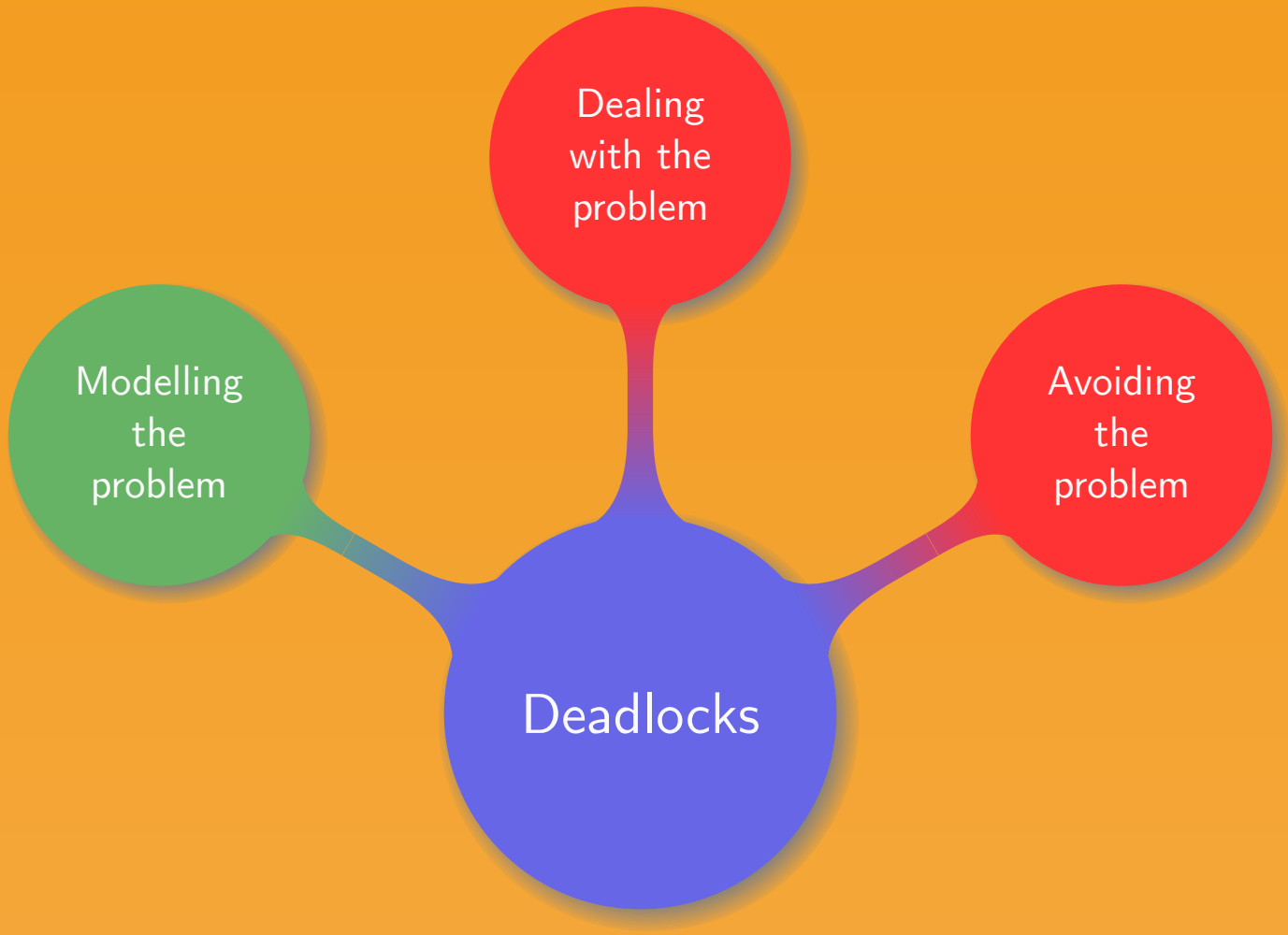
Introduction to Operating Systems

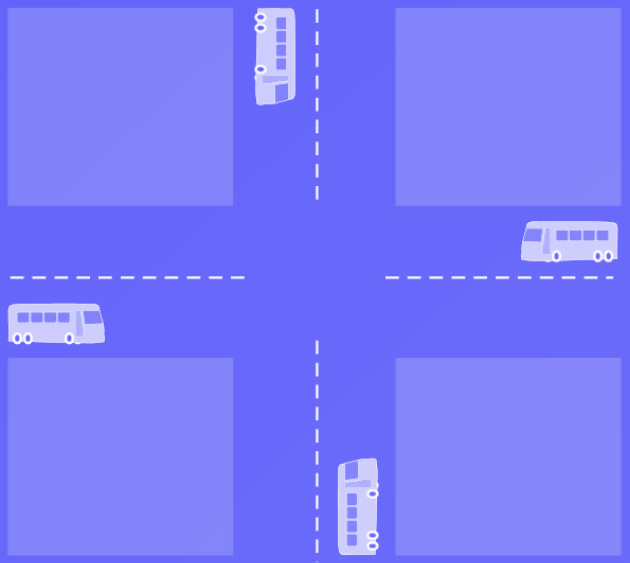
5. Deadlocks

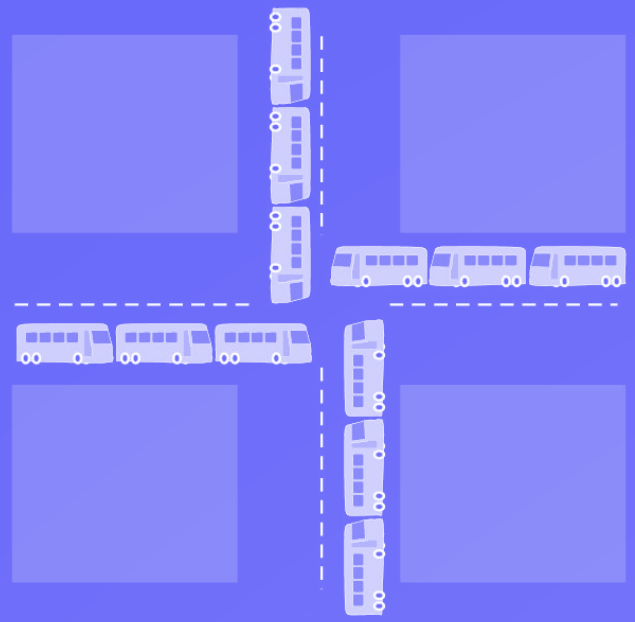
Manuel – Fall 2019

Chapter organisation



What is a deadlock?





A typical deadlock

Simple example:

- Process A needs to access resource R
- Process B needs to access resource S
- Process A wants to access resource S
- Process B wants to access resource R

A typical deadlock

Simple example:

- Process A needs to access resource R
- Process B needs to access resource S
- Process A wants to access resource S
- Process B wants to access resource R

Problem: neither A nor B releases the resources, both A and B will wait indefinitely

Types of resources

Preemptable

(Resources that can be safely taken away from a process)

- Total memory: 256 MB
- Processes A and B, 256MB each
- A loaded in memory and acquires the printer
- A exceeds its quantum
- B loaded in memory and tries to acquire the printer
- B fails, but has the memory
- Memory given to A
- A finishes its printing

Non-preemptable

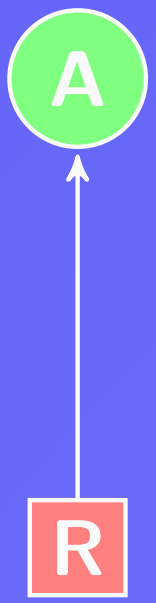
(Resources that cannot be safely taken away from a process)

- A is burning a DVD
- B wants the DVD
- DVD drive is not accessible
- Resource cannot be taken away from A

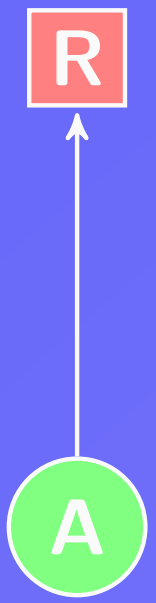
Representing deadlocks

Resource allocation represented using graphs:

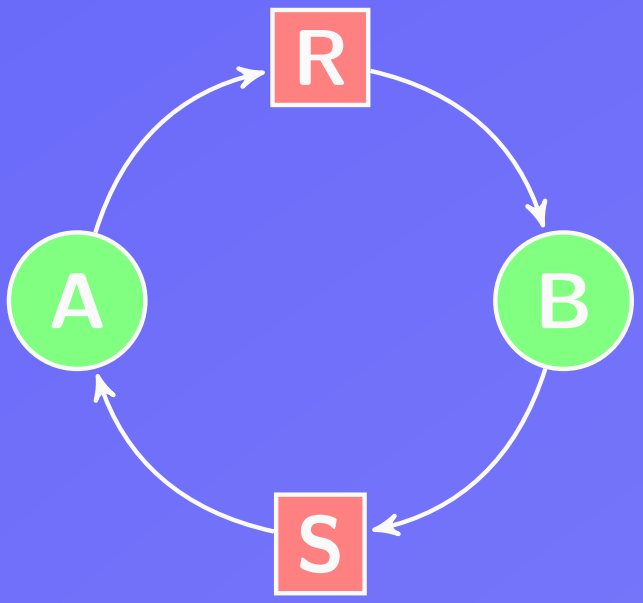
Resource R held by A



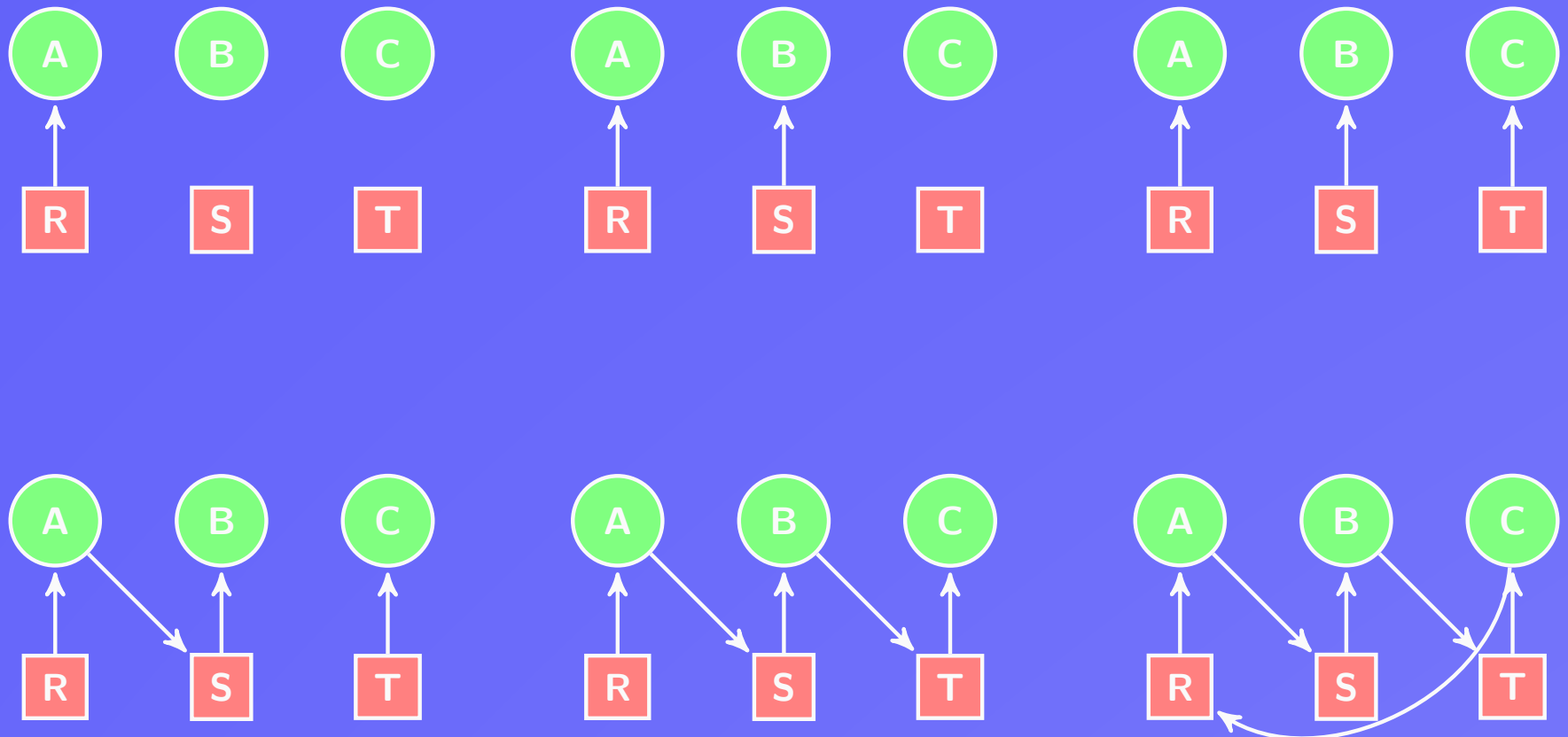
Resource R requested by A



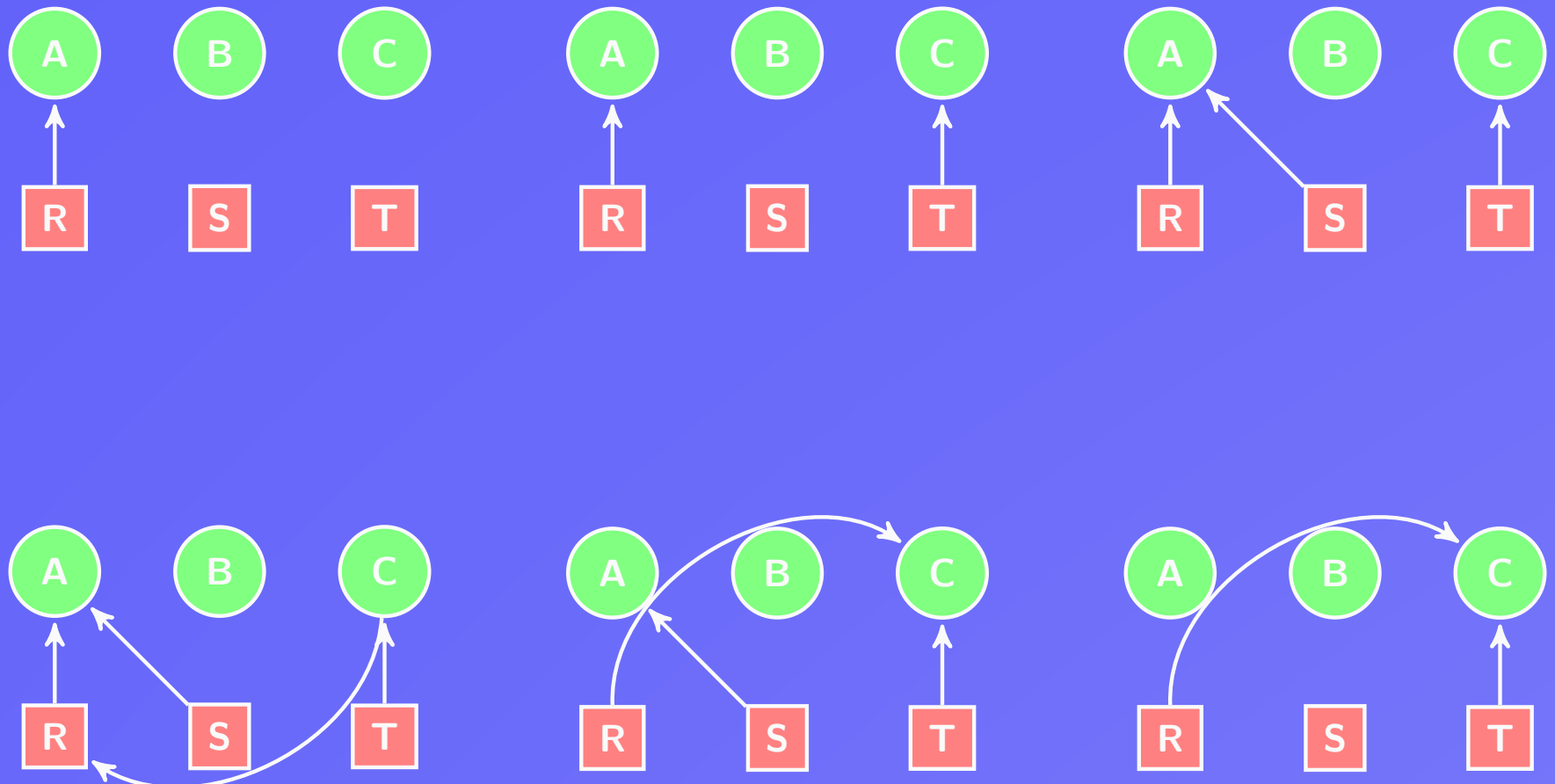
Deadlock



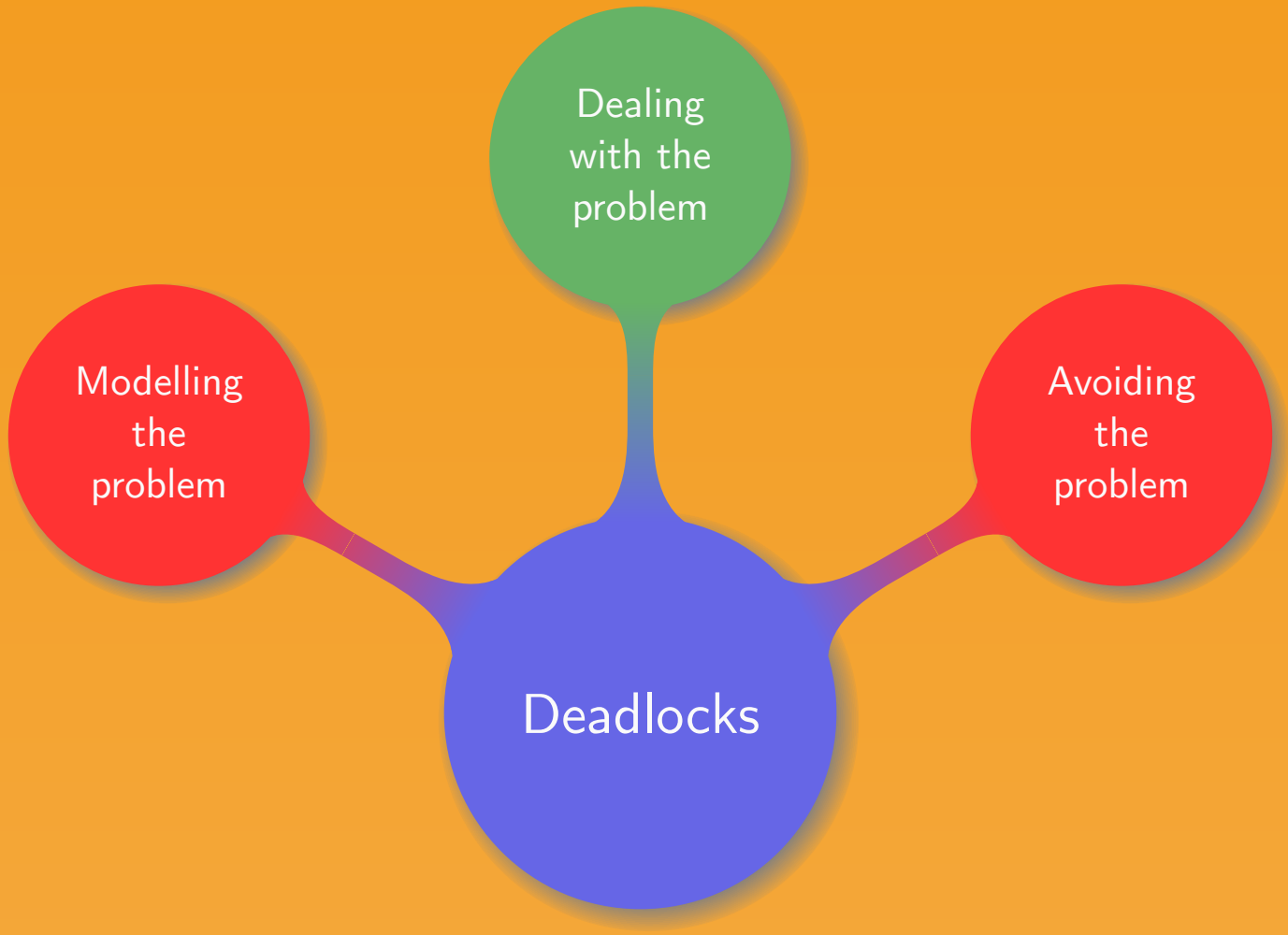
Example – Deadlock



Example – No deadlock



Chapter organisation



Ostrich algorithm

ALERT, ALERT

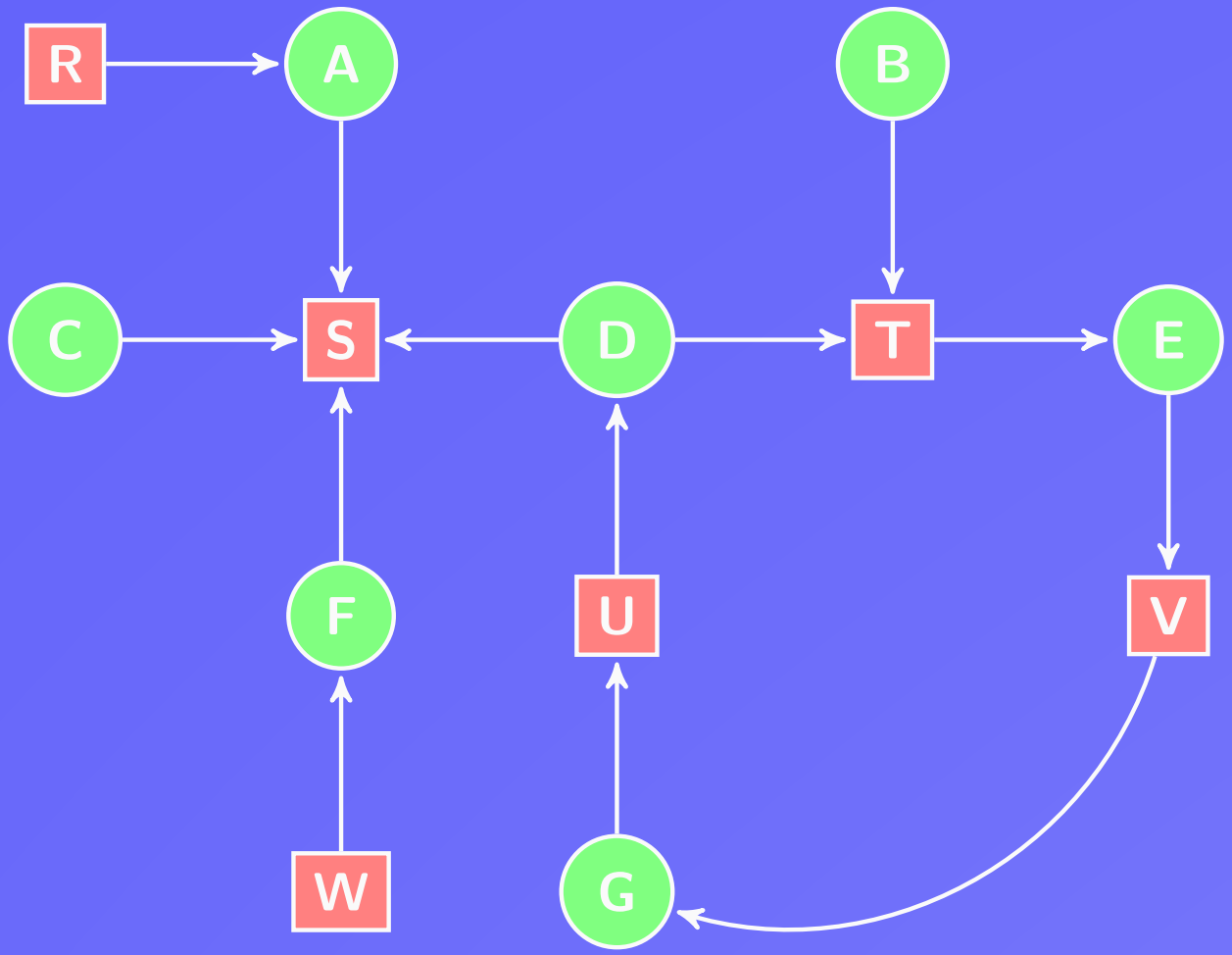
HIDE!!!!

Deadlock detection – Single resource

Example.
Is there any deadlock in the following system with seven processes (A–G) and six resources (R–W)?

- Process A holds R and wants S
- Process B wants T
- Process C want S
- Process D holds U and wants both S and T
- Process E holds T and wants V
- Process F holds W and wants S
- Process G holds V and wants U

Deadlock detection – Single resource



Deadlock detection – Multiple resources

Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 2 & 1 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

Deadlock detection – Multiple resources

Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 2 & 2 & 1 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Deadlock detection – Multiple resources

Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 2 & 2 & 1 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

Deadlock detection – Multiple resources

Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 2 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 3 & 0 & 2 & 1 \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

Deadlock detection – Multiple resources

Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 4 & 2 & 2 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

Deadlock detection – Multiple resources

Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 2 & 1 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

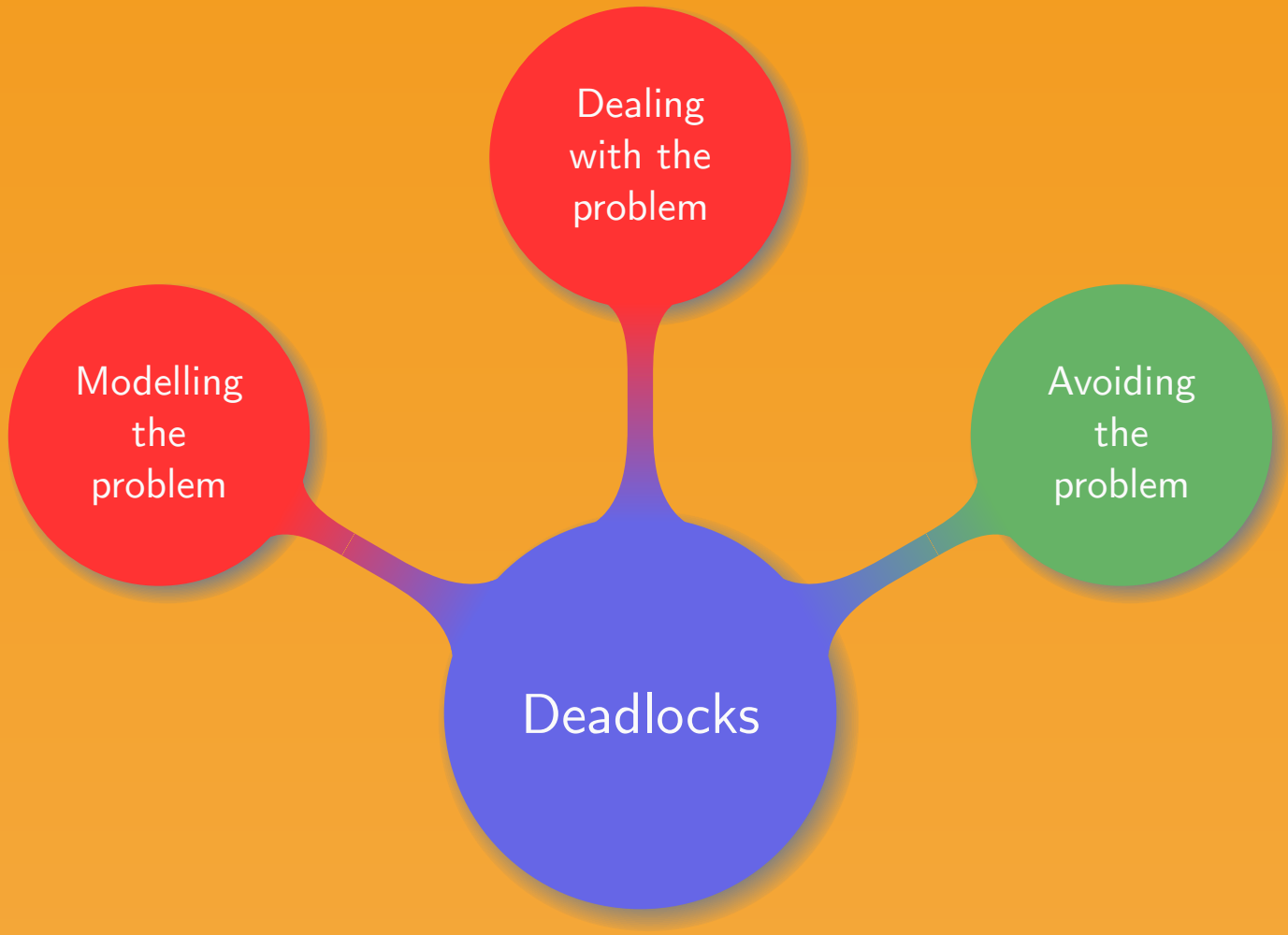
What if the second process requests two DVD burners?

Recovering from a deadlock

Three main recovery strategies:

- Preemption:
  - Take a resource from another process
  - Might require manual intervention (e.g. collect papers from printer, pile them up and resume printing later)
- Rollback:
  - Set periodical checkpoints on processes
  - Save process state at the checkpoints
  - Restart process at a checkpoint (from before the deadlock)
- Killing:
  - Simplest strategy
  - Kill a process that uses resources related to the deadlock
  - Pick a process that can be re-run from the beginning

Chapter organisation



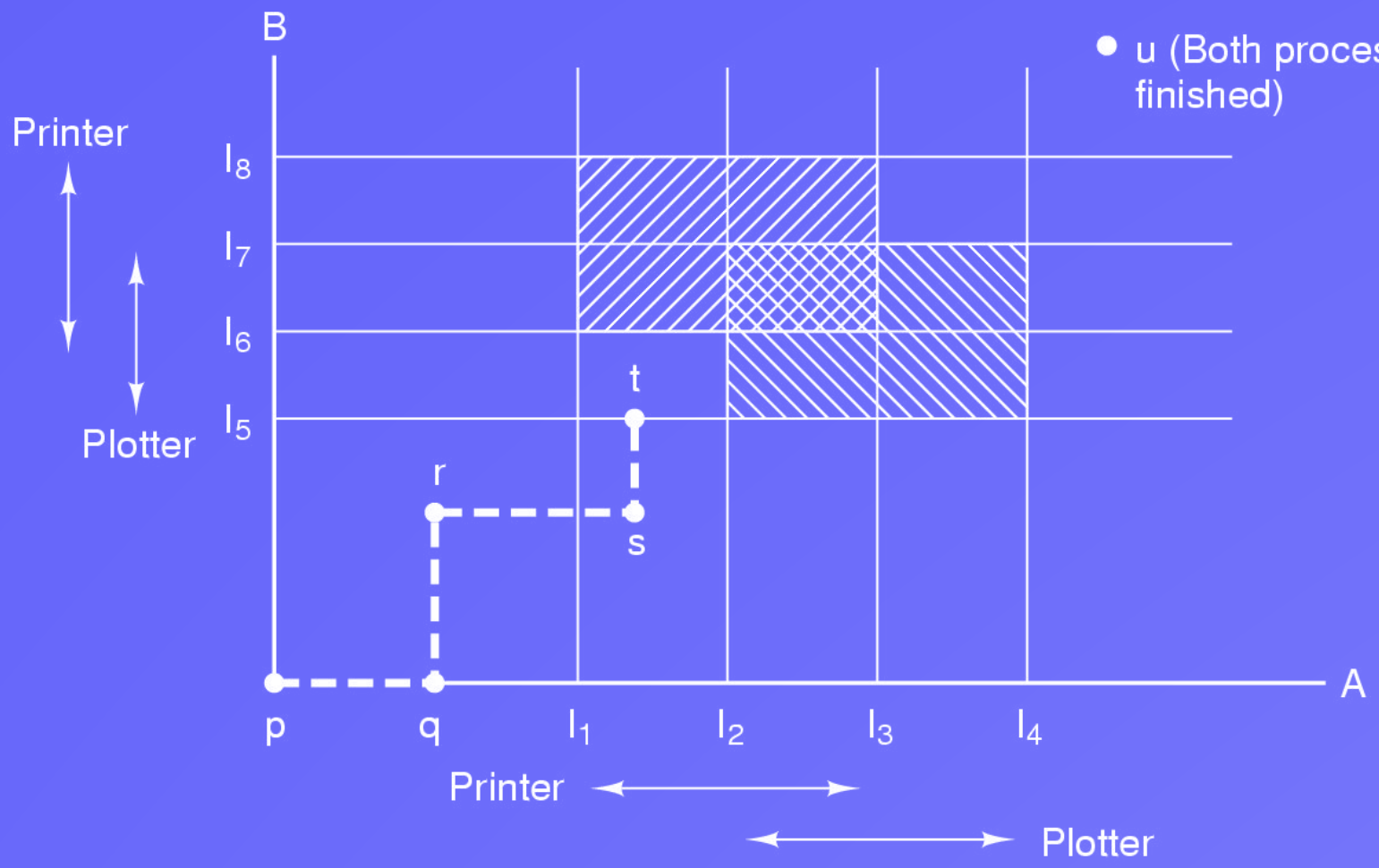
Avoidance vs. prevention

Two main strategies against deadlocks:

- Avoidance: resources are assigned and released one at a time.
  Is there any algorithm that can perform the right choice to avoid deadlocks?

- Prevention: when is a deadlock occurring?
  If it is possible to describe the circumstances under which deadlocks occur it might be possible to prevent them

Resources trajectories



Defining states

Using the matrices E, A, C and R, define:

- Safe state: there exists a scheduling order allowing all processes to complete, even if they suddenly all request their maximum number of resources. The system can guarantee that all processes can finish
- Unsafe state: the ability of the system not to deadlock depends on the order the resources are allocated/deallocated. There is no way to predict whether or not all the processes will finish

Defining states

Using the matrices E, A, C and R, define:

- Safe state: there exists a scheduling order allowing all processes to complete, even if they suddenly all request their maximum number of resources. The system can guarantee that all processes can finish
- Unsafe state: the ability of the system not to deadlock depends on the order the resources are allocated/deallocated. There is no way to predict whether or not all the processes will finish

Remark.
An unsafe state does not necessarily imply a deadlock; the system can still run for a while, or even complete all processes if some release their resources before requesting some more

The banker's algorithm

General idea:

- Dijkstra (1965)
- Based on the detection algorithm
- Idea: avoid deadlocks by avoiding to run into an unsafe state
- Any request leading to an unsafe state is denied

The banker's algorithm

General idea:

- Dijkstra (1965)
- Based on the detection algorithm
- Idea: avoid deadlocks by avoiding to run into an unsafe state
- Any request leading to an unsafe state is denied

Remark.
Mostly useless in practice since a process rarely knows the maximum resources it will need and the number of processes keeps varying. It also does not take into account hardware related issues (e.g. crashed printer)

The banker's algorithm

Deciding whether a state is safe or not:

1. Select a row in R whose resource request can be met. If no such row exists there is a possibility for a deadlock
2. When the process terminates it releases all its resource, and they can be added to the vector A
3. If all the processes terminate when repeating steps 1. and 2. then the sate is safe. If step 1. fails at any stage (not all the processes being finished) then the state is unsafe and the request should be denied

The banker's algorithm

Example.

Consider a system with 6 scanners, 3 plotters, 4 printers and 2 DVD drives:

$$E = \begin{pmatrix} 6 & 3 & 4 & 2 \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 1 & 2 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 3 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 \end{pmatrix}$$

The banker's algorithm

Example.

Consider a system with 6 scanners, 3 plotters, 4 printers and 2 DVD drives:

$$E = \begin{pmatrix} 6 & 3 & 4 & 2 \end{pmatrix} \qquad A = \begin{pmatrix} 2 & 2 & 2 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 3 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ . & . & . & . \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ . & . & . & . \\ 3 & 1 & 1 & 0 \end{pmatrix}$$

The banker's algorithm

Example.

Consider a system with 6 scanners, 3 plotters, 4 printers and 2 DVD drives:

$$E = \begin{pmatrix} 6 & 3 & 4 & 2 \end{pmatrix} \qquad A = \begin{pmatrix} 5 & 2 & 3 & 2 \end{pmatrix}$$

$$C = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ 0 & 2 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ 0 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 3 & 1 & 1 & 0 \end{pmatrix}$$

The banker's algorithm

Example.

Consider a system with 6 scanners, 3 plotters, 4 printers and 2 DVD drives:

$$E = \begin{pmatrix} 6 & 3 & 4 & 2 \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 1 & 2 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 3 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 3 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 \end{pmatrix}$$

Assuming that the first process requests 3 more scanners instead of 2, is the sate safe?

Deadlock prevention

Resource deadlocks only occur under the following four conditions:

- Mutual exclusion: a resource can be assigned to at most one process at a time
- Hold and wait: a process currently holding some resources can request some more
- No preemption: resources must be released by the process itself, i.e. they cannot be taken away by another process
- Circular wait: there is a circular chain of processes each of them waiting from some resources held by another process

Mutual exclusion condition

Preventing deadlocks:

- Not possible to remove it (e.g. two processes cannot print at the same time)
- Use daemon that handle specific output (e.g. printing daemon uses SPOOL)
- Deadlock can still happen (e.g. two processes fill up the SPOOL disk, without any of them being full)
- SPOOL cannot always be applied

Conclusion: not much can be done on this problem apart from carefully assigning resources

Hold and wait condition

Preventing deadlocks:

- Require processes to claim all the resources at once
- Not realistic, a process does not always know what resources will be necessary
- What if computation last for hours, and then the result is burnt on a DVD?
- Resources are not handle in an optimal way
- Alternative strategy: process has to release its resources before getting new ones

Conclusion: possible, but far from optimal

No preemption condition

Preventing deadlocks:

- Issue inherent to the hardware
- Often impossible to do anything (e.g. stop burning a DVD and resume later)
- Might require human intervention (e.g. stop a printer job, get the already printed pages, print another job and resume the first one)

Conclusion: not viable to break this condition

Circular wait condition

Preventing deadlocks:

- Order the resources
- Processes have to request resources in increasing order
- A process can only request a lower resource if it has released all the larger ones
- Is there an order satisfying everybody?

Conclusion: this is the best solution but it not always possible to use it in practice

A few more remarks

Deadlocks are not necessarily related to hardware resources:

- Database records: two-phase locking solution; lock all the records in phase one; if one fails release all the locks and retry later, otherwise proceed with phase two and manipulate the records
- Communication deadlocks: mutex can lead to deadlocks; no hardware resource involved; could be due to the loss of a message
- Livelock: lack of resources, process can not keep going so seat in tight loop and keeps trying not knowing it is hopeless
- Starvation: one long process delayed to let shorter ones run; might never run...

Key points

- How to detected deadlocks?
- How to fix a deadlock in a clean way?
- What are the four conditions that characterise a deadlock?
- What is a common practice regarding deadlocks?

Thank you!