Reachability and deadlocking in multi-stage scheduling

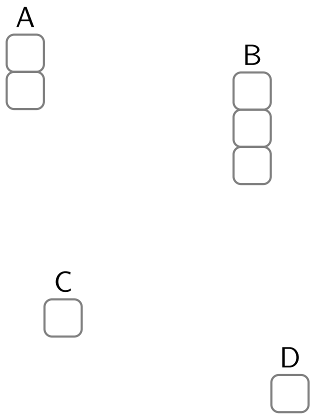
Christian Eggermont
Gerhard Woeginger

Eindhoven University of Technology

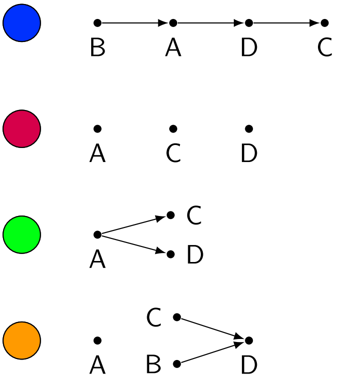
Wednesday September 28, 2011 RP 2011

Scheduling systems

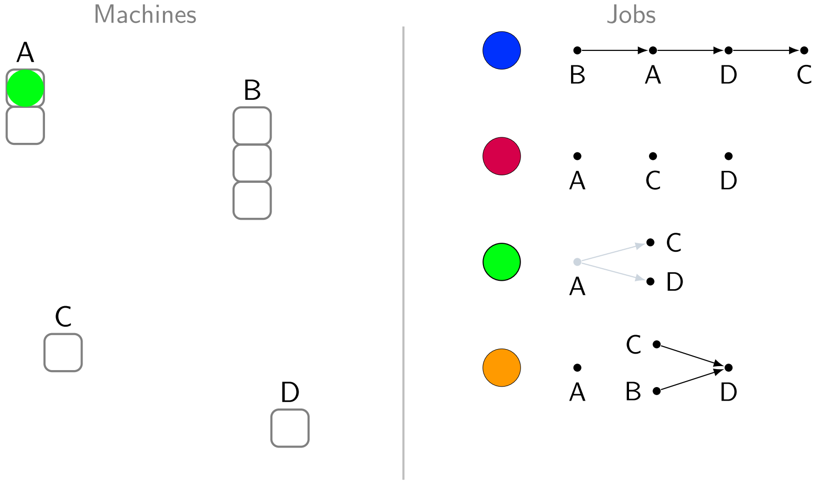
Machines



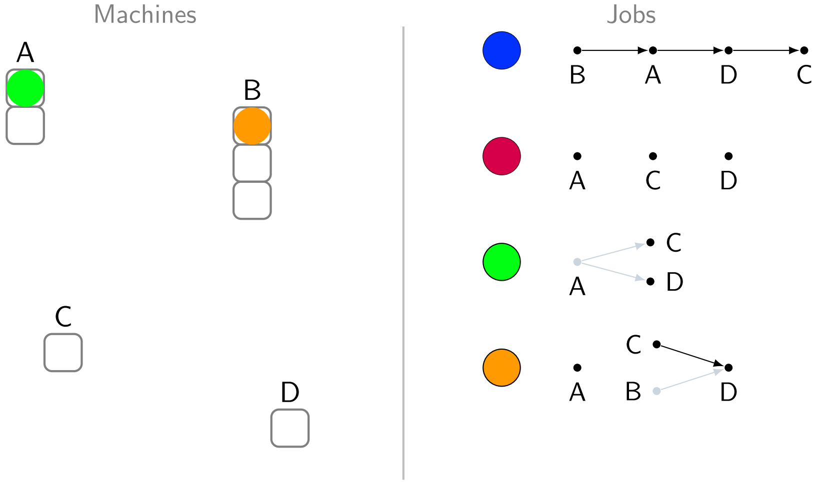
Jobs



Scheduling systems

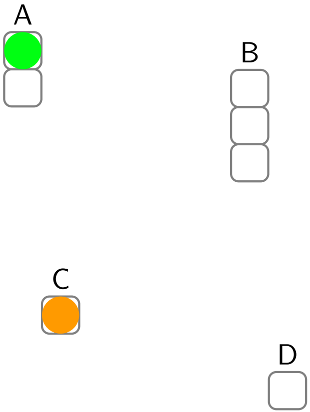


Scheduling systems

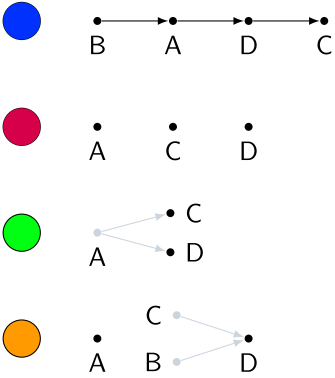


Scheduling systems

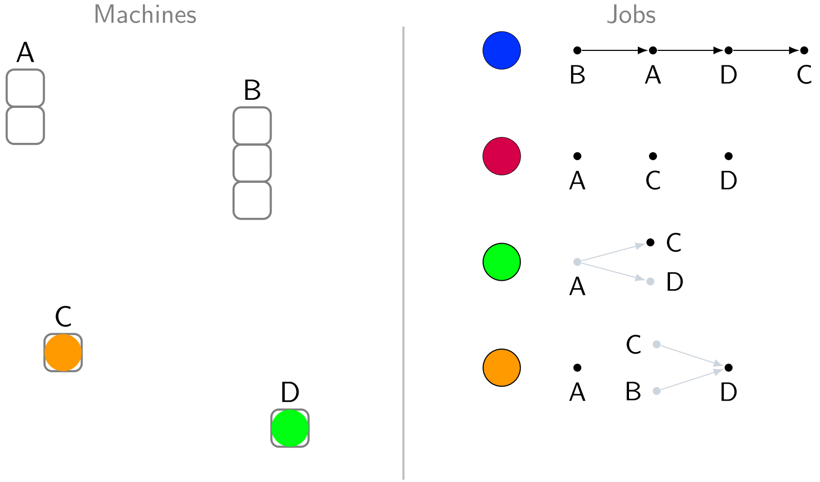
Machines



Jobs

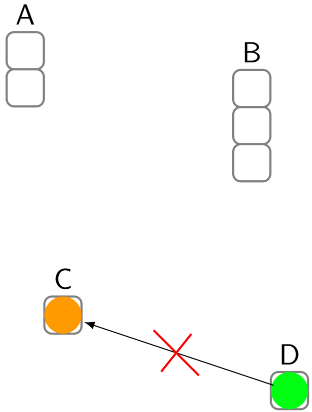


Scheduling systems

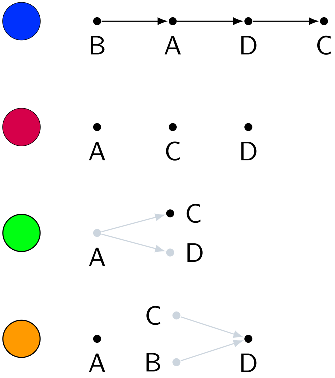


Scheduling systems

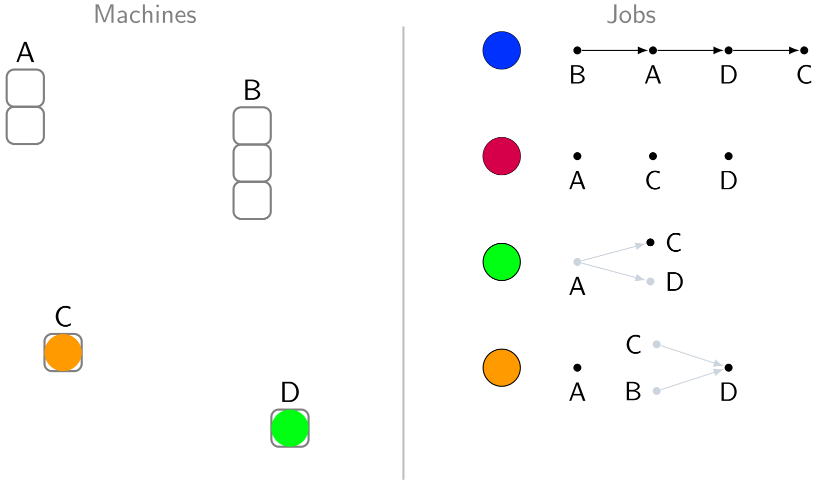
Machines



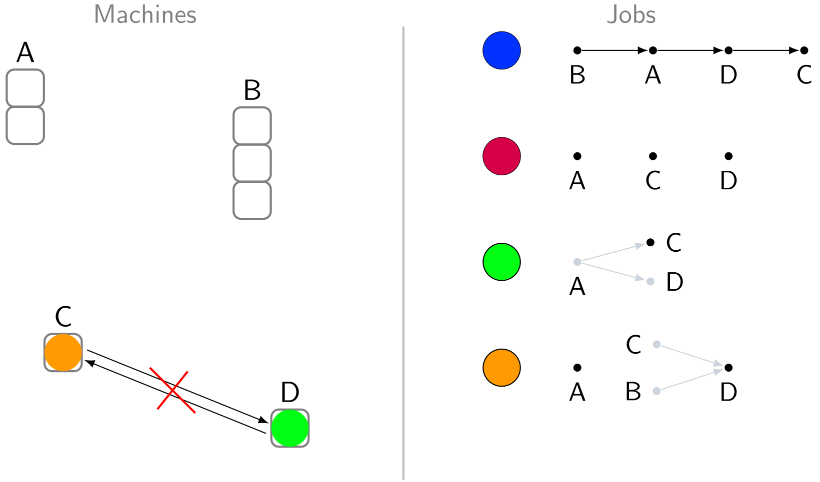
Jobs



Scheduling systems

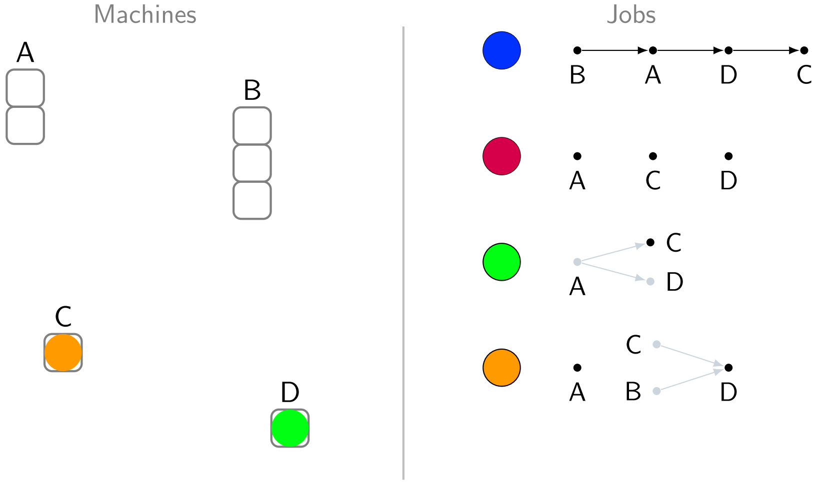


Scheduling systems

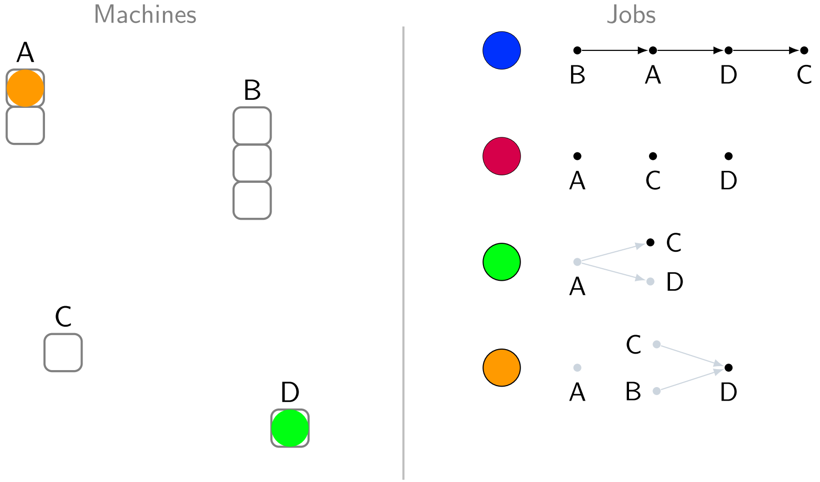


Unsynchronised job movements

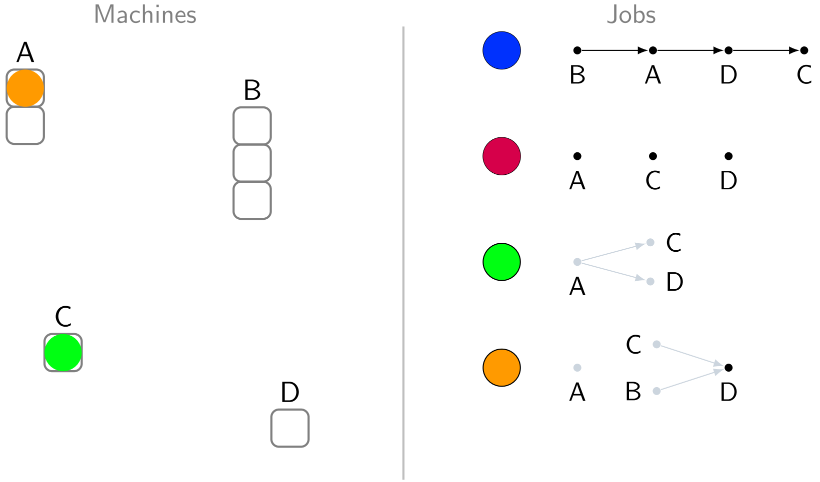
Scheduling systems



Scheduling systems

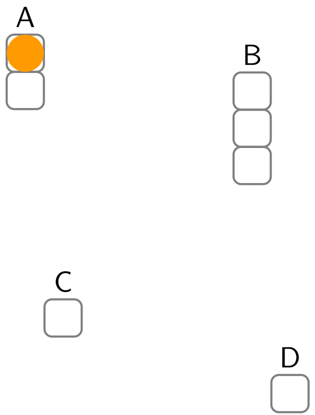


Scheduling systems

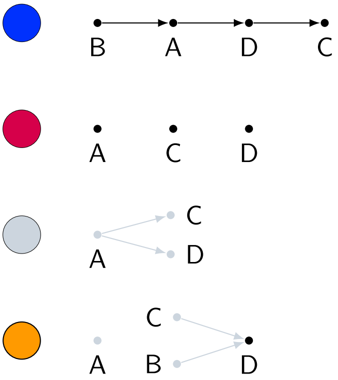


Scheduling systems

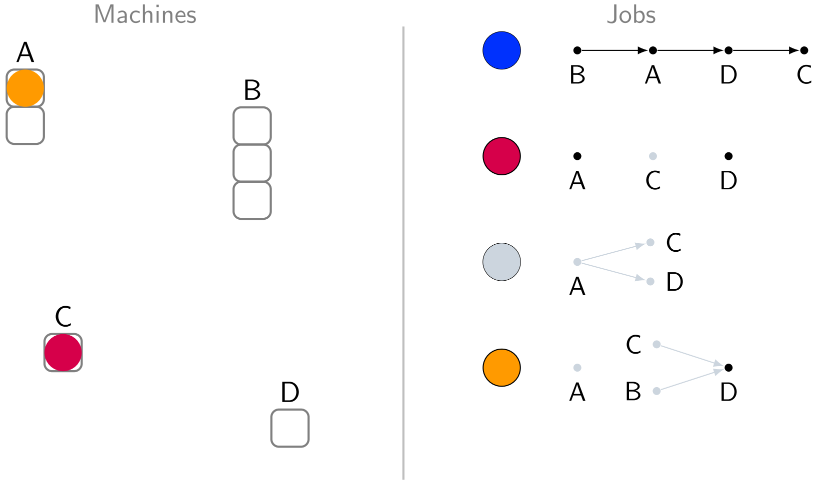
Machines



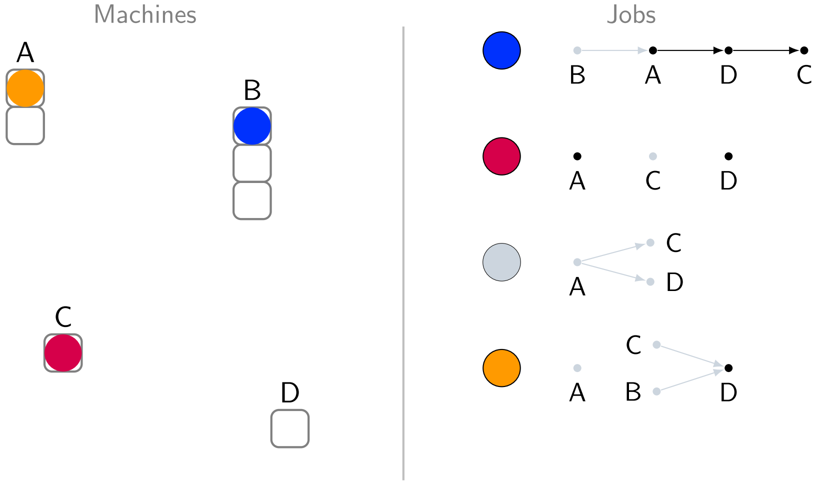
Jobs



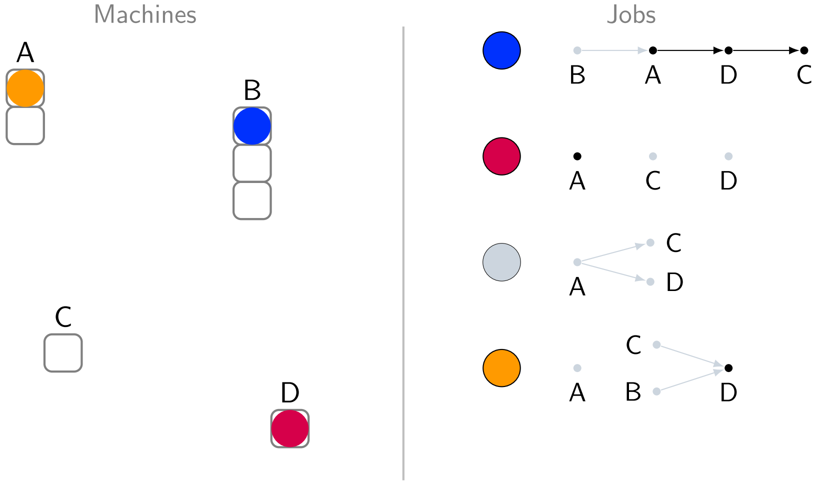
Scheduling systems



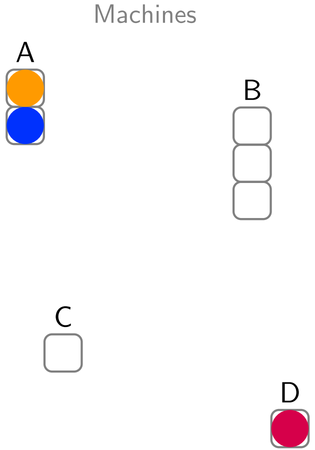
Scheduling systems

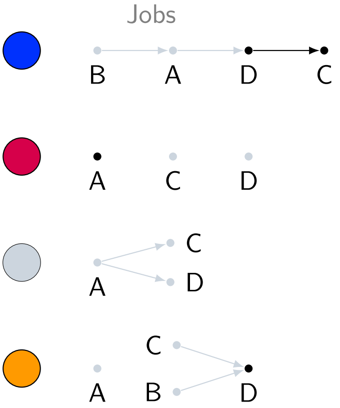


Scheduling systems

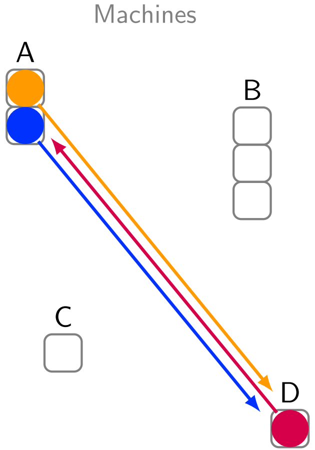


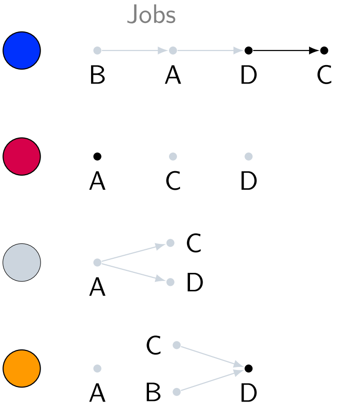
Scheduling systems





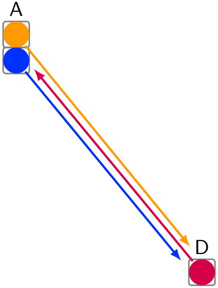
Scheduling systems





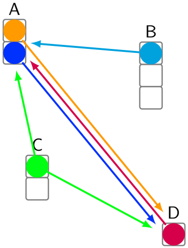
Basic Terms

- blocking set;
  - (sub-)set of machines $\mathcal{B} \neq \emptyset$ occupied to full capacity,
  - jobs on $\mathcal{B}$ need further processing, *next* only within $\mathcal{B}$



Basic Terms

- blocking set;
  - (sub-)set of machines $\mathcal{B} \neq \emptyset$ occupied to full capacity,
  - jobs on $\mathcal{B}$ need further processing, *next* only within $\mathcal{B}$
- deadlock; a system state with
  - not all jobs completed
  - no job can move



Basic Terms

- blocking set;
  - (sub-)set of machines $\mathcal{B} \neq \emptyset$ occupied to full capacity,
  - jobs on $\mathcal{B}$ need further processing, *next* only within $\mathcal{B}$
- deadlock; a system state with
  - not all jobs completed
  - no job can move
- unsafe state; a system state where deadlock is unavoidable

Basic Terms

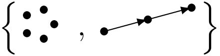
- blocking set;
  - (sub-)set of machines $\mathcal{B} \neq \emptyset$ occupied to full capacity,
  - jobs on $\mathcal{B}$ need further processing, *next* only within $\mathcal{B}$
- deadlock; a system state with
  - not all jobs completed
  - no job can move
- unsafe state; a system state where deadlock is unavoidable

Lemma

state contains blocking set $\implies$ *state unsafe*

Basic Terms

- blocking set;
  - (sub-)set of machines $\mathcal{B} \neq \emptyset$ occupied to full capacity,
  - jobs on $\mathcal{B}$ need further processing, *next* only within $\mathcal{B}$
- deadlock; a system state with
  - not all jobs completed
  - no job can move
- unsafe state; a system state where deadlock is unavoidable
- scheduling system restricted to family of digraphs F;
  each job is isomorphic to digraph in F.



Previous Related Work

- M. Lawley and S. Reveliotis (2001). Deadlock avoidance for sequential resource allocation systems: hard and easy cases. *The International Journal of Flexible Manufacturing Systems* 13, 385–404.
- W. Sulistyono and M. Lawley (2001). Deadlock avoidance for manufacturing systems with partially ordered process plans. *IEEE Transactions on Robotics and Automation 17*, 819–832.
- C.E.J. Eggermont, A. Schrijver, G.J. Woeginger (2011). Analysis of multi-stage open shop processing systems. *International Symposium on Theoretical Aspects of Computer Science* (STACS), LIPIcs 9, 484–494.

Problems

Problems

1. Recognize Safe State
   Instance : scheduling system, state s
   Question : Is state s safe?

Problems

1. RECOGNIZE SAFE STATE
   Instance : scheduling system, state s
   Question : Is state s safe?

2. REACHABILITY
   Instance : scheduling system, state s
   Question : Can the system reach state s?

Problems

1. RECOGNIZE SAFE STATE
   Instance : scheduling system, state s
   Question : Is state s safe?
2. REACHABILITY
   Instance : scheduling system, state s
   Question : Can the system reach state s?
3. DEADLOCK
   Instance : scheduling system
   Question : Can the system fall into a deadlock state?

Problems

How does restricting scheduling system affect complexity?

1. RECOGNIZE SAFE STATE
   Instance : scheduling system, state s
   Question : Is state s safe?

2. REACHABILITY
   Instance : scheduling system, state s
   Question : Can the system reach state s?

3. DEADLOCK
   Instance : scheduling system
   Question : Can the system fall into a deadlock state?

Results

Results: 1. Safety

RECOGNIZE SAFE STATE

Instance : scheduling system, state s

Question : Is state s safe?

Results: 1. Safety

RECOGNIZE SAFE STATE

Instance : scheduling system, state s

Question : Is state s safe?

Theorems (1996-2001)

For scheduling systems where either

(i) every machine capacity > 1 or

(ii) every job is unconstrained:

$$\text{state unsafe} \iff \text{state contains blocking set}$$

Results: 1. Safety

RECOGNIZE SAFE STATE

Instance : scheduling system, state s

Question : Is state s safe?

Theorems (1996-2001)

For scheduling systems where either

(i) every machine capacity > 1 or

(ii) every job is unconstrained:

$$\text{state unsafe} \iff \text{state contains blocking set}$$

Lemma

Whether state contains blocking set is decidable in polynomial time

Results: 1. Safety

Results: 1. Safety

Theorem

For scheduling system s.t. machines with capacity 1 occur only in unconstrained plans and out-stars:

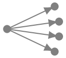
$$\textit{state unsafe} \iff \textit{state contains blocking set}$$

Results: 1. Safety

Theorem

For scheduling system s.t. machines with capacity 1 occur only in unconstrained plans and out-stars:

$$\text{state unsafe} \iff \text{state contains blocking set}$$



Results: 1. Safety

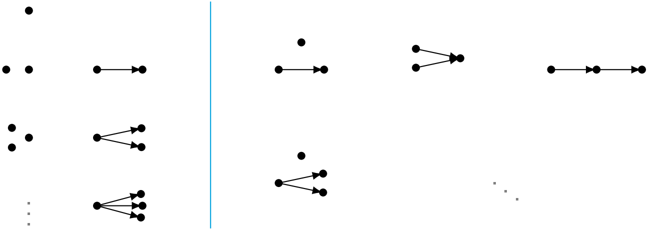
Theorem

For scheduling system s.t. machines with capacity 1 occur only in unconstrained plans and out-stars:

$$\text{state unsafe} \iff \text{state contains blocking set}$$

Theorem

RECOGNIZE SAFE STATE *NP-hard otherwise*



Results: 2. Reachability

REACHABILITY

Instance : scheduling system, state s

Question : Can the system reach state s?

Results: 2. Reachability

REACHABILITY

Instance : scheduling system, state s

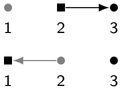
Question : Can the system reach state s?

Example:

$\mathcal{M}$: $\{1, 2, 3\}$

J_1:

J_2:



Results: 2. Reachability

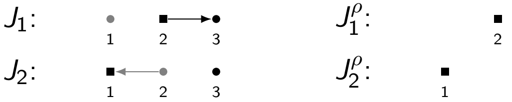
REACHABILITY

Instance : scheduling system, state s

Question : Can the system reach state s?

Example:

$\mathcal{M}$: $\{1, 2, 3\}$ $\qquad$ $\mathcal{M}^\rho$: $\{1, 2, 3\}$



Results: 2. Reachability

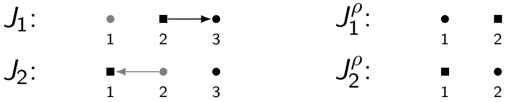
Reachability

Instance : scheduling system, state s

Question : Can the system reach state s?

Example:

$\mathcal{M}$: $\{1, 2, 3\}$ $\qquad$ $\mathcal{M}^\rho$: $\{1, 2, 3\}$



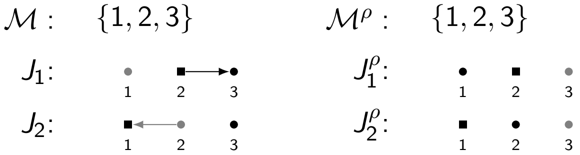
Results: 2. Reachability

REACHABILITY

Instance : scheduling system, state s

Question : Can the system reach state s?

Example:



Results: 2. Reachability

REACHABILITY

Instance : scheduling system, state s

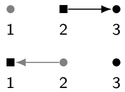
Question : Can the system reach state s?

Example:

$\mathcal{M}$: $\{1,2,3\}$

J_1:

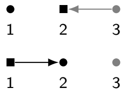
J_2:



$\mathcal{M}^\rho$: $\{1,2,3\}$

J_1^ρ:

J_2^ρ:



Results: 2. Reachability

REACHABILITY
Instance : scheduling system, state s
Question : Can the system reach state s?

Lemma

State s reachable $\iff$ state $\rho(s)$ is safe in new system

Results: 2. Reachability

REACHABILITY

Instance : scheduling system, state s

Question : Can the system reach state s?

Lemma

State s reachable $\Longleftrightarrow$ state $\rho(s)$ is safe in new system

Theorem

REACHABILITY *is decidable in polynomial time for scheduling system where machines with capacity 1 occur only in unconstrained plans and in-stars, and NP-hard otherwise.*

Results: 3. Deadlock

DEADLOCK

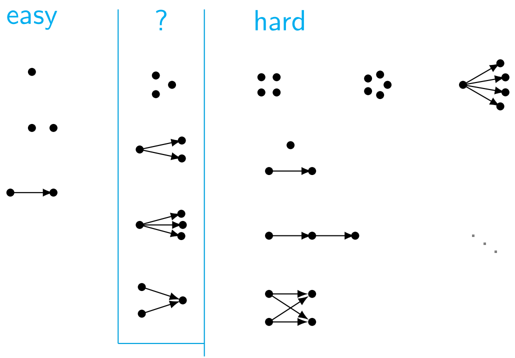
Instance : scheduling system

Question : Can the system fall into a deadlock?

Results: 3. Deadlock

Deadlock

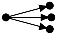
Instance : scheduling system
Question : Can the system fall into a deadlock?



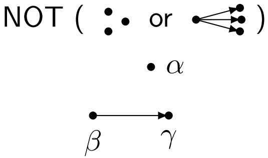
Proof: 1. Safety
sketch

Proof: 1. Safety

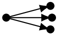
sketch

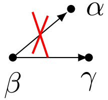
NOT ( or )

Proof: 1. Safety
sketch

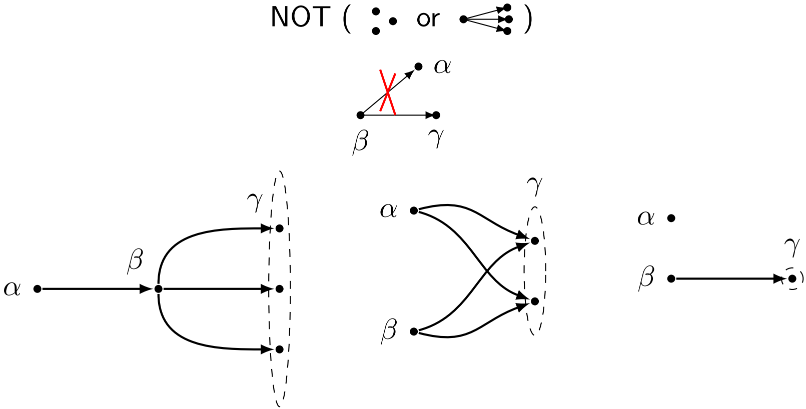


Proof: 1. Safety
sketch

NOT ( or )

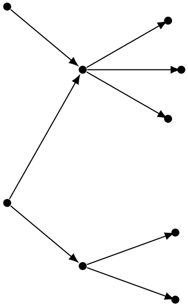


Proof: 1. Safety
sketch

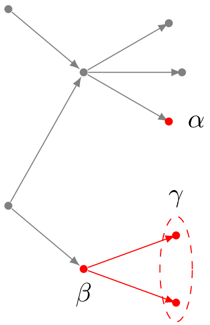


Decomposition: machine β s.t. set $\gamma := \text{succ}(\beta)$ minimal,
$\alpha \neq \beta$ sink on non-succesors, call rest δ.

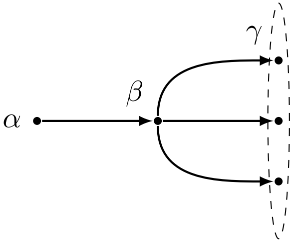
Proof: 1. Safety
sketch



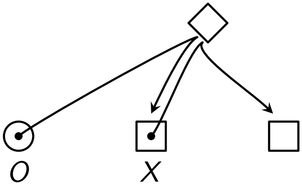
Proof: 1. Safety
sketch



Proof: 1. Safety
sketch

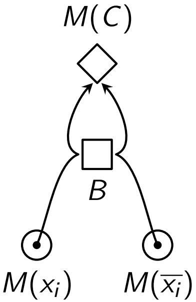


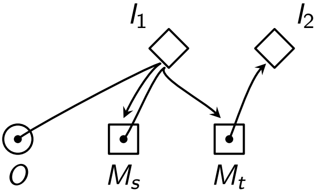
Proof: 1. Safety
sketch



Proof: 1. Safety

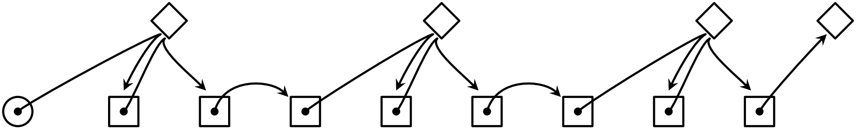
sketch



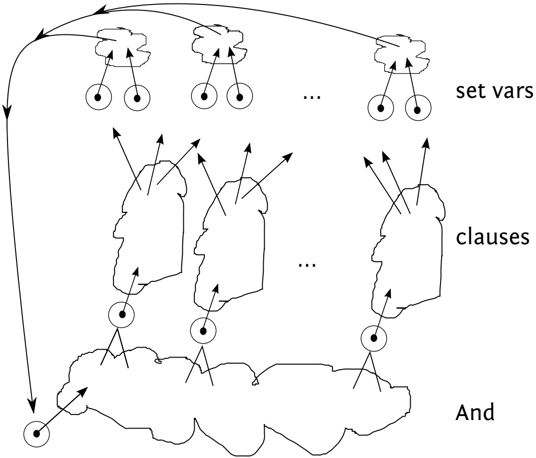


□ : internal machine
◇ : input machine
○ : output machine

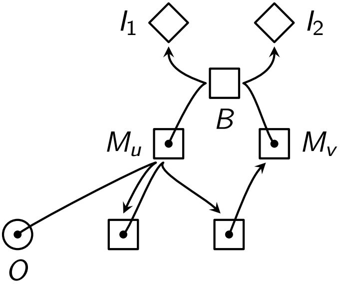
Proof: 1. Safety
sketch



Proof: 1. Safety
sketch



Proof: 1. Safety
sketch



Questions / Comments ?