Tutorials: Groups of 2/3

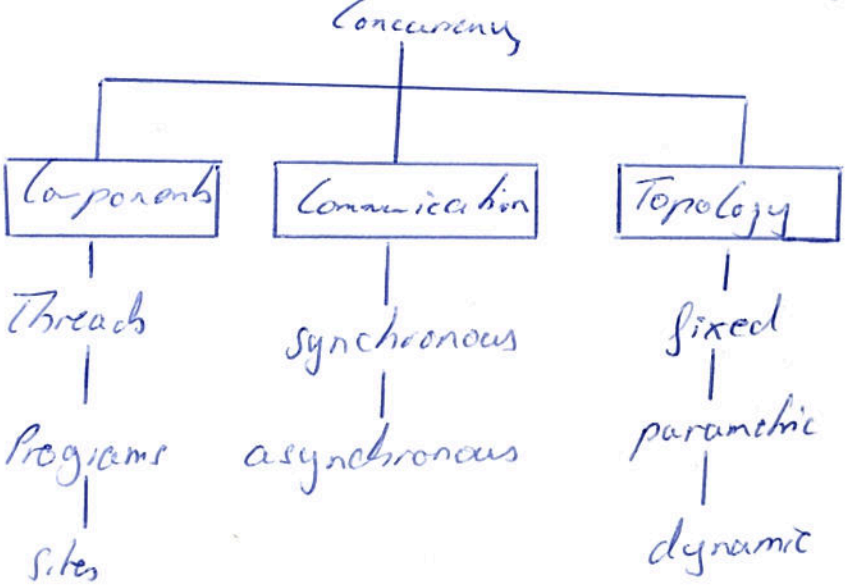
- Exercises: Out Tuesday, in Monday, Wednesday tutorial.
- Exercises marked + for correct answer (need not be perfect)
- 60% of exercises need + to be admitted to the exam
- Exercises start this week
  - ↳ Georgel gives more introduction to Petri nets.
- 2 presentations on the board required

Exam:

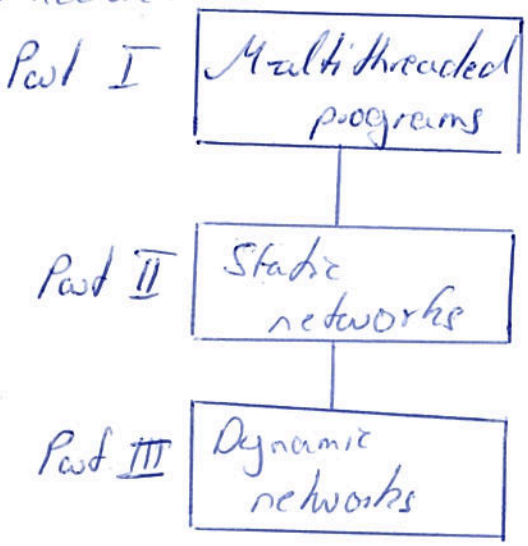
- Oral / written will be decided next week.

Concurrent systems:

Spectrum



This lecture:



Multithreaded programs

↳ Threads, synchronous/asynchronous communication, fixed/parametric topology

↳ Dekker's / Peterson's mutex algorithms

↳ Dijkstra's semaphore

Static networks:

↳ Networking sites, asynchronous message passing, fixed topology (parametric message channels)

↳ Alternating bit protocol

Dynamic networks (Remote objects):
↳ OO-programs, synchronous communication,
dynamic topologies
↳ Logic protocol for client-server architectures

Goal of this course:
- Learn about automata models for these concurrent systems

I-DFP:
↳ NFA/DFA for finite state programs
↳ Pushdown automata for recursive programs

Here:
↳ Automata for concurrent systems.

- Learn how to analyse them automatically:

Approach to verification:
↳ Start from Java RMI program,
class in concurrency library, ...
↳ Compile down to automata
↳ Apply procedures you learned to infer correctness.

Overview of the course content:

Part I: Multi-threaded programs

Goal: Verify multithreaded programs
Examples: • Mutual exclusion protocols
• Programs with shared memory, caches

Automata model: (Unbounded) Petri nets
↳ finitely many types of threads/components
↳ potentially unboundedly many instances (parameter)
↳ programs do not use heap structures (lists, trees, ...),
no recursion

Verification problems:

↳ Safety properties: Is an undesirable state reachable?

- Mutex fails if two threads in critical sections
- Bounded Petri nets (finite state): Fast algorithms
- Unbounded Petri nets (infinite state): Decidability.

↳ Invariants: Does some relation among variables always hold?

Example: Dijkstra's semaphore

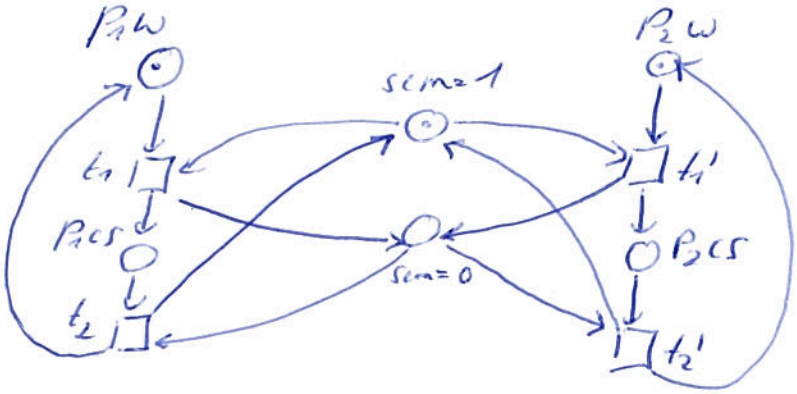
Semaphore: Boolean variable, initially 1

2 threads: ↳ Try to enter their critical sections

↳ Pass test semaphore from 1 to 0

↳ Free test semaphore from 0 to 1

Automaton model:



Places ○ ≈ control locations of finite automata

Transitions □ + arcs ≈ transitions of finite automata + their synchronisation (Remember parallel product $T_1 \| T_2$)

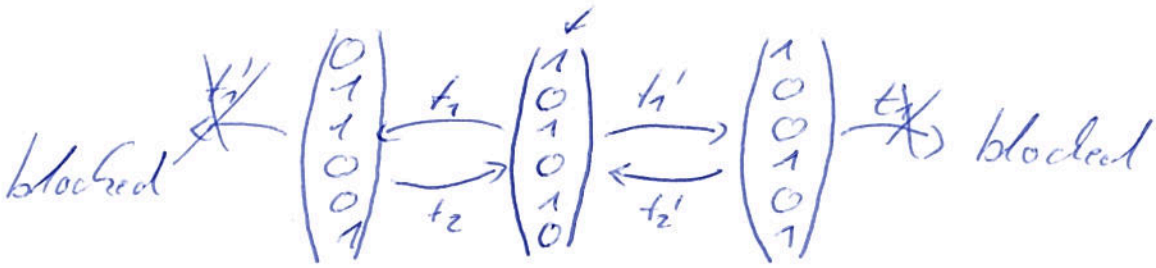
Marking ≈ state of an automaton

↳ Tokens • are not ordered.

A little bit of semantics:

↳ Markings are vectors $\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ $\begin{matrix} P_1W \\ P_1CS \\ P_2W \\ P_2CS \\ sem=1 \\ sem=0 \end{matrix}$ over $\mathbb{N}^{|Places|}$

↳ Transitions
- remove a token for every incoming arc
- produce a token for every outgoing arc
- block if there are not enough tokens available



Reachability-problem:

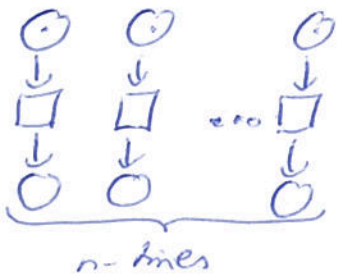
Given: PN N and a (bad) marking M.

Question: Is M reachable from initial marking M_0 of N?

Study algorithmic techniques to answer this question.

Unfoldings:

Problem: State space grows exponentially in degree of concurrency



yields 2^n reachable markings.

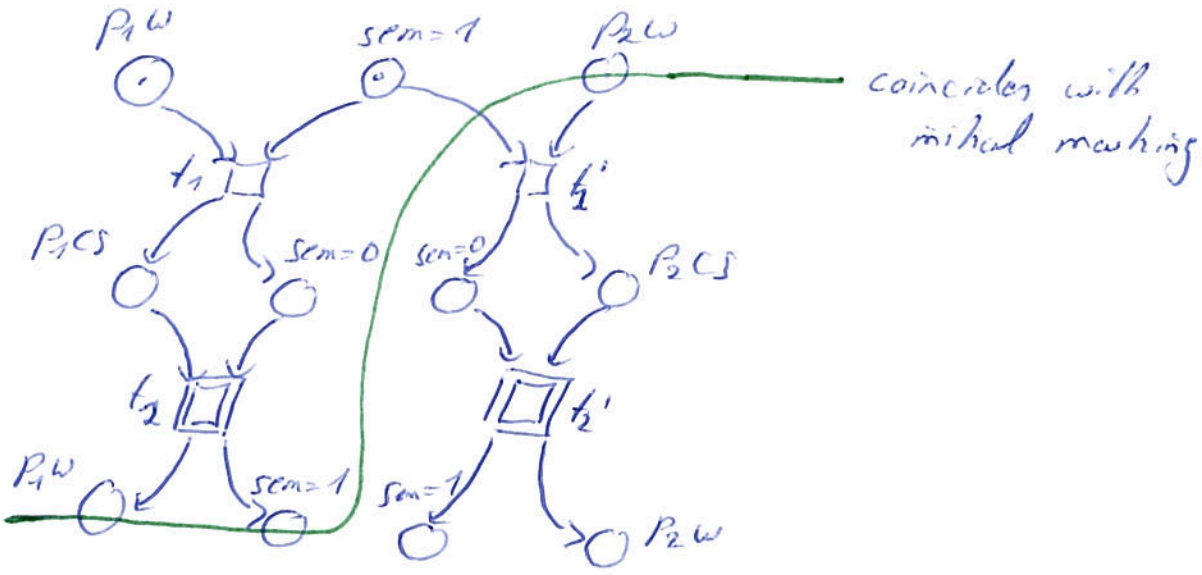
Not a good idea to compute them explicitly.

Idea: Perform changes locally.

Technically:

Compute an acyclic version of the net
↳ Every token yields a place
↳ Instead fresh instances of transitions.

In the example:



... ...

from here, the unfolding starts repeating and can be truncated.

Encode unfolding into SAT:

$$\mathcal{U} = \underbrace{(t_2 \Rightarrow t_1) \wedge (t_2' \Rightarrow t_1')}_{\text{causally closed}} \wedge \underbrace{(t_1 \Rightarrow \neg t_1') \wedge (t_1' \Rightarrow \neg t_1)}_{\text{conflict free}}$$

Is $M = \begin{pmatrix} ? \\ 1 \\ ? \\ ? \\ 1 \\ 2 \\ ? \end{pmatrix}$ reachable?

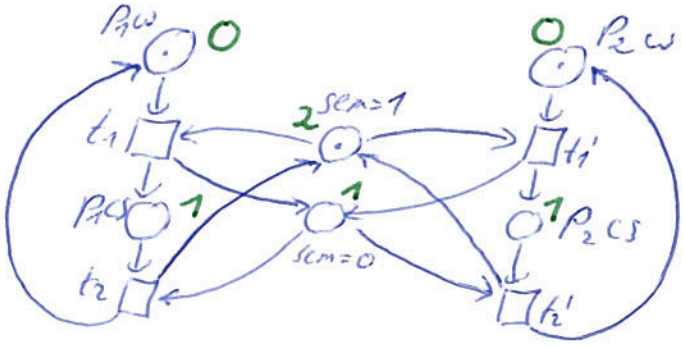
$\Leftrightarrow$ t_1 and t_1' can be fired without t_2 or t_2' in between in some sequence.

But $\mathcal{U} \wedge t_1 \wedge t_1' \equiv$ false

Return "unreachable". □

Invariants:

Weight tokens on places



Weighted sum of tokens stays constant in every reachable marking M:

$$2M(sem=1) + M(p_1cs) + M(p_2cs) + M(sem=0) = 2$$

Init: ✓

If t_1: $2 \cdot 0 + 1 + 0 + 1 = 2$

If t_1': $2 \cdot 0 + 0 + 1 + 1 = 2$

Proving mutual exclusion (with invariants)

Is M reachable with $M(p_1cs) = 1 = M(p_2cs) = M(sem=0)$?

Equation has to be satisfied:

$$2M(sem=1) + M(p_1cs) + M(p_2cs) + M(sem=0)$$
$$= 2M(sem=1) + 1 + 1 + 1$$
$$\geq 3$$
$$\neq 2.$$

Return "marking is not reachable, mutual exclusion holds".

How to compute the weights?

Use linear algebra.

Connectivity matrix

$$C = \begin{array}{c} \\ p_1w \\ p_1cs \\ p_2w \\ p_2cs \\ sem=1 \\ sem=0 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_1' & t_2 & t_2' \end{array} \\ \begin{pmatrix} -1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 \end{pmatrix} \end{array}$$

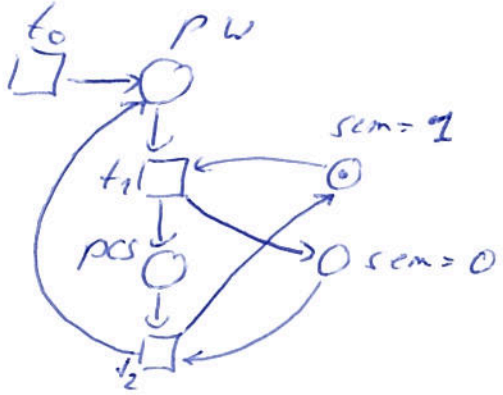
Invariants are solutions to $C^T x = 0$.

In the example:

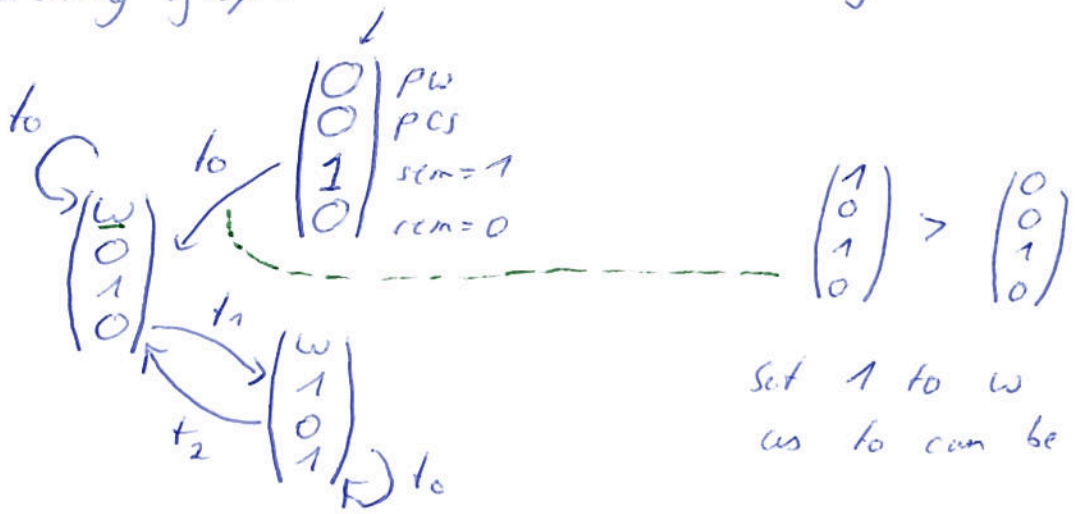
$$\begin{pmatrix} -1 & 1 & 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \checkmark$$

Coverability graphs:

Consider an infinite state version of the example:



Coverability graph is an abstract version of the transition system



Mutual exclusion still holds. ☐