

1. Indicate whether the following statements are true or false. Justify your answers.

(a) In the dining-philosopher problem with five philosophers, if we allow at most four of them to be hungry simultaneously, deadlock may still occur.

(b) It is impossible to have a deadlock involving only one single process.

(c) If a resource allocation graph contains a cycle, then a deadlock has occurred.

1

1b) It is impossible to have a deadlock involving only one single process.

→ True.

Justification: The hold-and-wait condition can never be satisfied.

3

1a) In the dining-philosopher problem with five philosophers, if we allow at most four of them to be hungry simultaneously, deadlock may still occur.

→ False.

Justification: Deadlock does not occur, since at least one philosopher can get both chopsticks.

2

1c) If a resource allocation graph contains a cycle, then a deadlock has occurred.

→ False.

Justification: It depends on the number of instances per resource type.

4

2. Use resource-allocation graphs to model the following situations, and determine if deadlock occurs in each case. There are three resource types, R, S, and T, each having a single instance.

Case 1

P1 requests R
P2 requests T
P1 requests S
P2 requests S
P1 releases R
P1 releases S

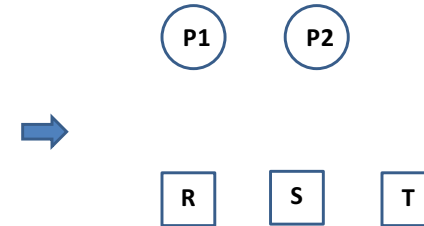
Case 2

P1 requests R
P2 requests T
P1 requests S
P2 requests S
P1 requests T

Case 1: Resource Graph

Case 1

P1 requests R
P2 requests T
P1 requests S
P2 requests S
P1 releases R
P1 releases S



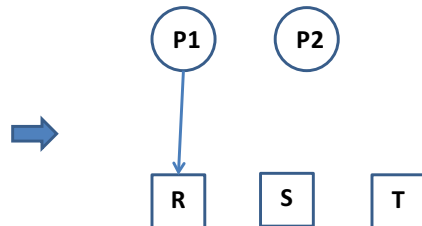
5

6

Case 1: Resource Graph

Case 1

→ P1 requests R
P2 requests T
P1 requests S
P2 requests S
P1 releases R
P1 releases S

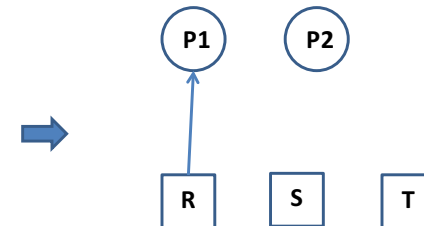


7

Case 1: Resource Graph

Case 1

→ P1 requests R
P2 requests T
P1 requests S
P2 requests S
P1 releases R
P1 releases S

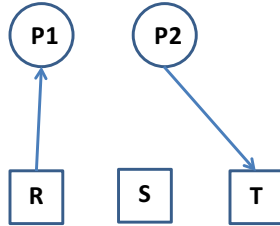


8

Case 1: Resource Graph

Case 1

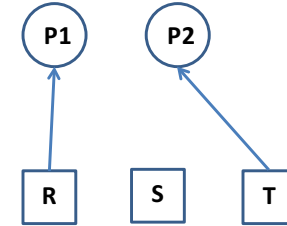
P1 requests R
 ➔ P2 requests T
 P1 requests S
 P2 requests S
 P1 releases R
 P1 releases S



Case 1: Resource Graph

Case 1

P1 requests R
 ➔ P2 requests T
 P1 requests S
 P2 requests S
 P1 releases R
 P1 releases S



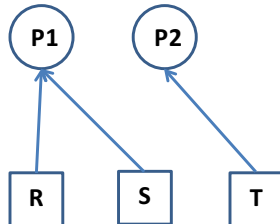
9

10

Case 1: Resource Graph

Case 1

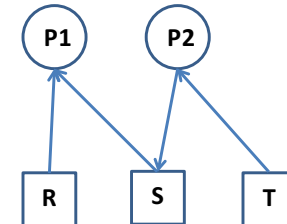
P1 requests R
 P2 requests T
 ➔ P1 requests S
 P2 requests S
 P1 releases R
 P1 releases S



Case 1: Resource Graph

Case 1

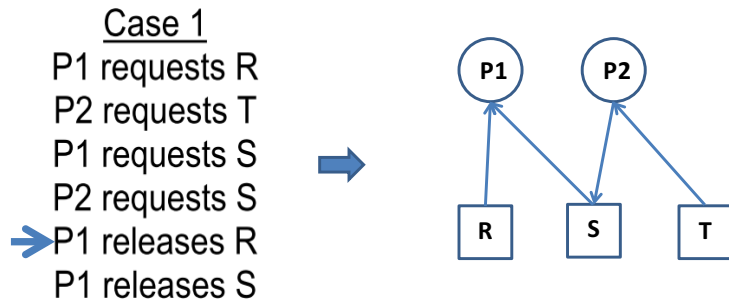
P1 requests R
 P2 requests T
 P1 requests S
 ➔ P2 requests S
 P1 releases R
 P1 releases S



11

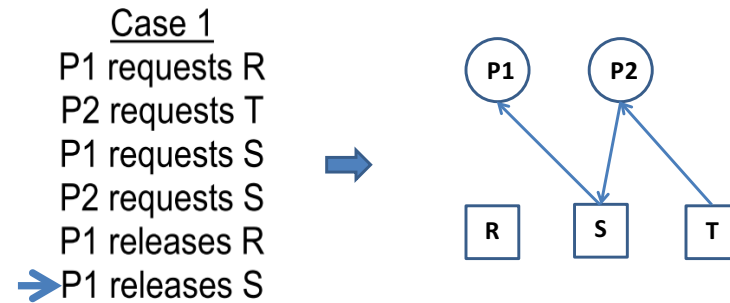
12

Case 1: Resource Graph



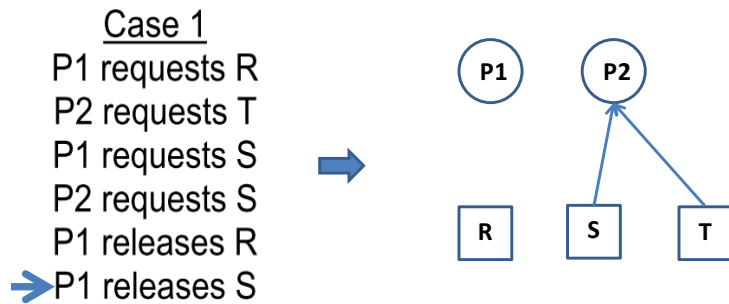
13

Case 1: Resource Graph



14

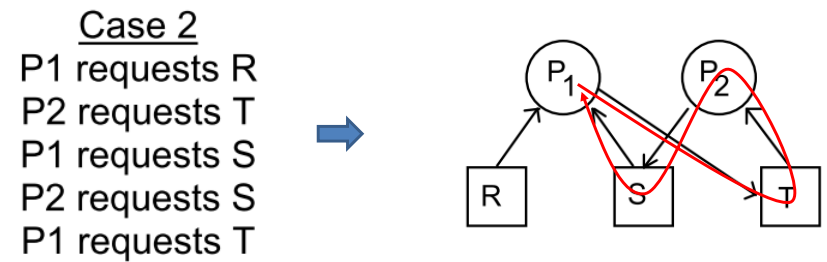
Case 1: Resource Graph



No deadlock.

15

Case 2: Resource Graph



Deadlock!

16

If PROCESS 4 requests for one more unit...

3. A resource-allocation state is given below. Assume Available = 2

| | Allocation | Max | Need |
|-----------|------------|-----|------|
| PROCESS 1 | 1 | 6 | 5 |
| PROCESS 2 | 1 | 5 | 4 |
| PROCESS 3 | 2 | 4 | 2 |
| PROCESS 4 | 4 | 7 | 3 |

(a) If PROCESS 4 requests for one more unit of the resource, does this lead to a safe state or an unsafe one?

(b) If PROCESS 3 requests for one more unit of the resource, does this lead to a safe state or an unsafe one?

Available = 2



| | Allocation | Max | Need |
|-----------|------------|-----|------|
| PROCESS 1 | 1 | 6 | 5 |
| PROCESS 2 | 1 | 5 | 4 |
| PROCESS 3 | 2 | 4 | 2 |
| PROCESS 4 | 5 | 7 | 2 |

Available = 1

→ unsafe, since $Need_i > Available$ for all i .

17

18

If PROCESS 3 requests for one more unit...

| | Allocation | Max | Need |
|-----------|------------|-----|------|
| PROCESS 1 | 1 | 6 | 5 |
| PROCESS 2 | 1 | 5 | 4 |
| PROCESS 3 | 2 | 4 | 2 |
| PROCESS 4 | 4 | 7 | 3 |

Available = 2



| | Allocation | Max | Need |
|-----------|------------|-----|------|
| PROCESS 1 | 1 | 6 | 5 |
| PROCESS 2 | 1 | 5 | 4 |
| PROCESS 3 | 3 | 4 | 1 |
| PROCESS 4 | 5 | 7 | 2 |

Available = 1

→ Safe, since sequence <PROCESS 3, PROCESS 2, PROCESS 1, PROCESS 4> satisfies safety requirement.

4. Consider the following snapshot of a system's state, with four processes (P0, P1, P2 and P3) and three resource types (A, B and C). The current Allocation, Need and Available matrices are shown in the below table. Compute the minimum value for x , so that this system state is safe. Justify your answer.

| Process | Allocation | Available | | |
|---------|------------|-----------|---|-----|
| | | A | B | C |
| | | 0 | 1 | x |
| | | Need | | |
| | | A | B | C |
| P0 | 2 1 1 | 0 | 1 | 0 |
| P1 | 1 1 0 | 2 | 1 | 2 |
| P2 | 1 1 1 | 2 | 0 | 1 |
| P3 | 1 1 1 | 4 | 1 | 0 |

19

20

The minimum value of x is 0 with the safe sequence <P0,P2,P1,P3>.

Running the safety algorithm, Available matrix evolves as follows.

| <u>Available</u> | <u>Need</u> | <u>Allocation</u> | <u>Process completion</u> |
|------------------|-------------|-------------------|---------------------------|
| | A B C | A B C | |
| 0 1 x=0 | 0 1 0 | 2 1 1 | P0 |
| 2 2 1 | 2 0 1 | 1 1 1 | P2 |
| 3 3 2 | 2 1 2 | 1 1 0 | P1 |
| 4 4 2 | 4 1 0 | 1 1 1 | P3 |
| 5 5 3 | | | |