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

Concurrency Errors (1)

lecture 21 (2020-05-13)

Master in Computer Science and Engineering

— Concurrency and Parallelism / 2019-20 —

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

Agenda

- Assigning Semantics to Concurrent Programs
- Concurrency Errors
  - Detection of data races
  - Detection of high-level data races and stale value errors
  - Detection of deadlocks
- Reading list:
  - TBD

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

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

- 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        a = Y
Y = 2        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
```

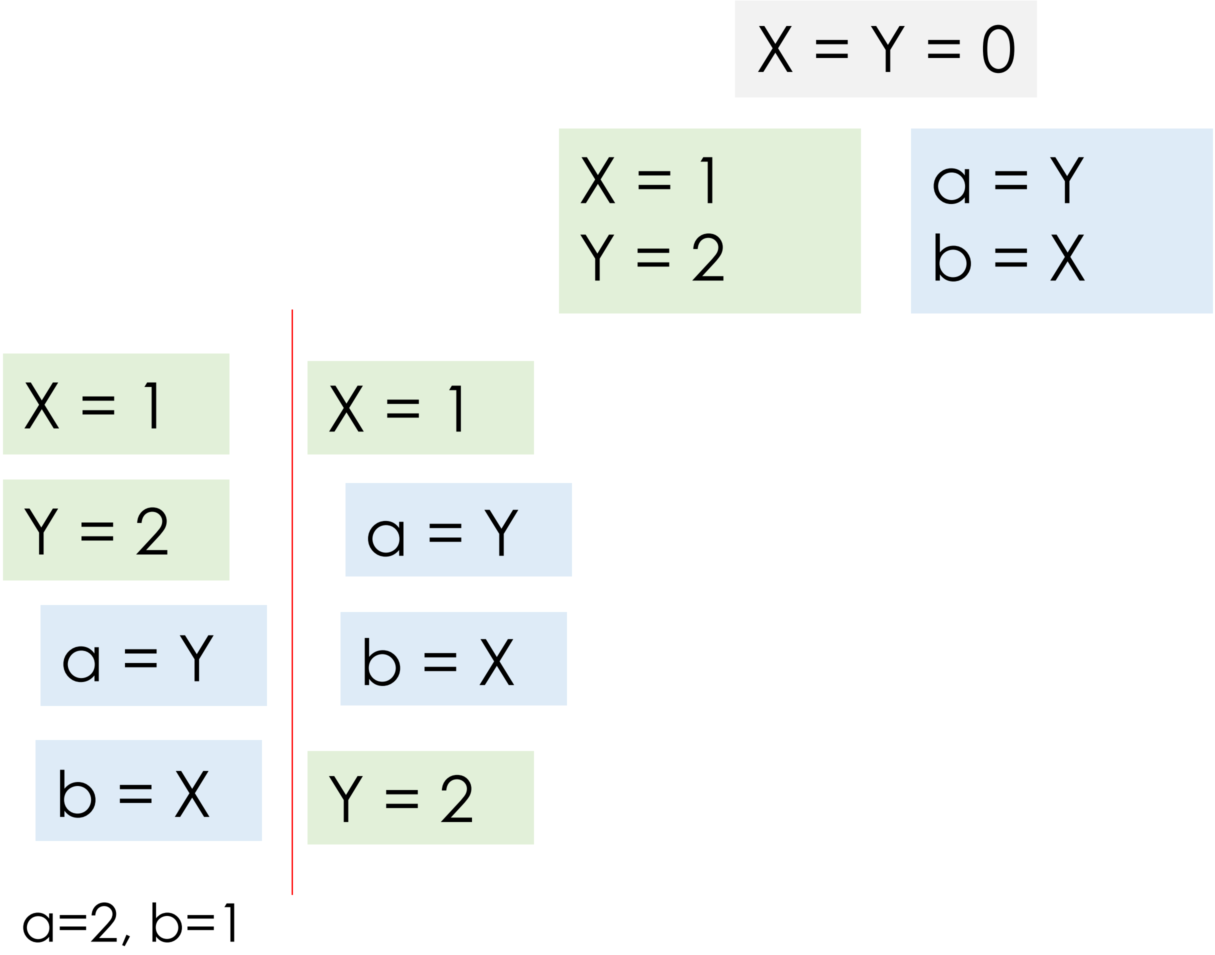
```
a = Y
b = X
```

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

a=2, b=1

Assigning Semantics to Concurrent Programs

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



X = Y = 0

```
X = 1        a = Y
Y = 2        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

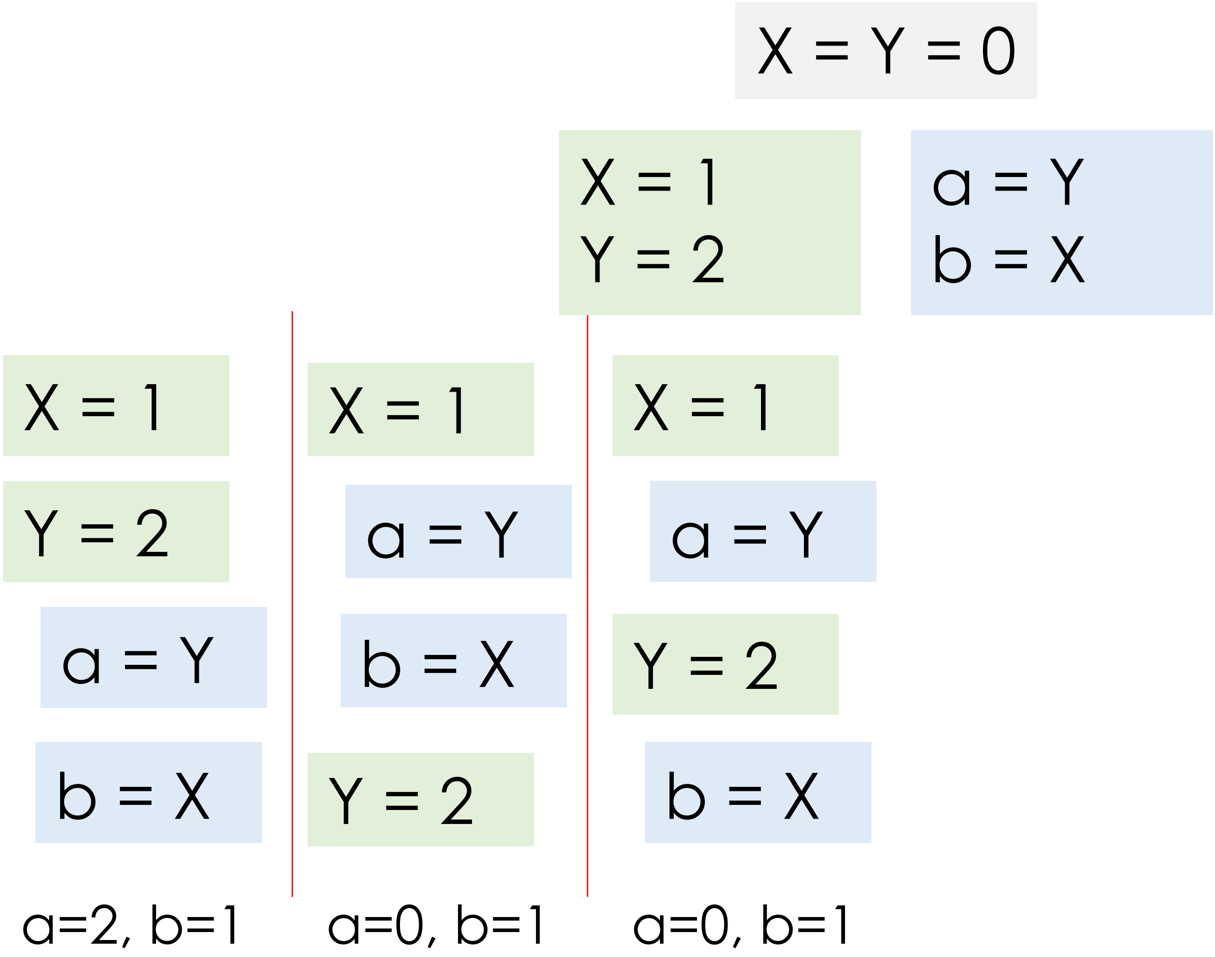
X, Y => Global Vars
a, b => Local Vars

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



```
X = Y = 0
```

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

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

a=2, b=1

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

a=0, b=1

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

a=0, b=1

Assigning Semantics to Concurrent Programs

X = Y = 0

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

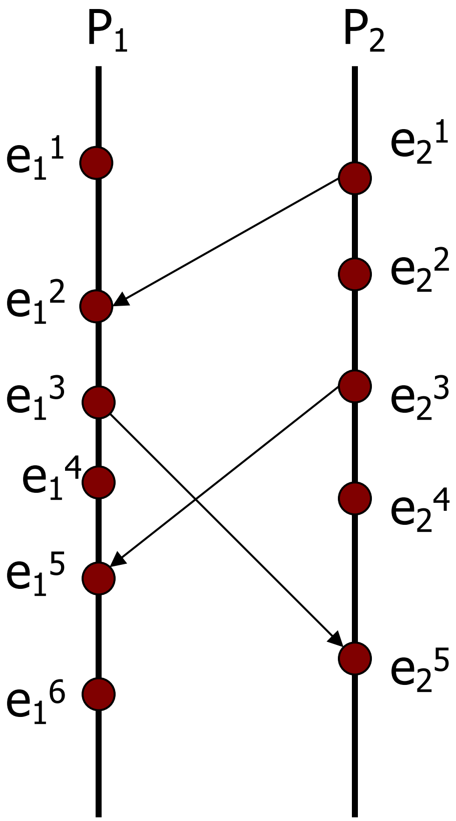
X = 1
Y = 2
	a = Y
	b = X

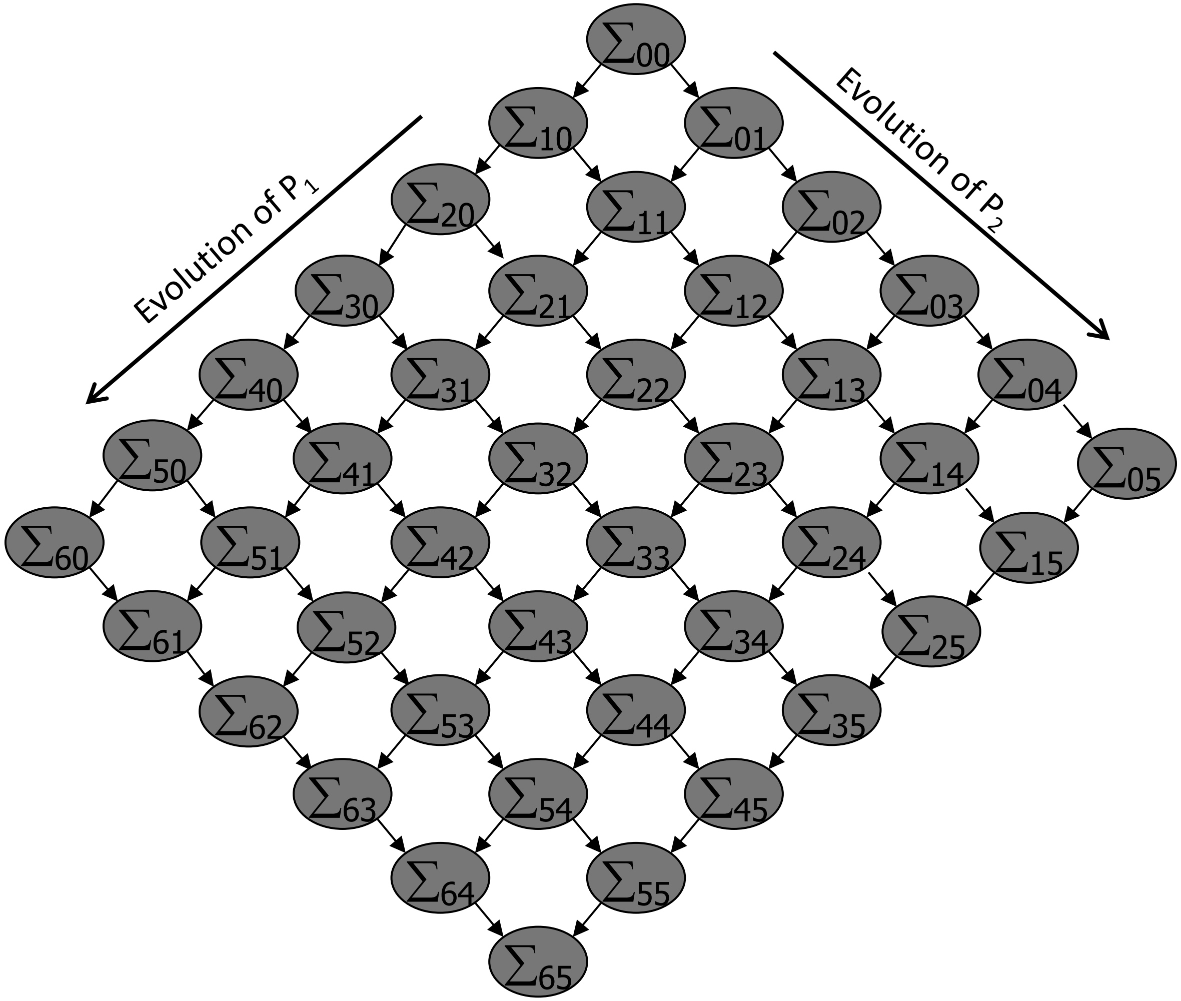
X = 1	X = 1	X = 1	a = Y	a = Y	a = Y
Y = 2	a = Y	a = Y	b = X	X = 1	X = 1
a = Y	b = X	Y = 2	X = 1	Y = 2	b = X
b = X	Y = 2	b = X	Y = 2	b = X	Y = 2
a=2, b=1	a=0, b=1	a=0, b=1	a=0, b=0	a=0, b=1	a=0, b=1

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!

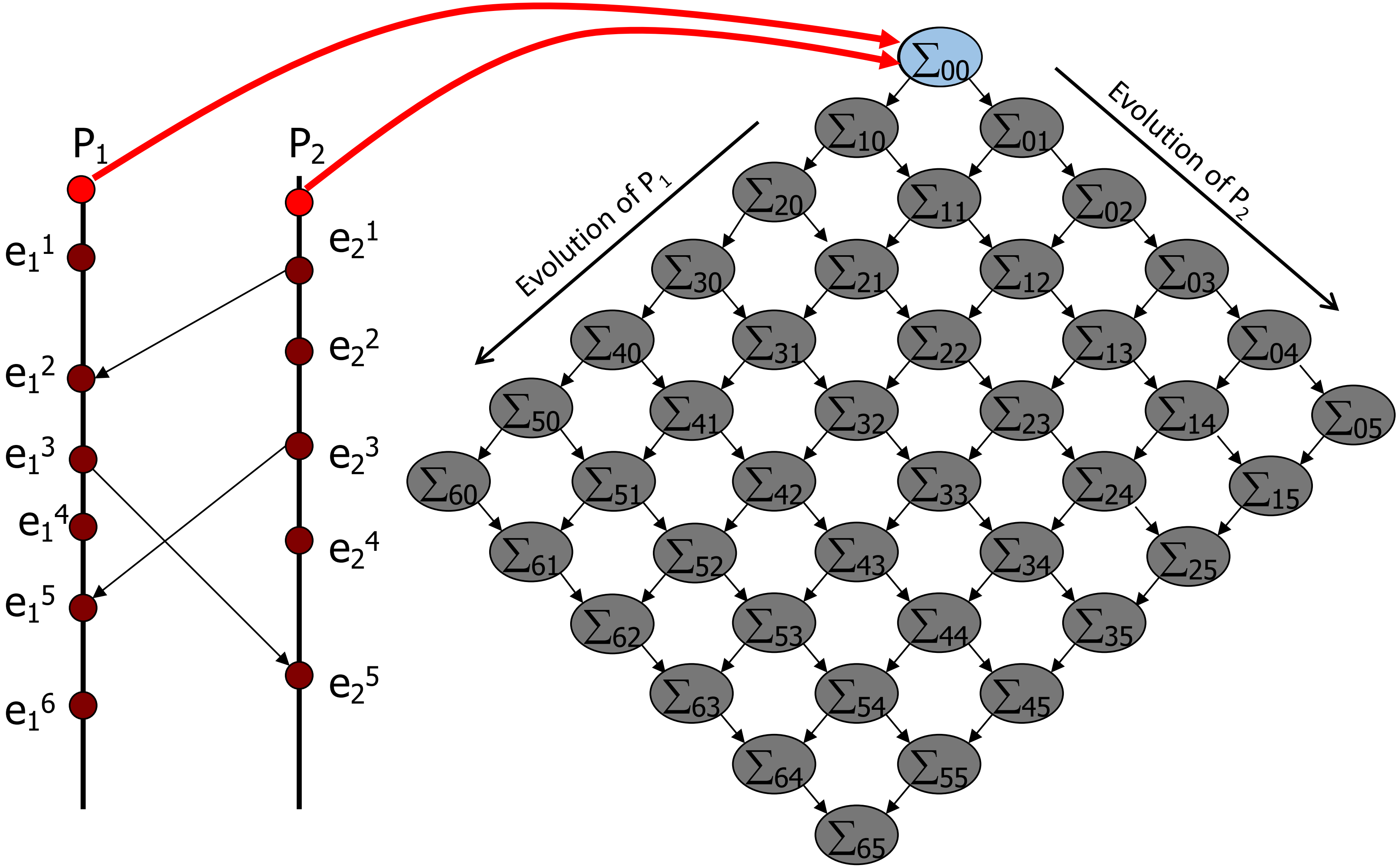
State explosion in concurrent programs





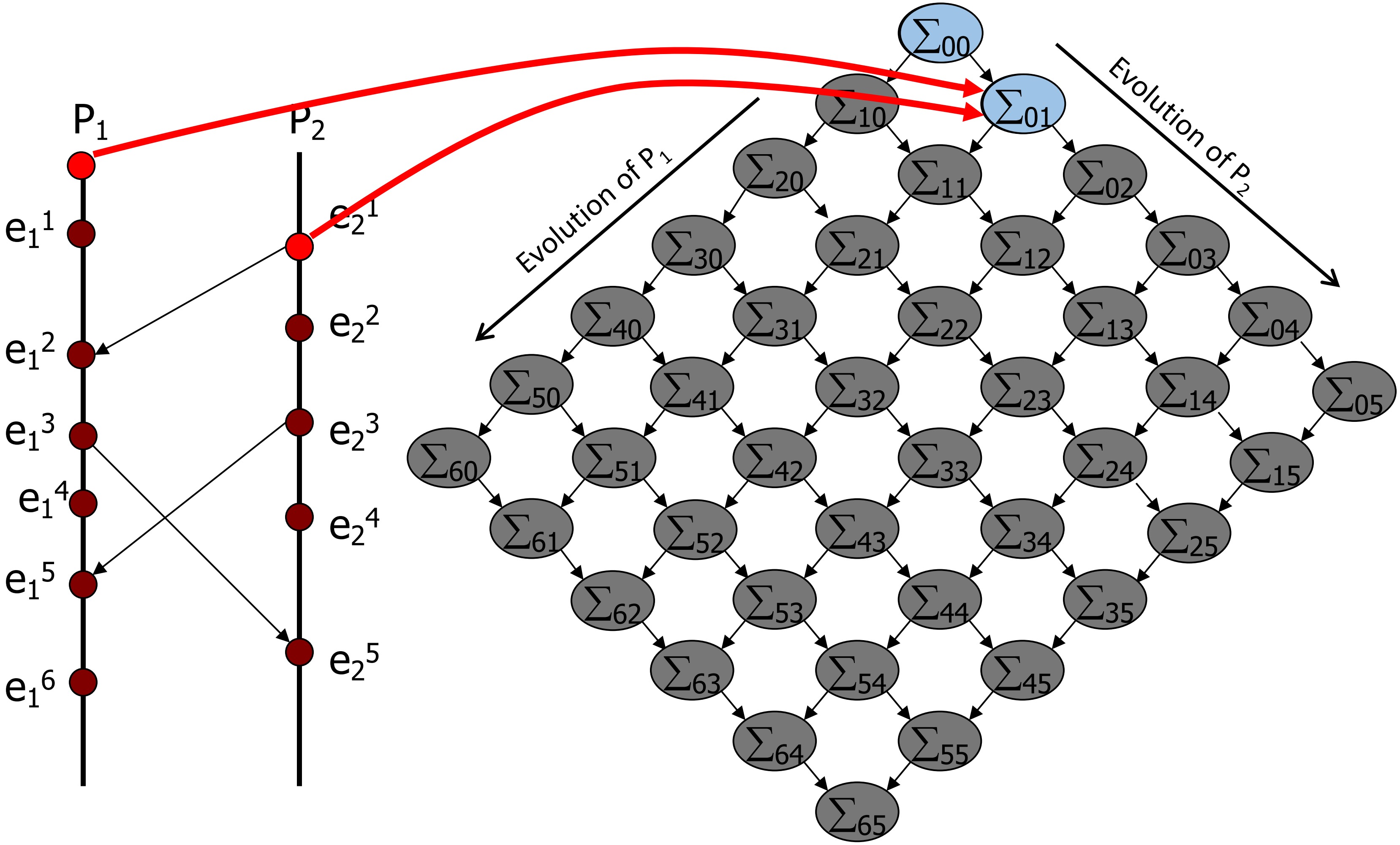
Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}

State explosion in concurrent programs

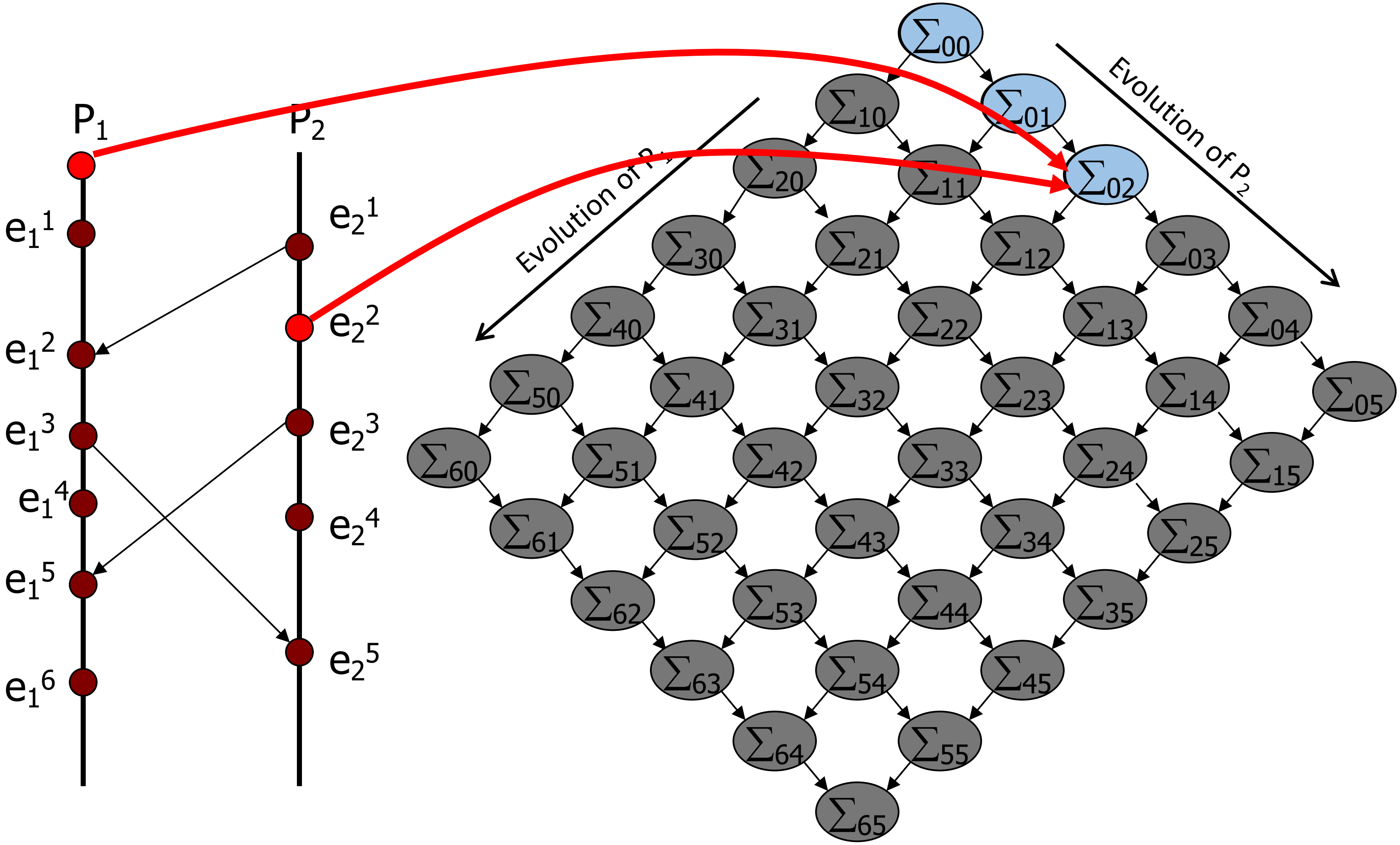


P_1
P_2
e_1^1
e_1^2
e_1^3
e_1^4
e_1^5
e_1^6
e_2^1
e_2^2
e_2^3
e_2^4
e_2^5
Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}

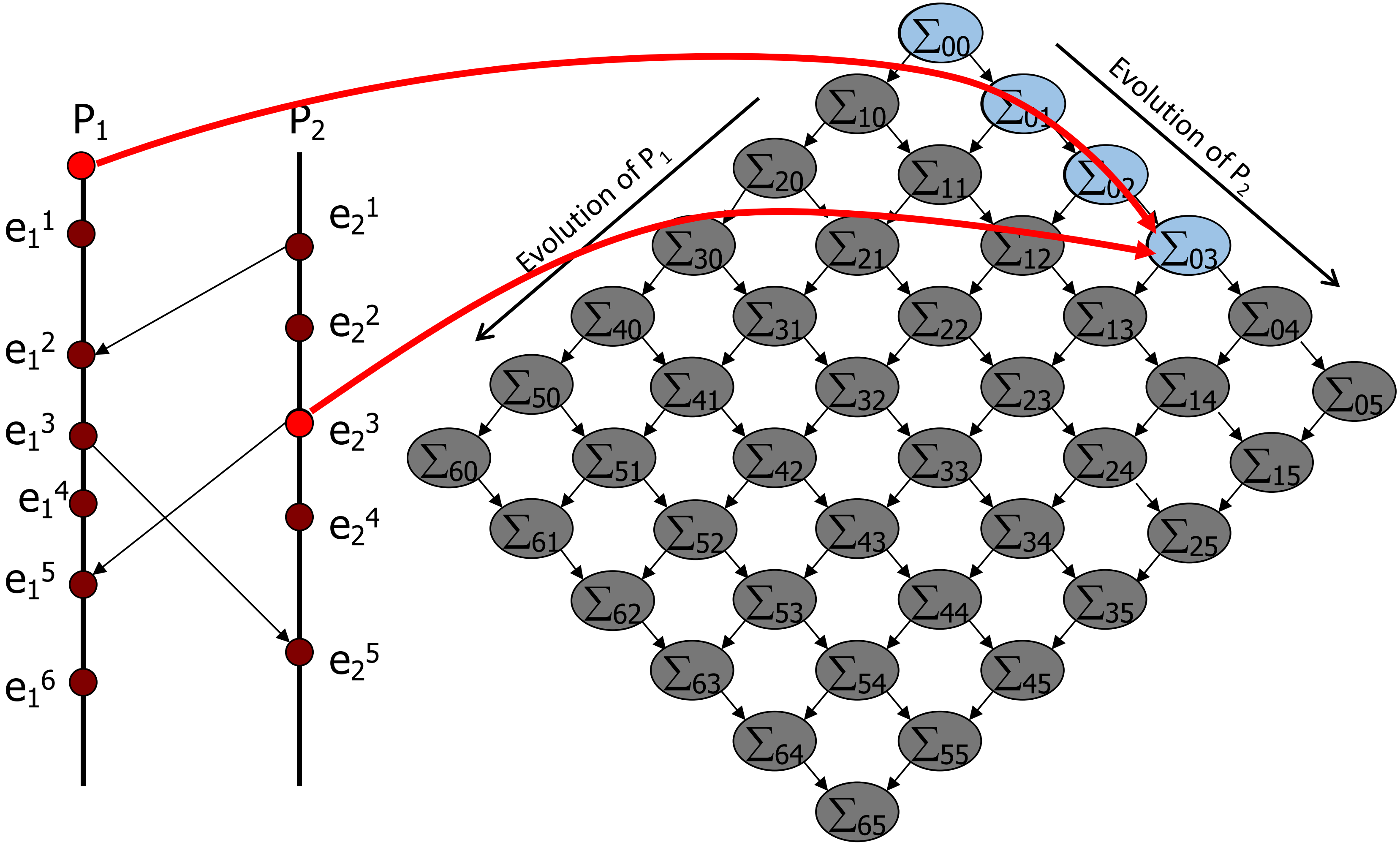
State explosion in concurrent programs


P_1
P_2
e_1^1
e_1^2
e_1^3
e_1^4
e_1^5
e_1^6
e_2^1
e_2^2
e_2^3
e_2^4
e_2^5
Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}

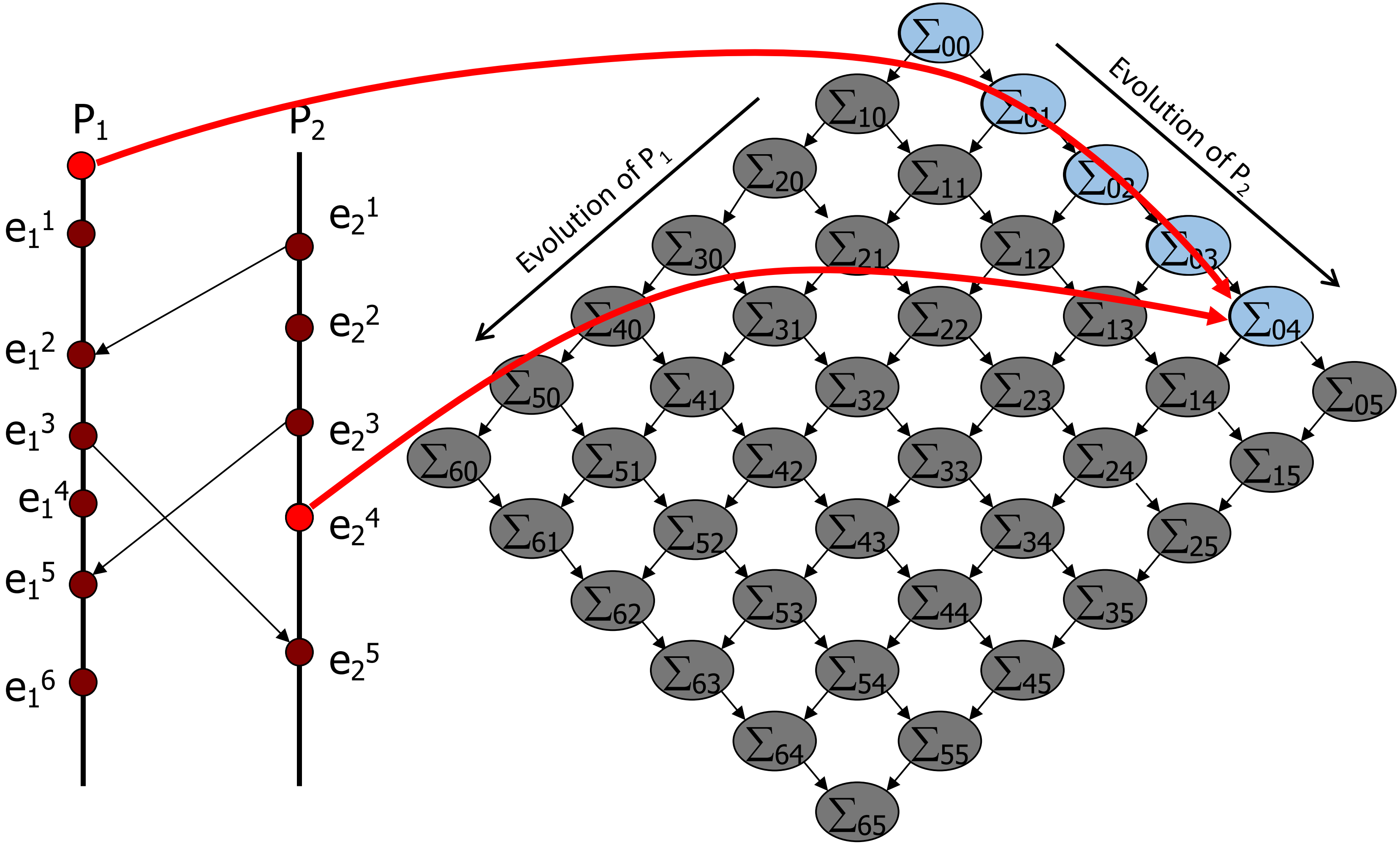
State explosion in concurrent programs



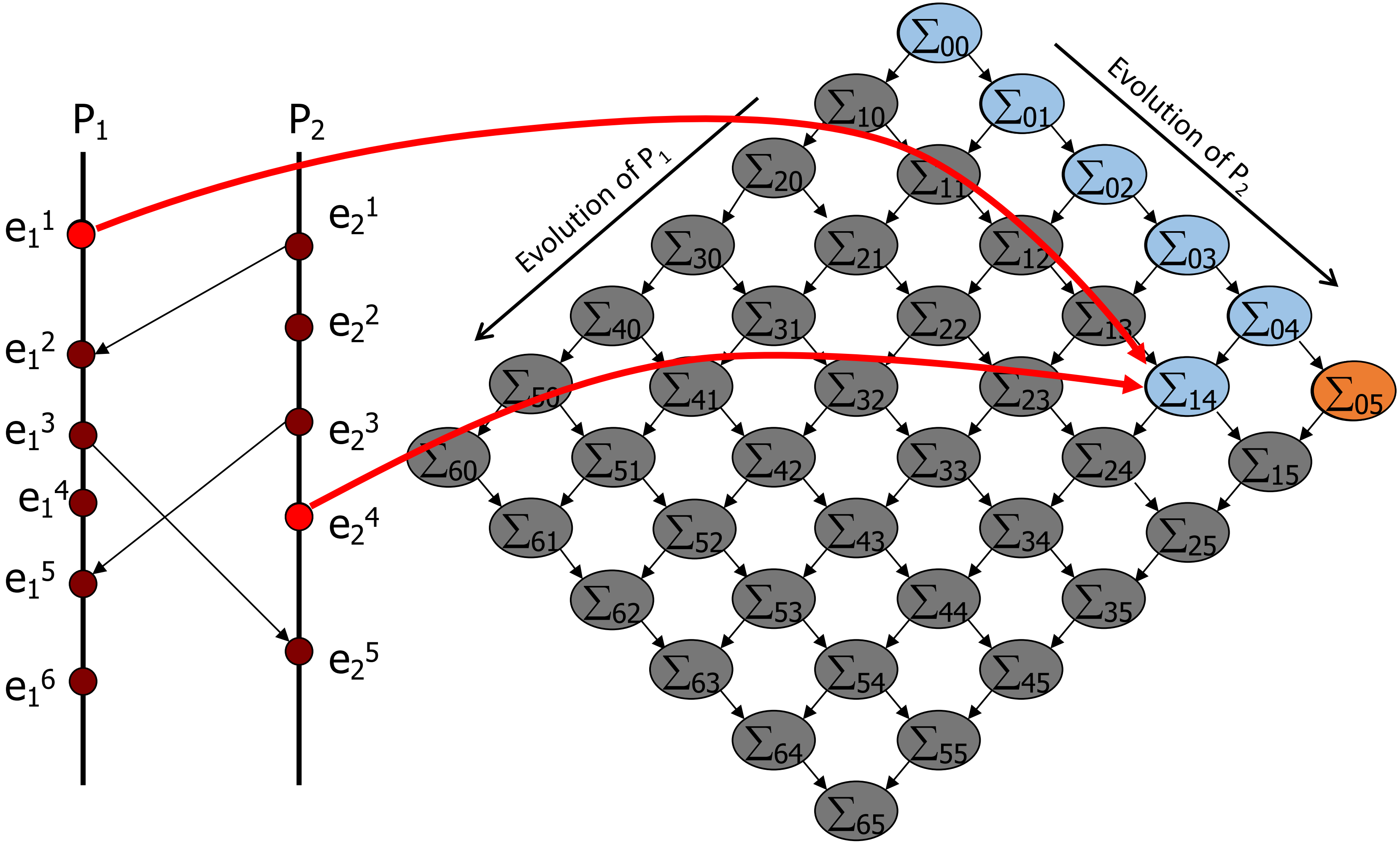
State explosion in concurrent programs



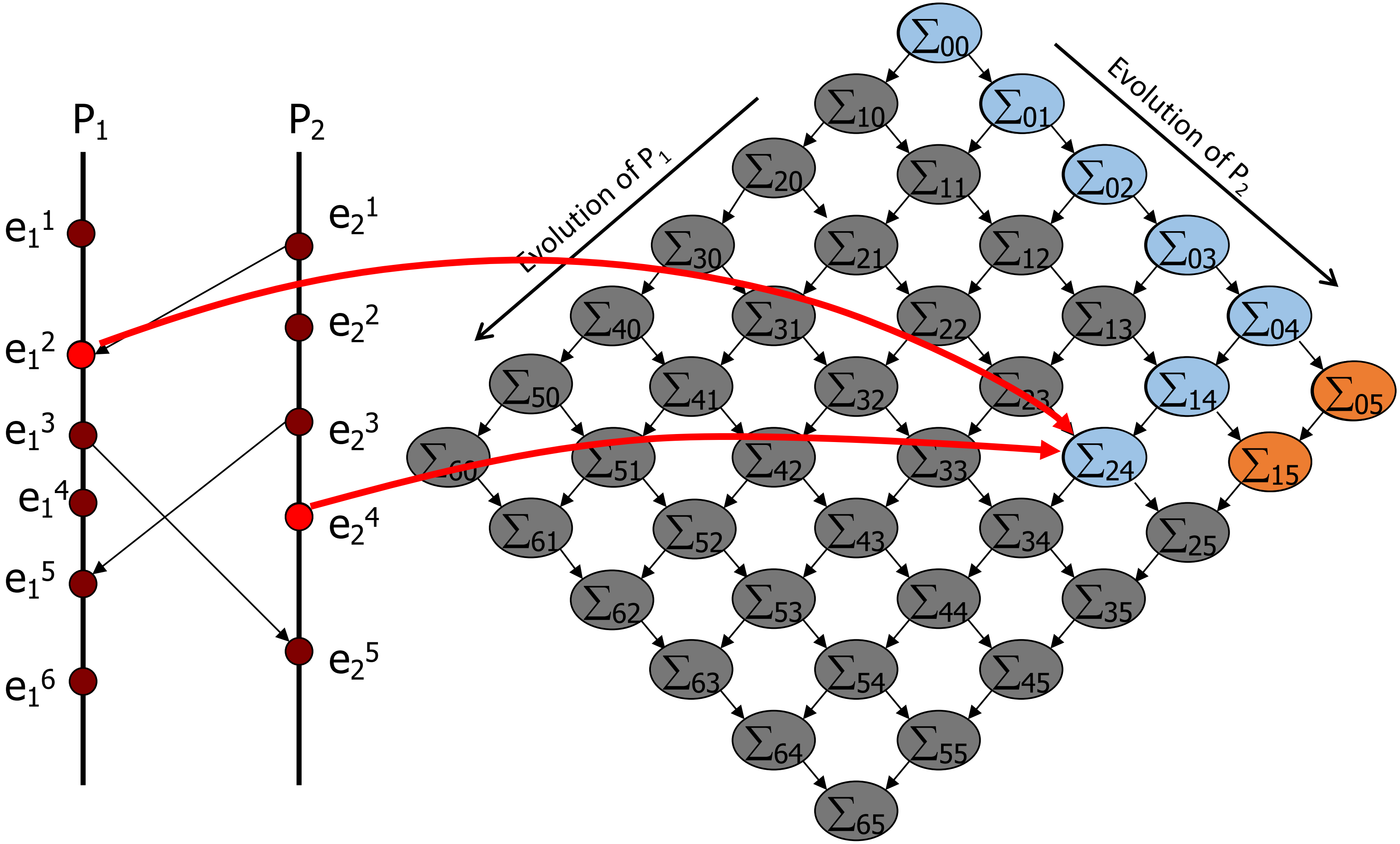
State explosion in concurrent programs


P_1
P_2
e_1^1
e_1^2
e_1^3
e_1^4
e_1^5
e_1^6
e_2^1
e_2^2
e_2^3
e_2^4
e_2^5
Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}

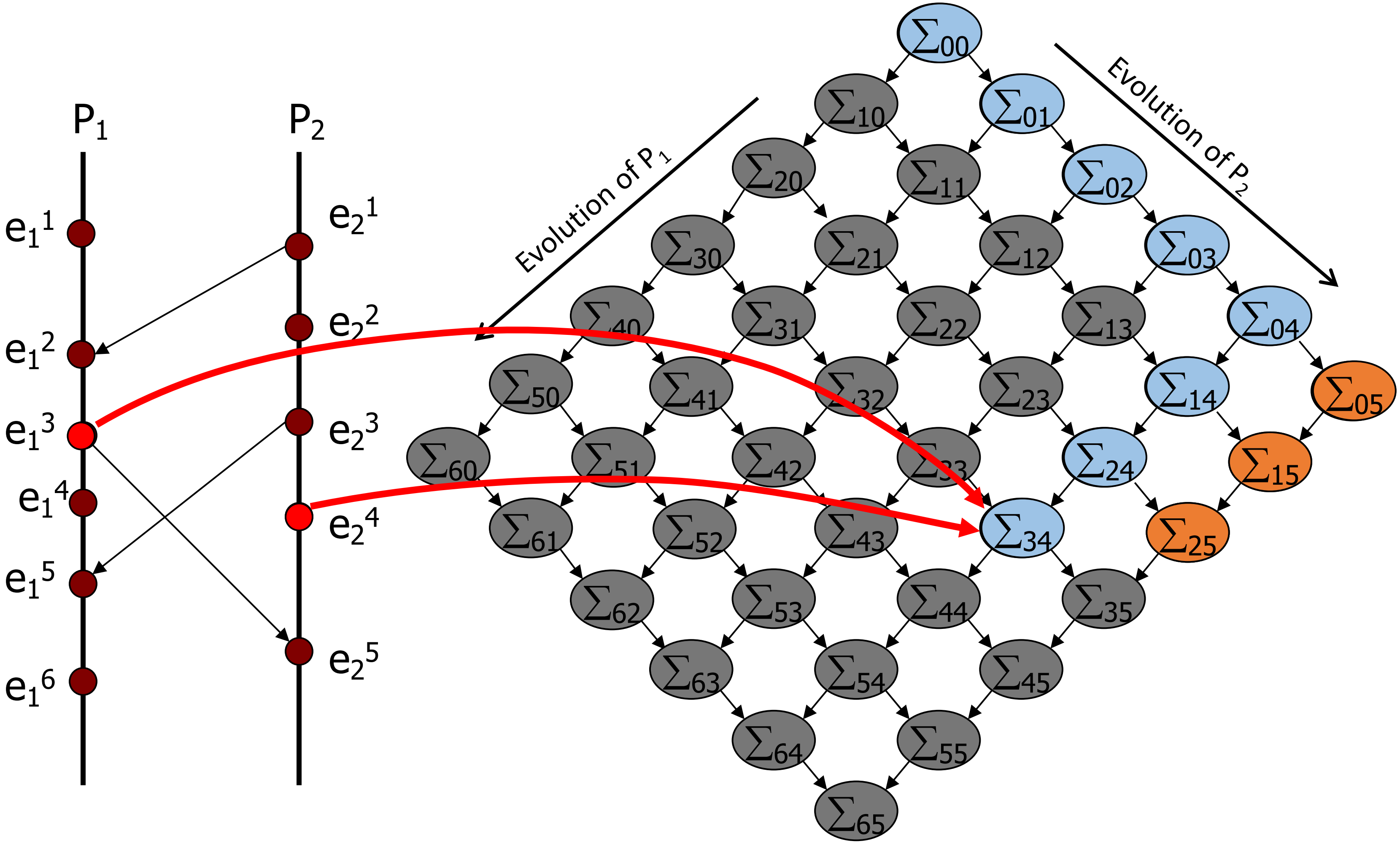
State explosion in concurrent programs



State explosion in concurrent programs

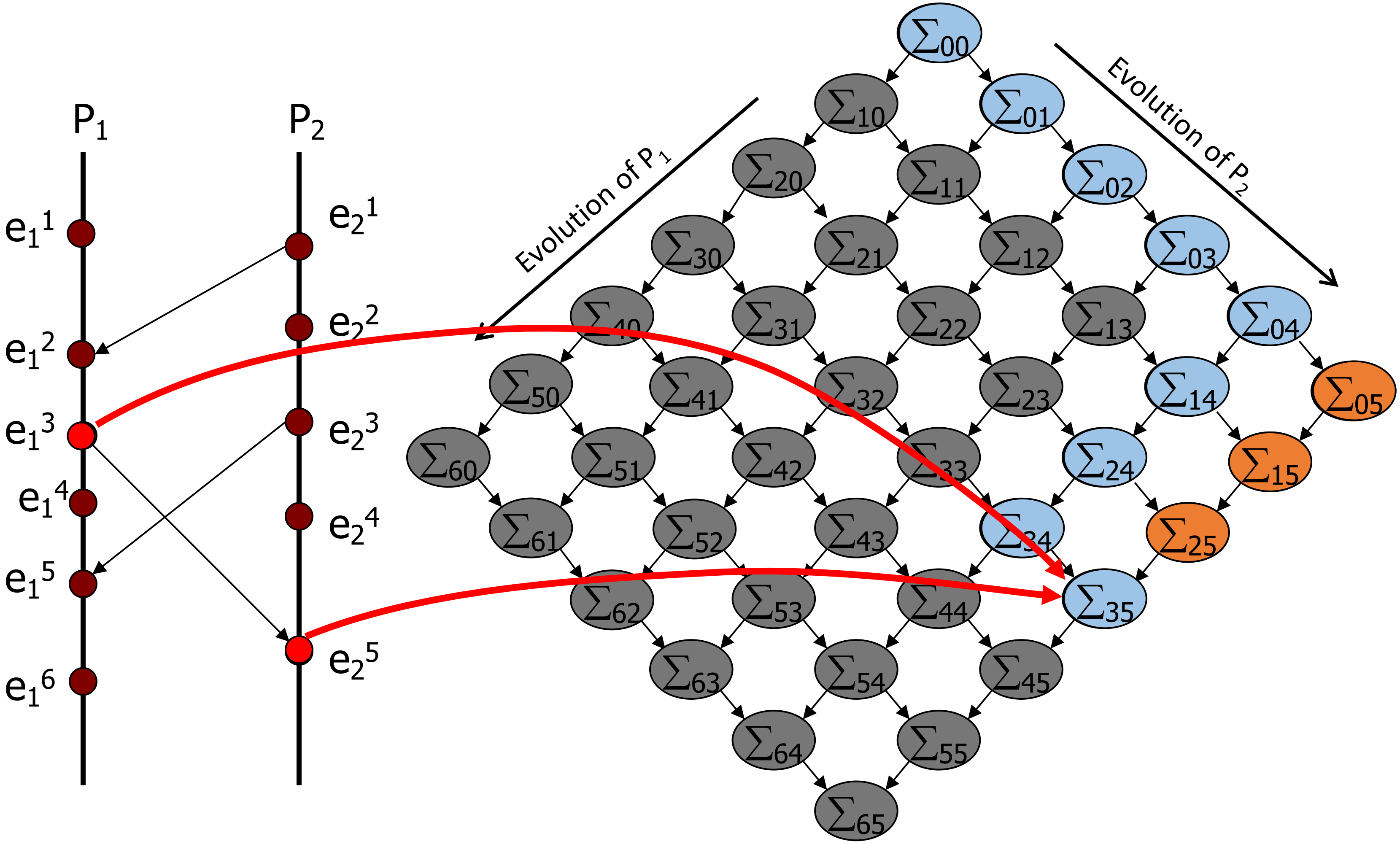


State explosion in concurrent programs

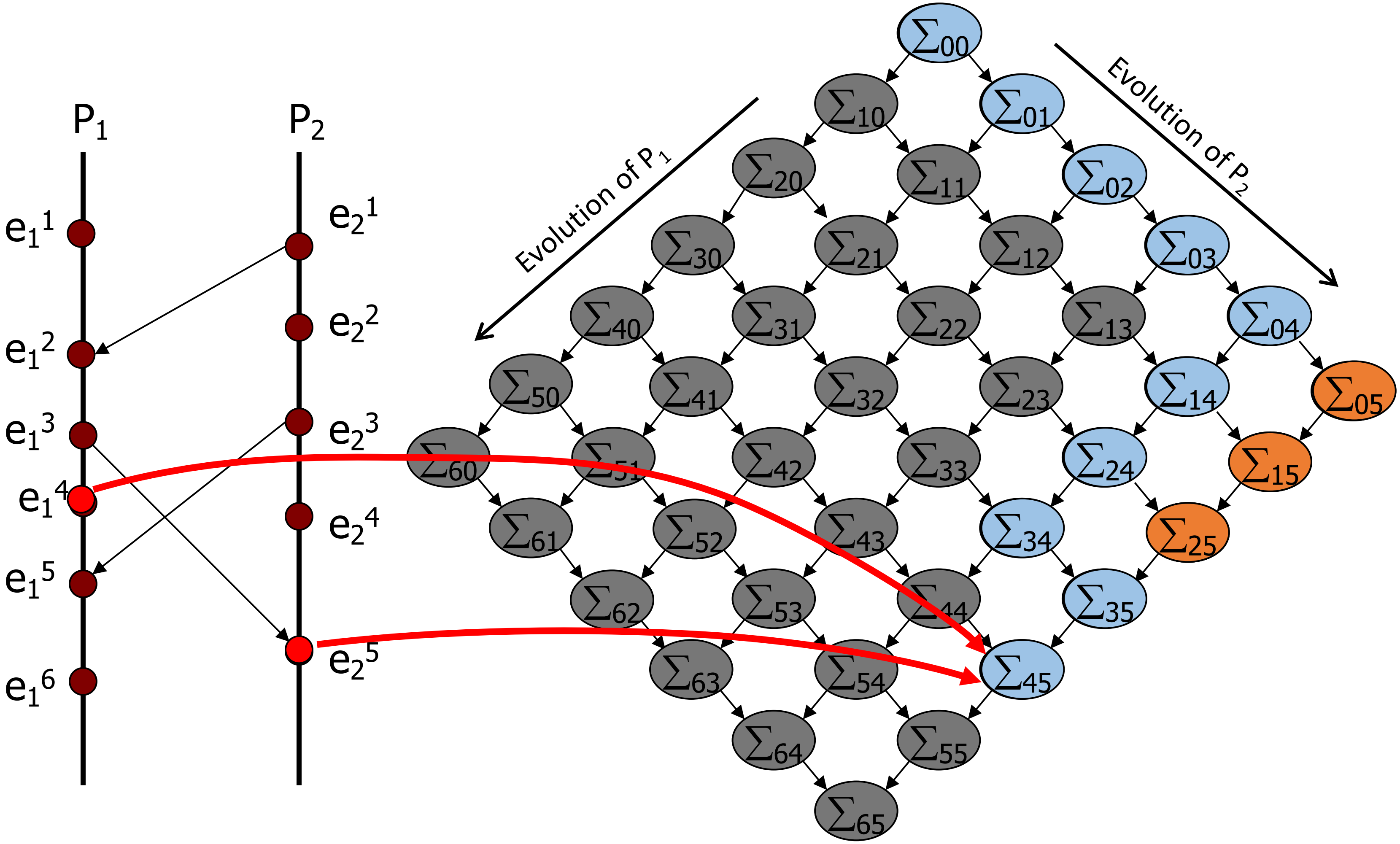


P_1
P_2
e_1^1
e_1^2
e_1^3
e_1^4
e_1^5
e_1^6
e_2^1
e_2^2
e_2^3
e_2^4
e_2^5
Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}

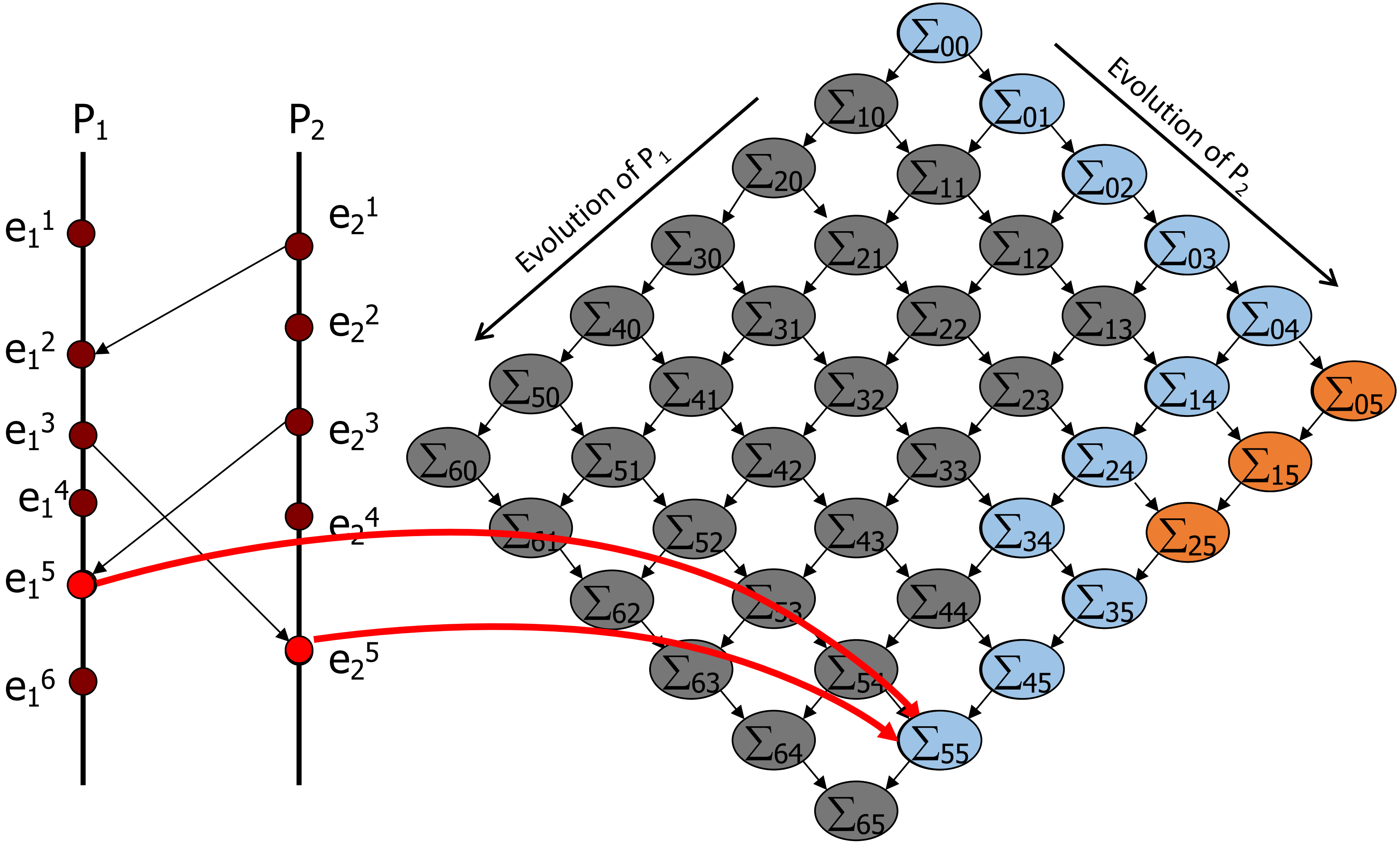
State explosion in concurrent programs


P_1
P_2
e_1^1
e_1^2
e_1^3
e_1^4
e_1^5
e_1^6
e_2^1
e_2^2
e_2^3
e_2^4
e_2^5
Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}

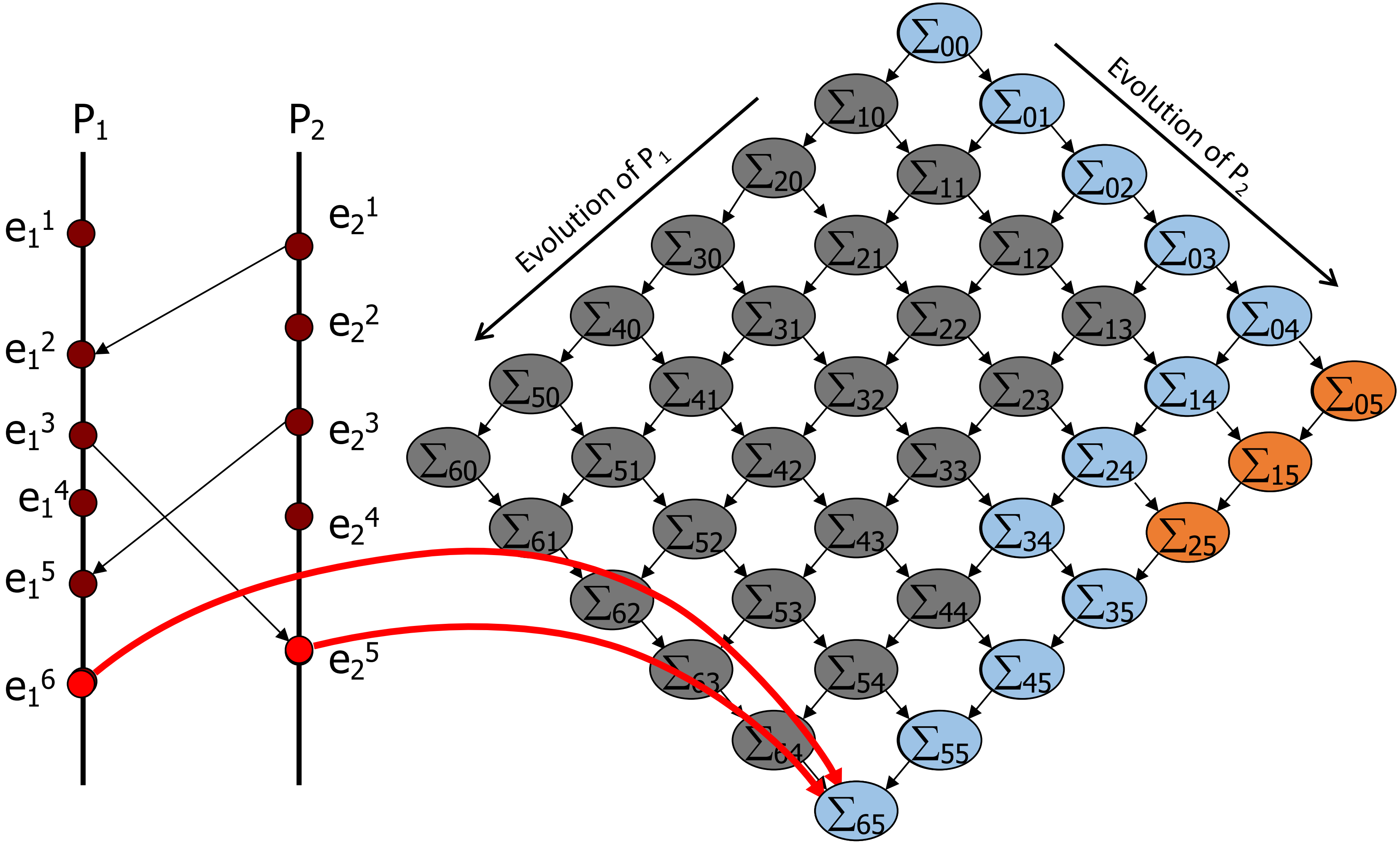
State explosion in concurrent programs



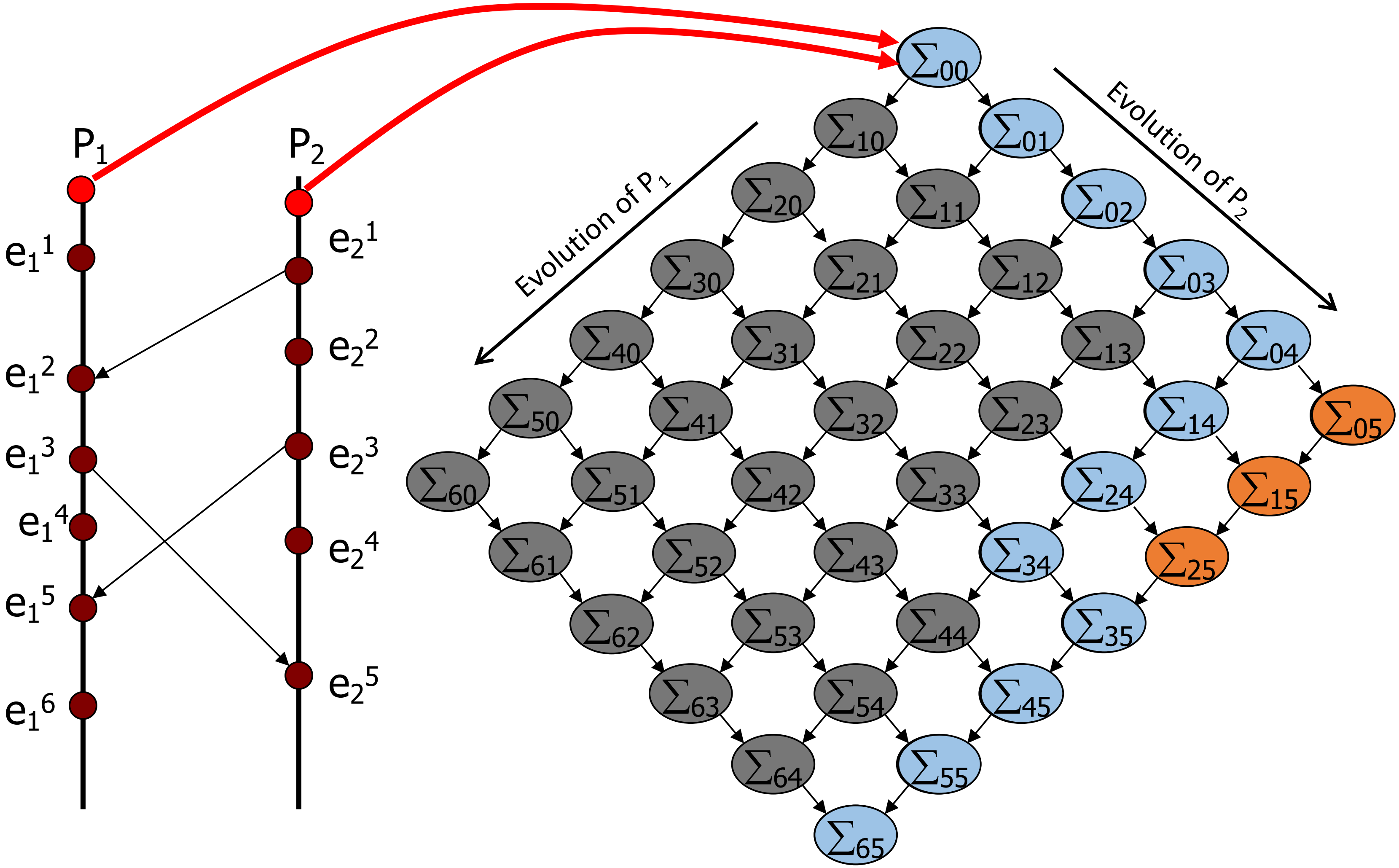
State explosion in concurrent programs



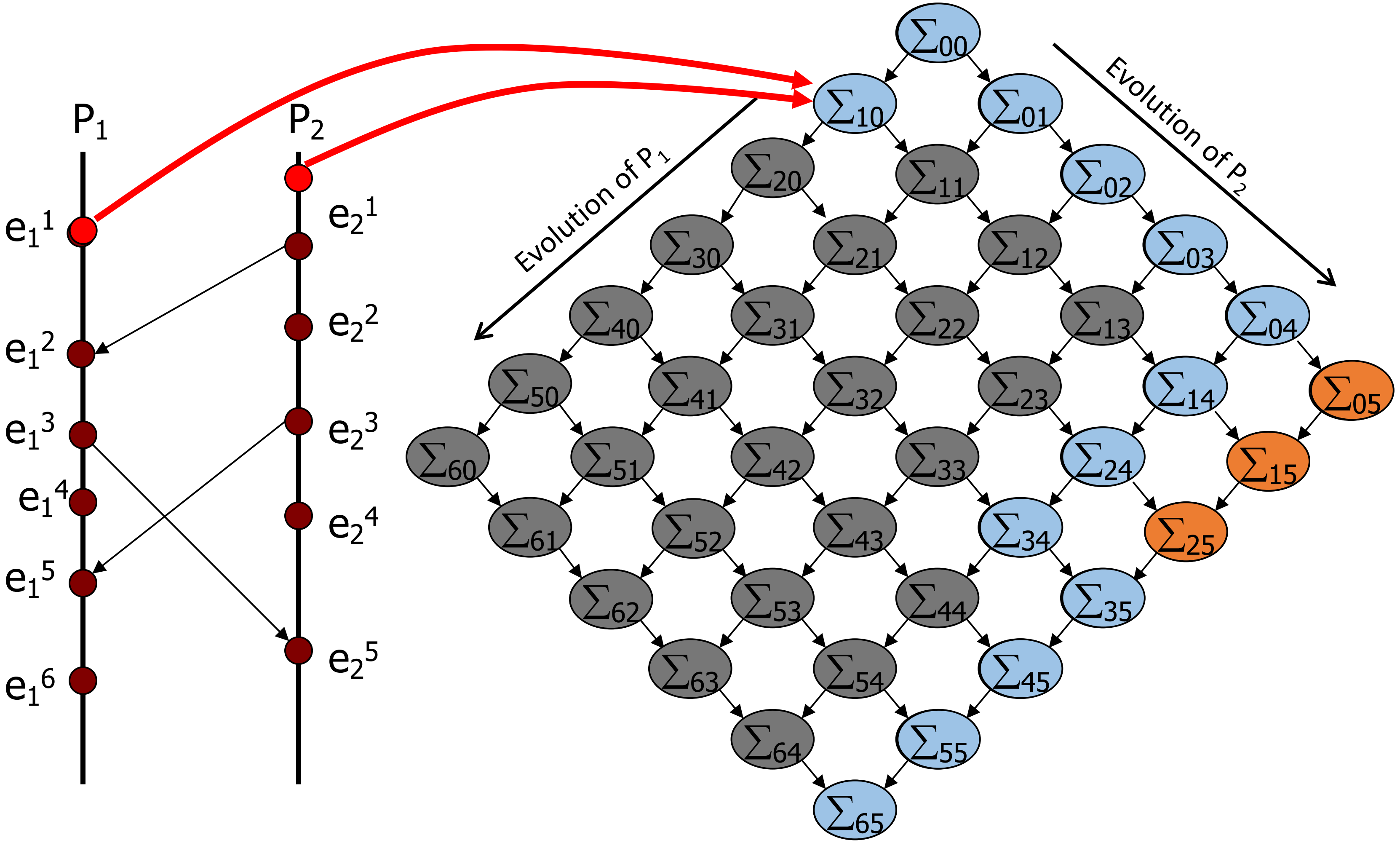
State explosion in concurrent programs


P_1
P_2
e_1^1
e_1^2
e_1^3
e_1^4
e_1^5
e_1^6
e_2^1
e_2^2
e_2^3
e_2^4
e_2^5
Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}

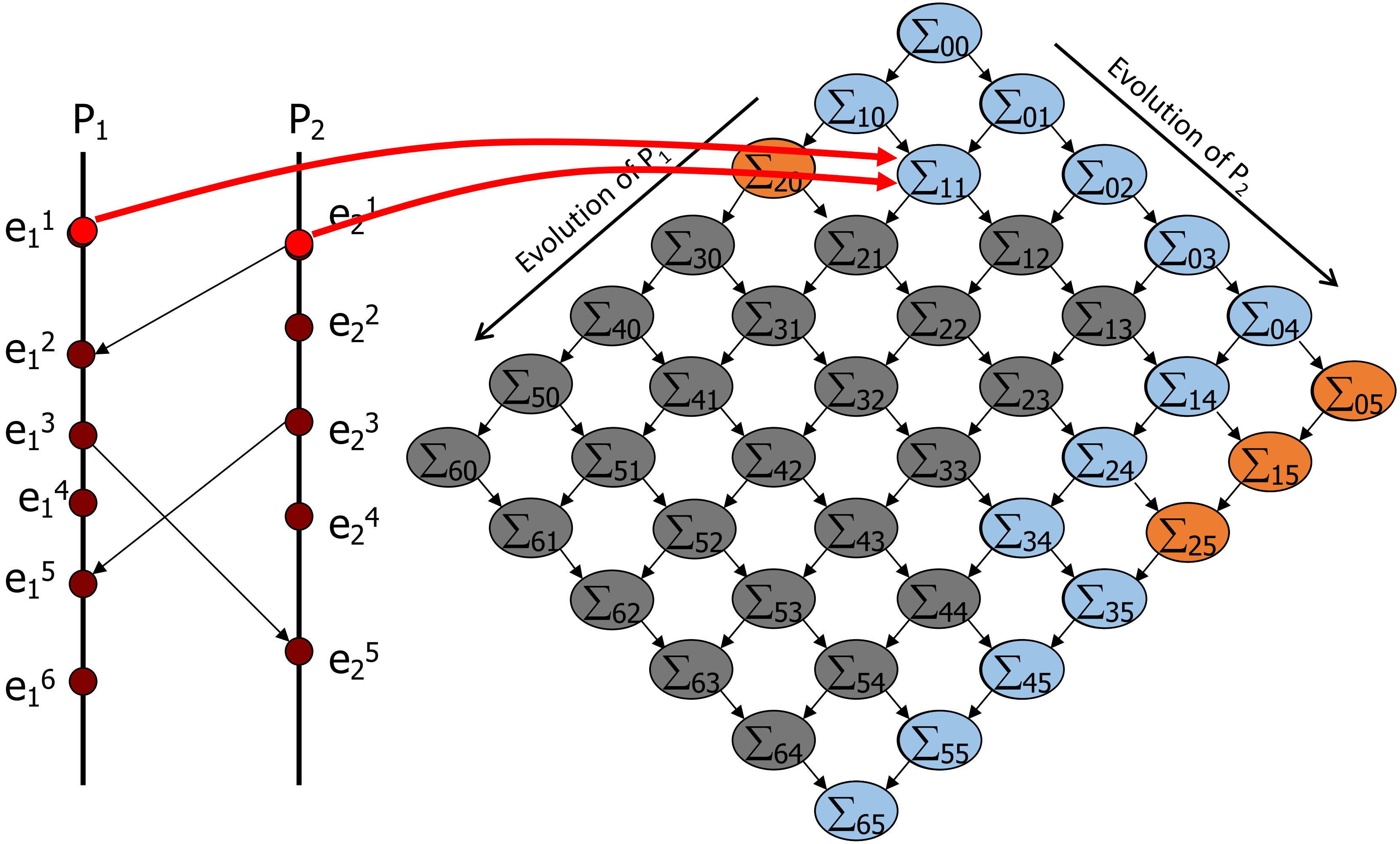
State explosion in concurrent programs



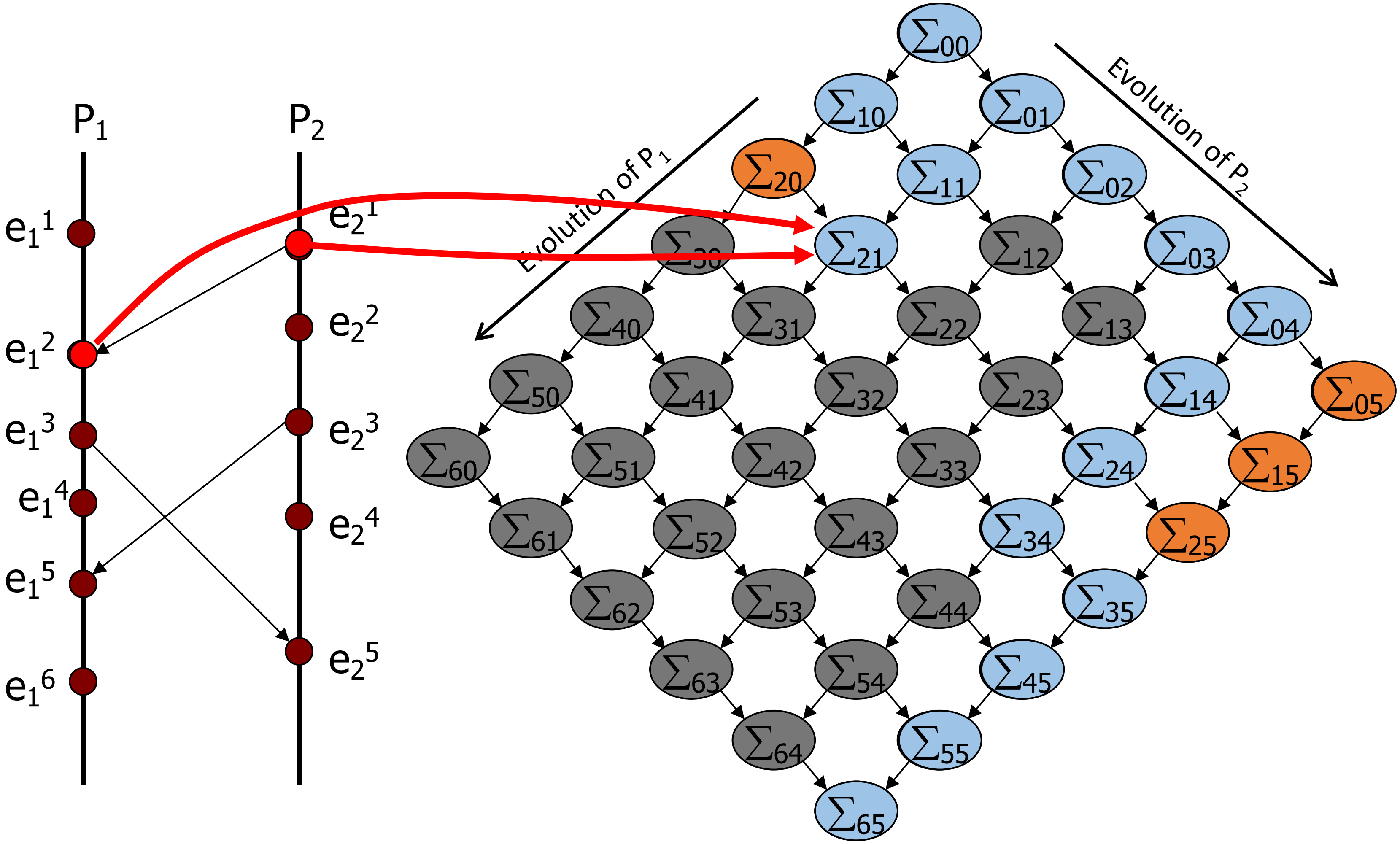
State explosion in concurrent programs



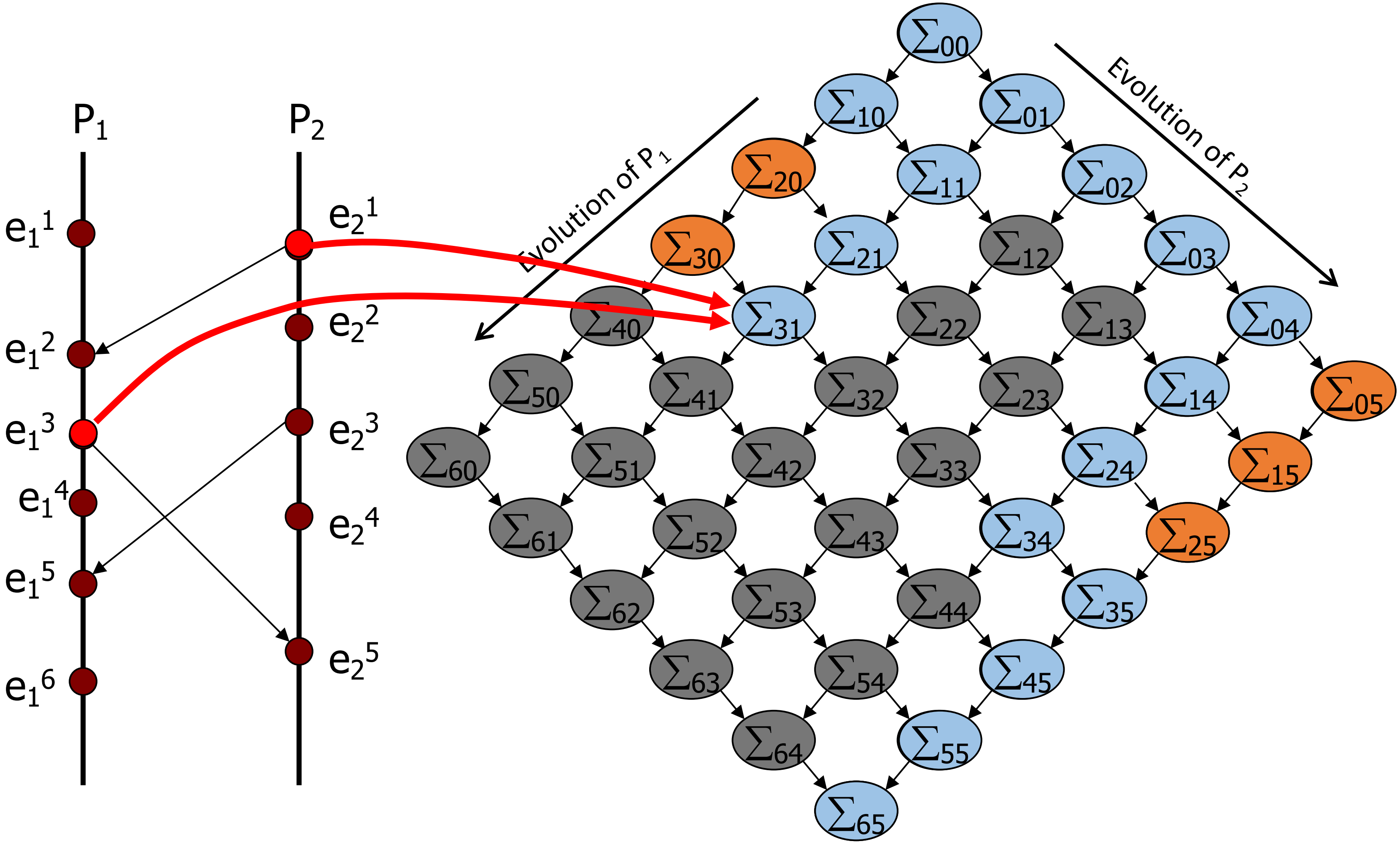
State explosion in concurrent programs



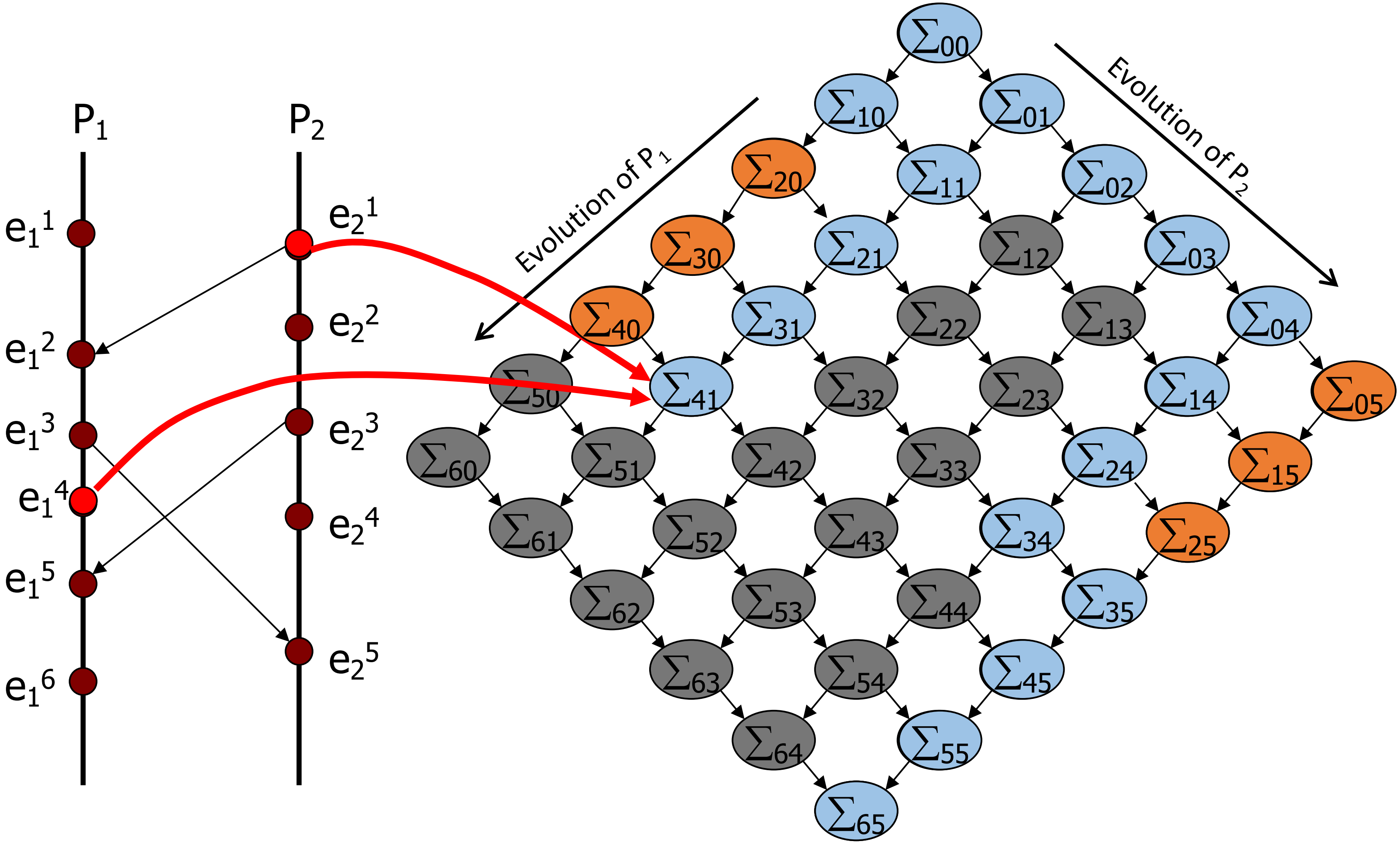
State explosion in concurrent programs



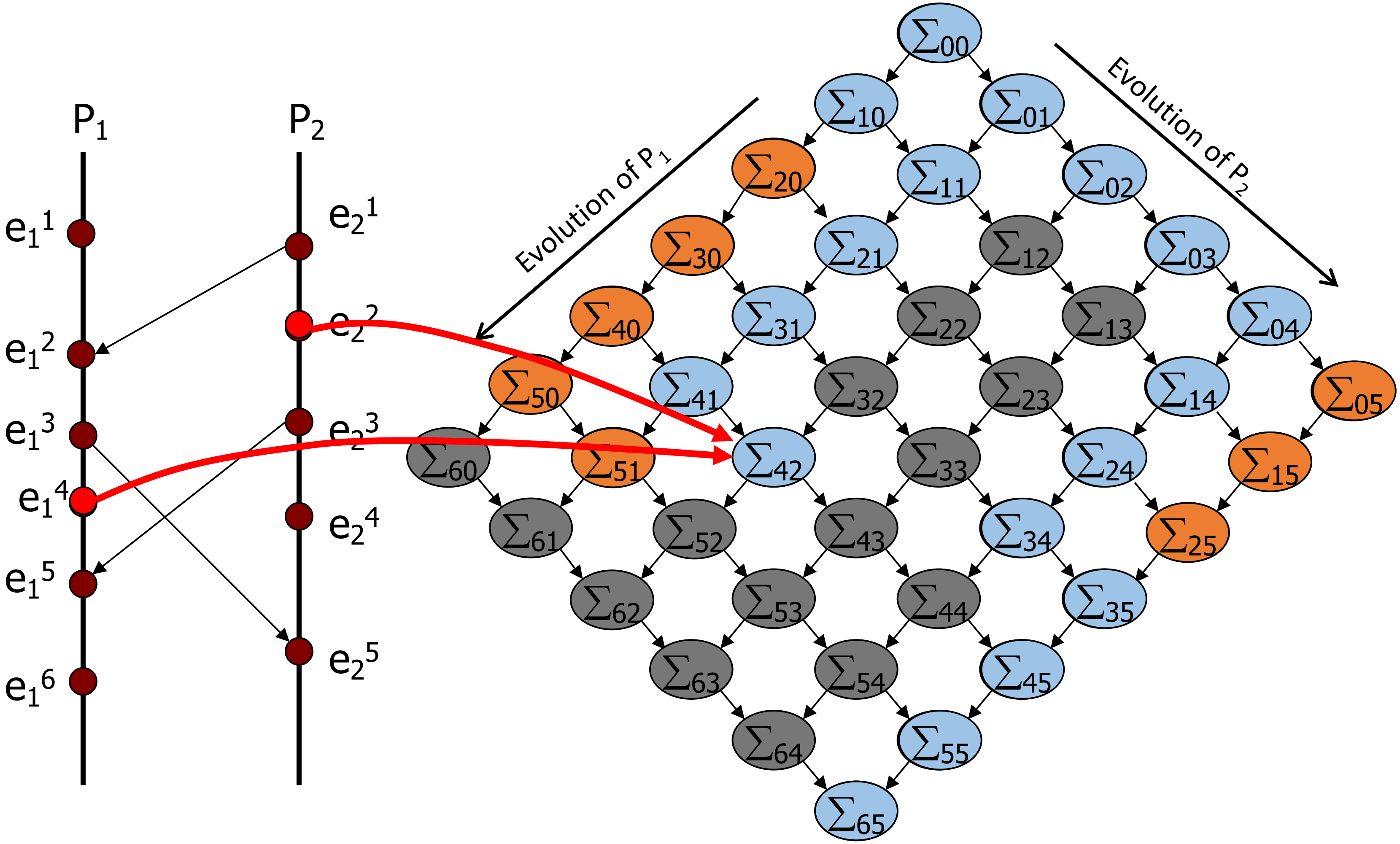
State explosion in concurrent programs



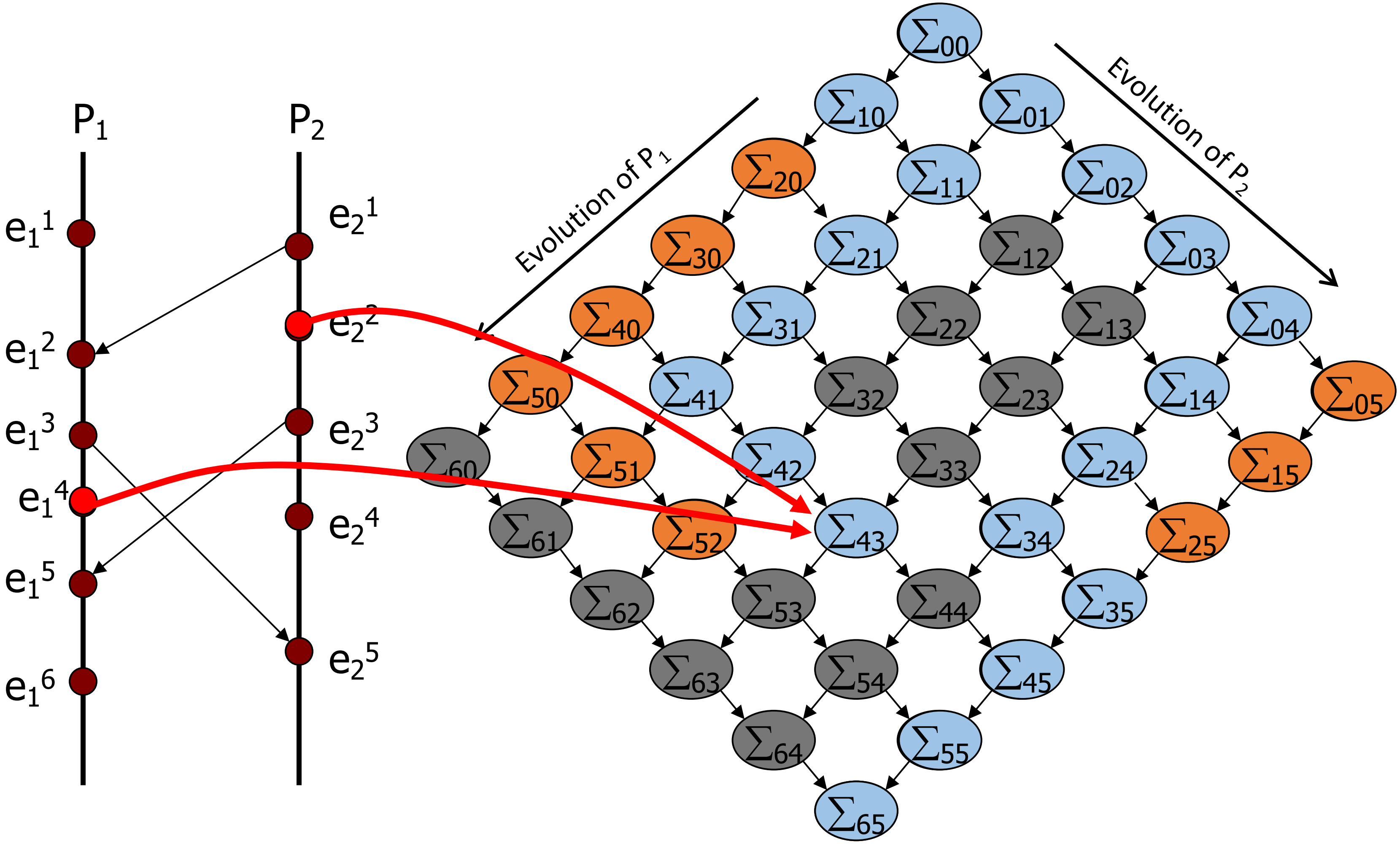
State explosion in concurrent programs



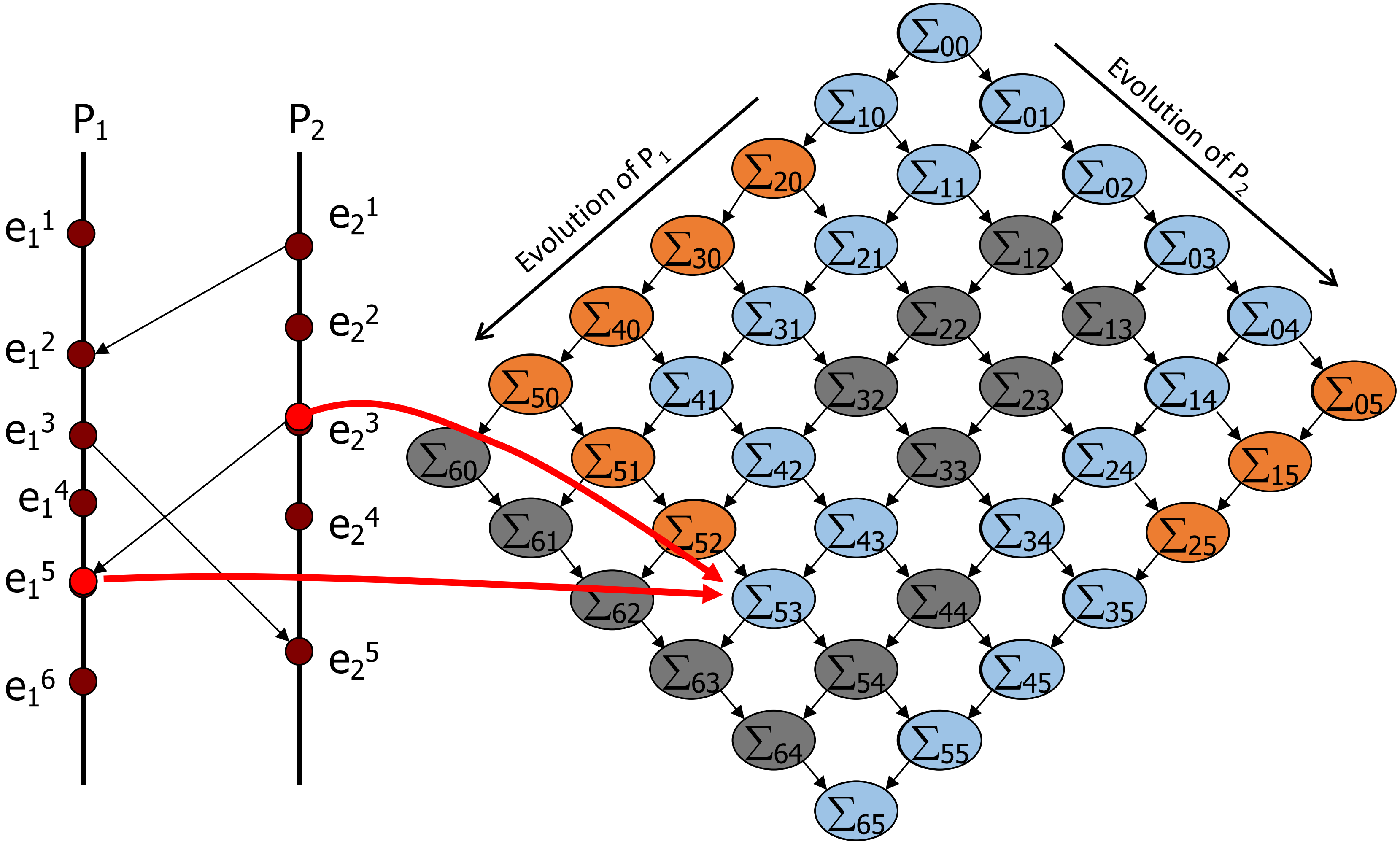
State explosion in concurrent programs



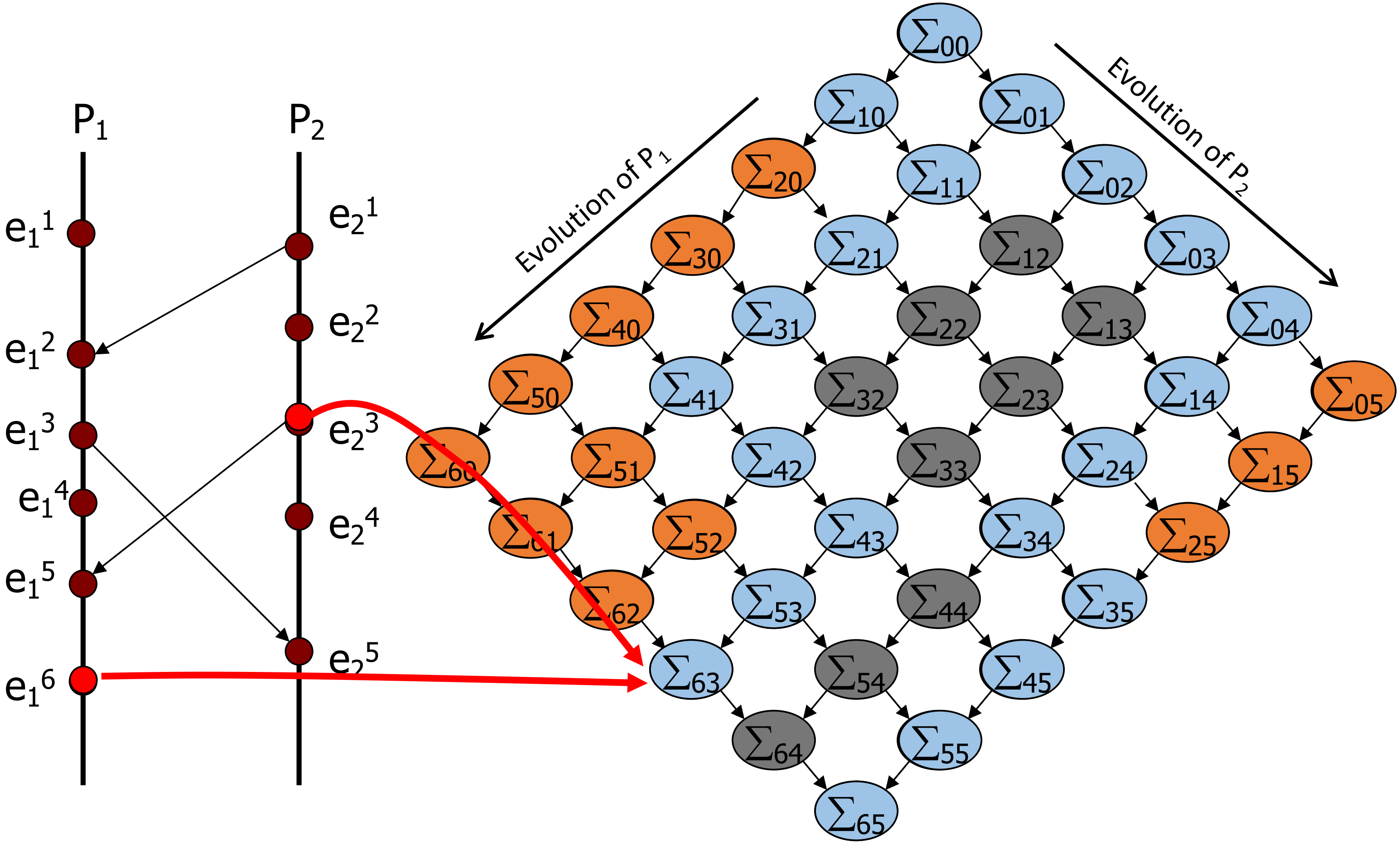
State explosion in concurrent programs



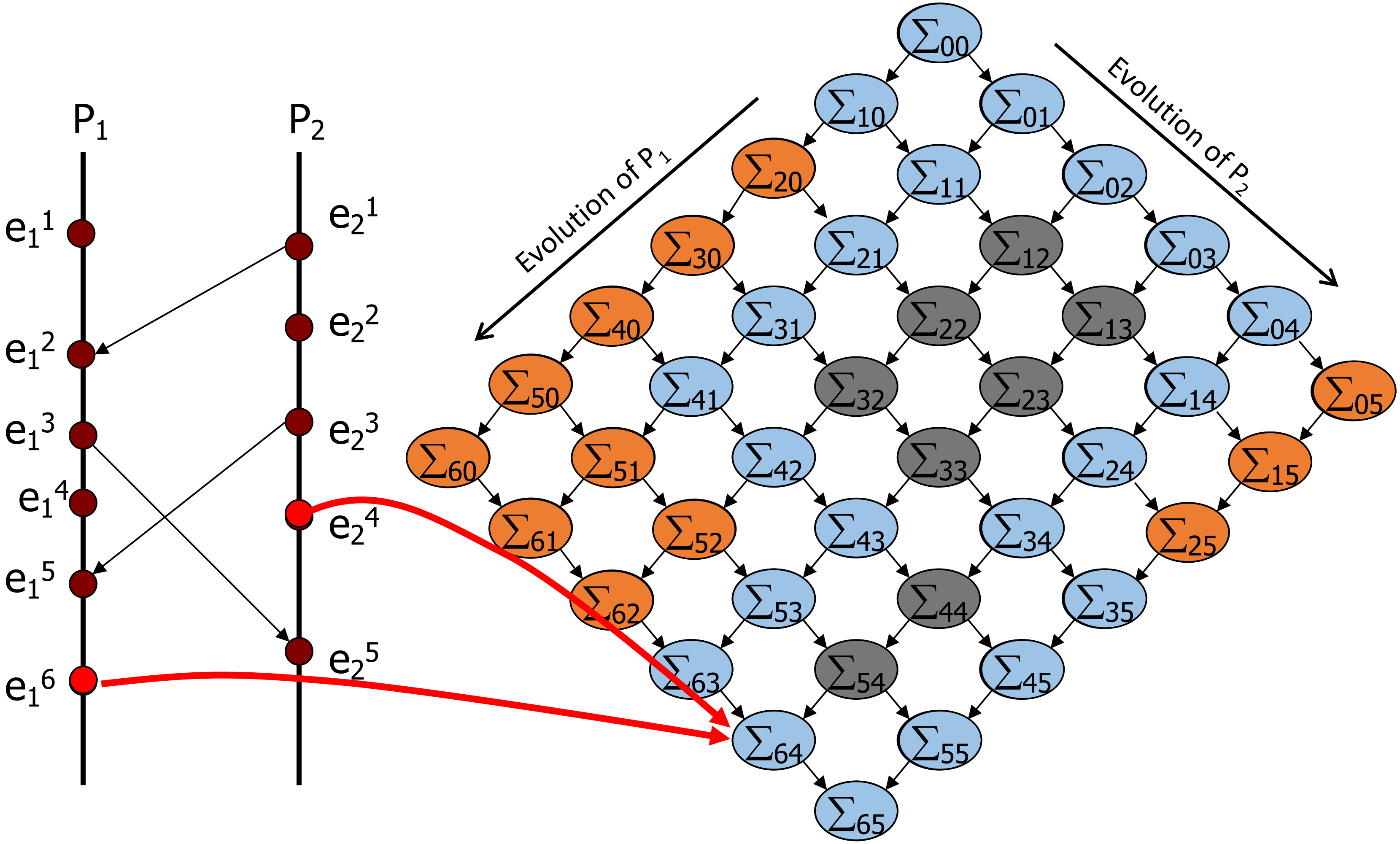
State explosion in concurrent programs



State explosion in concurrent programs

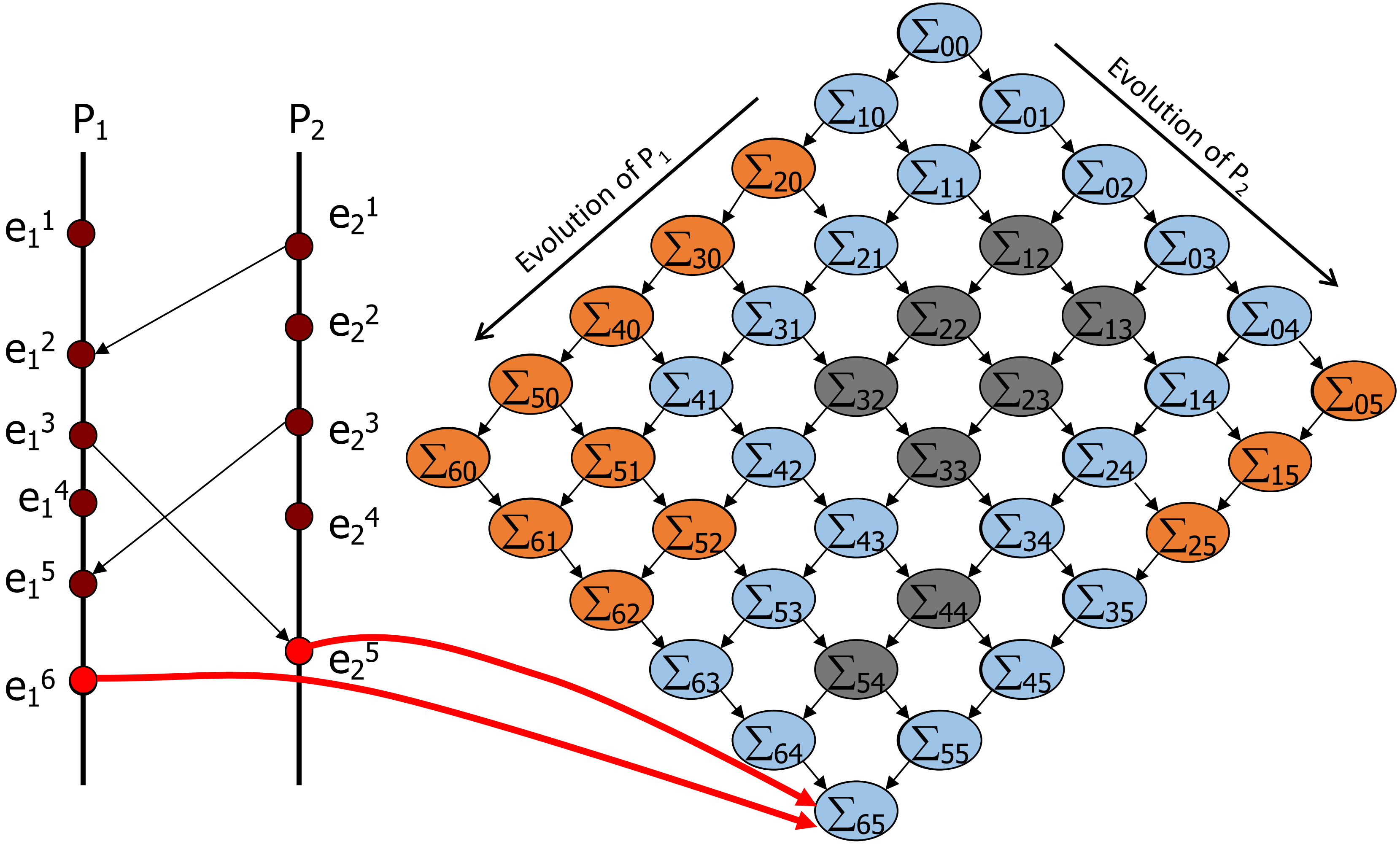


State explosion in concurrent programs

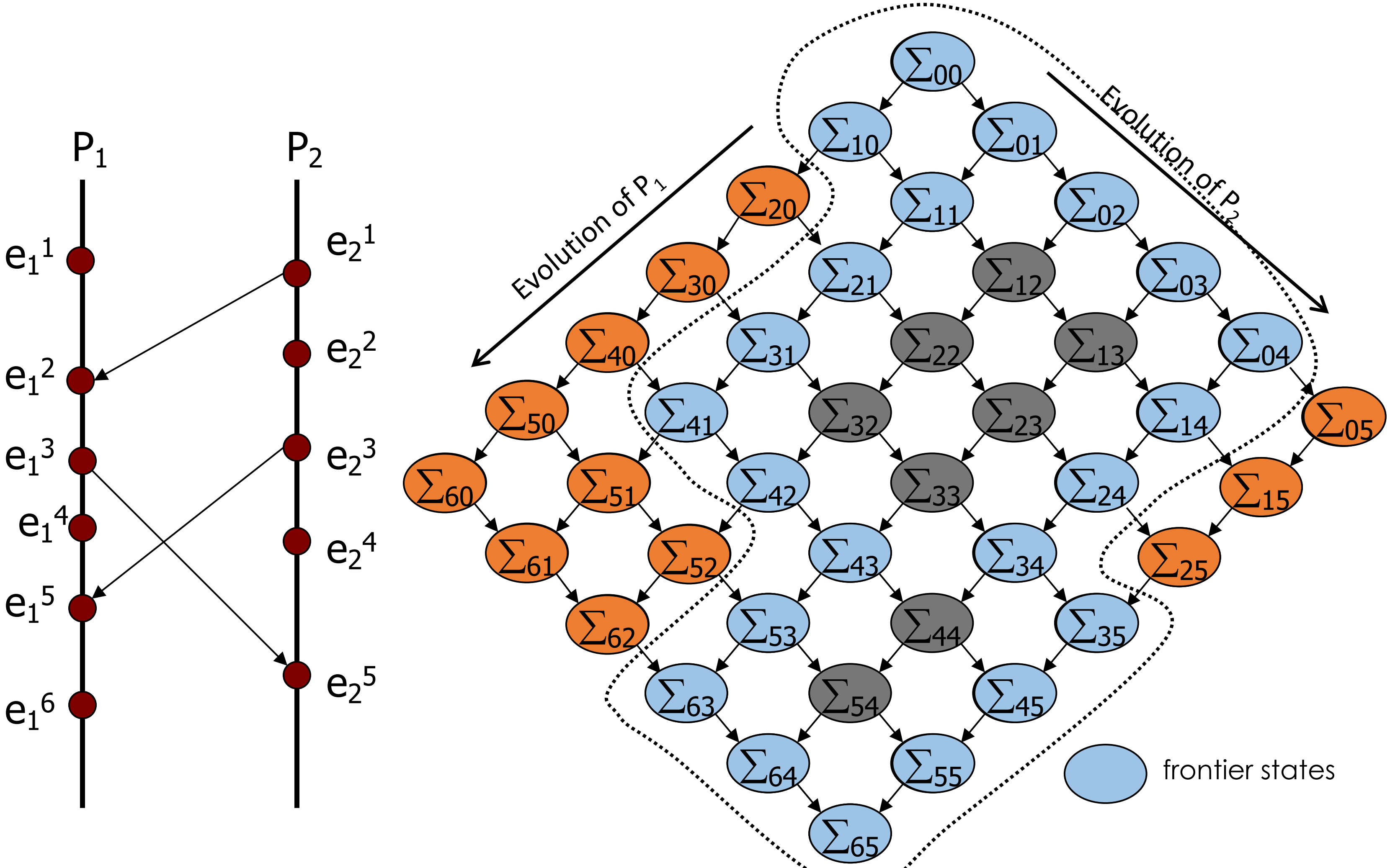


P_1
P_2
e_1^1
e_1^2
e_1^3
e_1^4
e_1^5
e_1^6
e_2^1
e_2^2
e_2^3
e_2^4
e_2^5
Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}

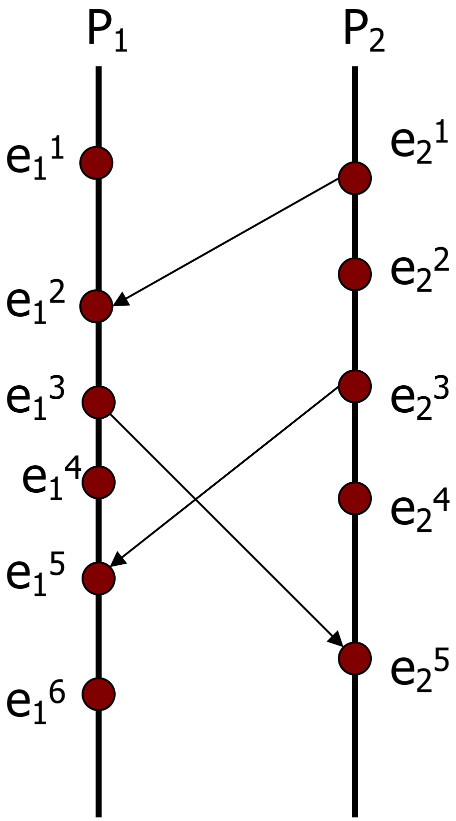
State explosion in concurrent programs

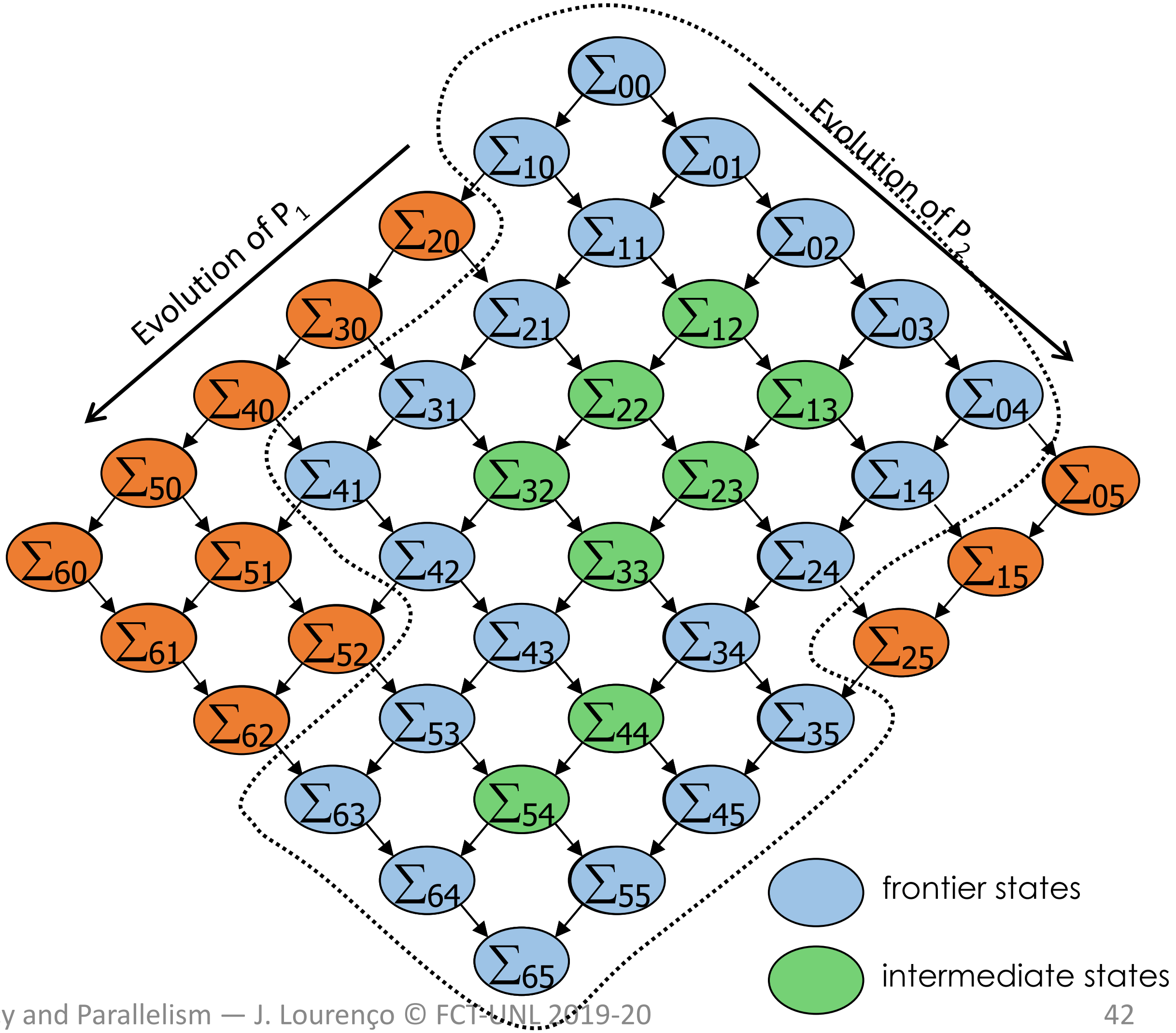

P_1
P_2
e_1^1
e_1^2
e_1^3
e_1^4
e_1^5
e_1^6
e_2^1
e_2^2
e_2^3
e_2^4
e_2^5
Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}

State explosion in concurrent programs

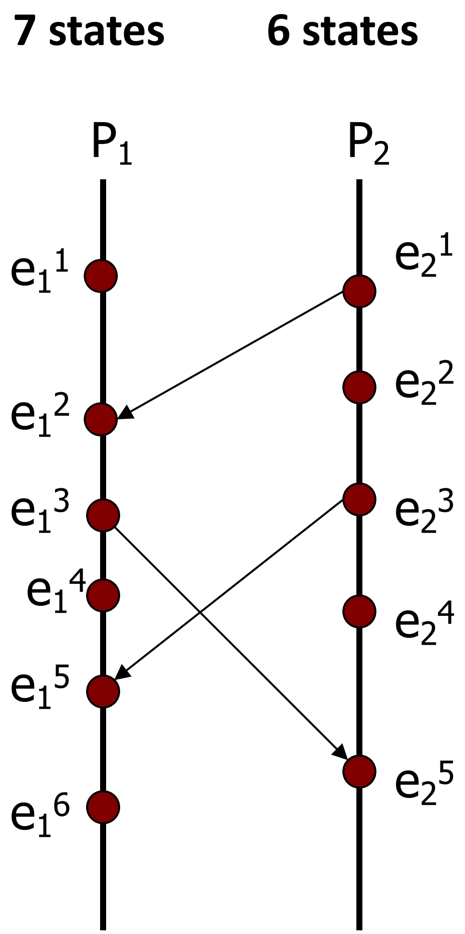


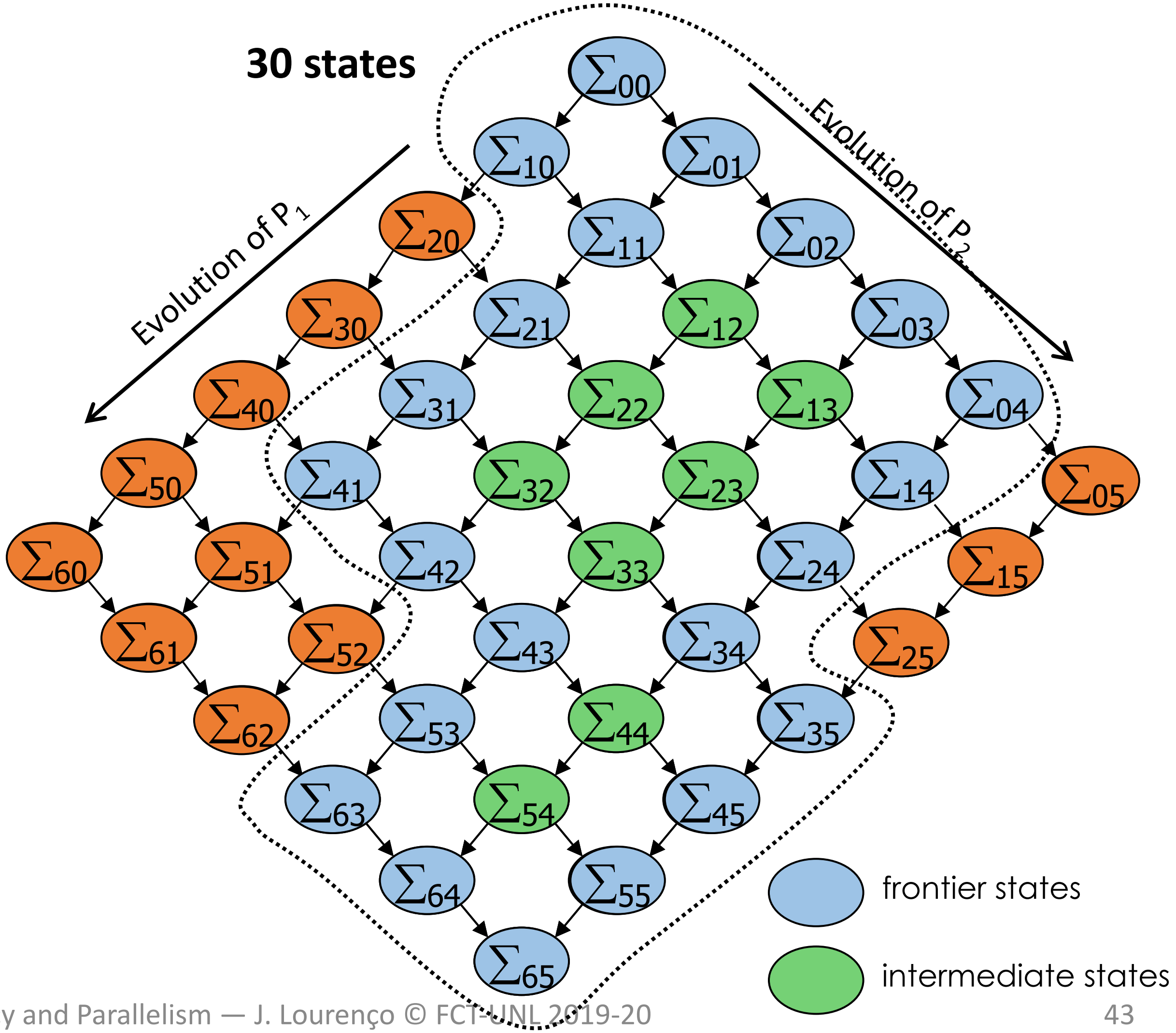
State explosion in concurrent programs



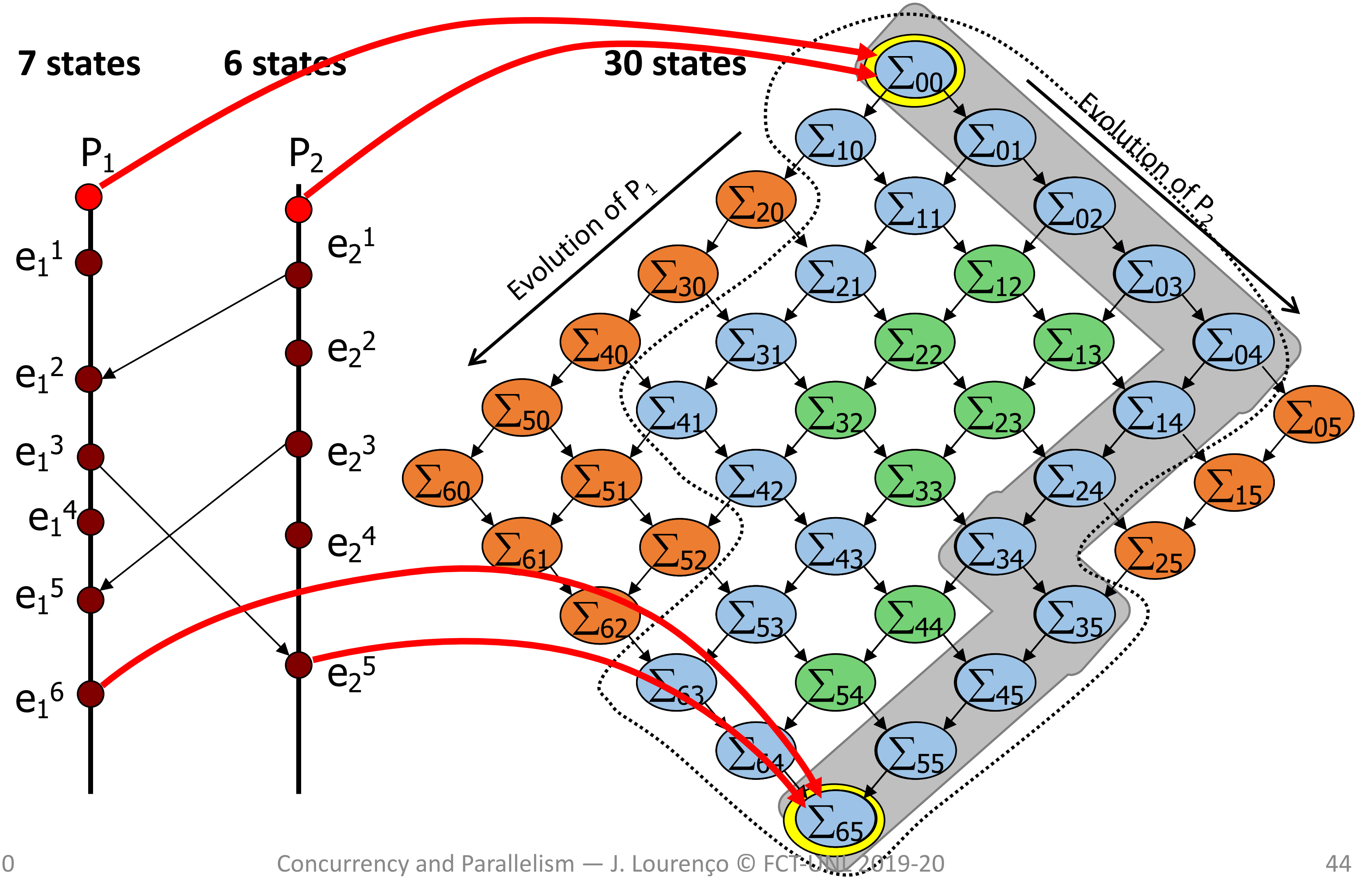


State explosion in concurrent programs



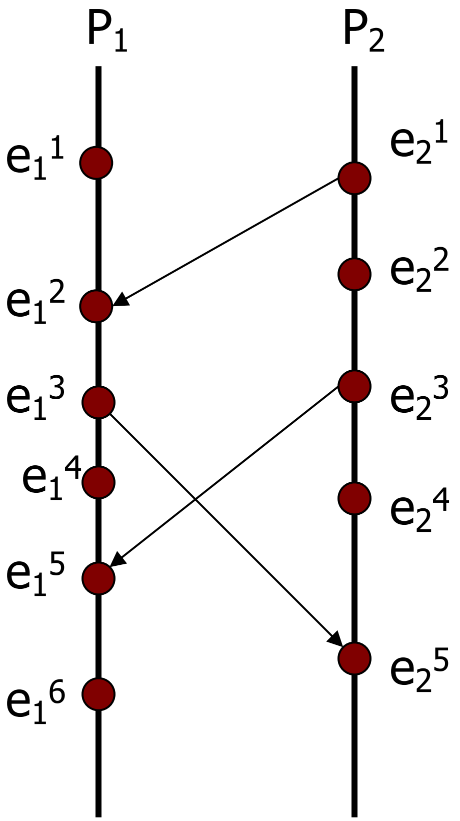

30 states
Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}
frontier states
intermediate states

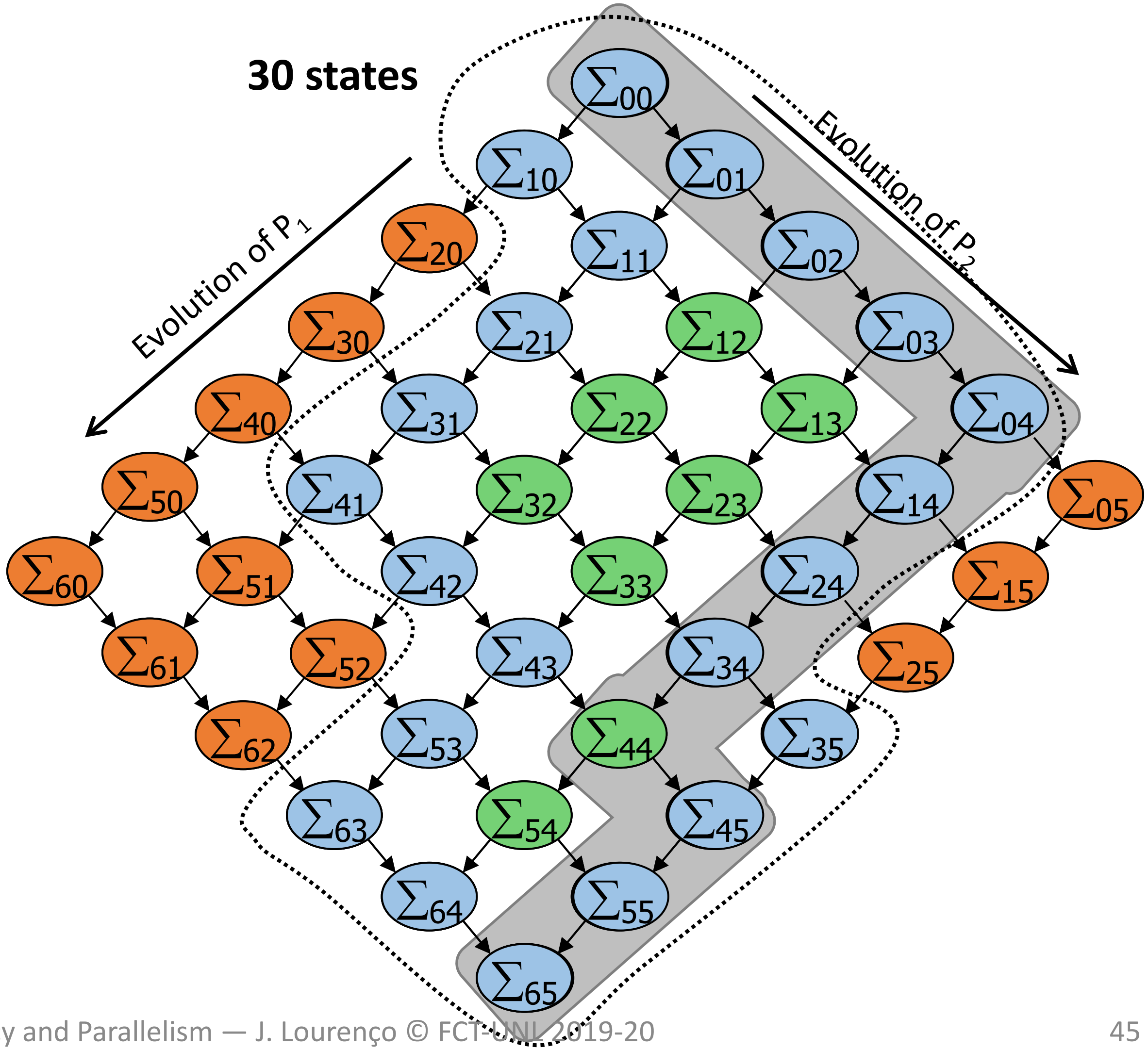
Consistent run: valid path Σ_{00} to Σ_{65}



Consistent run

7 states **6 states** **30 states**

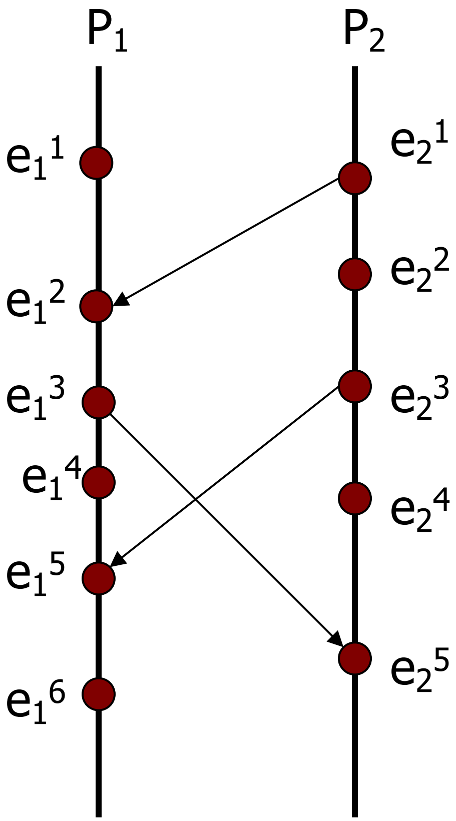


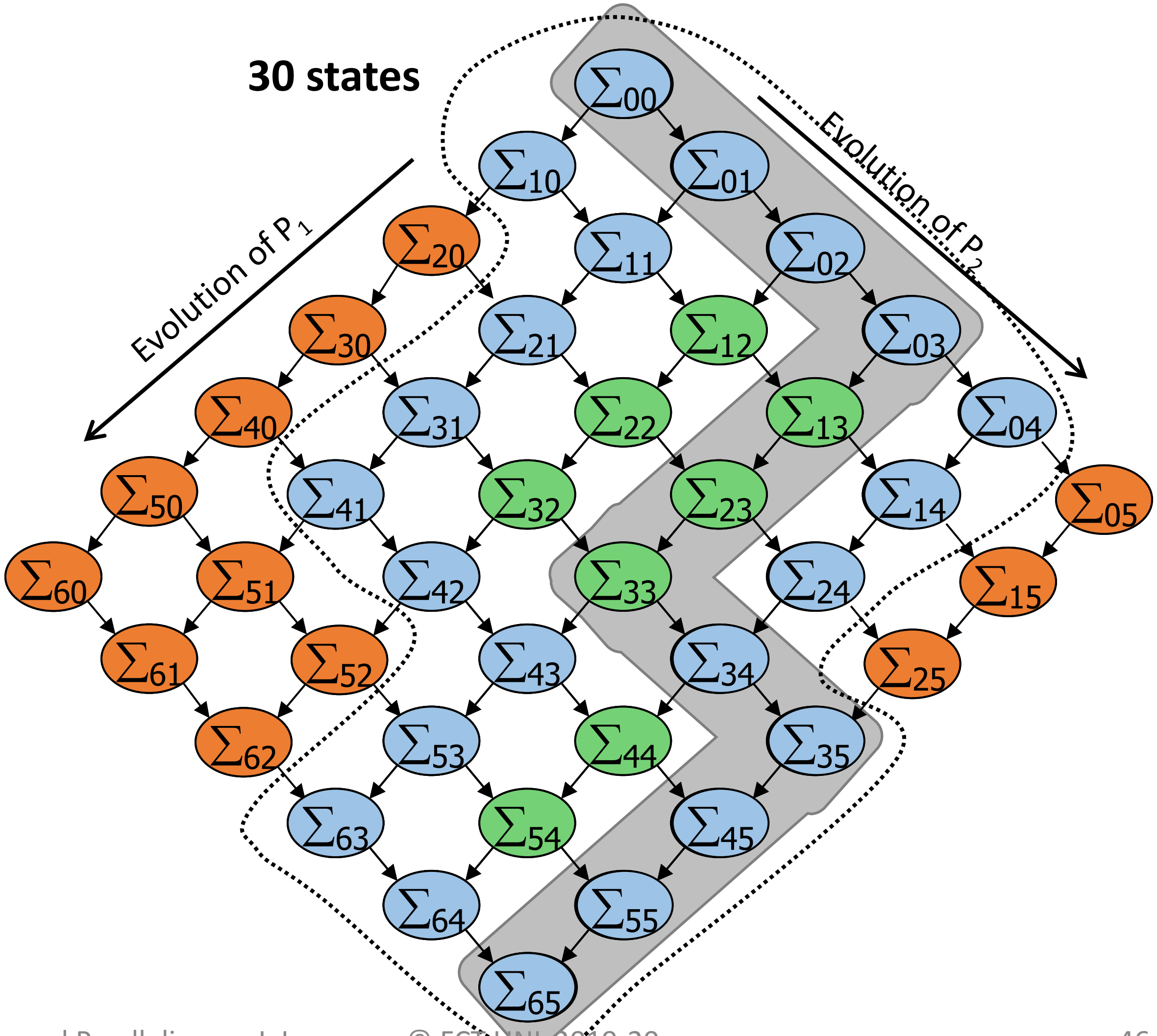


Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}

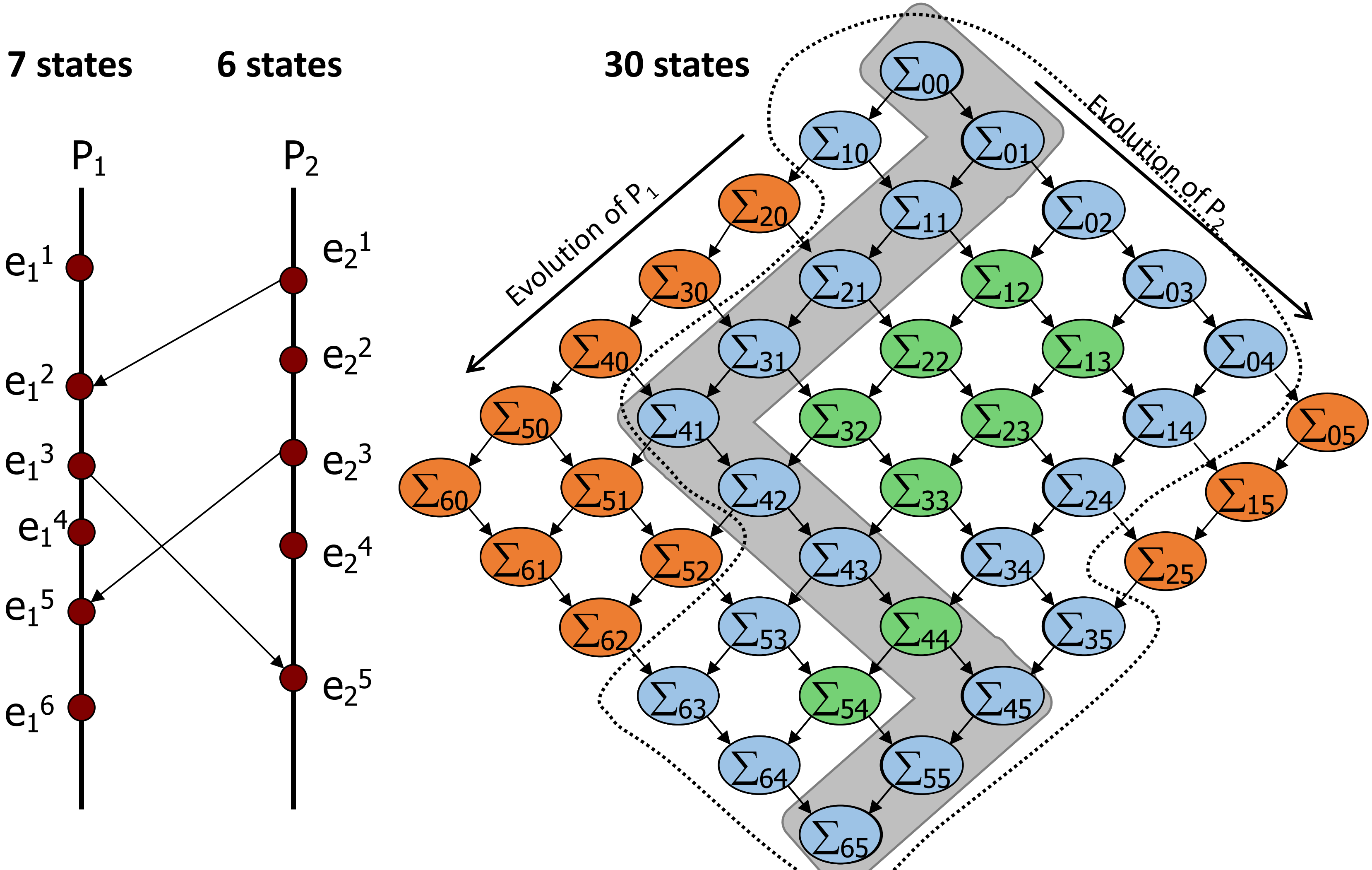
Consistent run

7 states **6 states** **30 states**



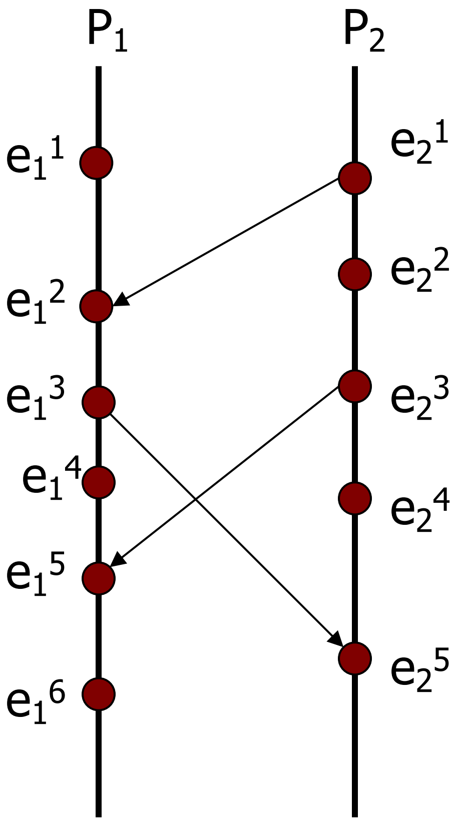


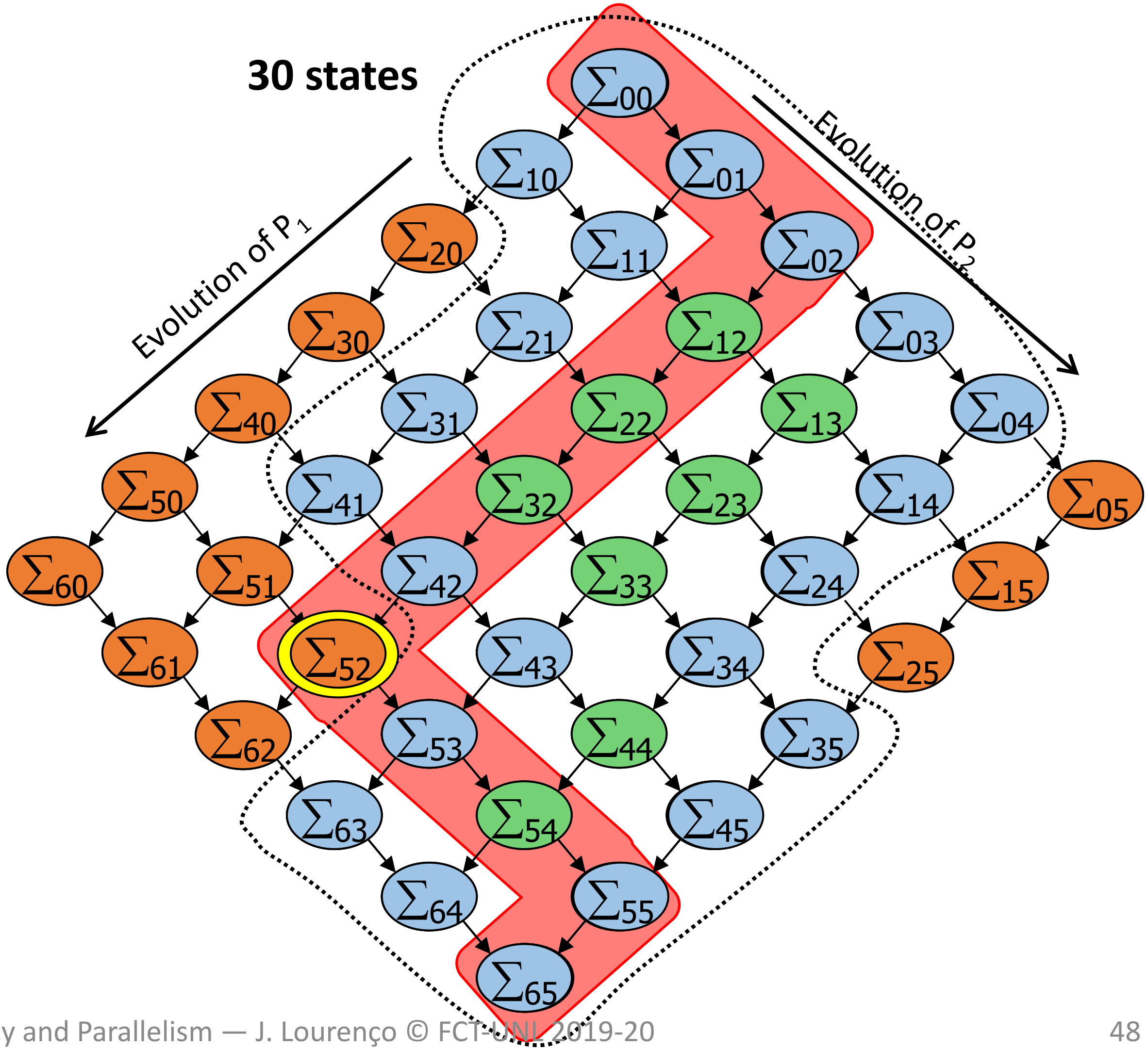
Consistent run



Inconsistent run – program error

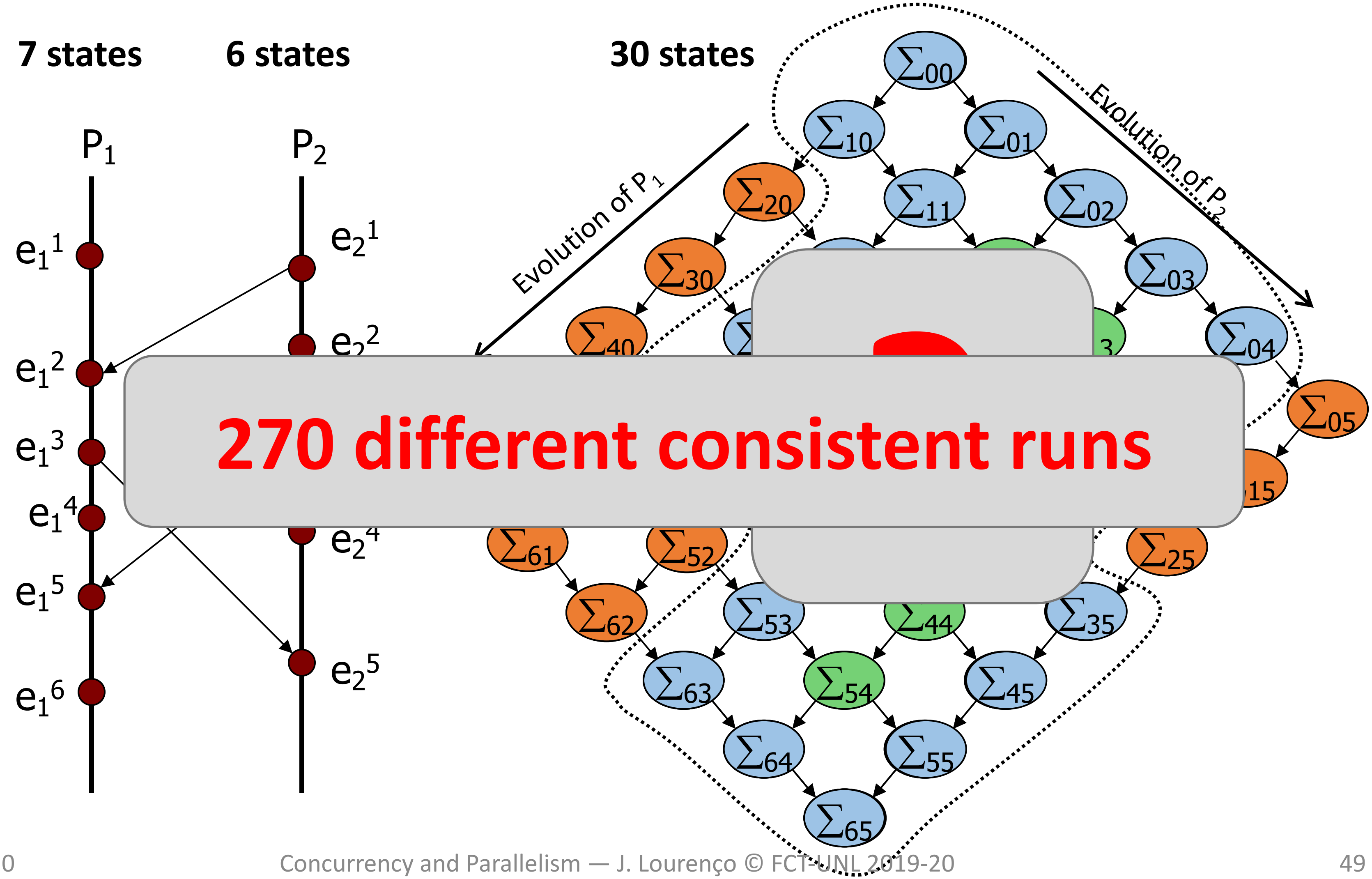
7 states **6 states** **30 states**





Evolution of P_1
Evolution of P_2
Σ_{00}
Σ_{10} Σ_{01}
Σ_{20} Σ_{11} Σ_{02}
Σ_{30} Σ_{21} Σ_{12} Σ_{03}
Σ_{40} Σ_{31} Σ_{22} Σ_{13} Σ_{04}
Σ_{50} Σ_{41} Σ_{32} Σ_{23} Σ_{14} Σ_{05}
Σ_{60} Σ_{51} Σ_{42} Σ_{33} Σ_{24} Σ_{15}
Σ_{61} Σ_{52} Σ_{43} Σ_{34} Σ_{25}
Σ_{62} Σ_{53} Σ_{44} Σ_{35}
Σ_{63} Σ_{54} Σ_{45}
Σ_{64} Σ_{55}
Σ_{65}

Consistent runs – How many?



270 different consistent runs

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

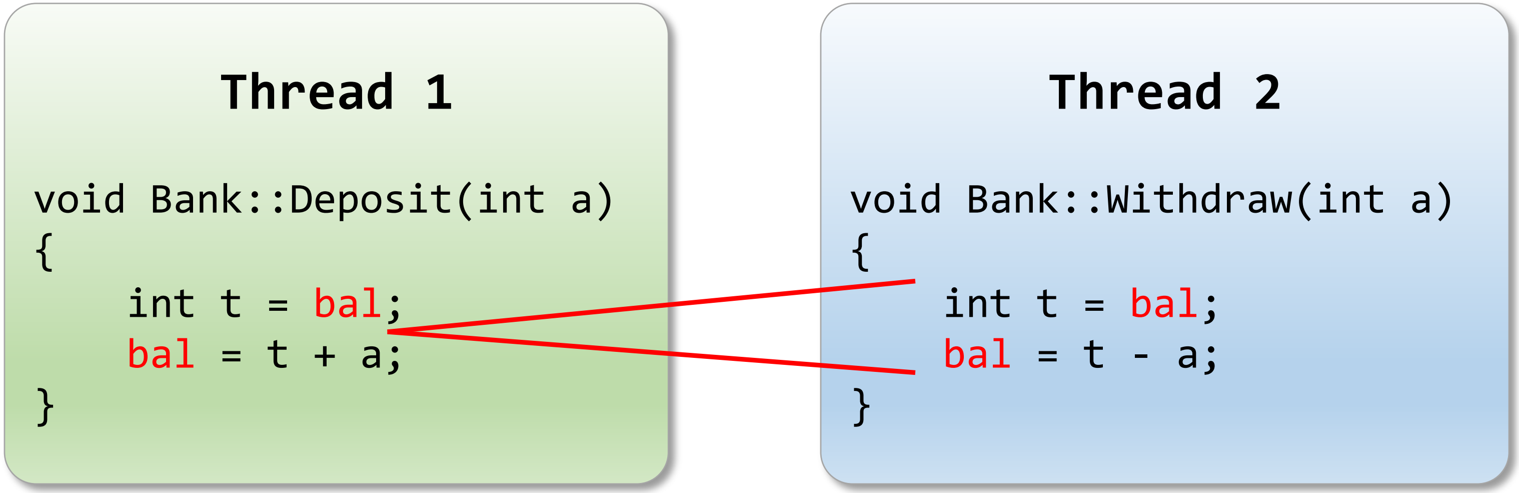
```
void Bank::Deposit(int a)
{
    int t = bal;
    bal = t + a;
}
```

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



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;
}
```

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

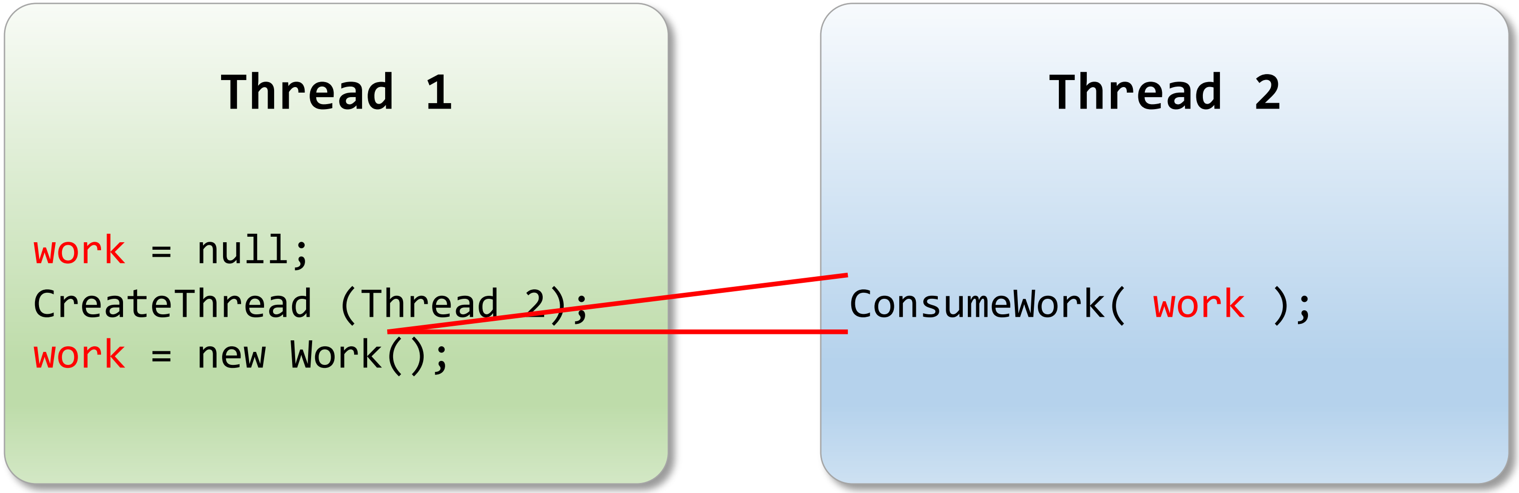
```
work = null;
CreateThread (Thread 2);
work = new Work();
```

Thread 2

```
ConsumeWork( work );
```

Ordering Violation

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



Thread 1

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

Thread 2

```
ConsumeWork( work );
```

'work' is not initialized yet!

Unintended Sharing

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

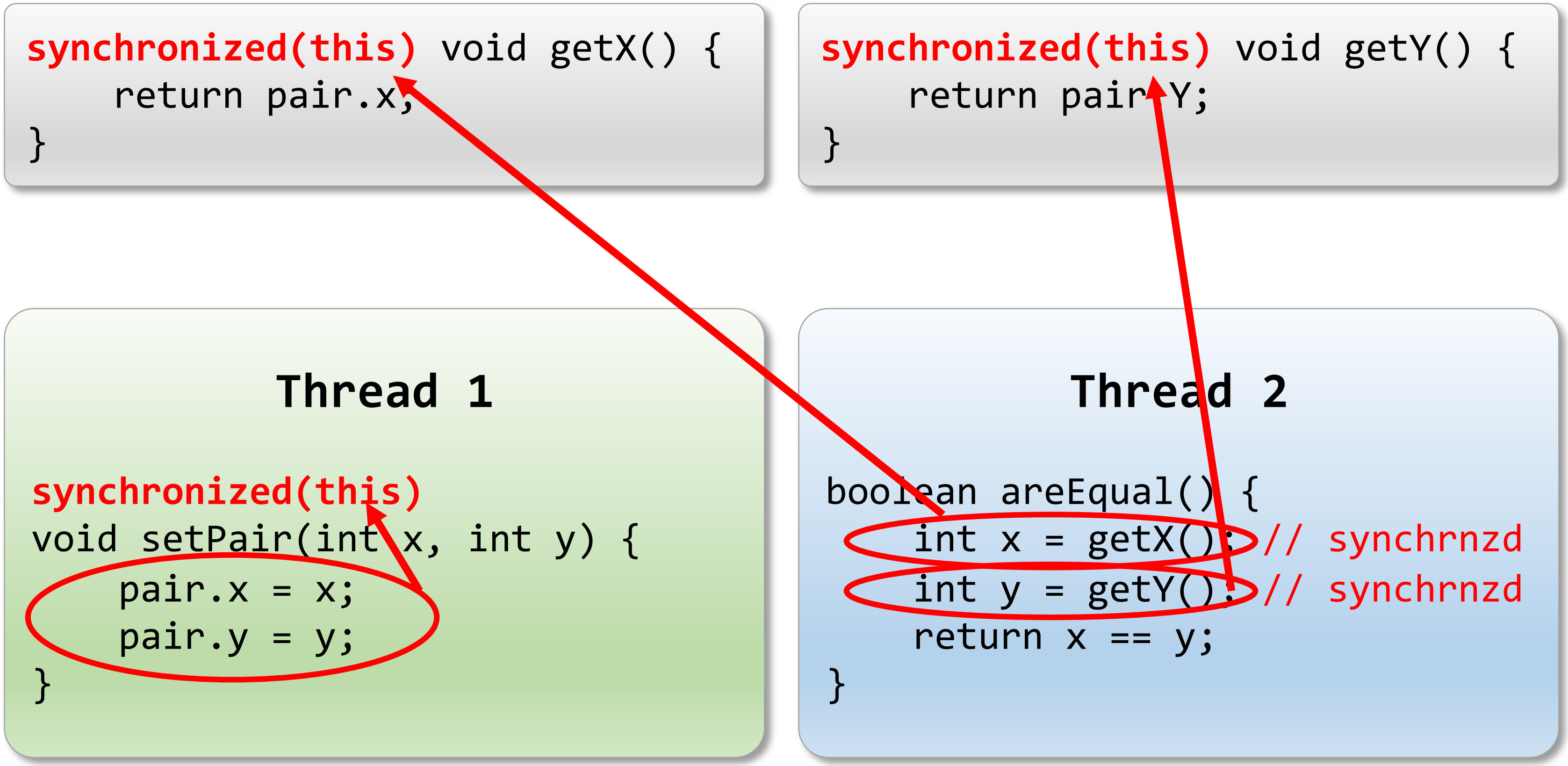
```
void work() {
   static int local = 0;
   …
   local += …
   …
}
```

```
Thread 1
…
work()
…
```

```
Thread 2
…
work()
…
```

High-Level Data Race

- Wrongly defined atomic blocks



```
synchronized(this) void getX() {
    return pair.x;
}
```

```
synchronized(this) void getY() {
    return pair.Y;
}
```

Thread 1

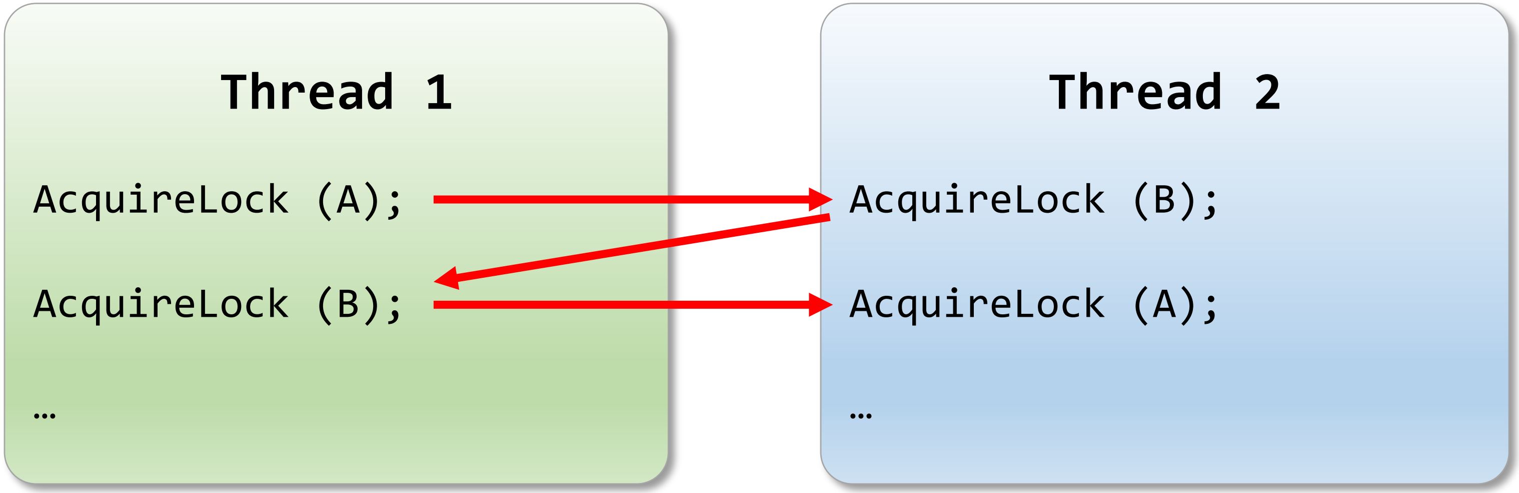
```
synchronized(this)
void setPair(int x, int y) {
    pair.x = x;
    pair.y = y;
}
```

Thread 2

```
boolean areEqual() {
    int x = getX(); // synchrnzd
    int y = getY(); // synchrnzd
    return x == y;
}
```

Deadlock

- Processes are waiting forever for each other



Thread 1

```
AcquireLock (A);

AcquireLock (B);

…
```

Thread 2

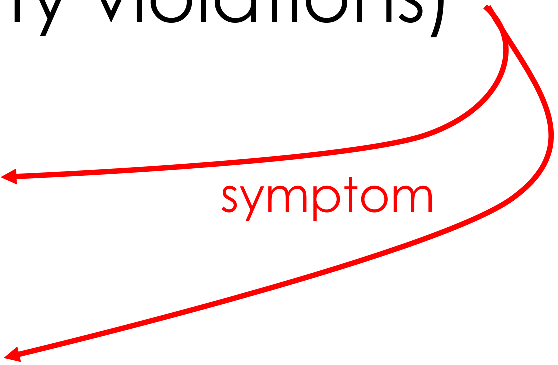
```
AcquireLock (B);

AcquireLock (A);

…
```

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

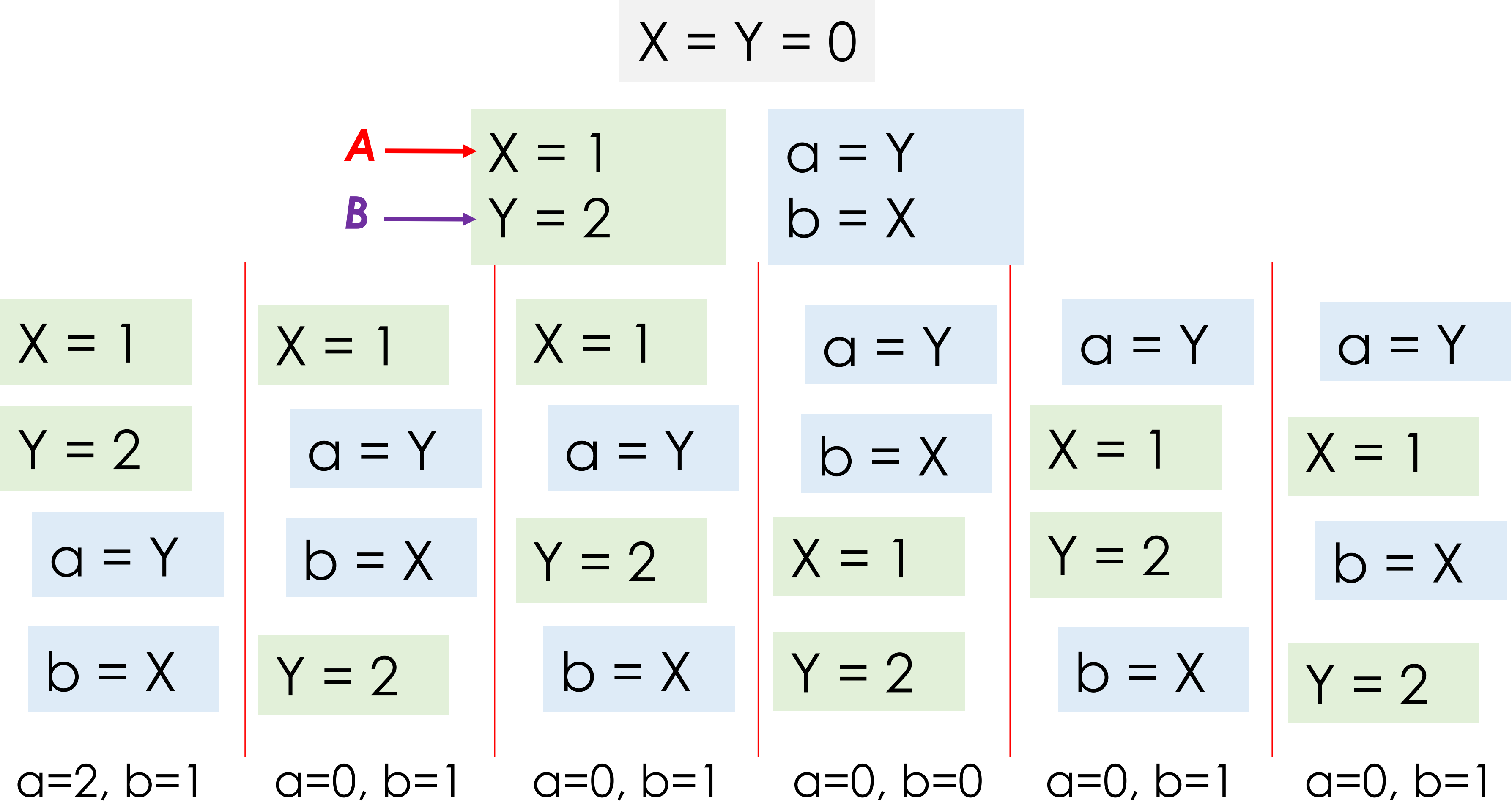
- Which means:
  - They access the same memory location
  - At least one is an update (write)

- 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



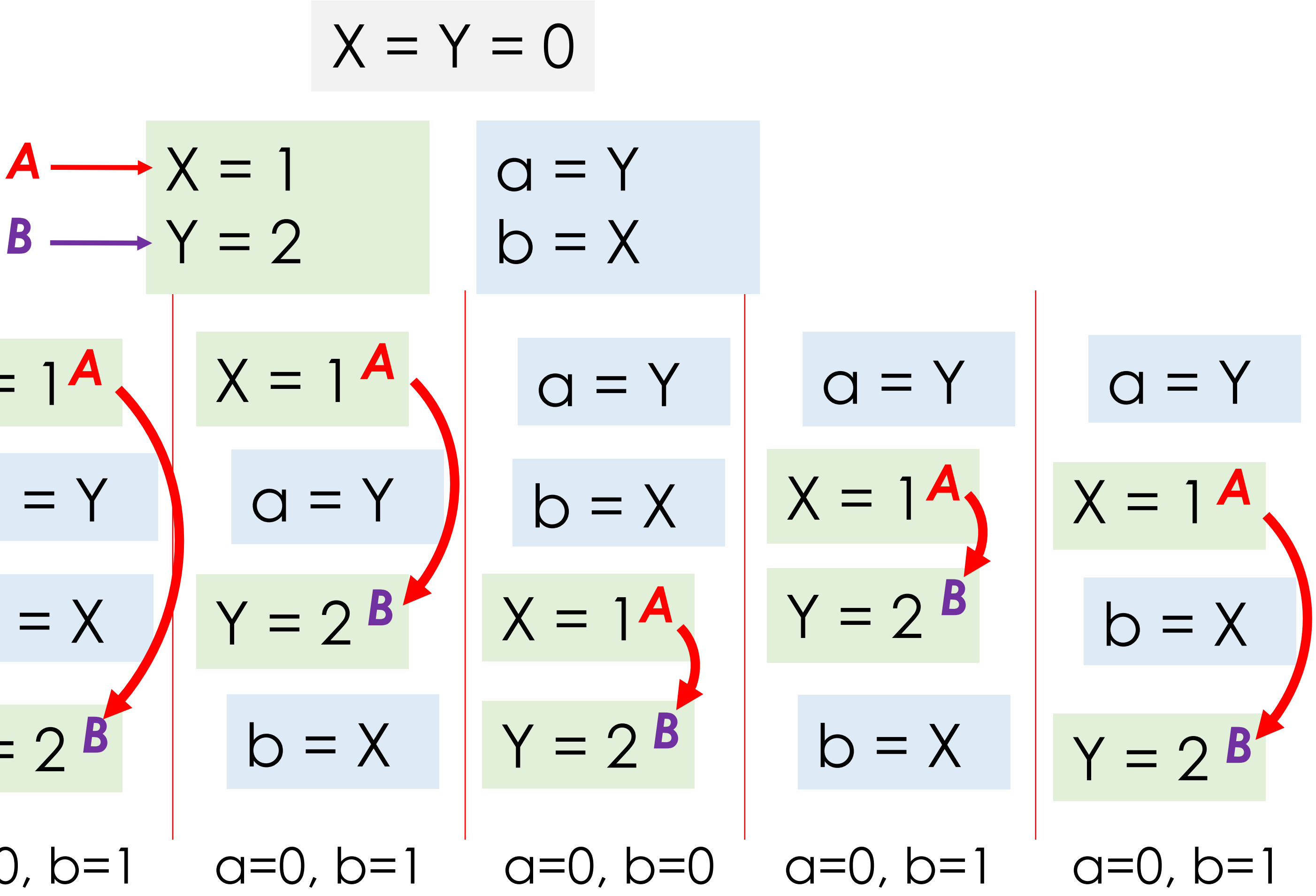
X = Y = 0

A → X = 1
B → Y = 2

a = Y
b = X

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

Assigning Semantics to Concurrent Programs



X = Y = 0

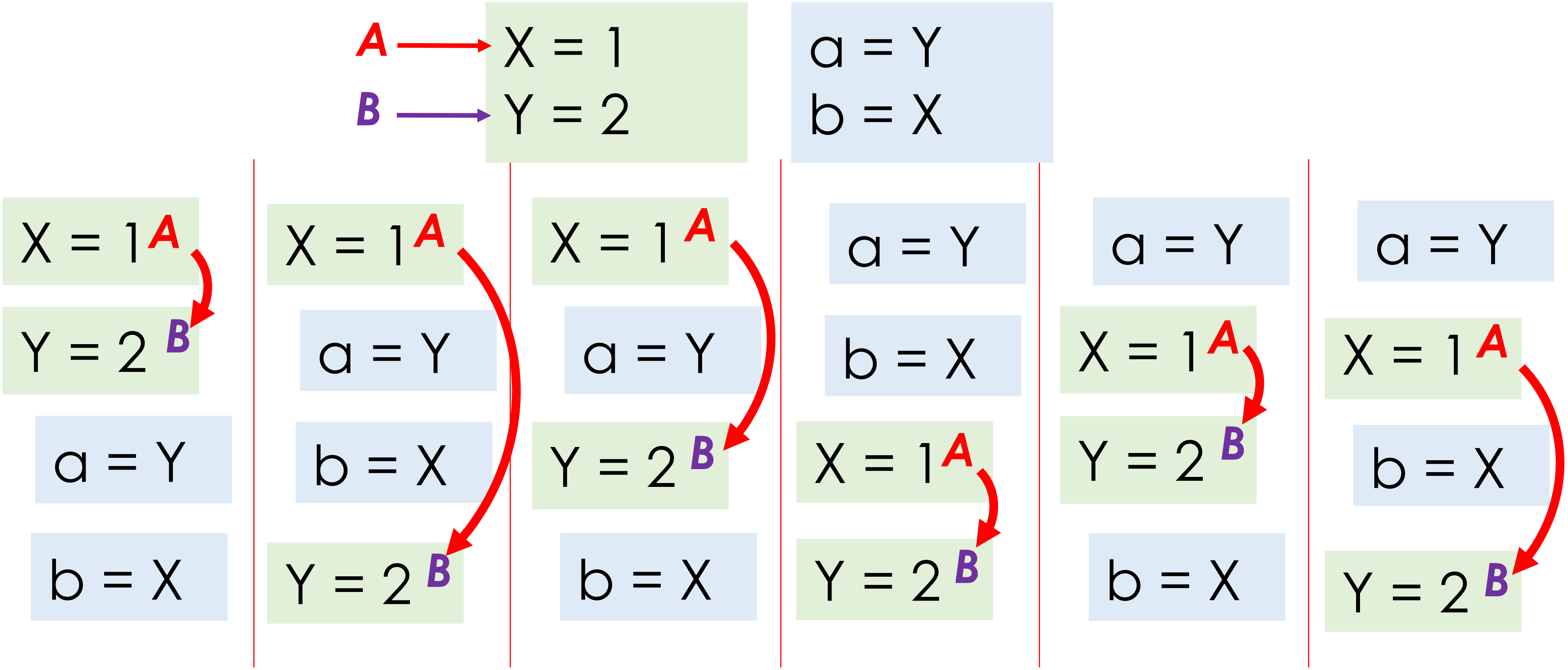
A → X = 1
B → Y = 2

a = Y
b = X

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

Assigning Semantics to Concurrent Programs

X = Y = 0



A → X = 1
B → Y = 2

a = Y
b = X

X = 1 A
Y = 2 B
a = Y
b = X

X = 1 A
a = Y
b = X
Y = 2 B

X = 1 A
a = Y
Y = 2 B
b = X

a = Y
b = X
X = 1 A
Y = 2 B

a = Y
X = 1 A
Y = 2 B
b = X

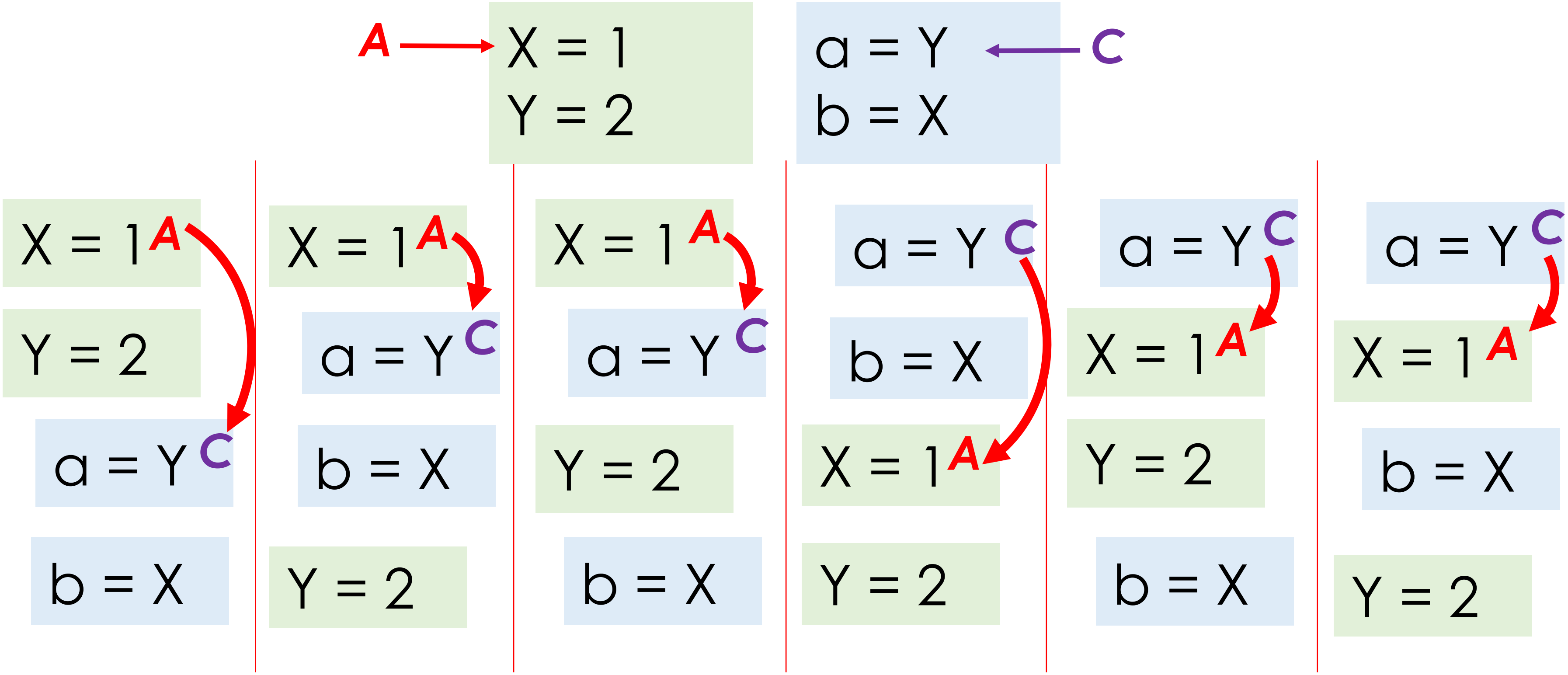
a = Y
X = 1 A
b = X
Y = 2 B

a=2, b=1	a=0, b=1	a=0, b=1	a=0, b=0	a=0, b=1	a=0, b=1

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

Assigning Semantics to Concurrent Programs

X = Y = 0



A → X = 1
Y = 2

a = Y ← C
b = X

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

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

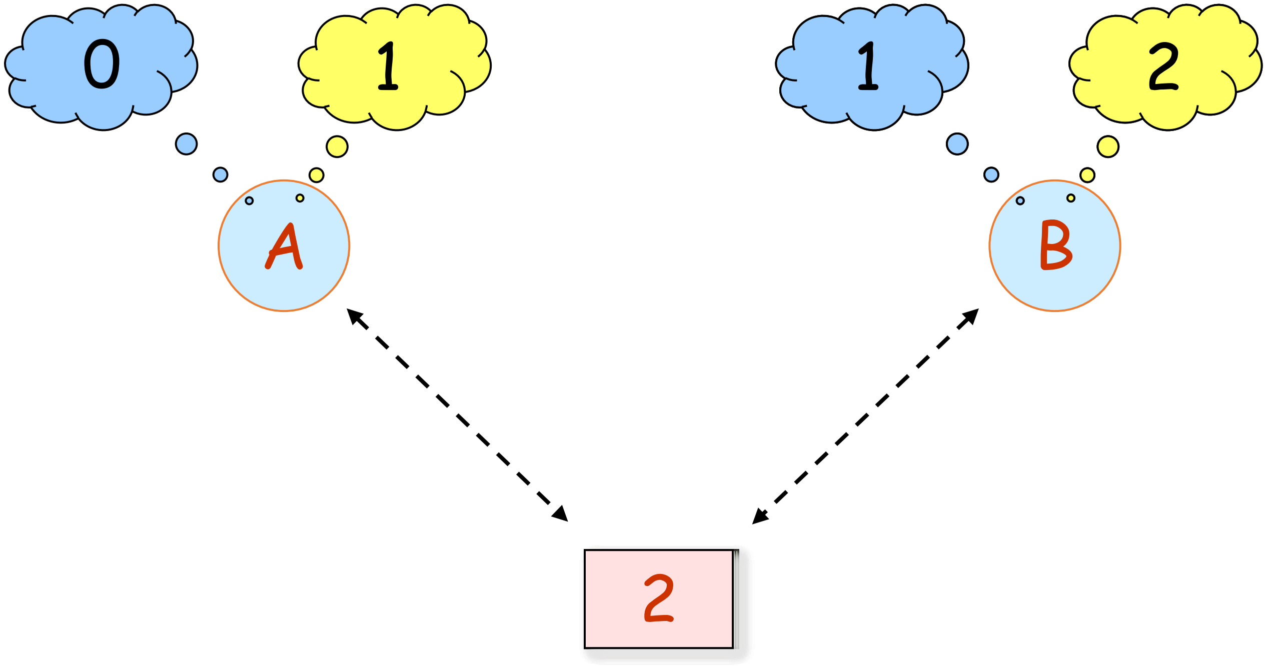
```
for (i = 0; i < 5;  i++) {
    x = x + 1
}
```

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