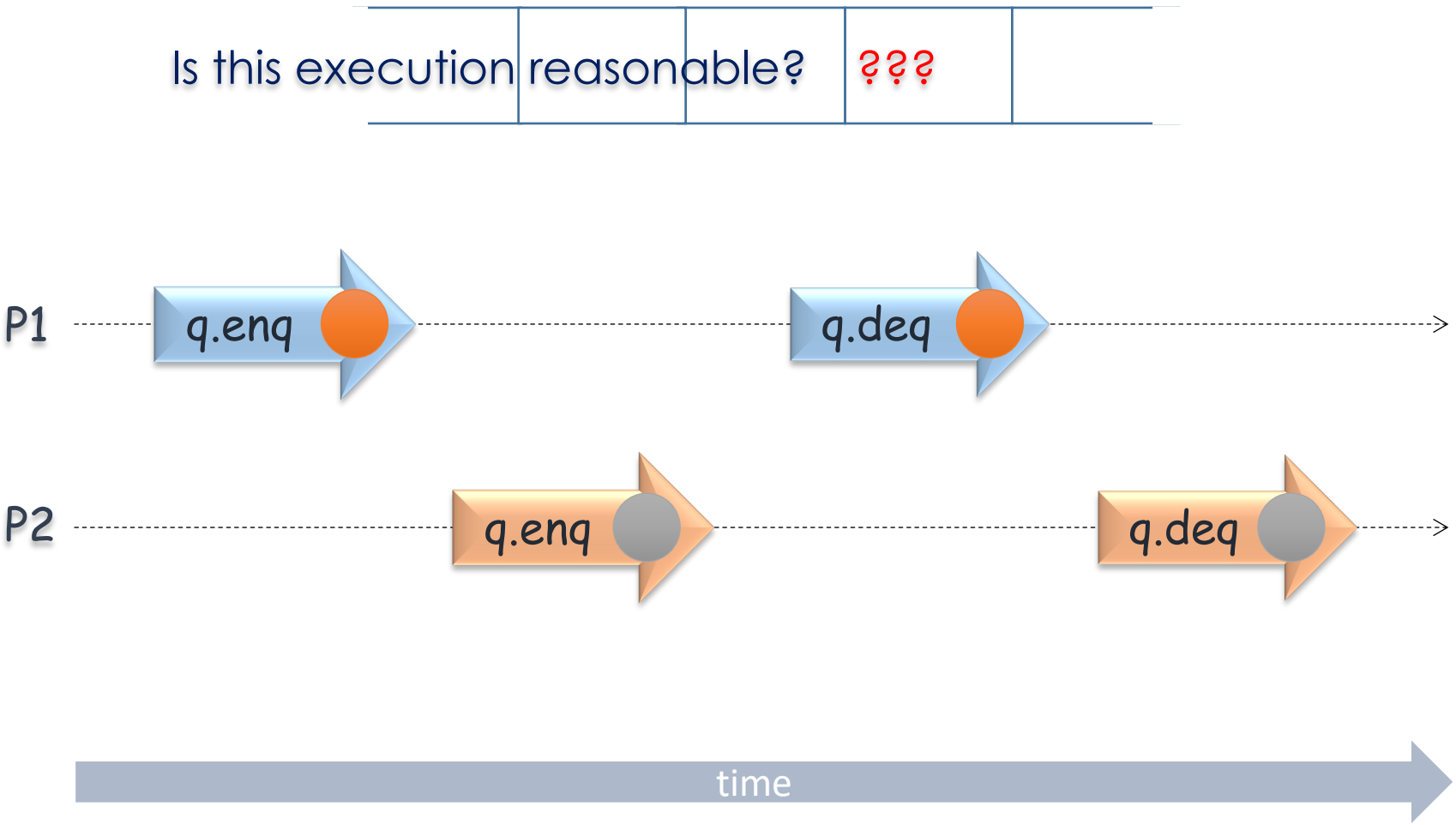
q.deq( )

Sequential Execution

Is this execution reasonable? Yes

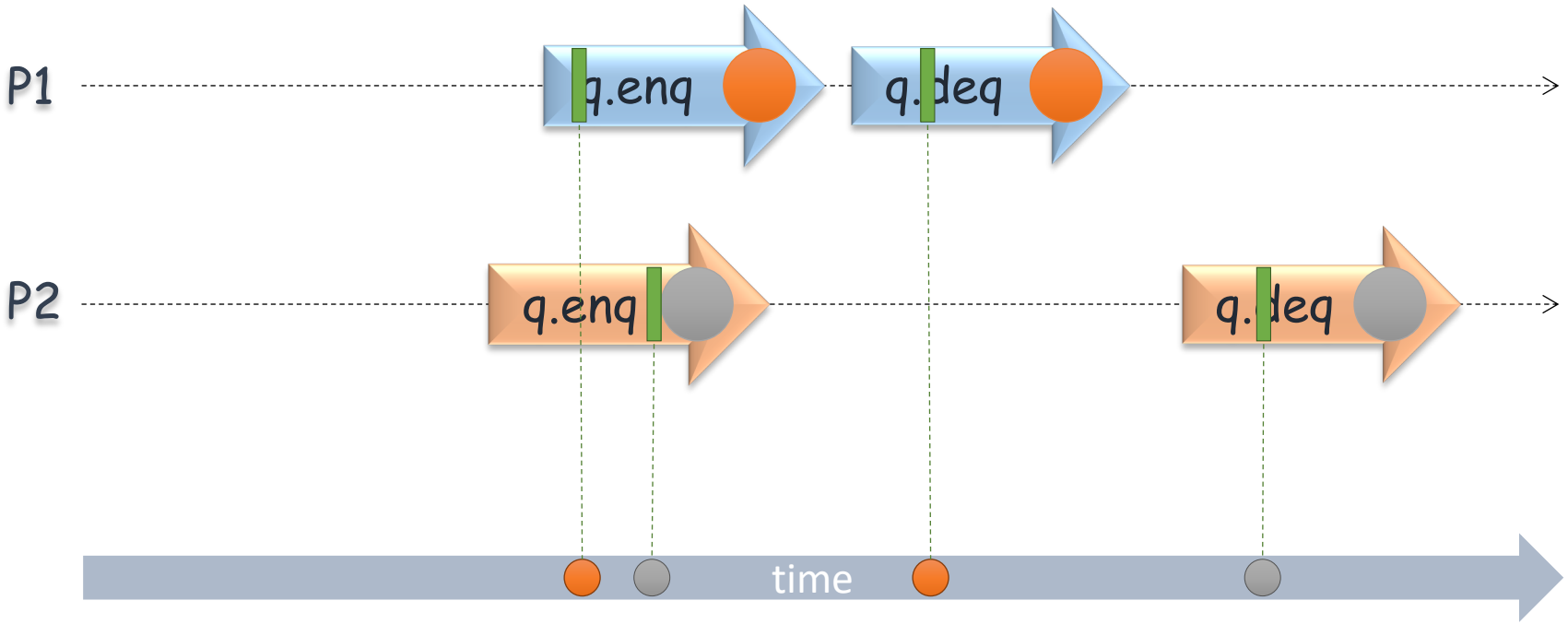


Concurrent Execution



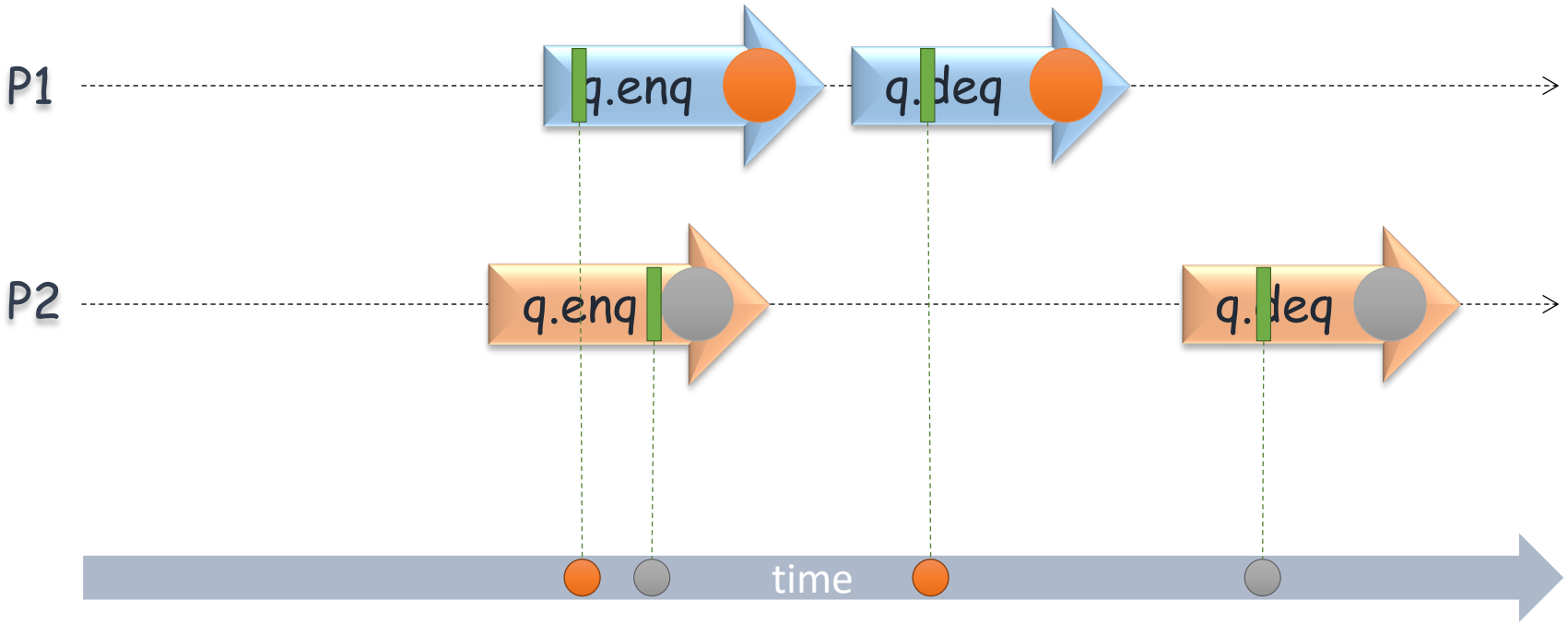
Concurrent Execution

Is this execution reasonable? Yes



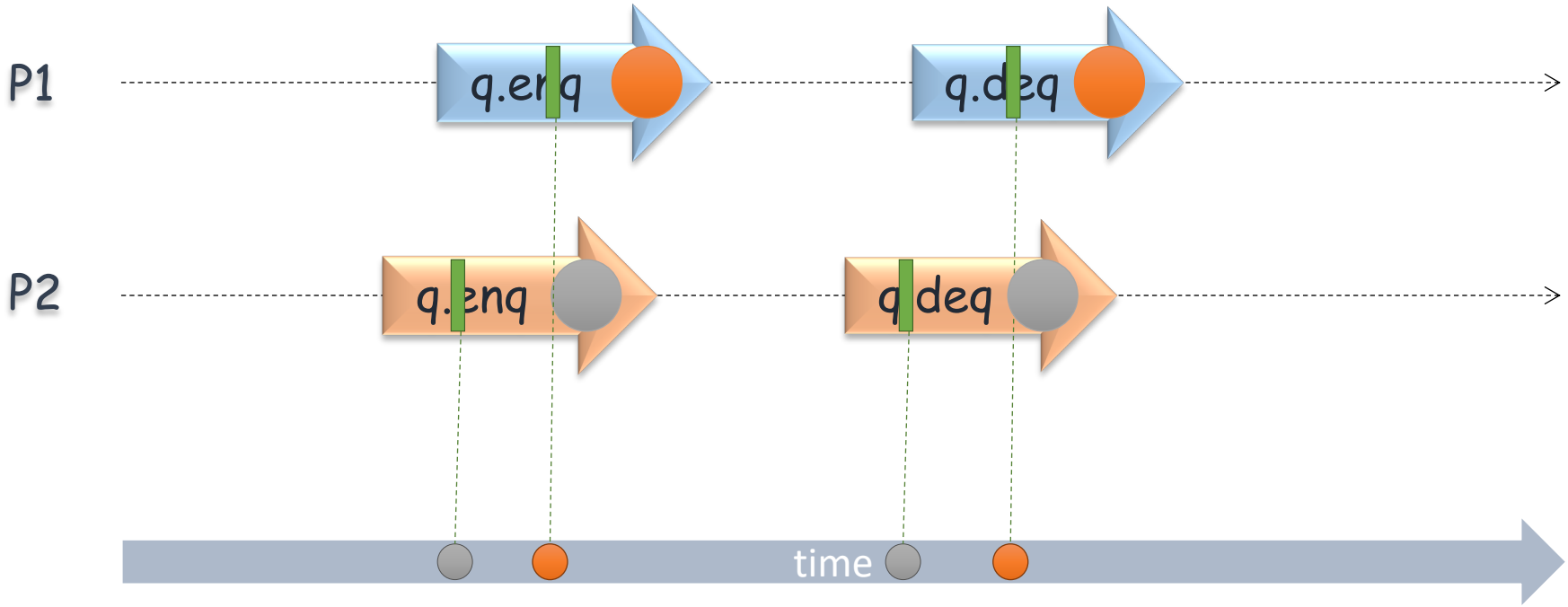
Concurrent Execution

Is this execution reasonable? Yes



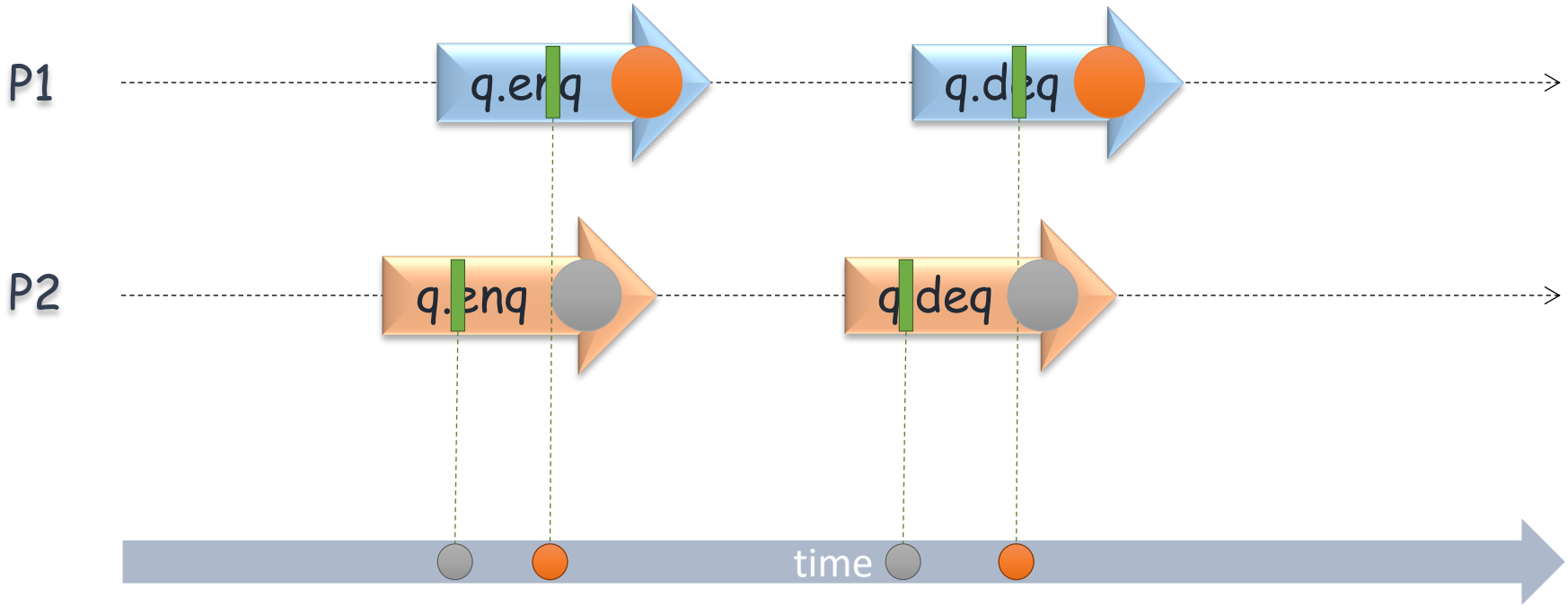
Linearizability

Is this execution **linearizable**? Yes



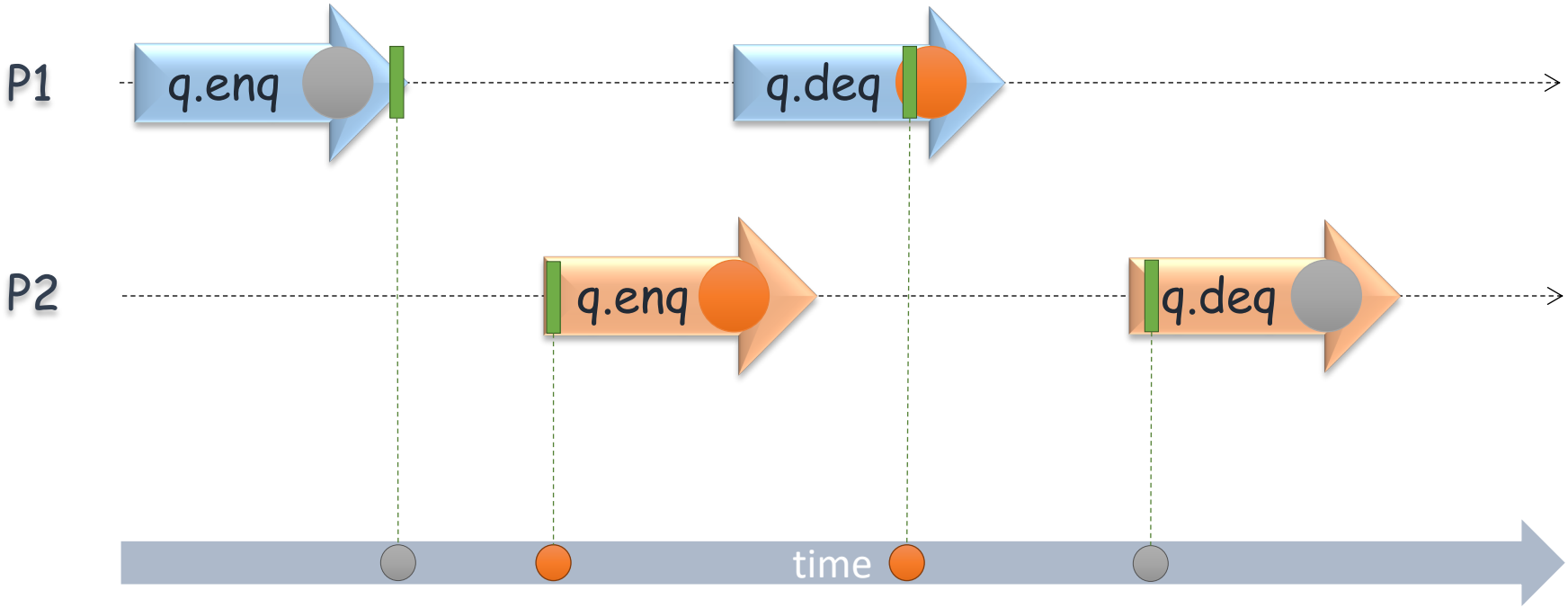
Linearizability

Is this execution **linearizable**? Yes



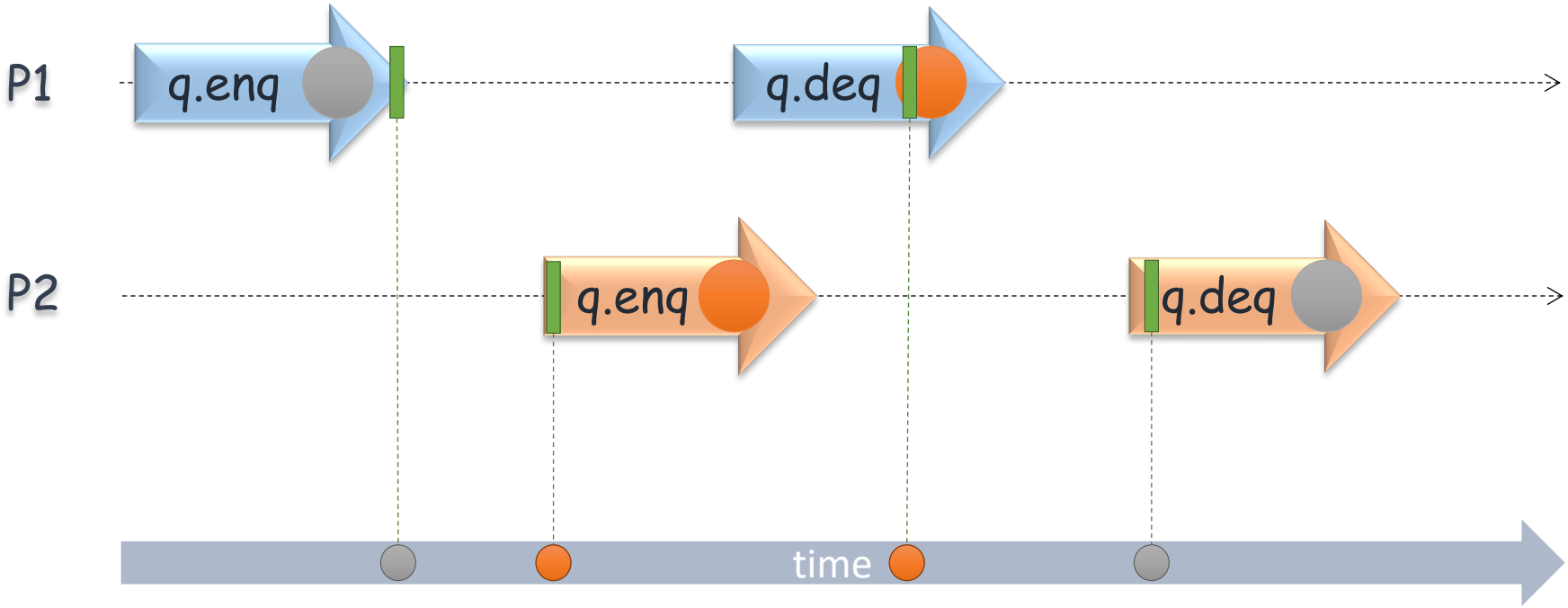
Linearizability

Is this execution **linearizable**? No, BUT …



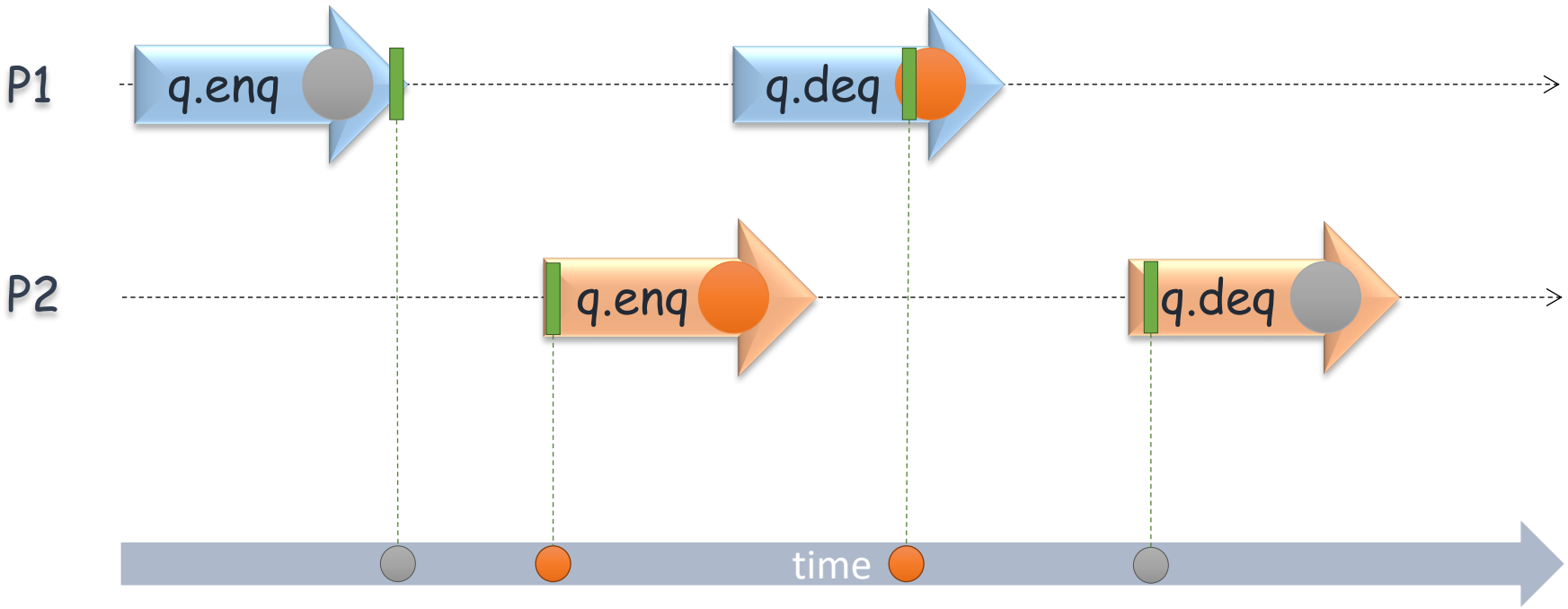
Linearizability

Is this execution **linearizable**? No, BUT …



~~Linearizability~~ Sequential Consistency

Is this execution **linearizable**? No, BUT …

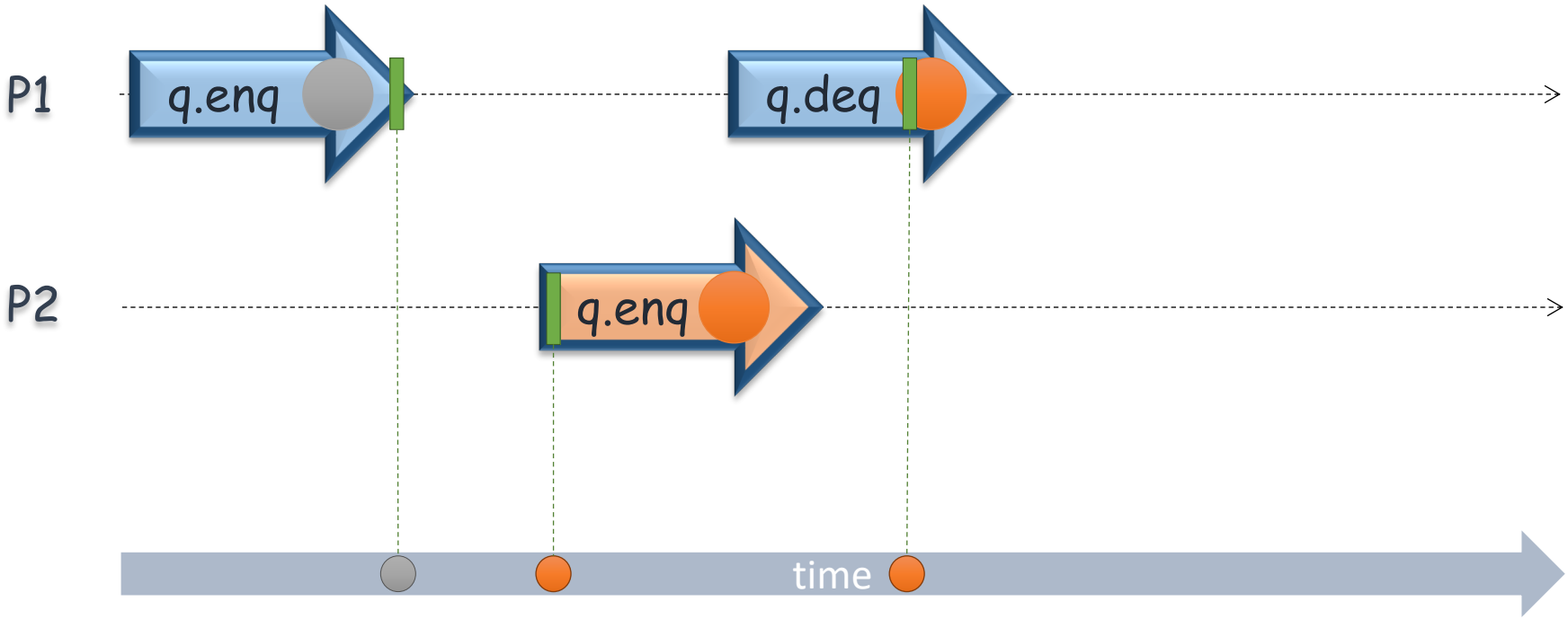


Sequential Consistency

- No need to preserve real time order
  - Operations can "slide" within the same thread but not be reordered
  - Parallel operations can be reordered as convenient
- Each operation should "take effect" instantaneously
- The result of the concurrent execution of a program P must be equivalent to some sequential execution of the same program P

Sequential Consistency

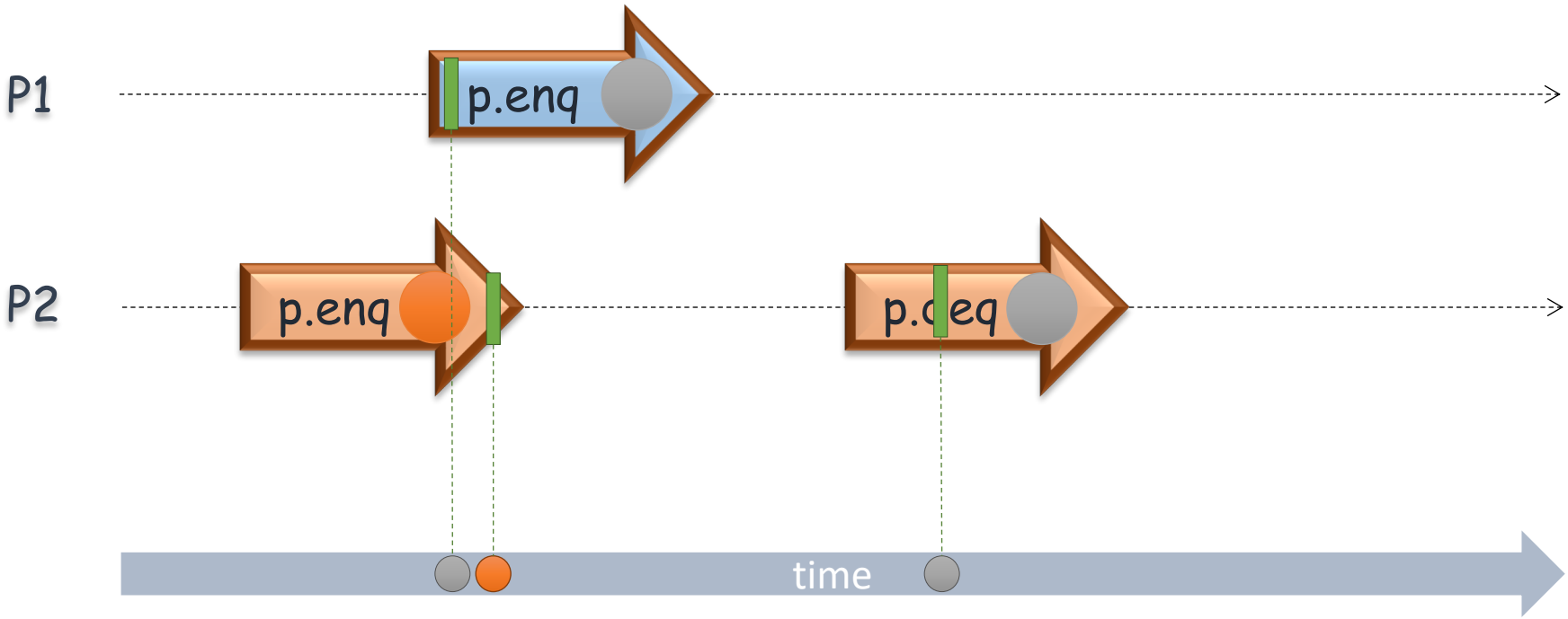
Is this execution sequentially consistent? Yes



Sequential Consistency

Is this execution sequentially consistent? Yes

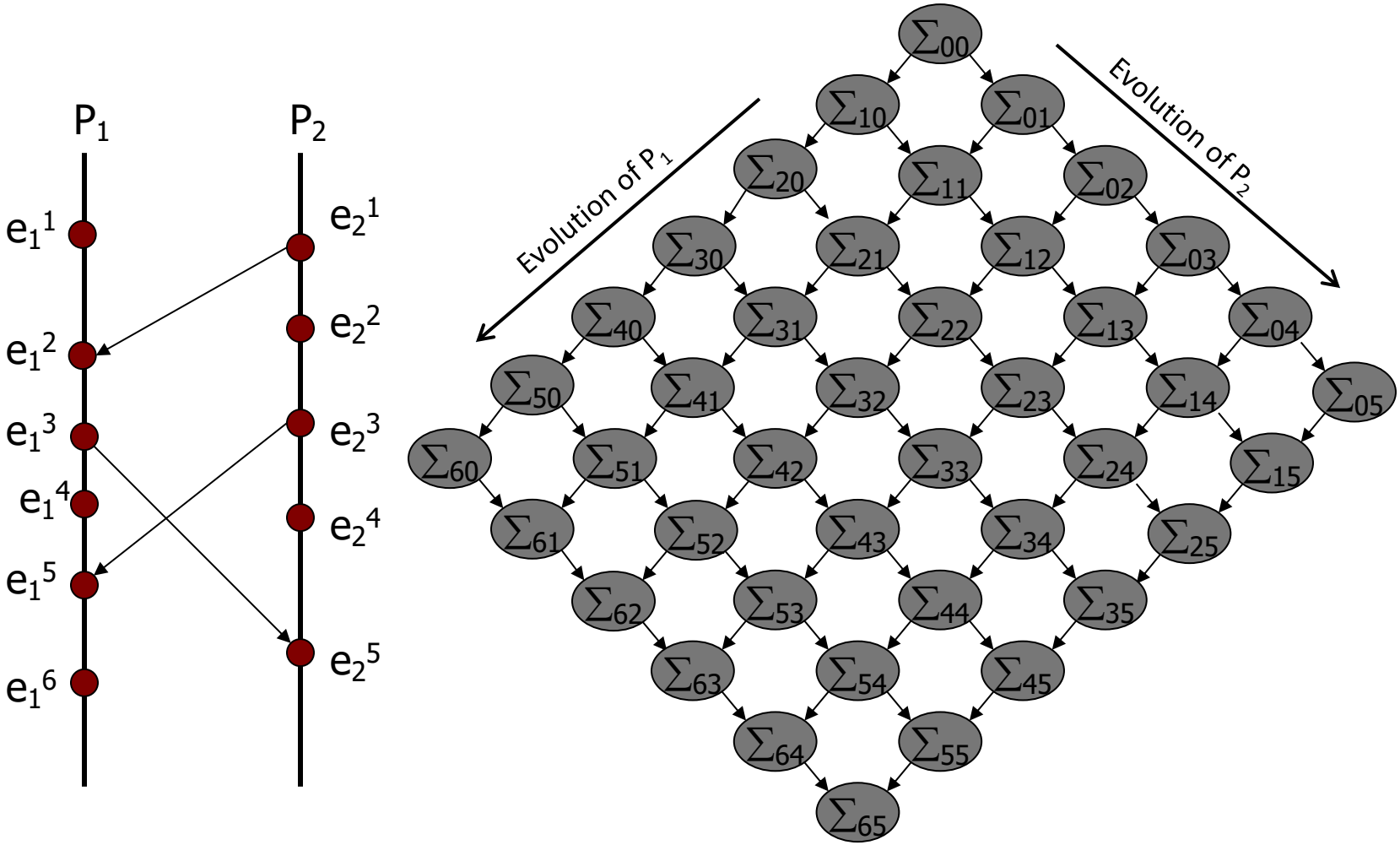
It is also linearizable!



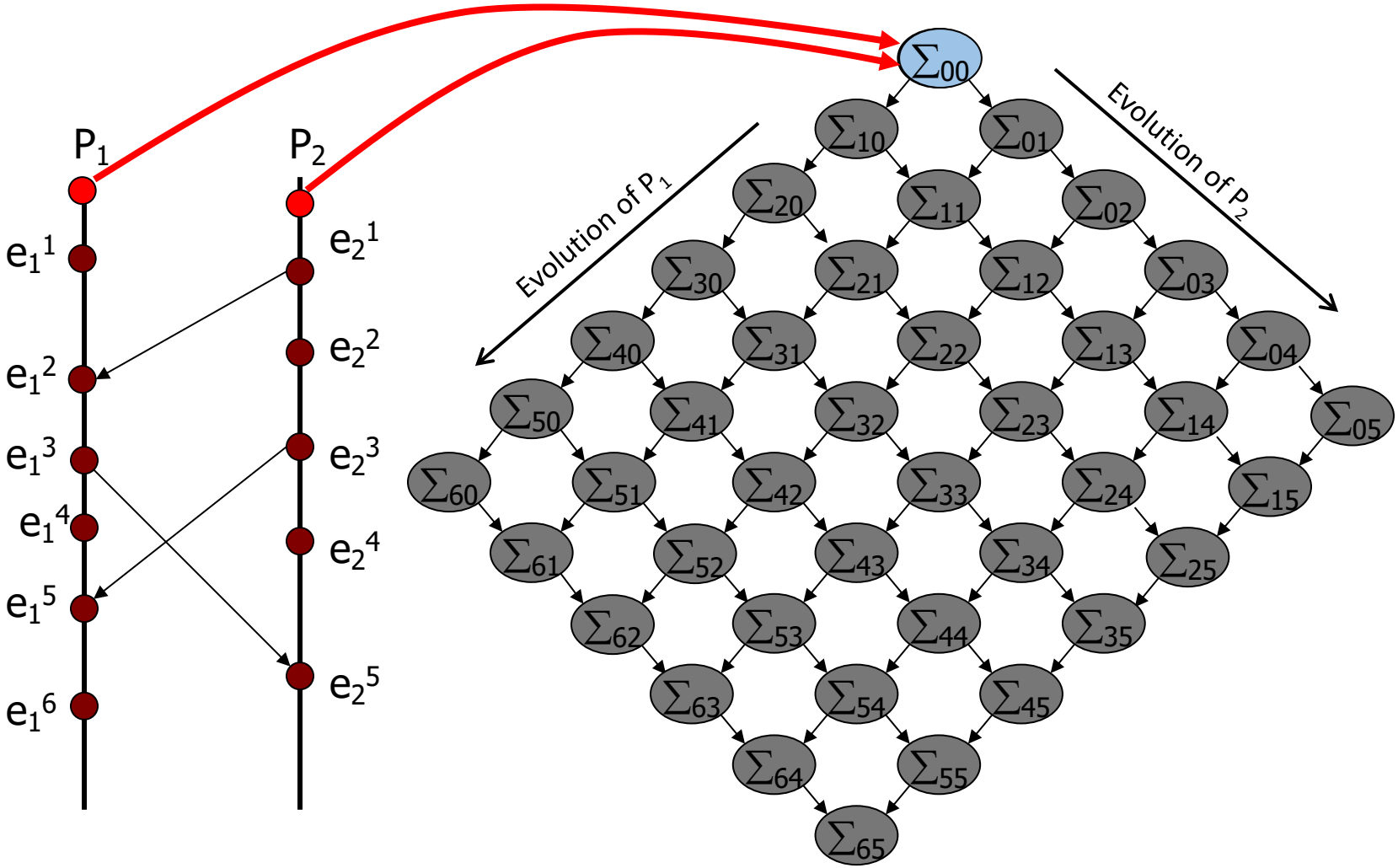
Program State and Runs

- Program state as a N-dimensional lattice
- Runs
  - Consistent
  - Inconsistent

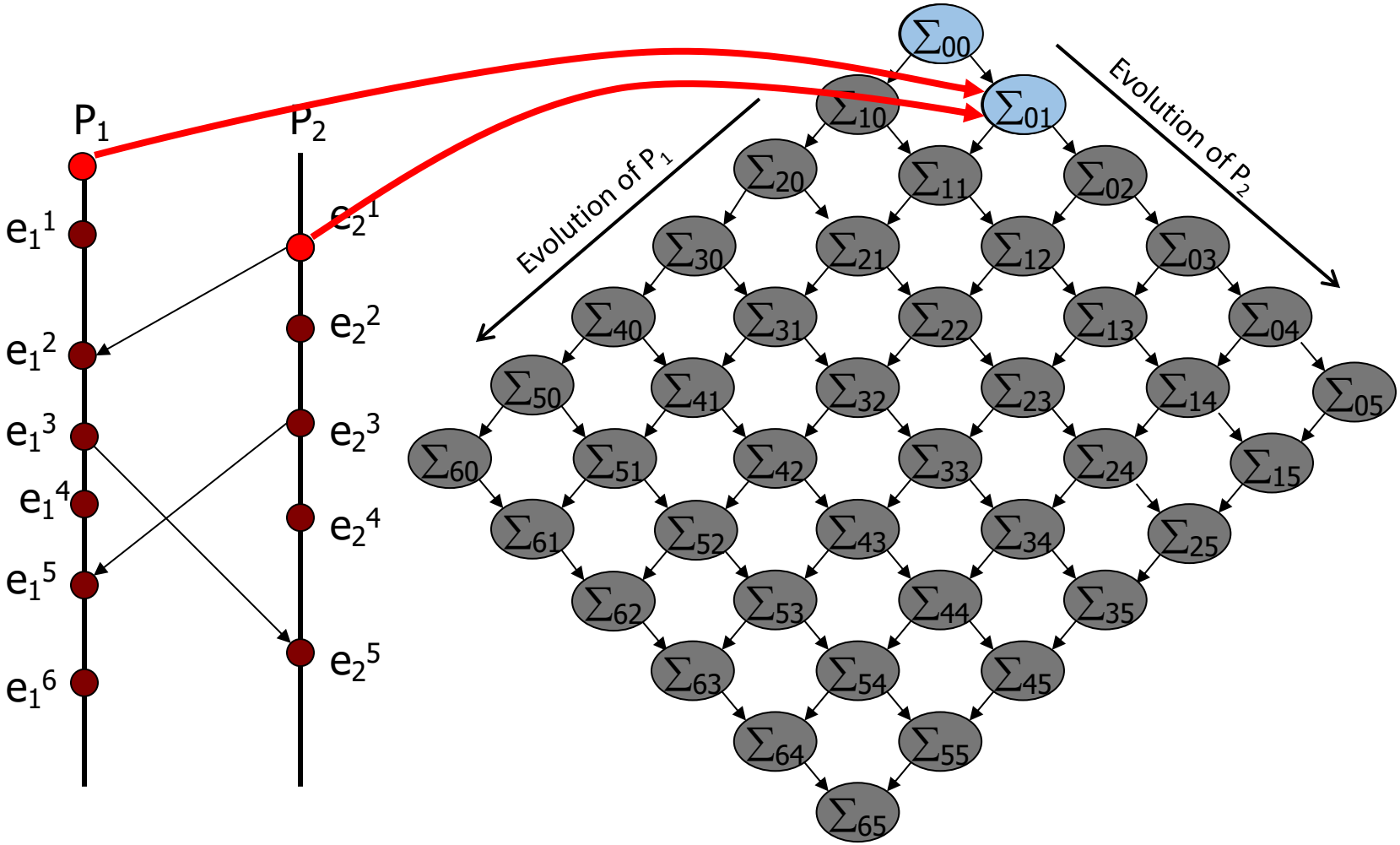
State explosion in concurrent programs



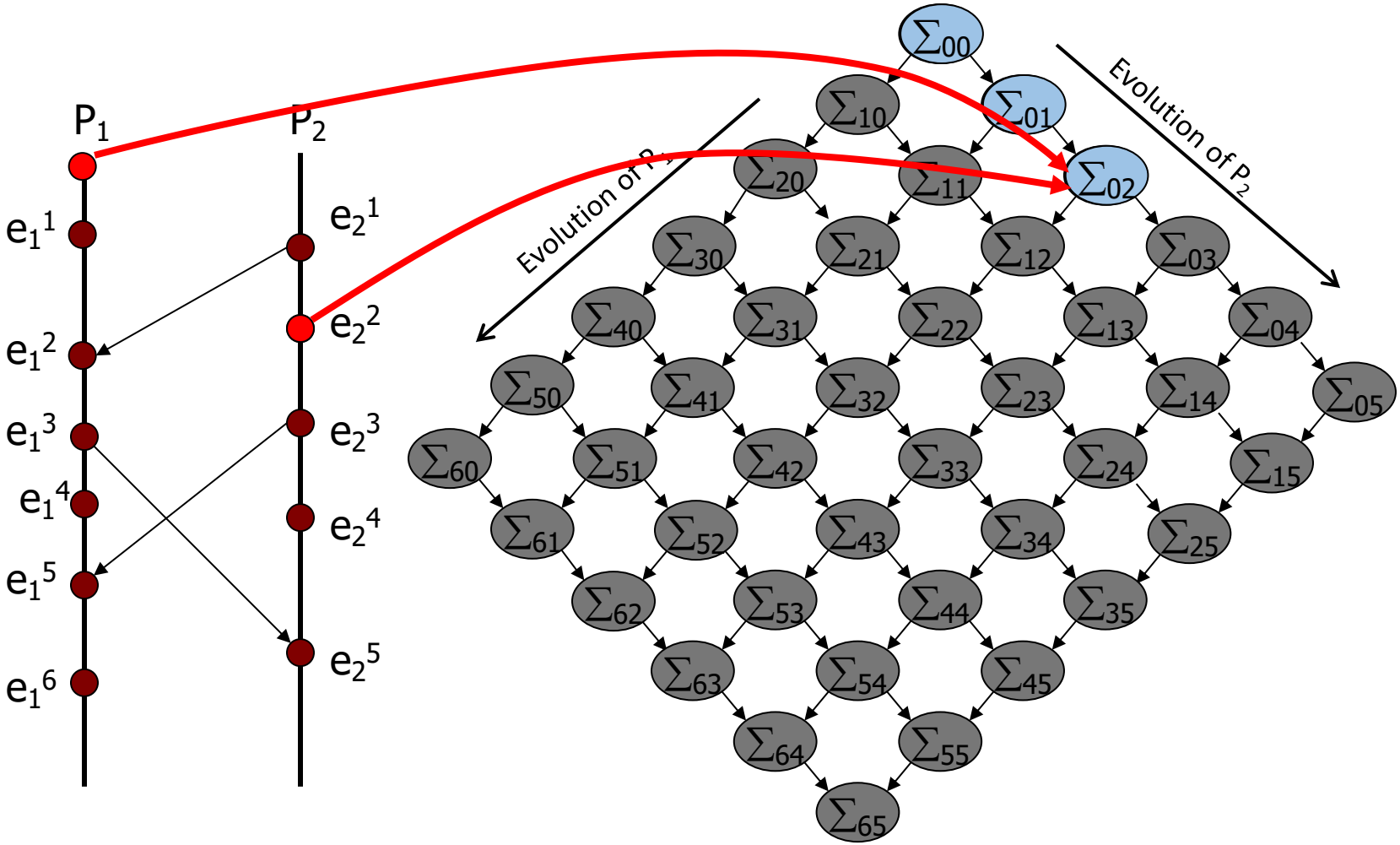
State explosion in concurrent programs



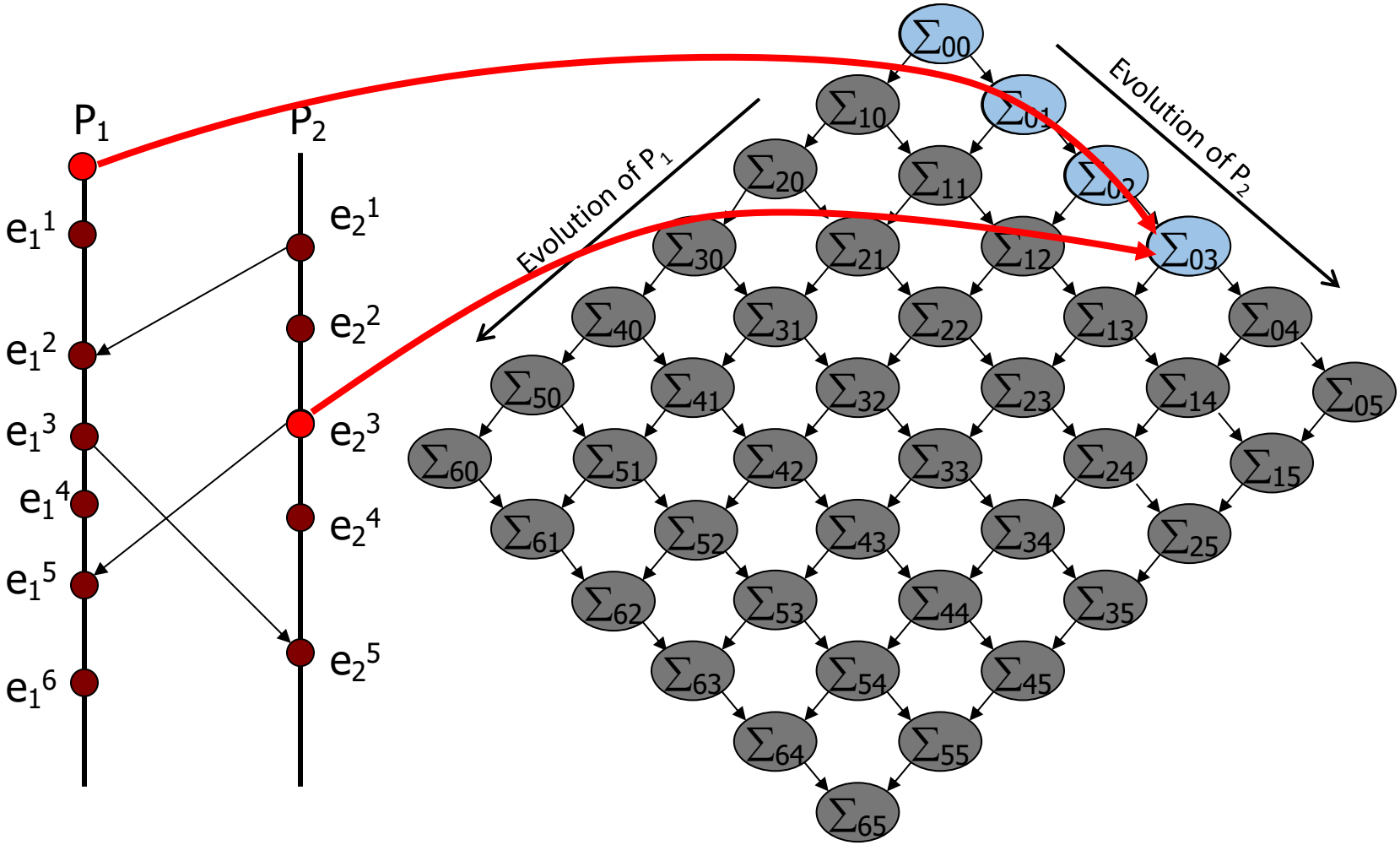
State explosion in concurrent programs



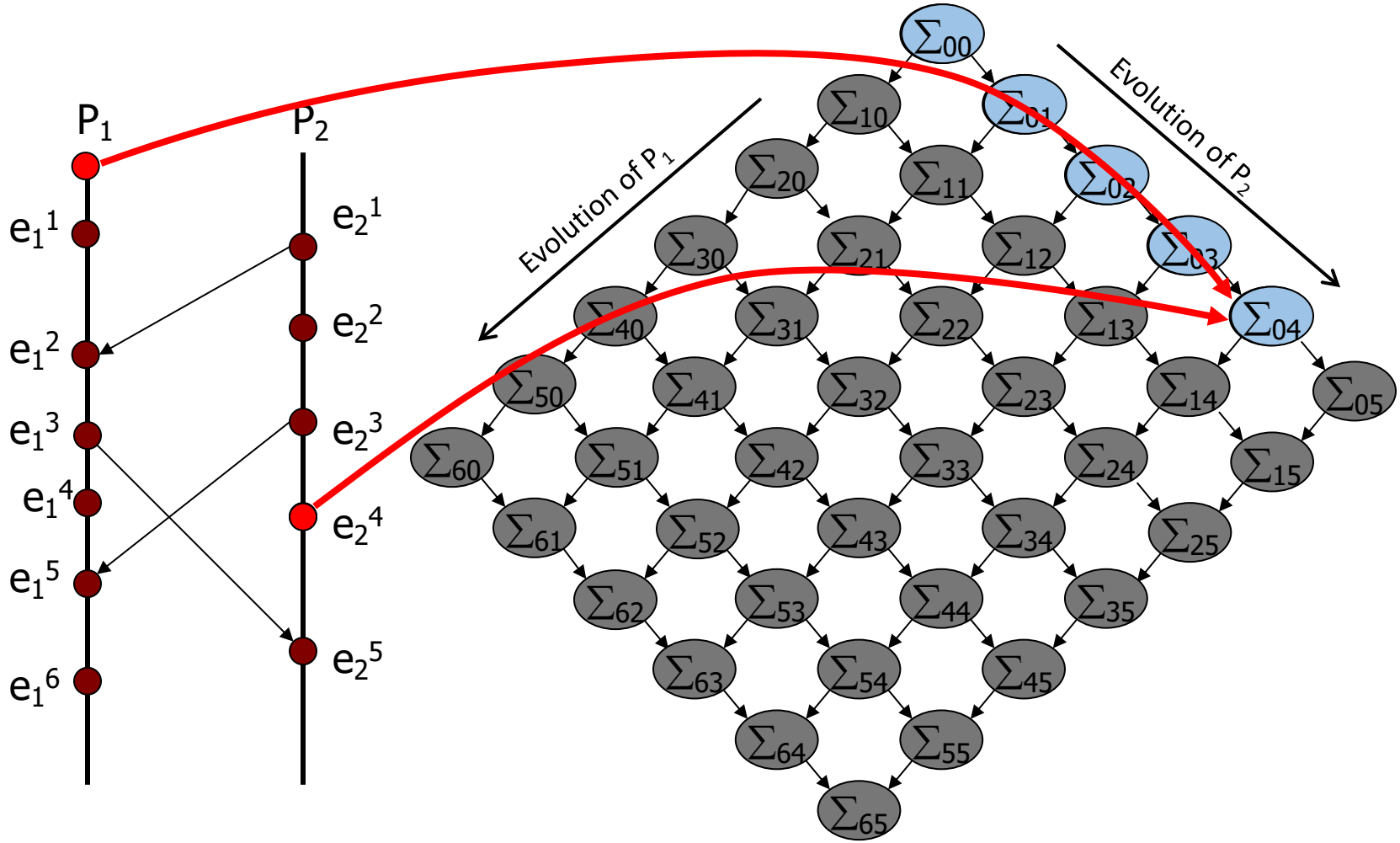
State explosion in concurrent programs



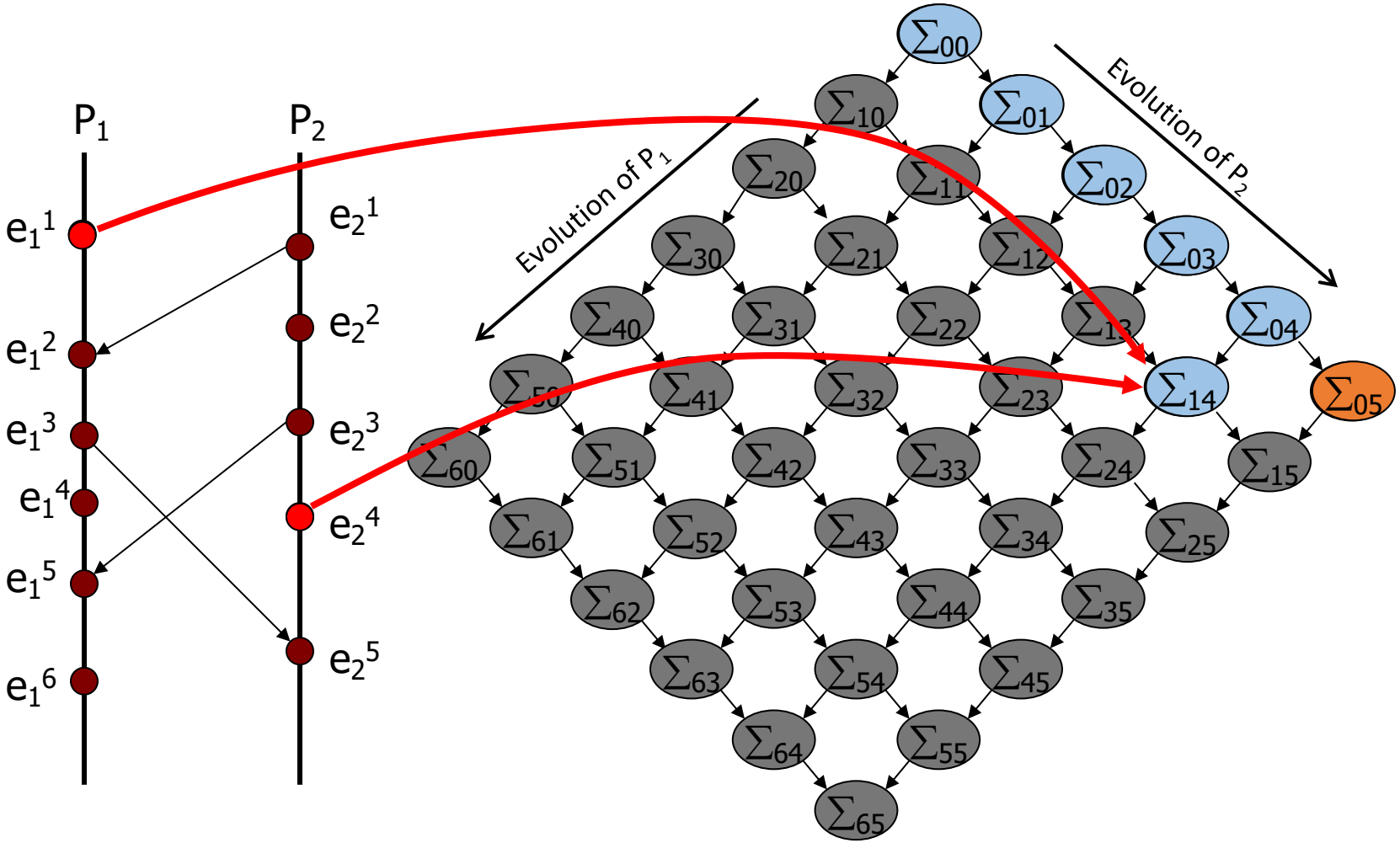
State explosion in concurrent programs



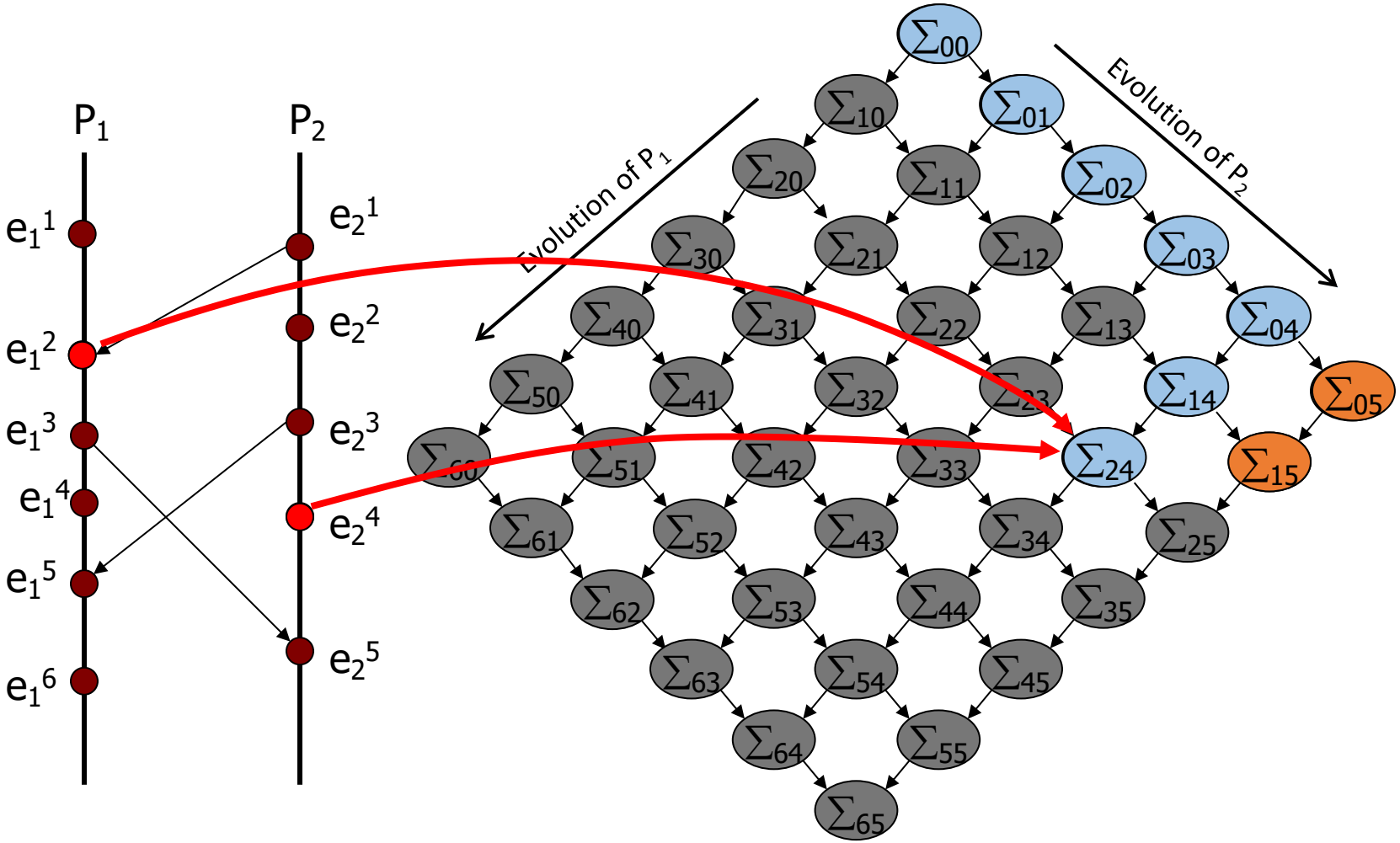
State explosion in concurrent programs



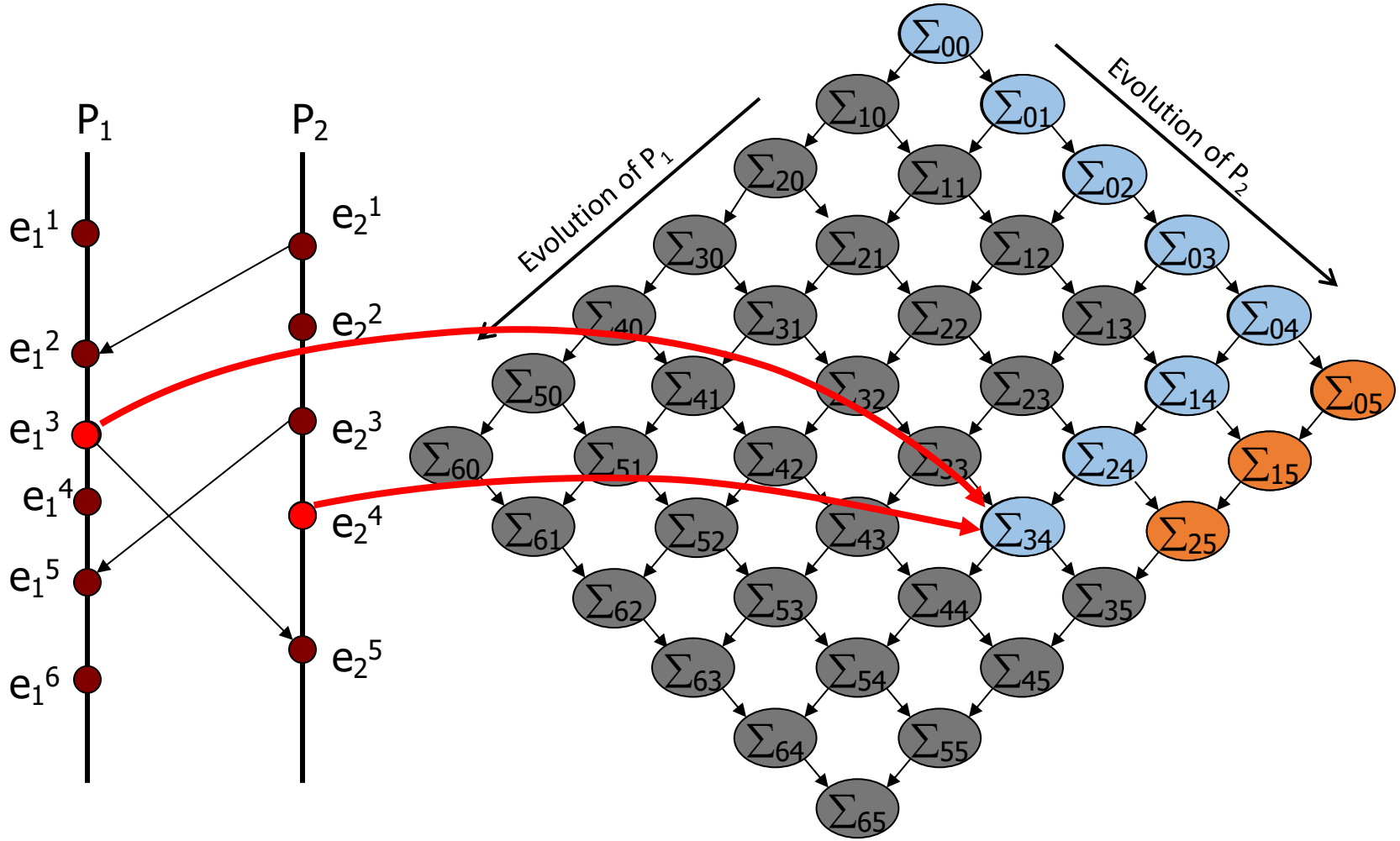
State explosion in concurrent programs



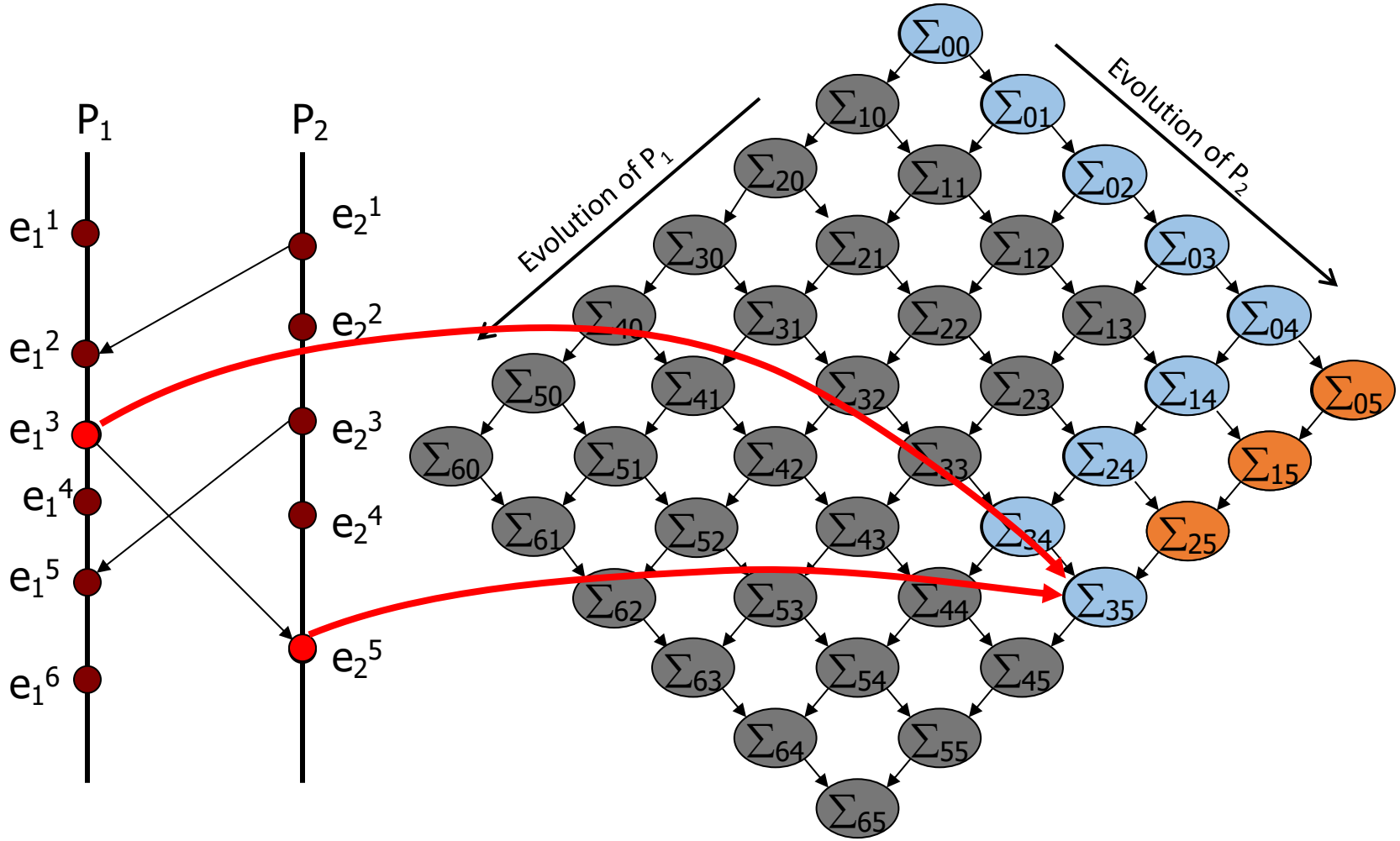
State explosion in concurrent programs



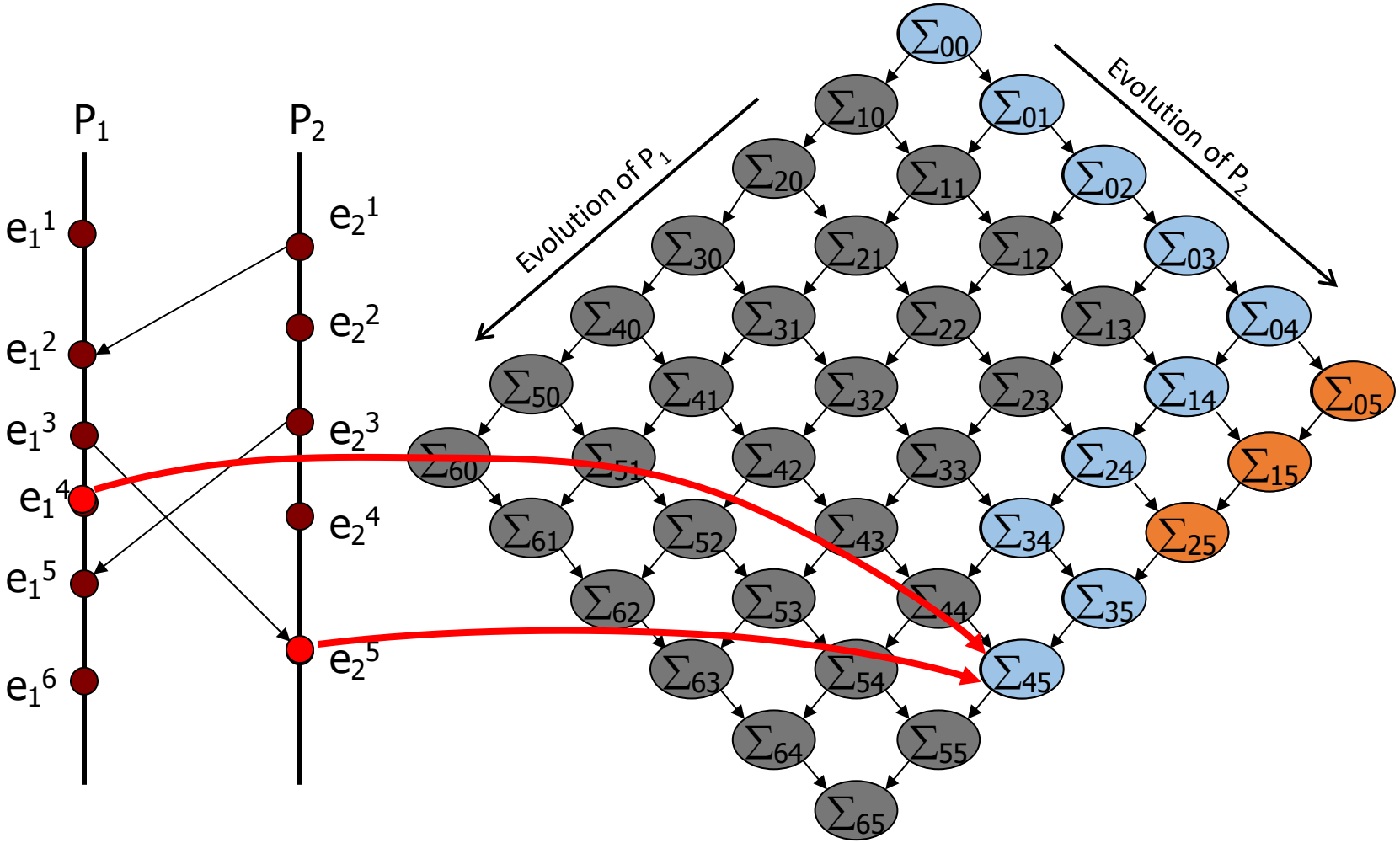
State explosion in concurrent programs



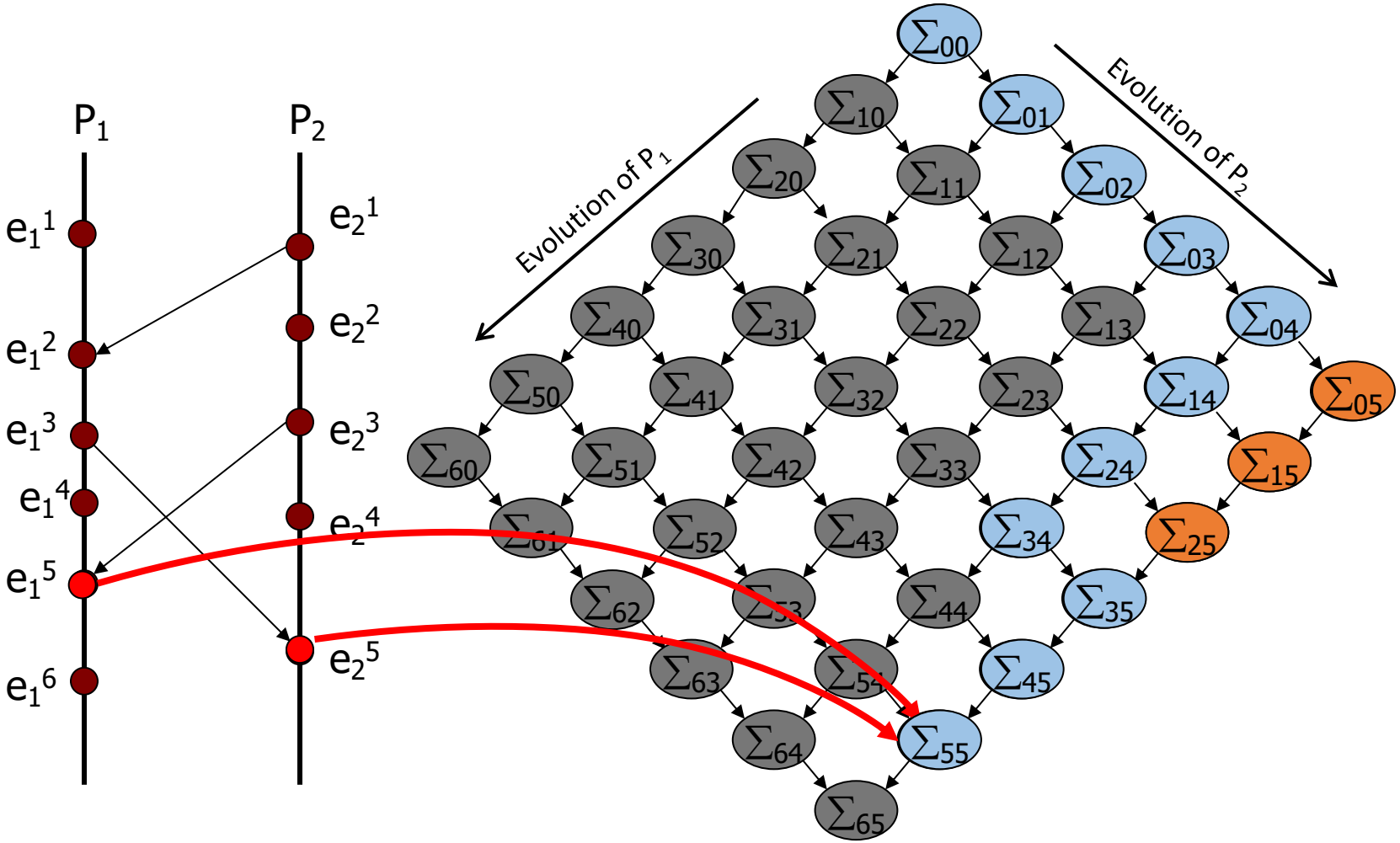
State explosion in concurrent programs



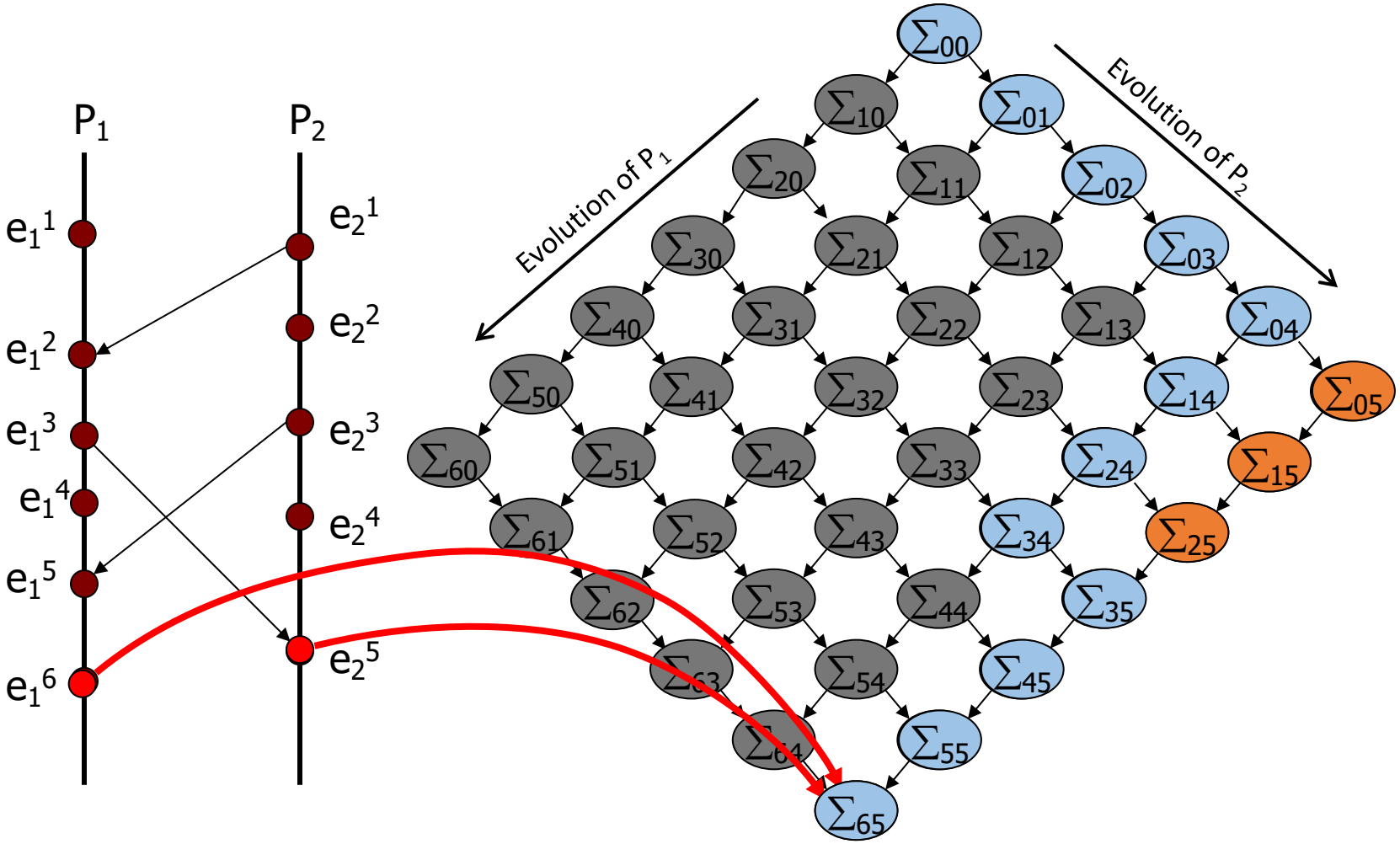
State explosion in concurrent programs



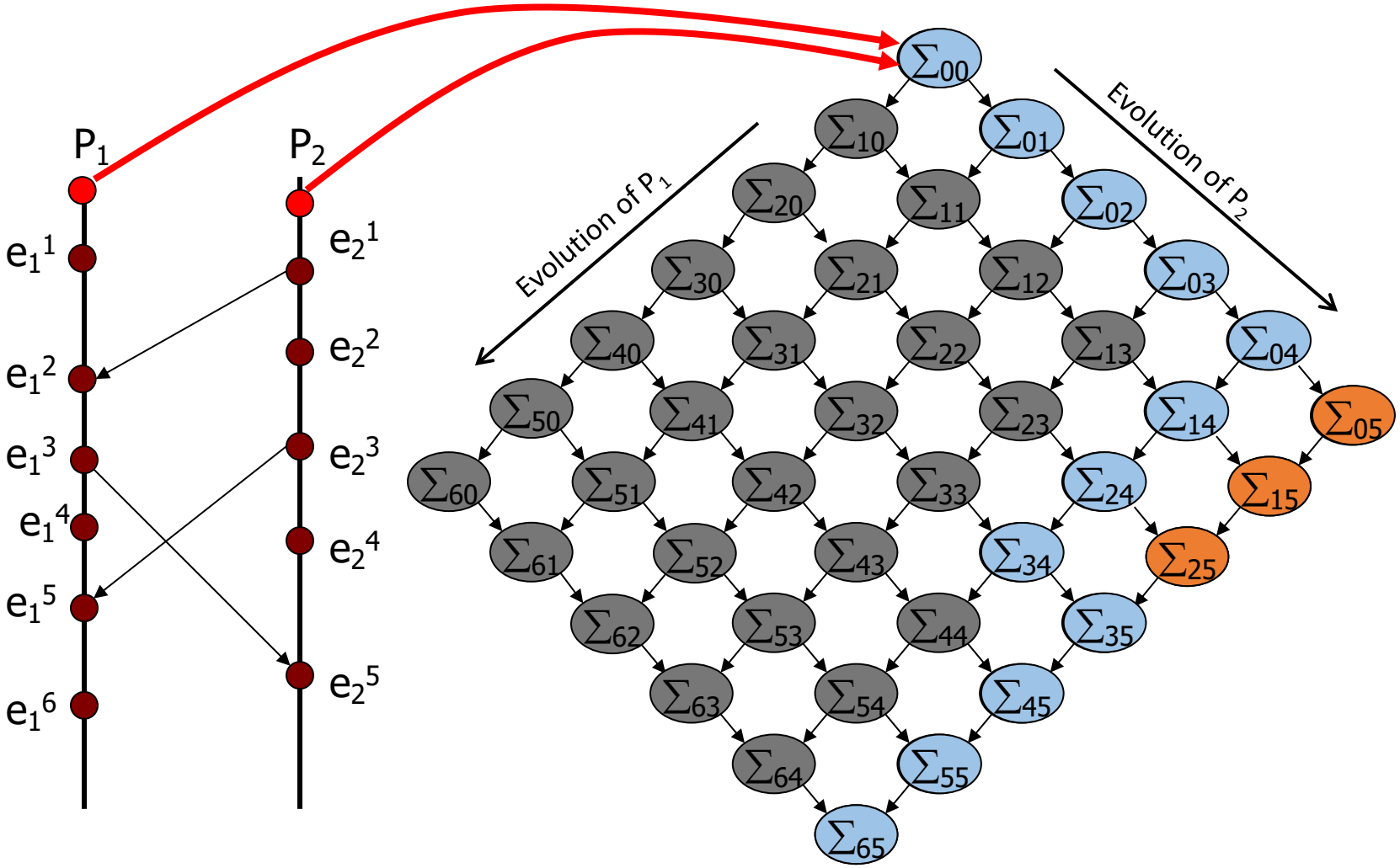
State explosion in concurrent programs



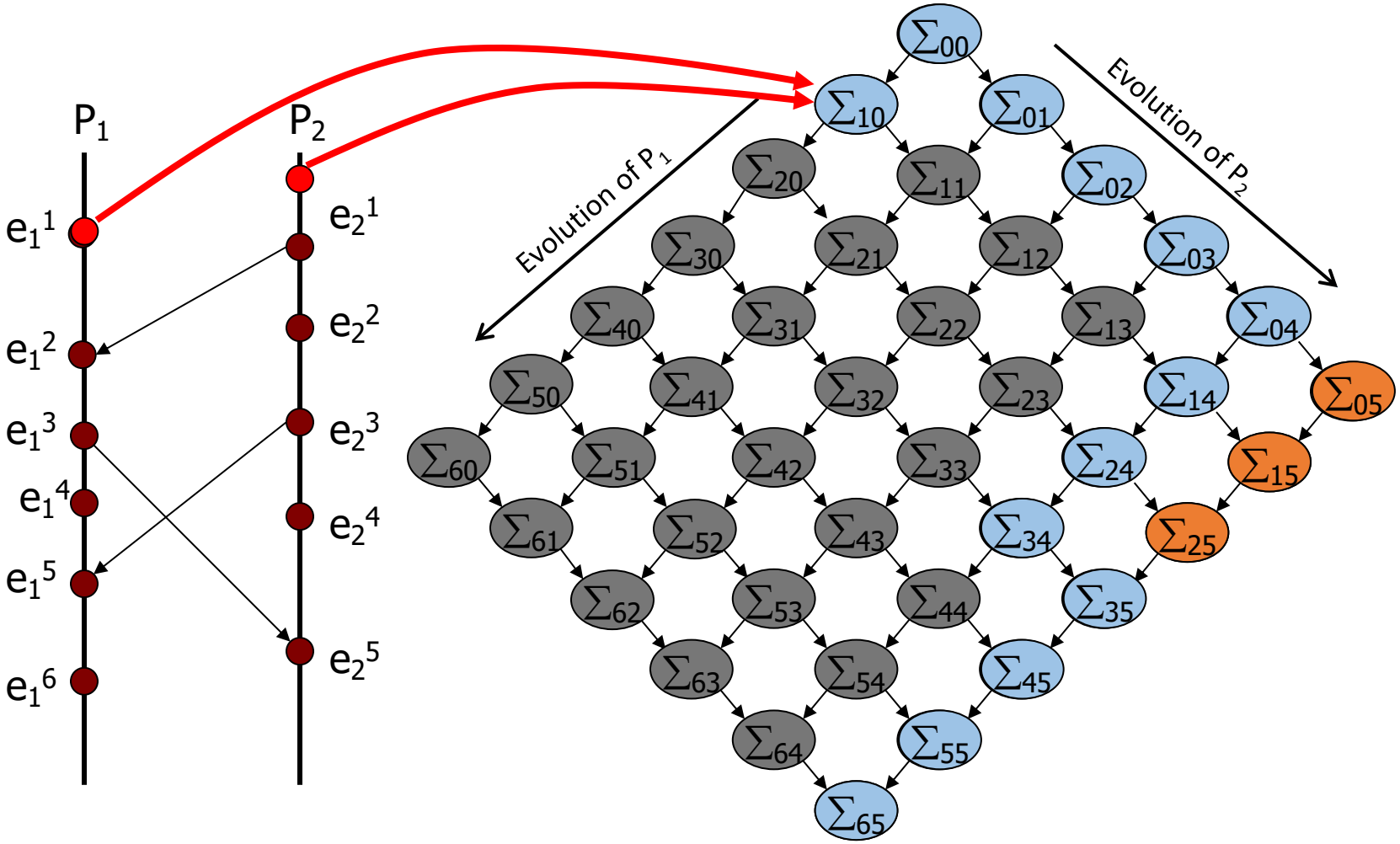
State explosion in concurrent programs



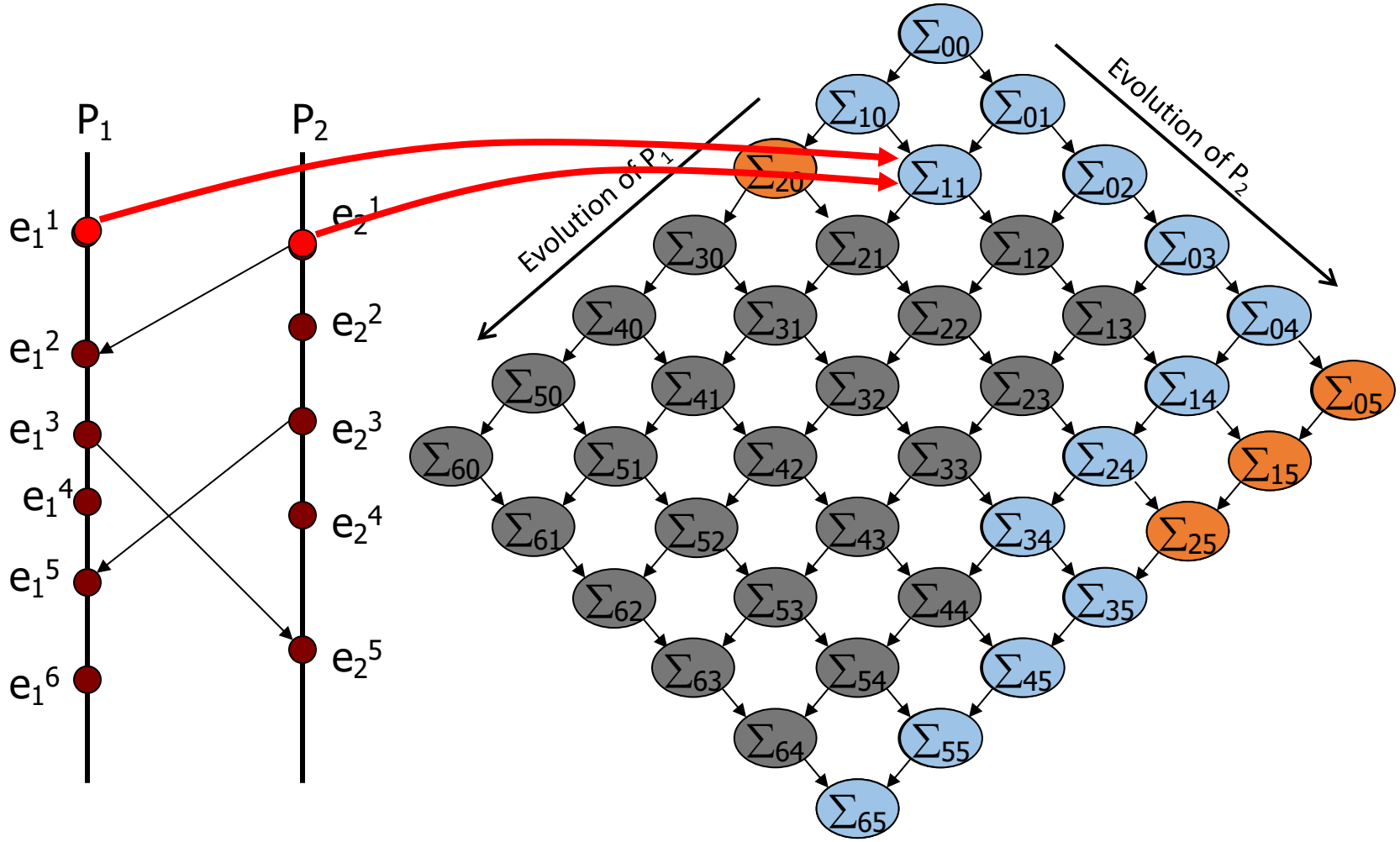
State explosion in concurrent programs



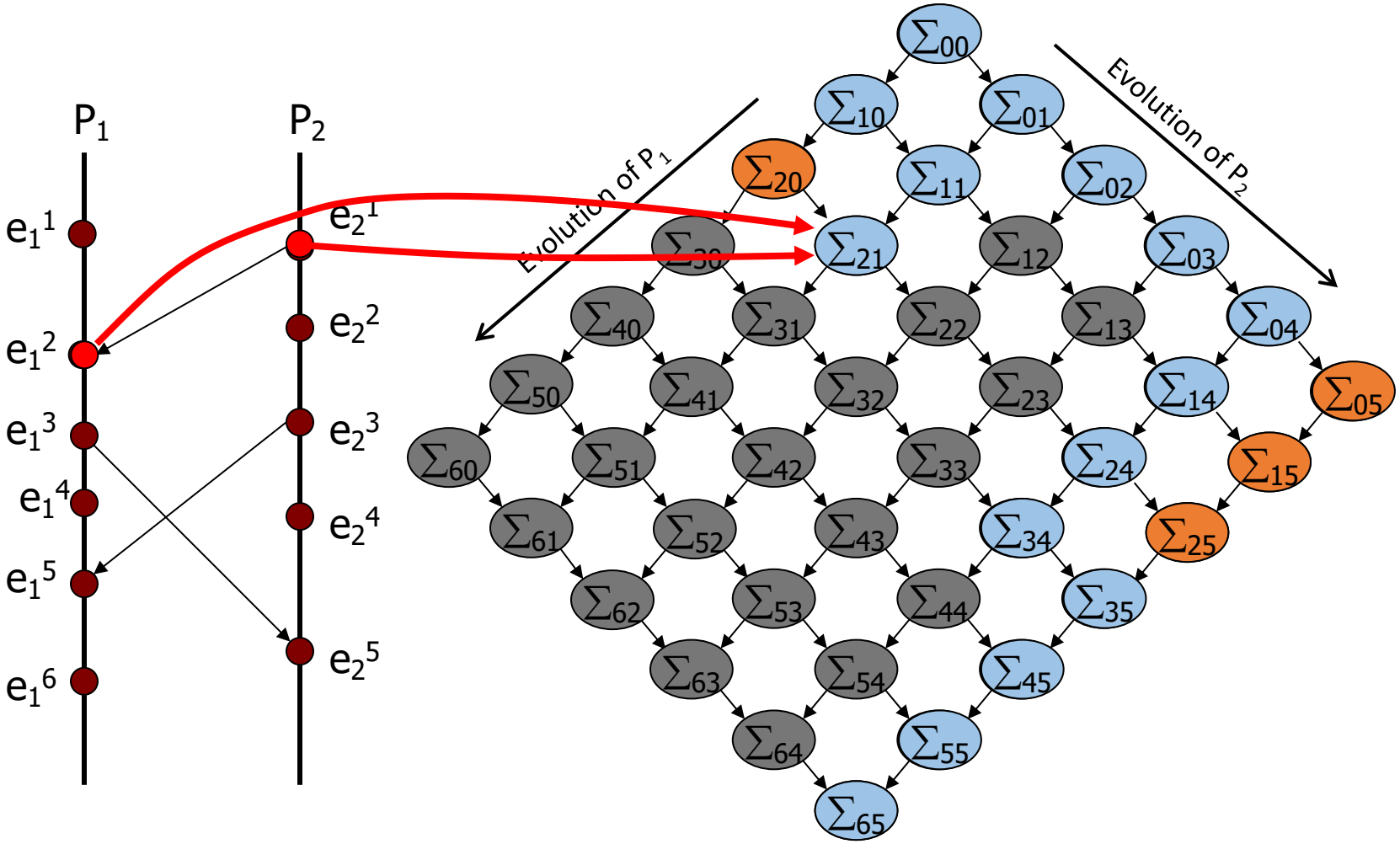
State explosion in concurrent programs



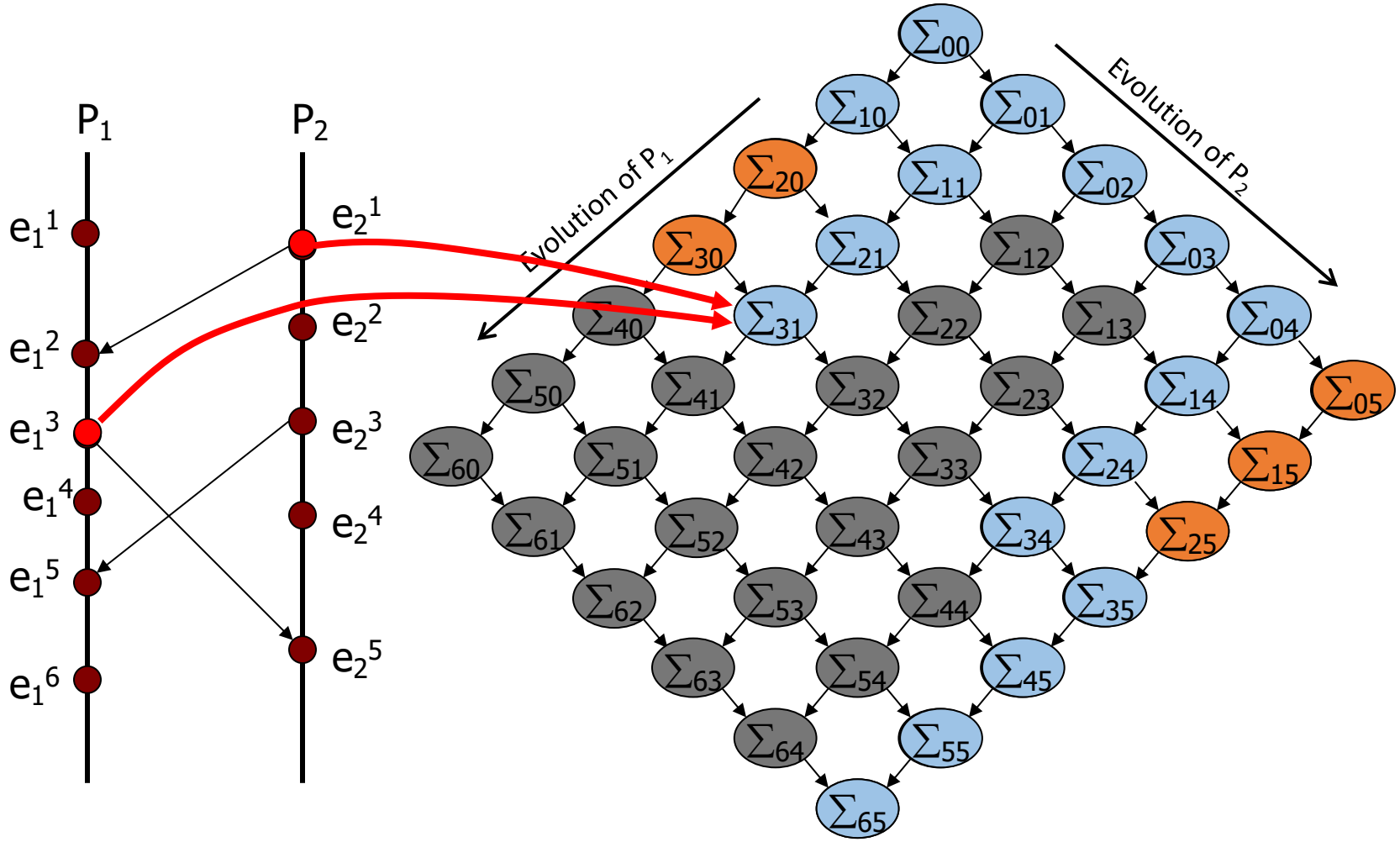
State explosion in concurrent programs



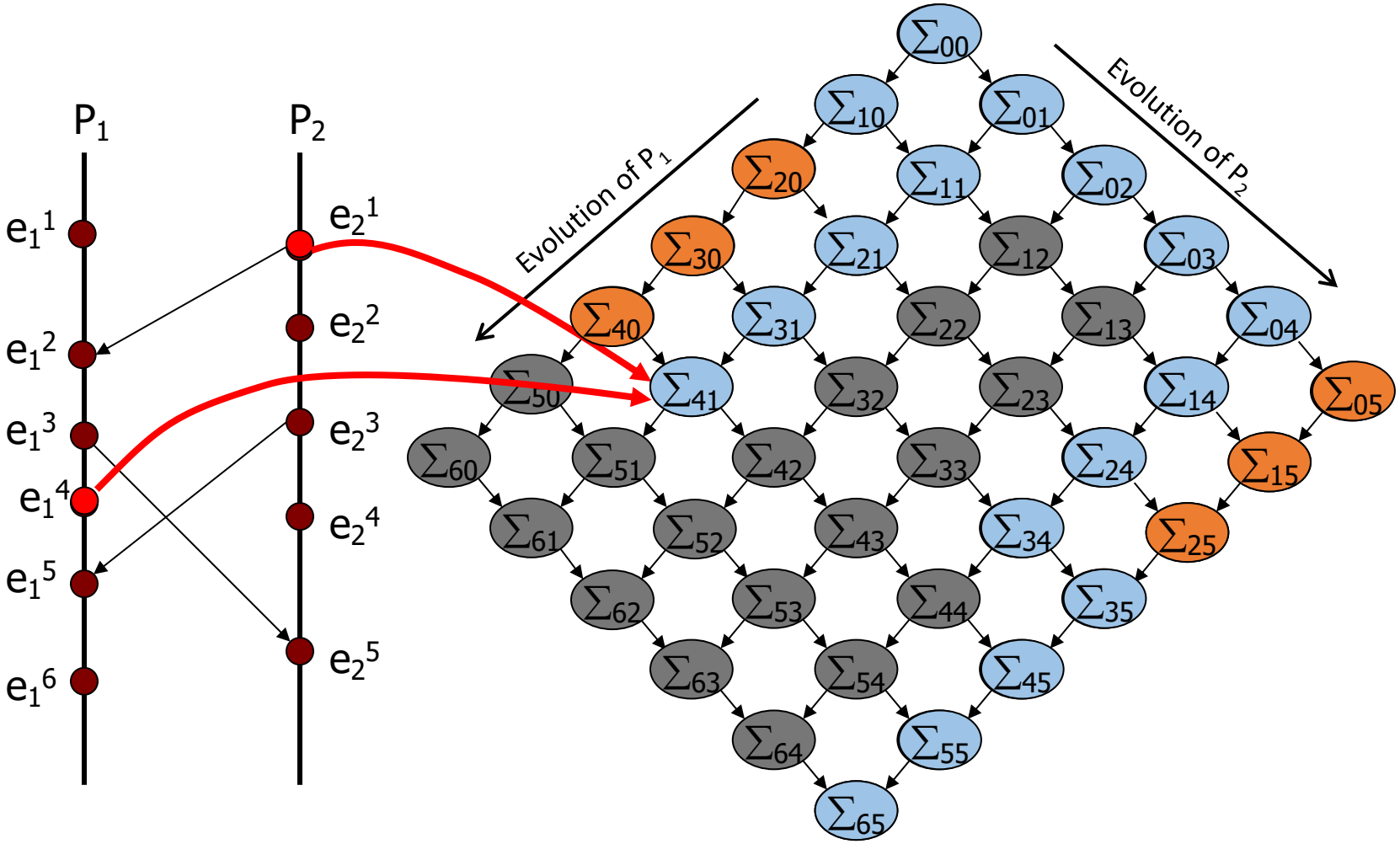
State explosion in concurrent programs



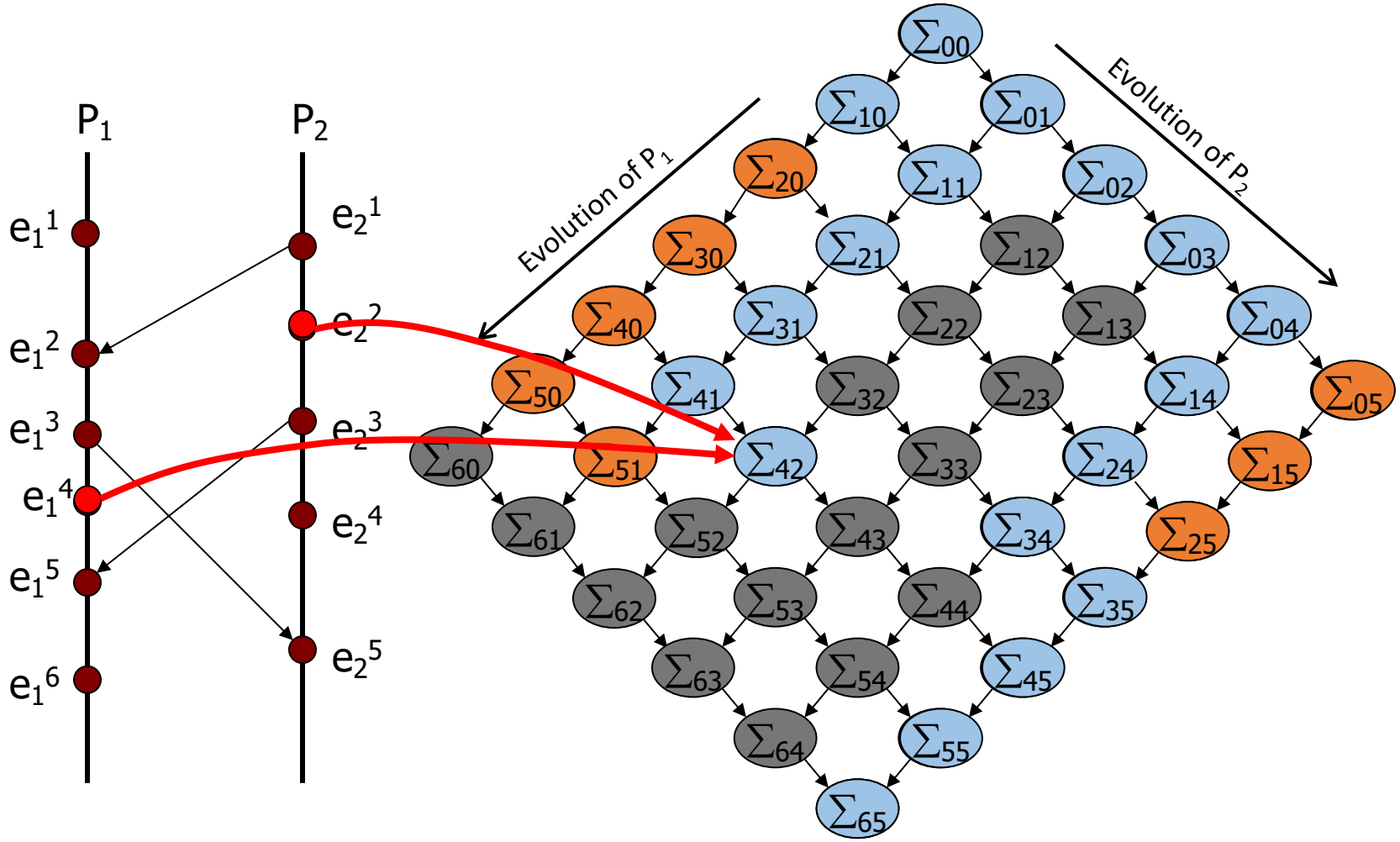
State explosion in concurrent programs



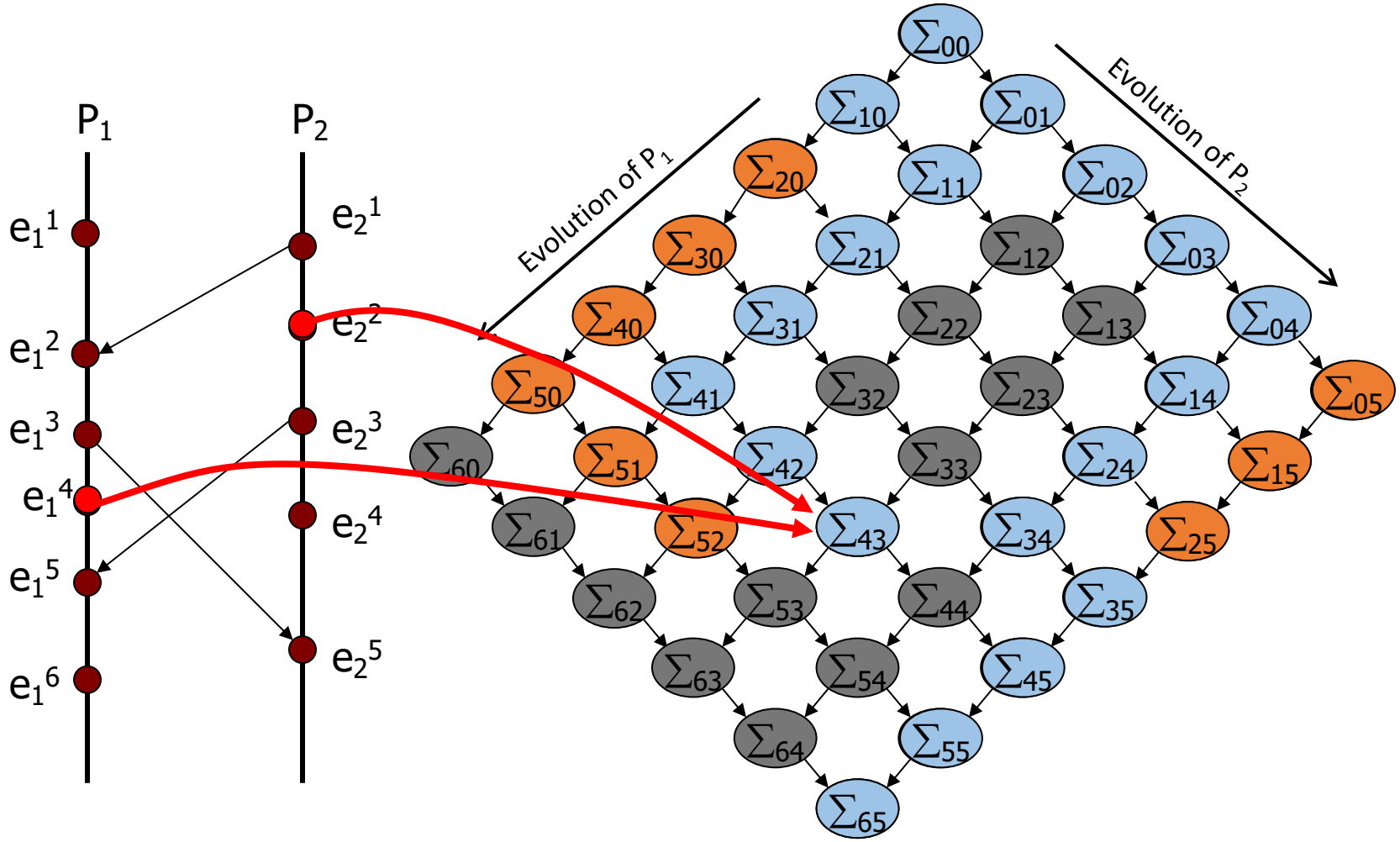
State explosion in concurrent programs



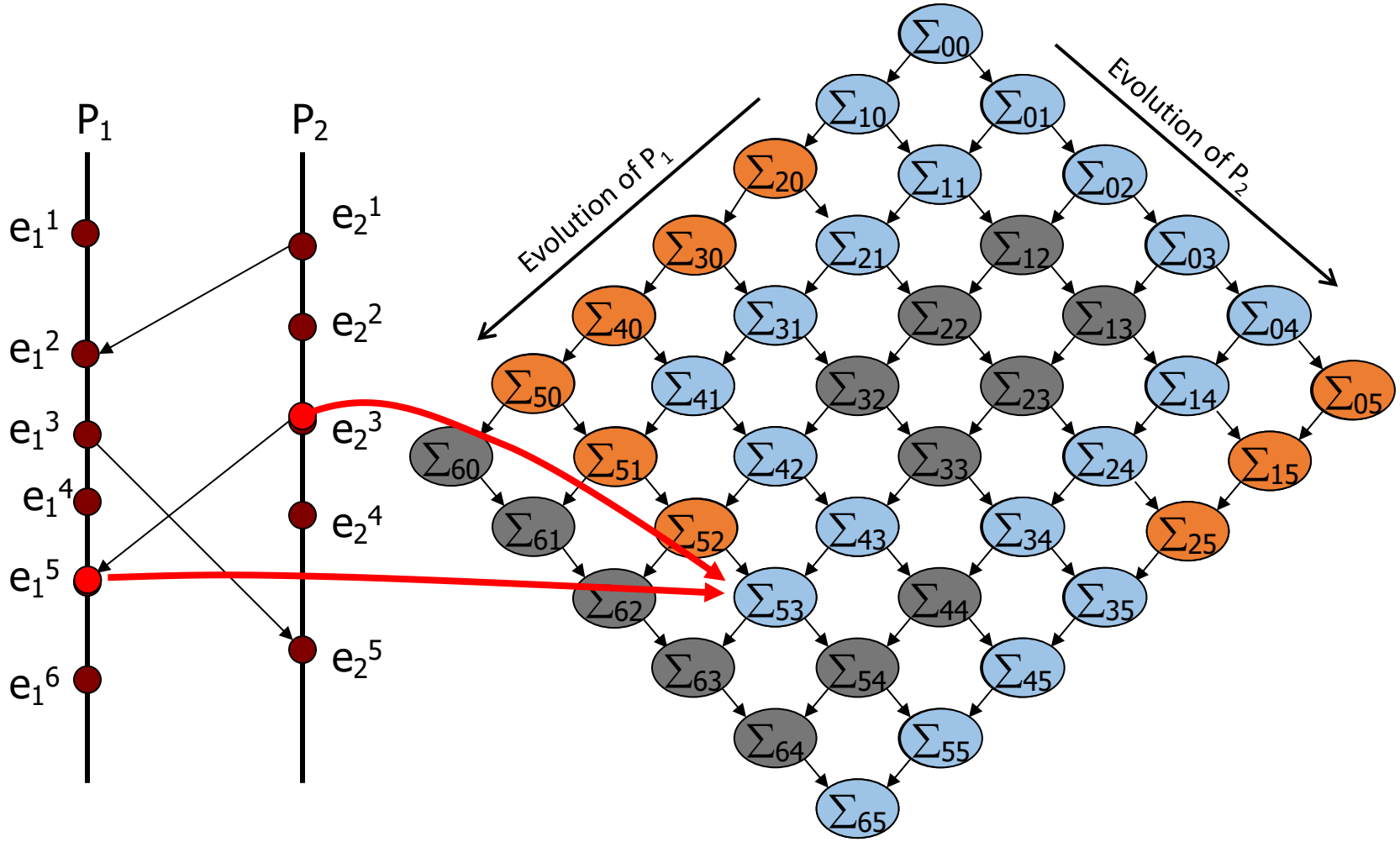
State explosion in concurrent programs



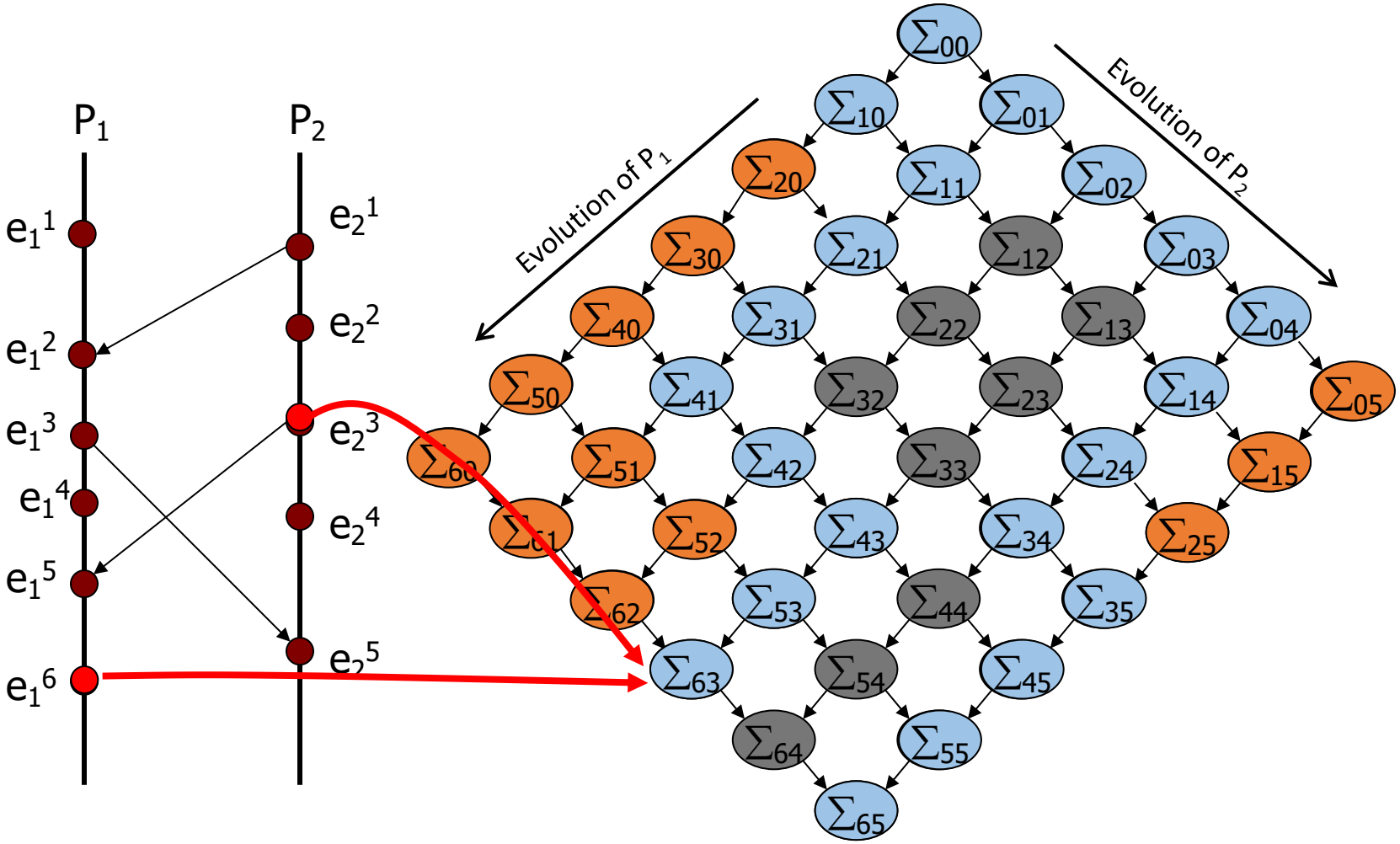
State explosion in concurrent programs



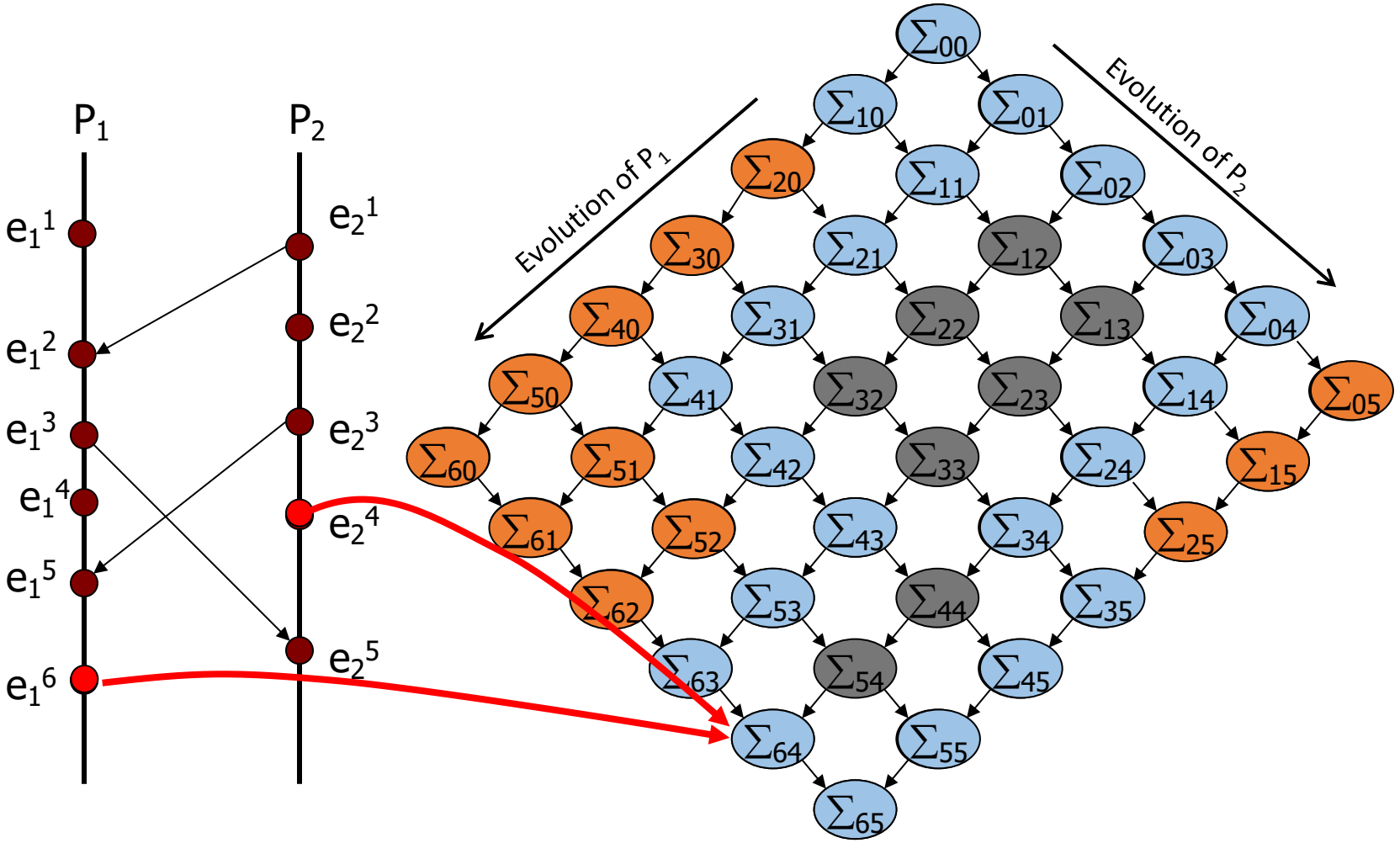
State explosion in concurrent programs



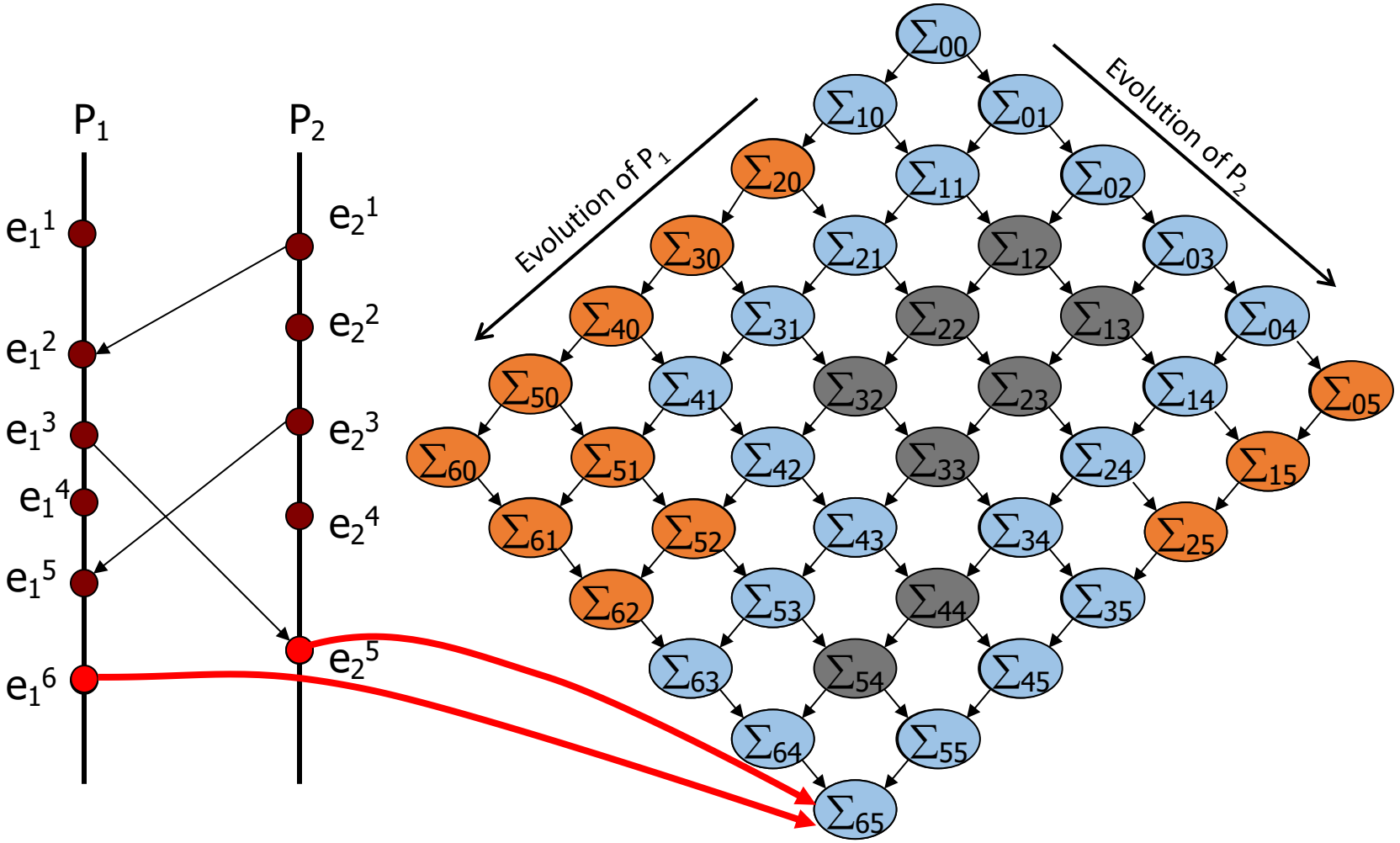
State explosion in concurrent programs



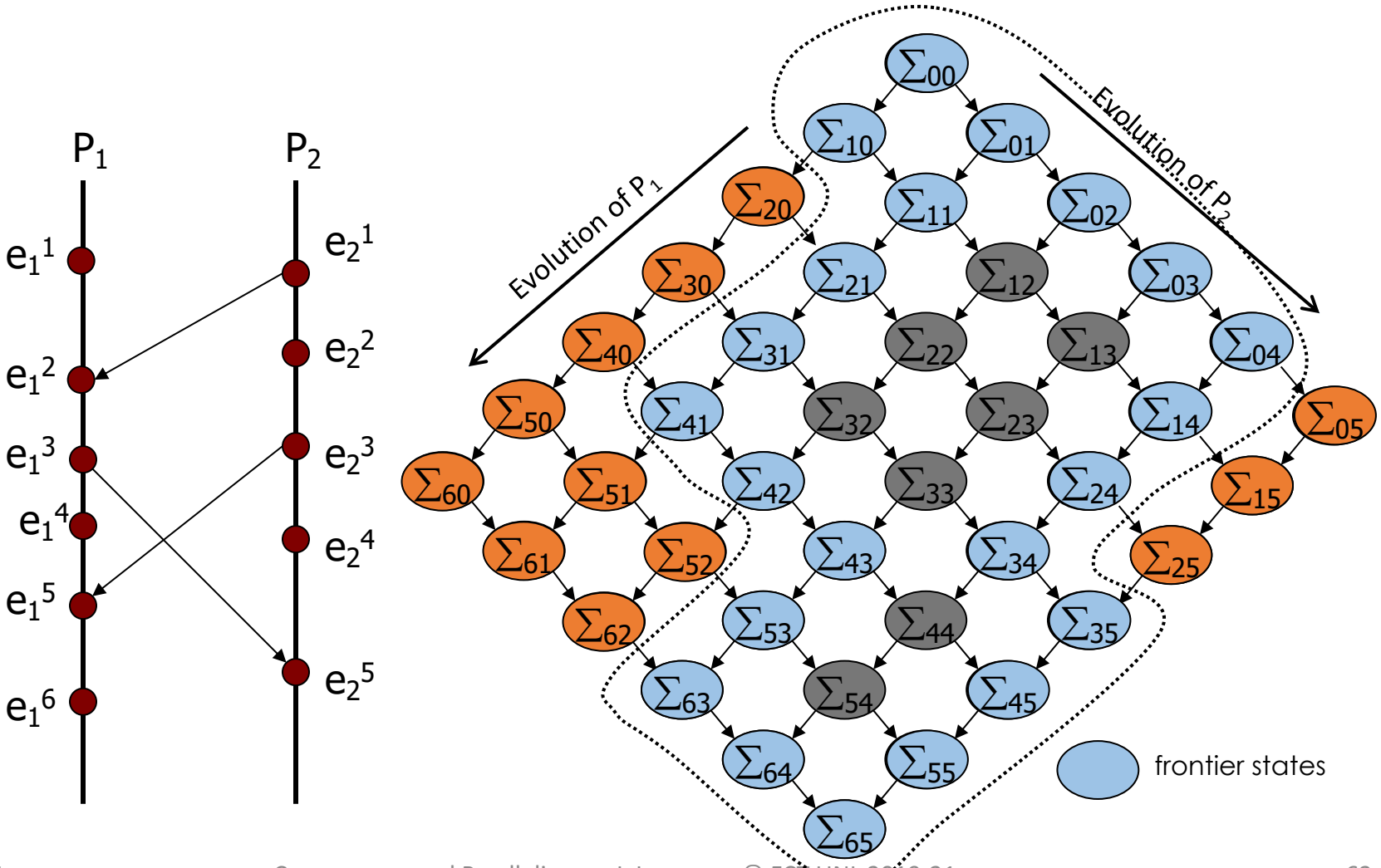
State explosion in concurrent programs



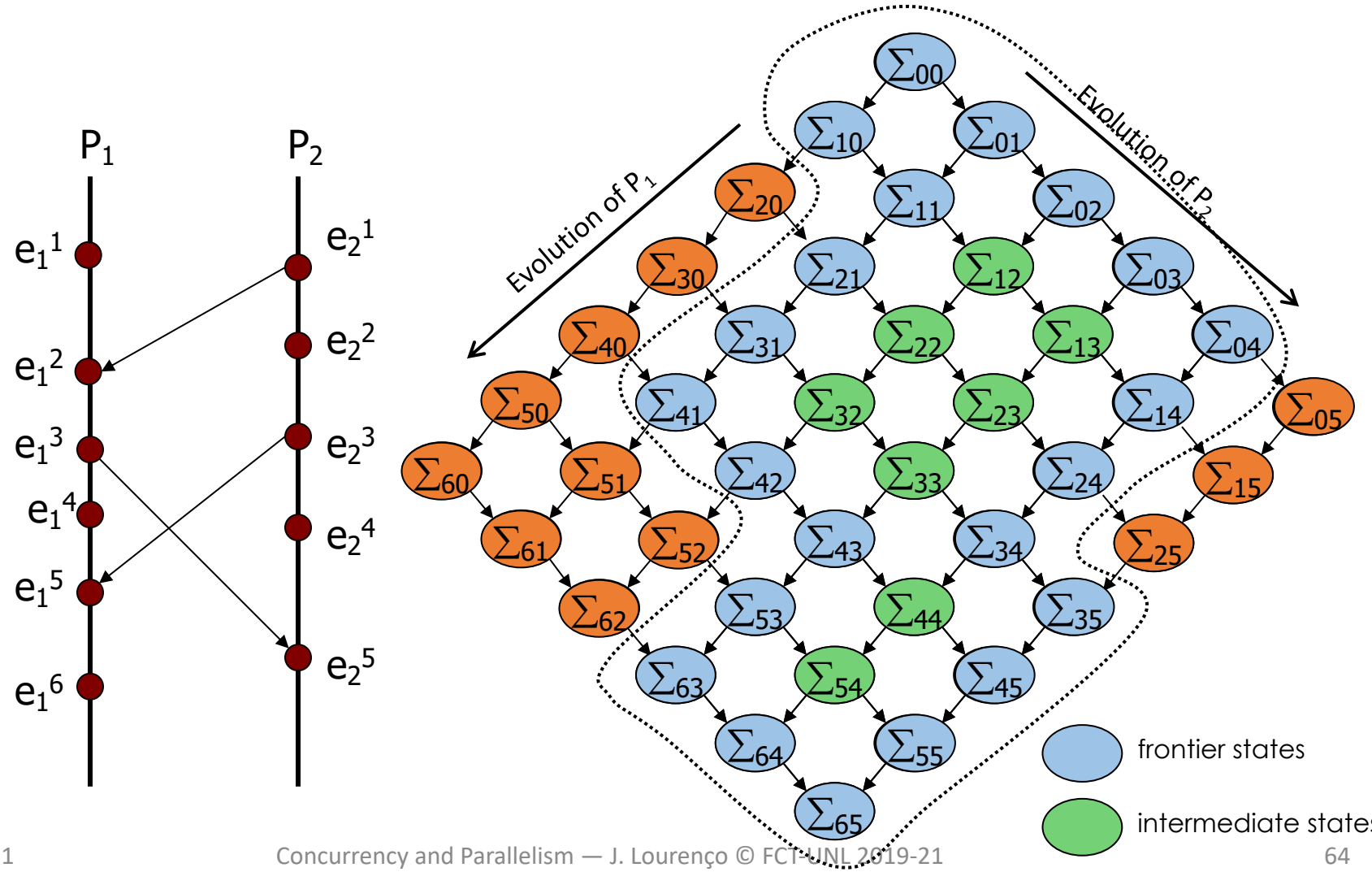
State explosion in concurrent programs



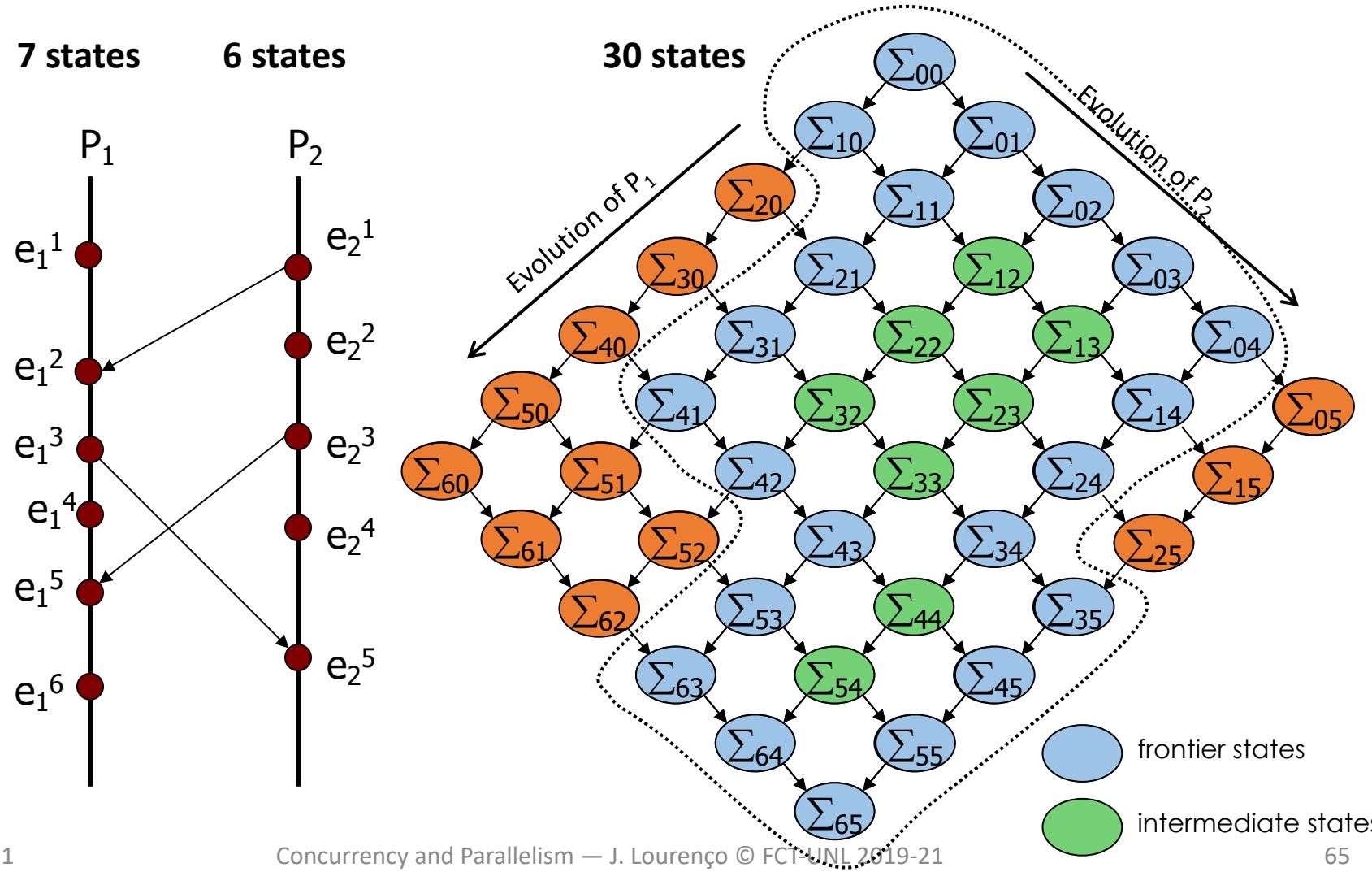
State explosion in concurrent programs



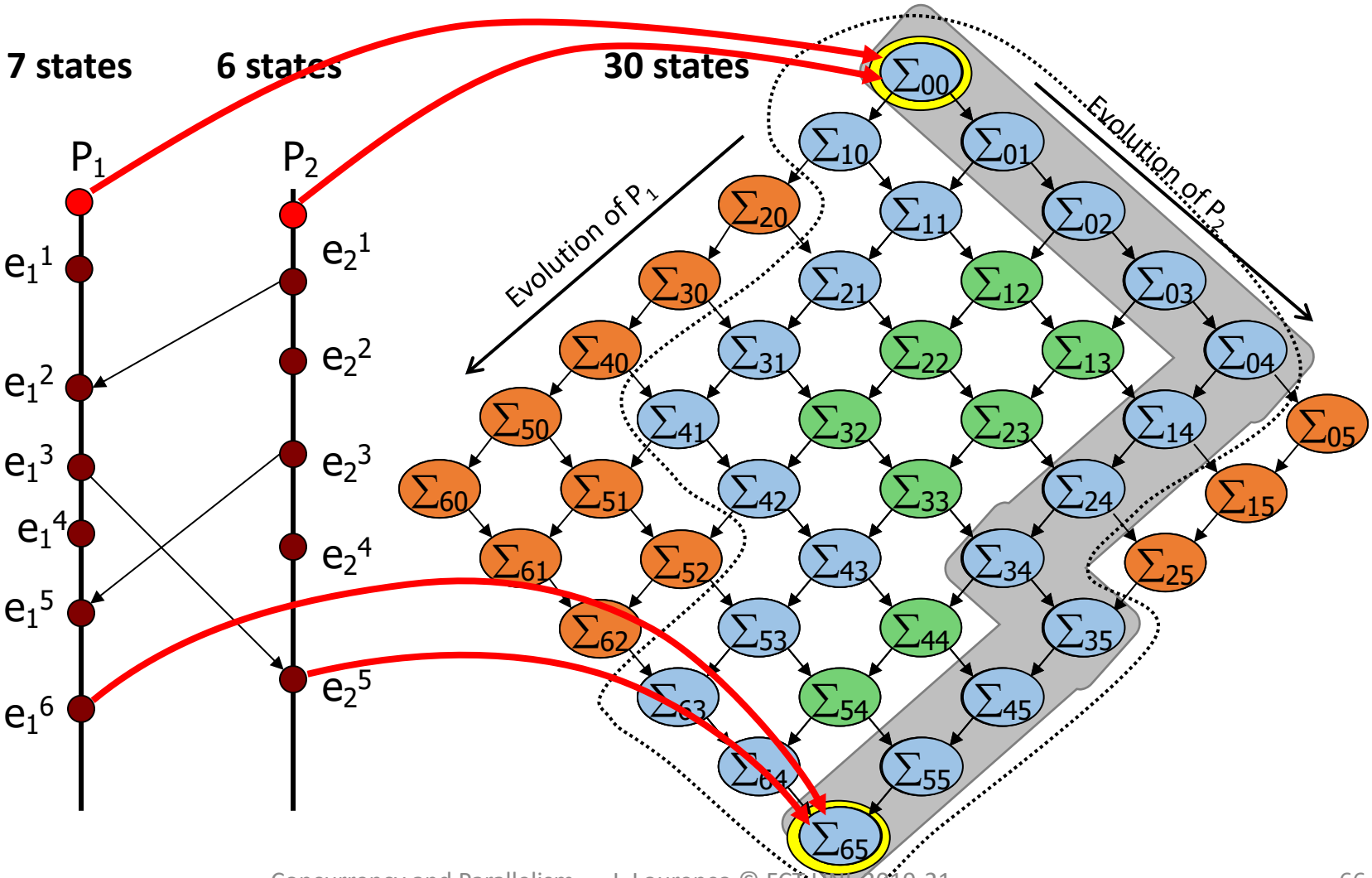
State explosion in concurrent programs



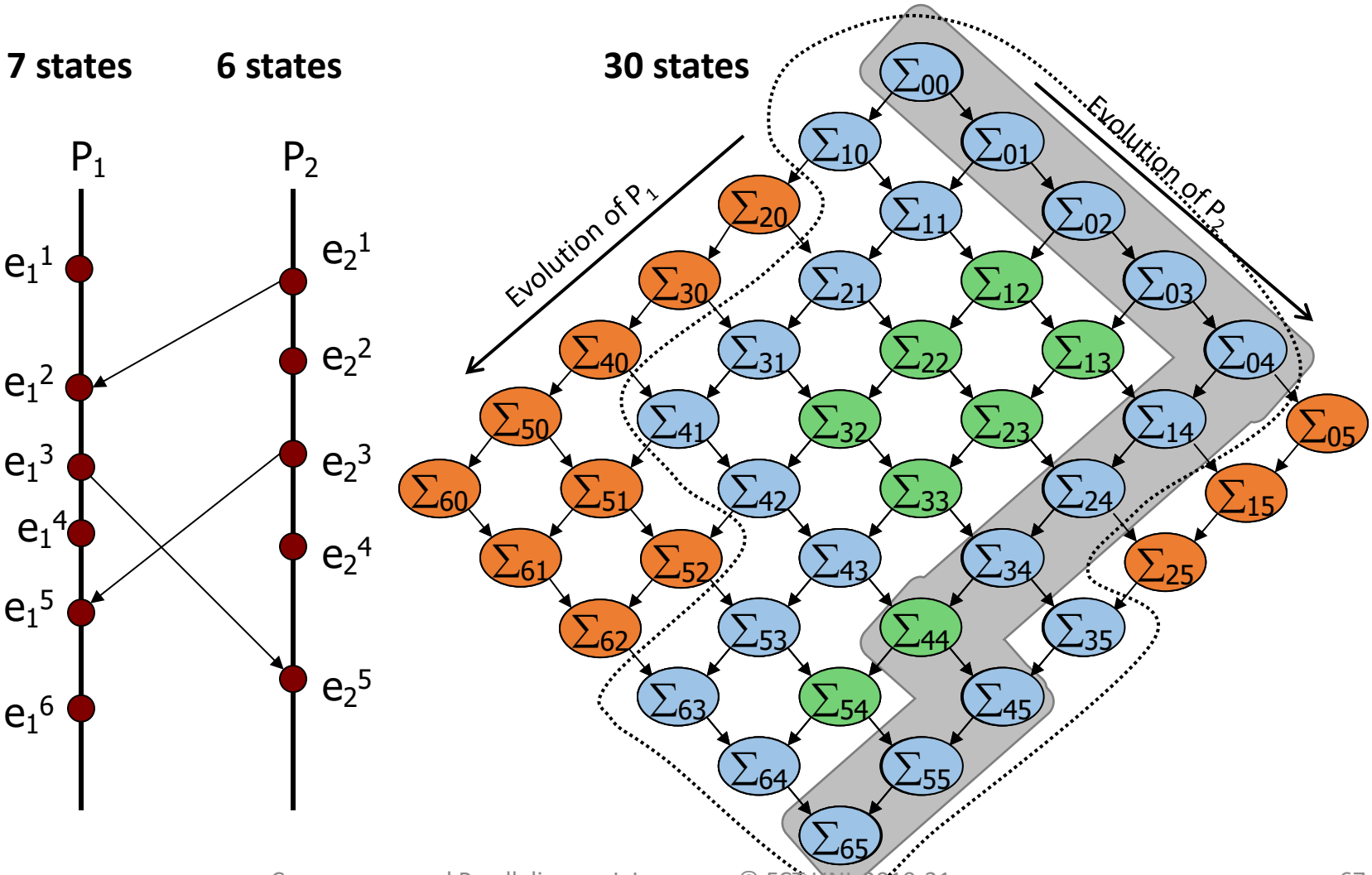
State explosion in concurrent programs



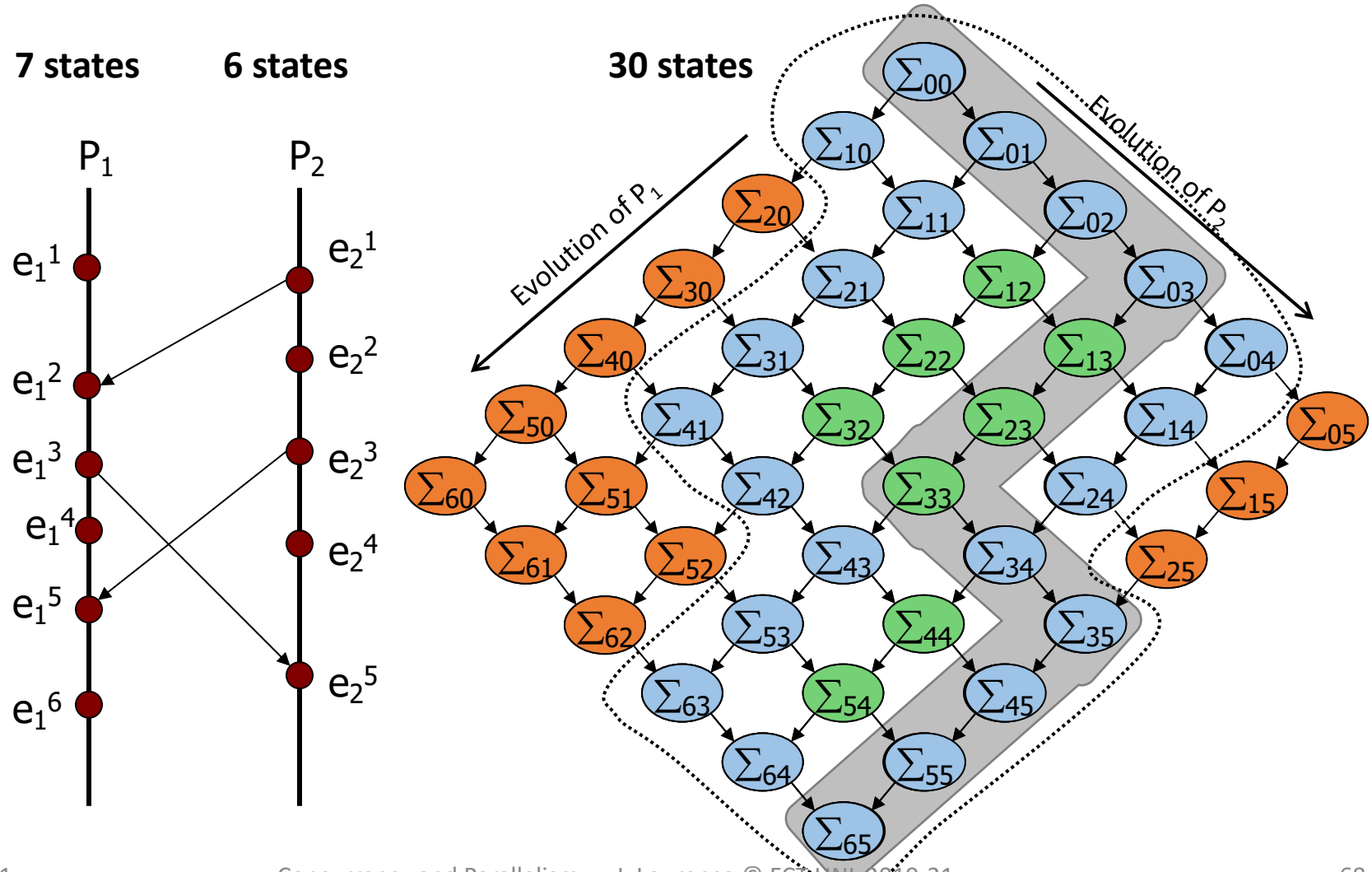
Consistent run: valid path Σ_{00} to Σ_{65}



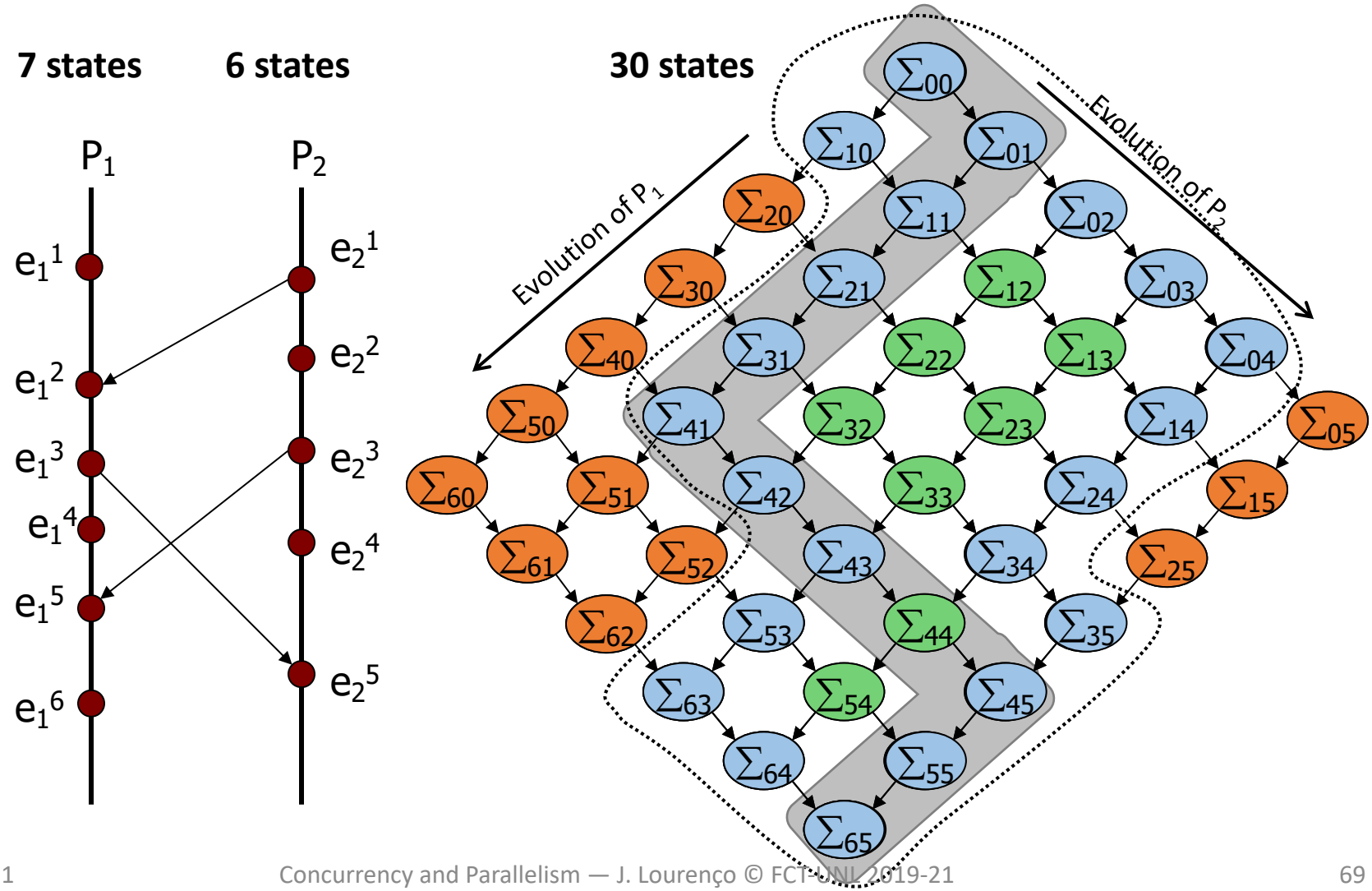
Consistent run



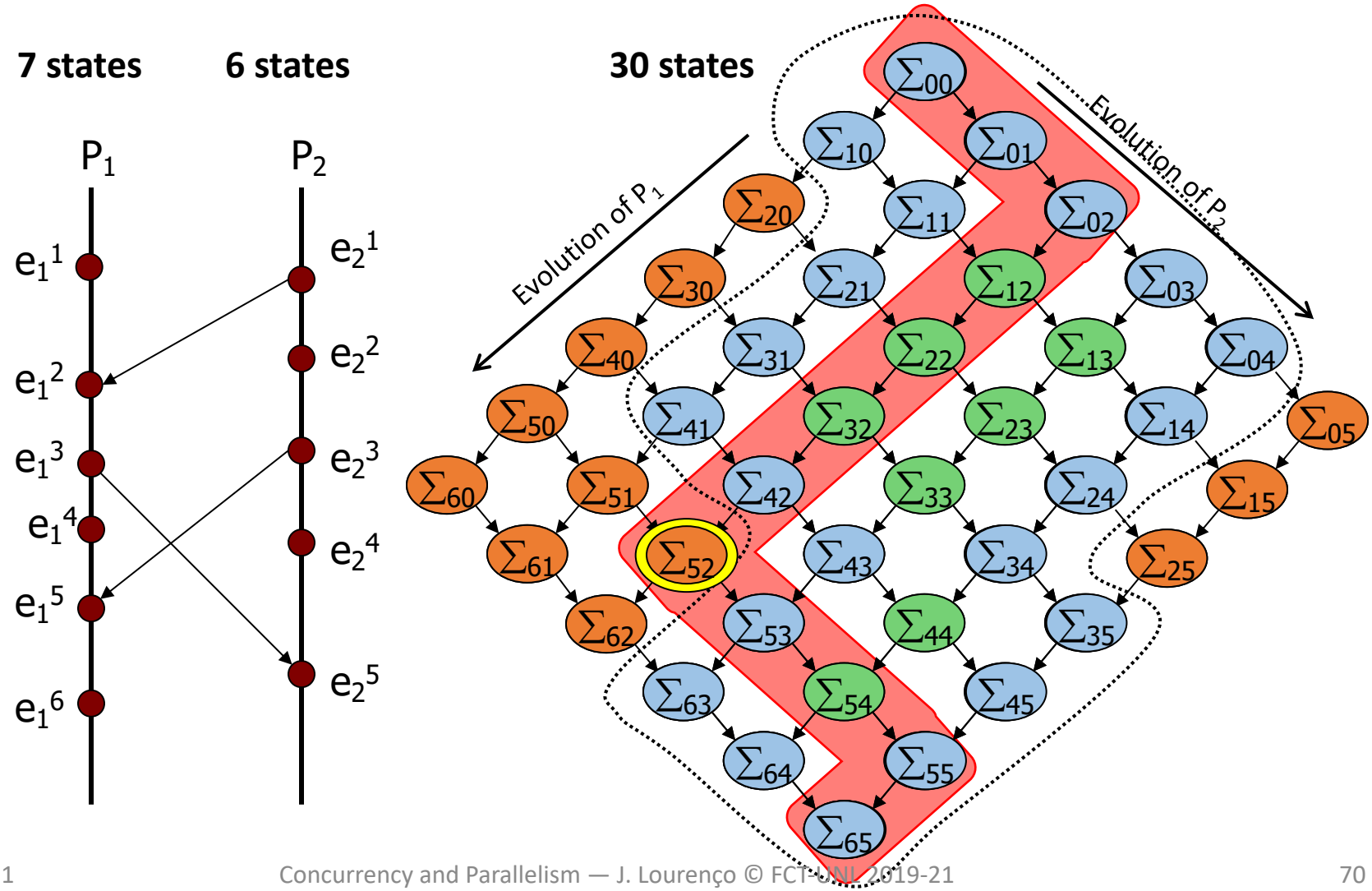
Consistent run



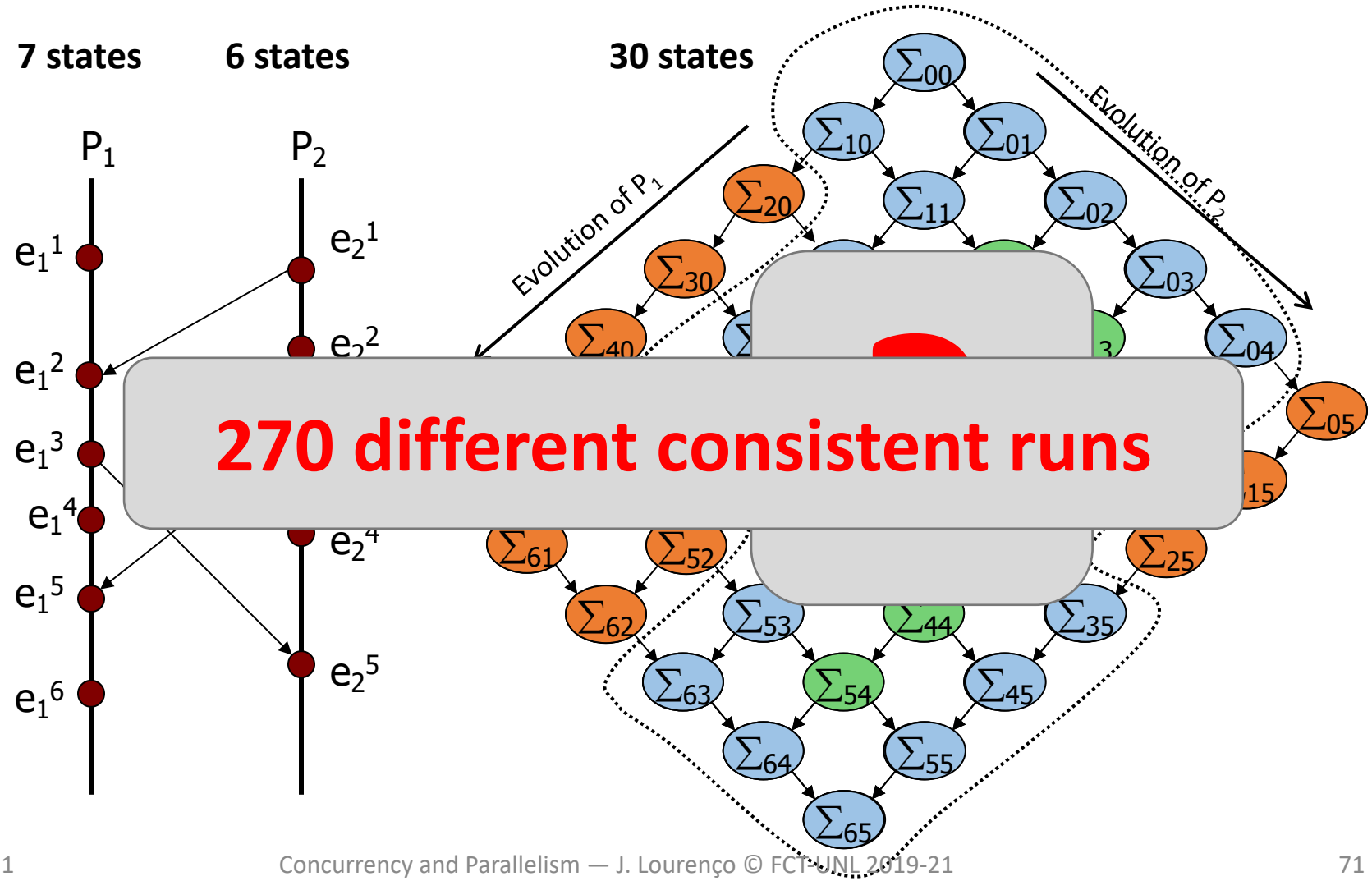
Consistent run



Inconsistent run – program error



Consistent runs – How many?



The END