Deadlock

- A computer system can be abstractly represented by a pair of sets (Σ, Π), where
  - Σ = {All possible allocation states of all system resources}
  - Π = {Threads}
- Threads behave like functions, mapping one system state to another as they execute
- We say that a thread is blocked if it is in a system state from which it cannot run
- We say that a thread is deadlocked if a thread is blocked in the current system state, and in all future states the system can ever reach

Deadlock

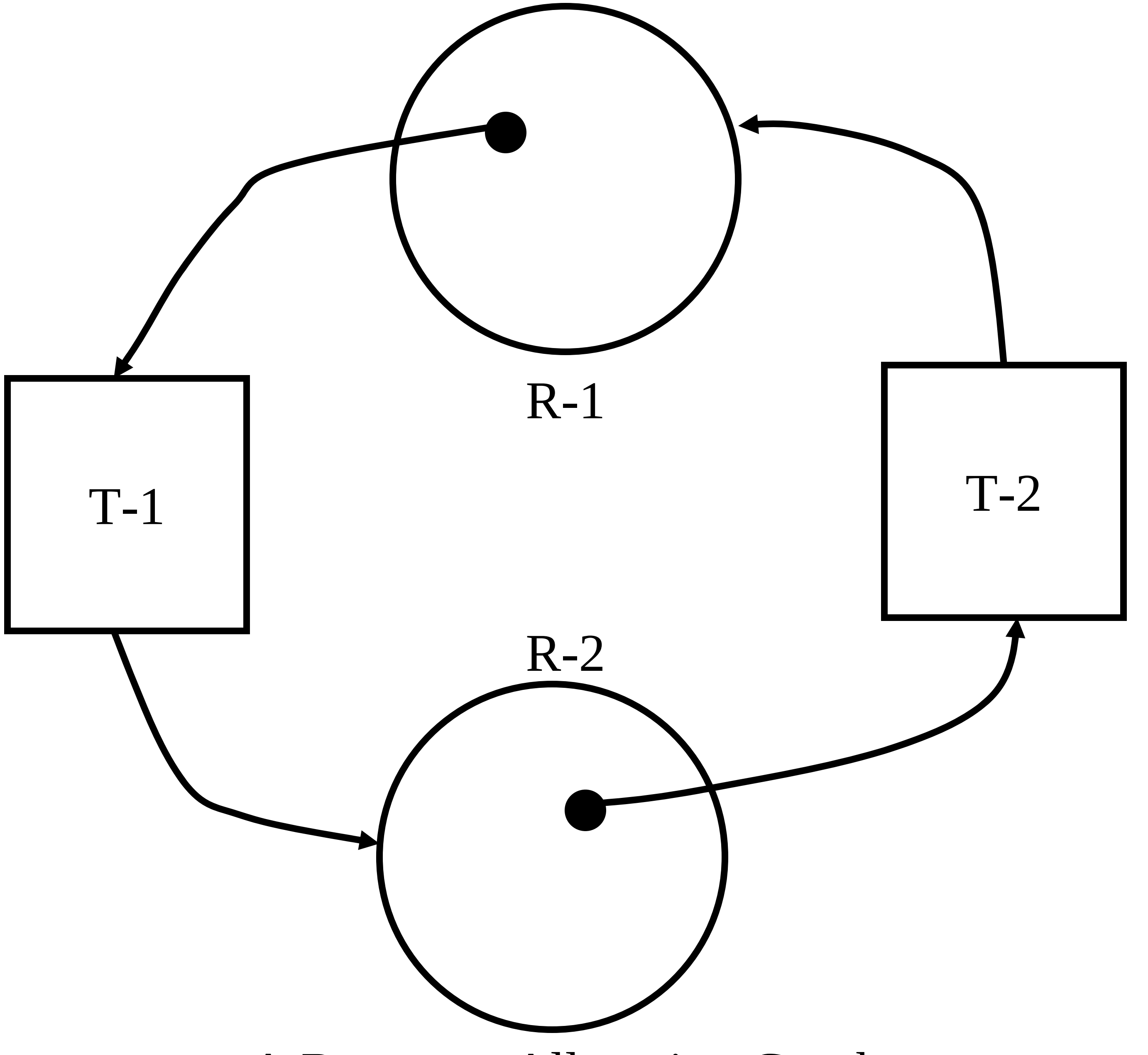
- There are 4 necessary conditions for a deadlock to occur
  - The existence of mutually exclusive resources in a system (the mutex condition)
    - Such resources are broadly characterized as either serially reusable, or consumable
  - A hold-and-wait condition in the system
  - A no-preemption condition in the system
  - A circular wait condition in the system

Deadlock

- There are 4 areas of deadlock study that have been researched extensively:
  - Deadlock prevention
  - Deadlock avoidance
  - Deadlock detection
    - Deadlock recovery as an extension of detection

Deadlock

- Prevention involves denying a necessary condition and is always “expensive”
- Avoidance employs policy decisions which may hold-back resources to maintain “safe states”
- Detection is generally achieved by the construction and reduction of Resource Allocation Graphs (RAGs … bipartite graphs with thread and resource nodes)
- Recovery generally involves thread termination and is often based on ad-hoc policies at a given site



A Resource Allocation Graph

Deadlock

- Prevention may be achieved by denying any one of the necessary conditions:
  - Exclusively accessed resources
    - since things as basic as a memory location can fall in this category, we have to live with this condition
  - Hold and wait condition
    - a-priori resource allocation (the policy employed can lead to its own deadlock)
    - resource under-utilization (RU)

Deadlock

- Prevention (continued)
  - No preemption
    - lost work
    - indefinite postponement (IP)
  - Circular wait
    - appropriate resource ordering
    - RU
    - changes may go all the way back to application sources

Deadlock

- Avoidance
  - Safe and unsafe states
    - no single resource allocation can lead directly to deadlock from a safe state
    - consider the following system of 3 threads and 10 tape drives:

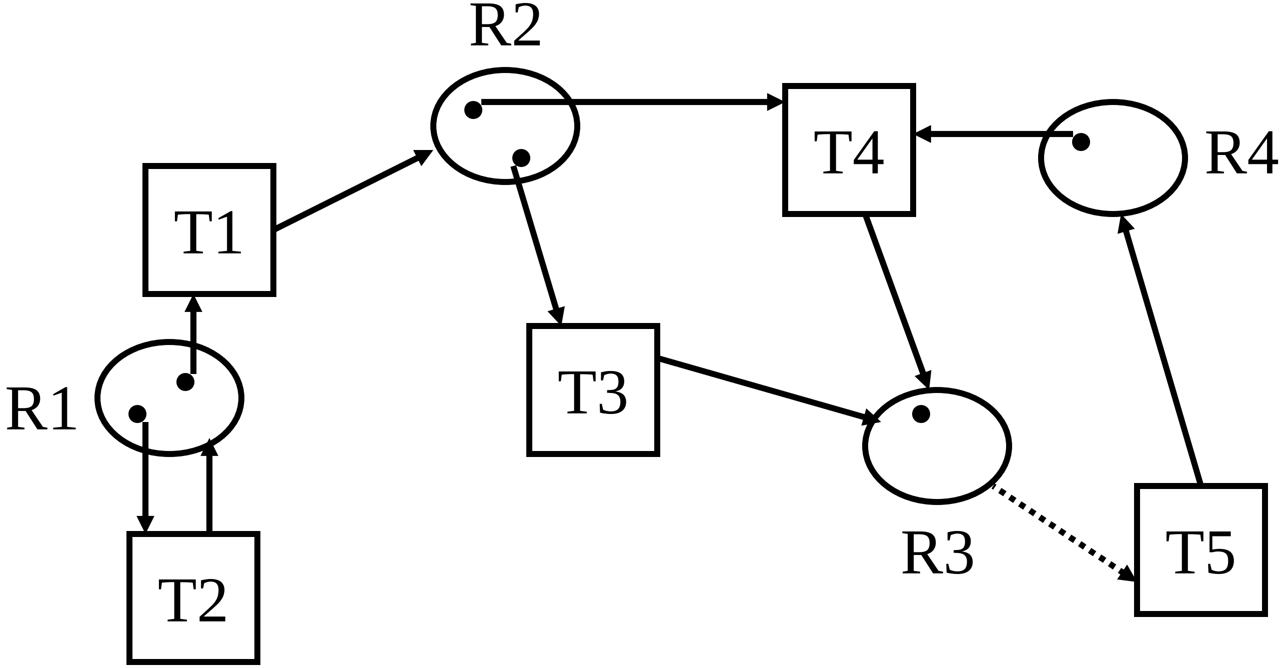
THREAD	CURRENT	MAX	BALANCE
A	2	4	2
B	3	6	3
C	3	8	5

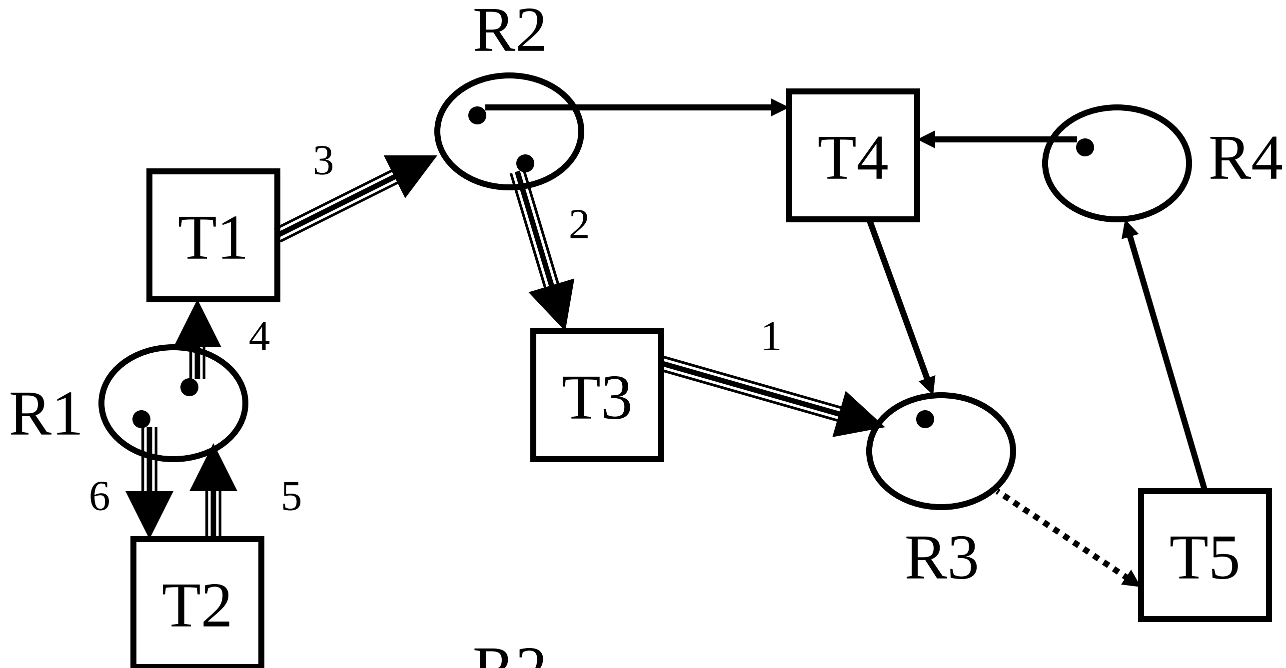
If A asks for 1 drive should the request be granted ?

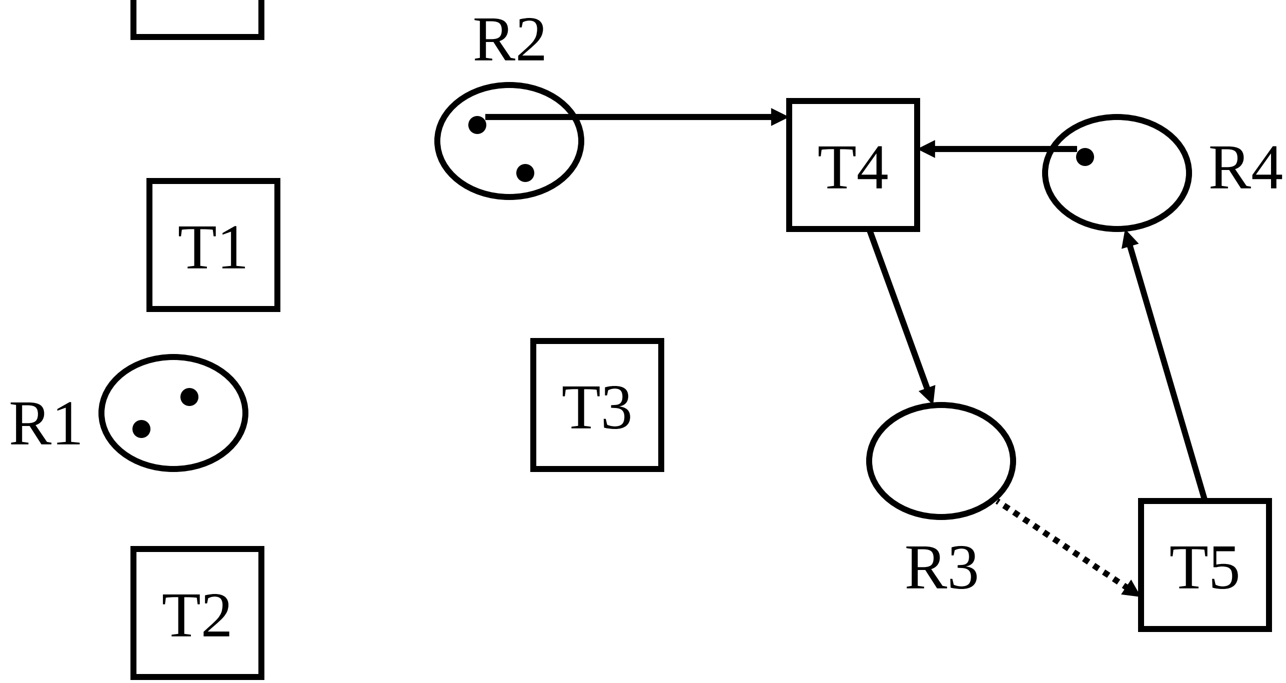
If B asks for 1 drive should the request be granted ?

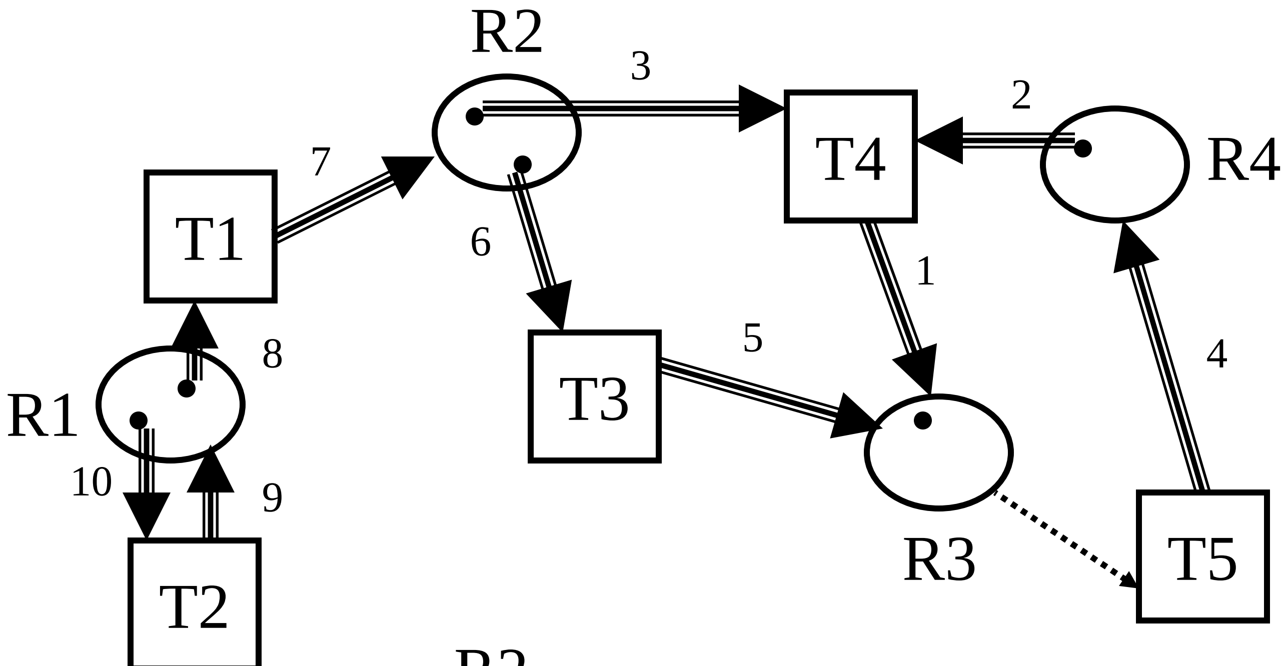
Deadlock

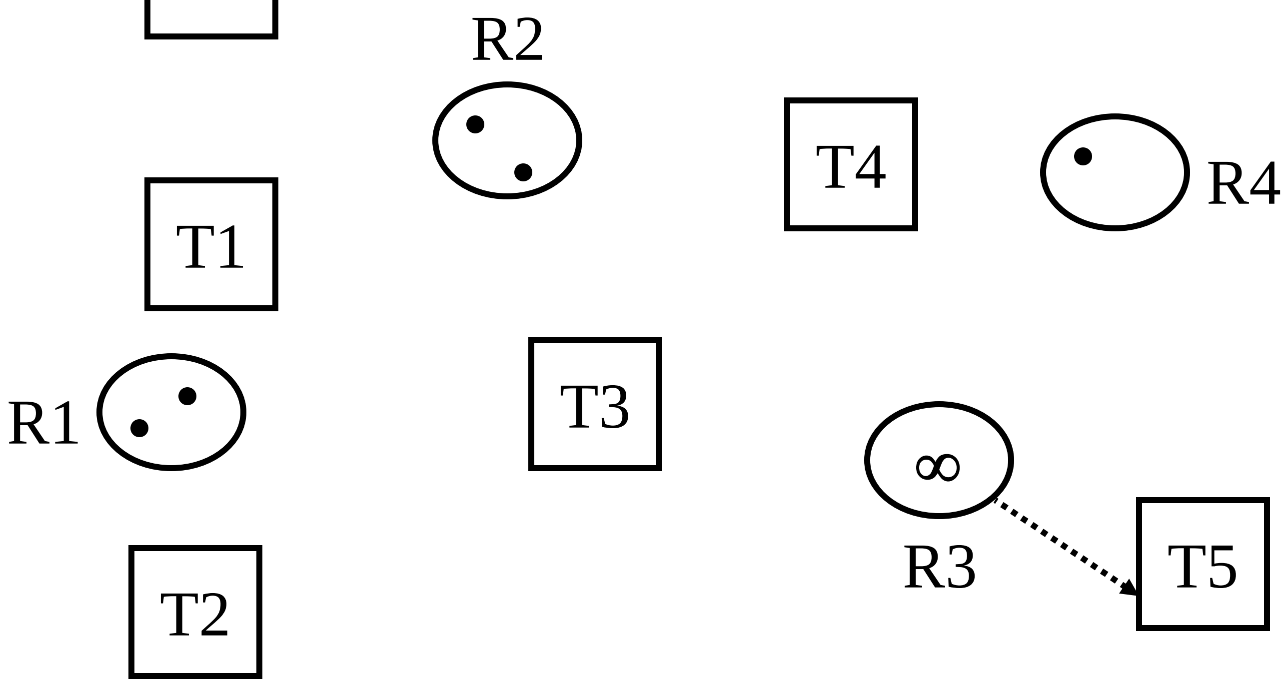
- Detection
  - RAG reduction … bipartite, M res, N threads
  - Cycle is necessary condition for deadlock in all cases, but is sufficient in AND model reusable only systems

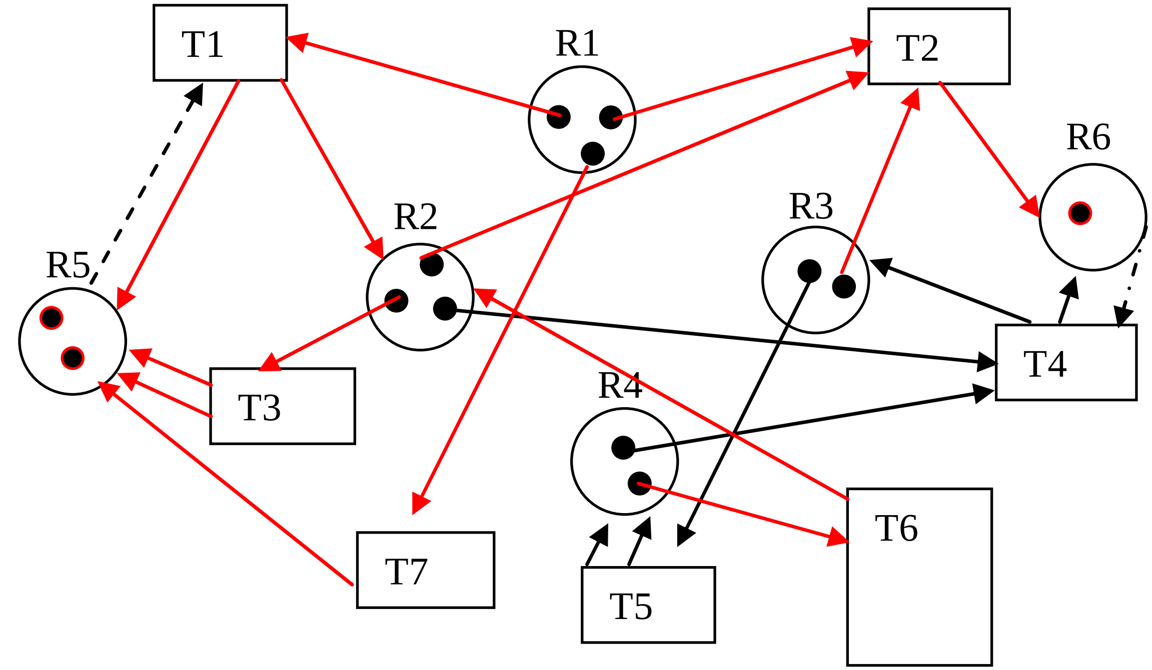


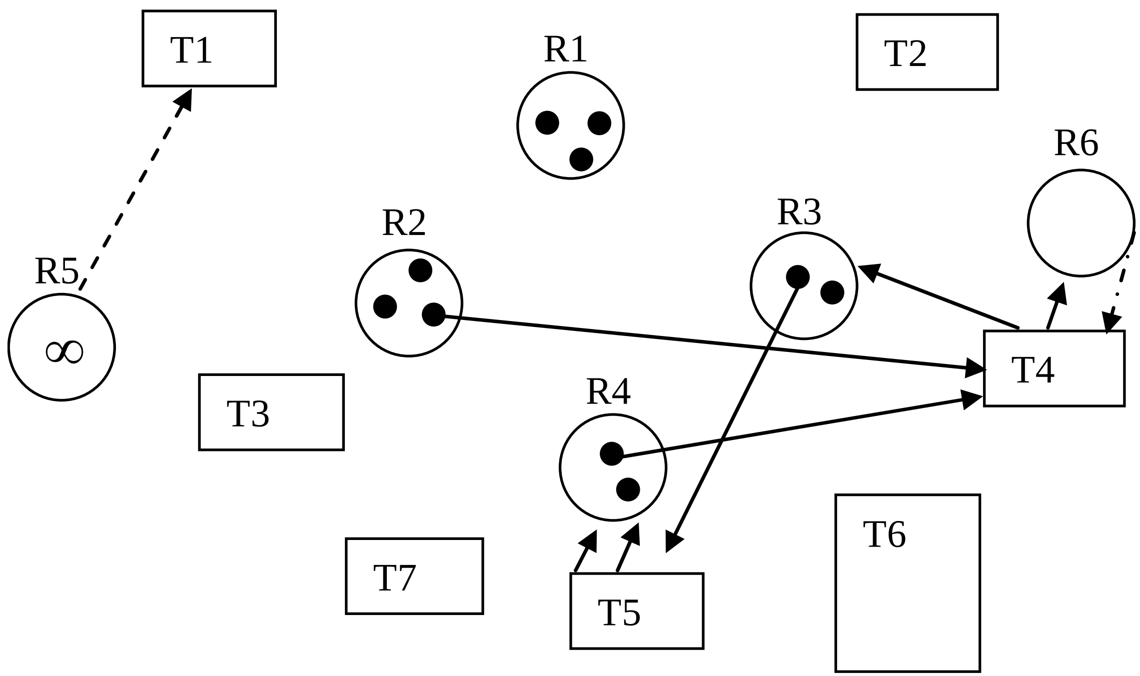

R2
T4
R4
3
T1
2
4
T3
1
R1
6
5
R3
T5
T2


R2
T4
R4
T1
T3
R1
R3
T5
T2


R2
3
T4
2
R4
7
T1
6
1
8
R1
T3
5
4
10
9
R3
T5
T2


R2
T4
R4
T1
R1
T3
∞
R3
T5
T2





T4 and T5 in DL

Deadlock

- Complexity of reduction:
  - For GENERAL graphs:
    - O(MN!)
  - For REUSABLE ONLY graphs:
    - O(MN)