Reliability of Technical Systems

Advanced Methods for Systems Modelling and Simulation I : Petri Nets

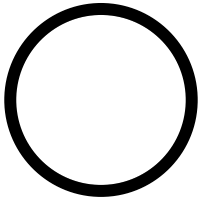
Petri Nets - Overview

- Introduction
- Basic elements and rules
- Notation
- Extensions/Tools
- Applications / Examples
- References

Introduction

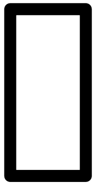
- Modelling and analysis of complicated systems
- Illustration and simulation of parallel processes
- Developed in1962 by Carl A. Petri
- In the 60‘s use in informatics
- Later also in other areas like risk analysis
- Today: Highly accepted and often used tool in risk analysis

Basic elements



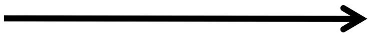
Place (storage)

- Places describe certain system states like a state of a machine



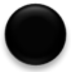
Transition (action)

- Transitions stand for events, which can influence one or several system states



Edge (link)

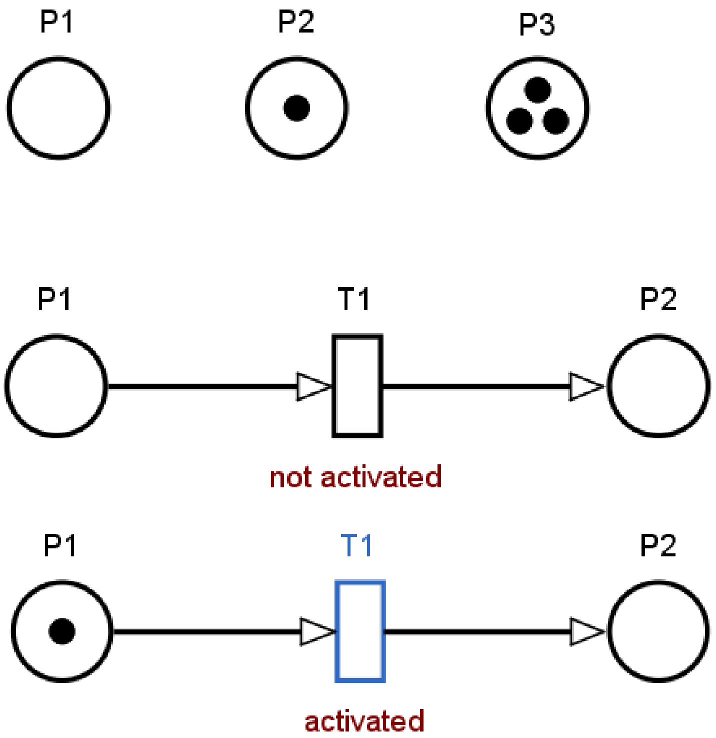
- Edges are directed graphs linking places and transitions



Token (marker)

- Tokens are marker, which illustrates the actual system state

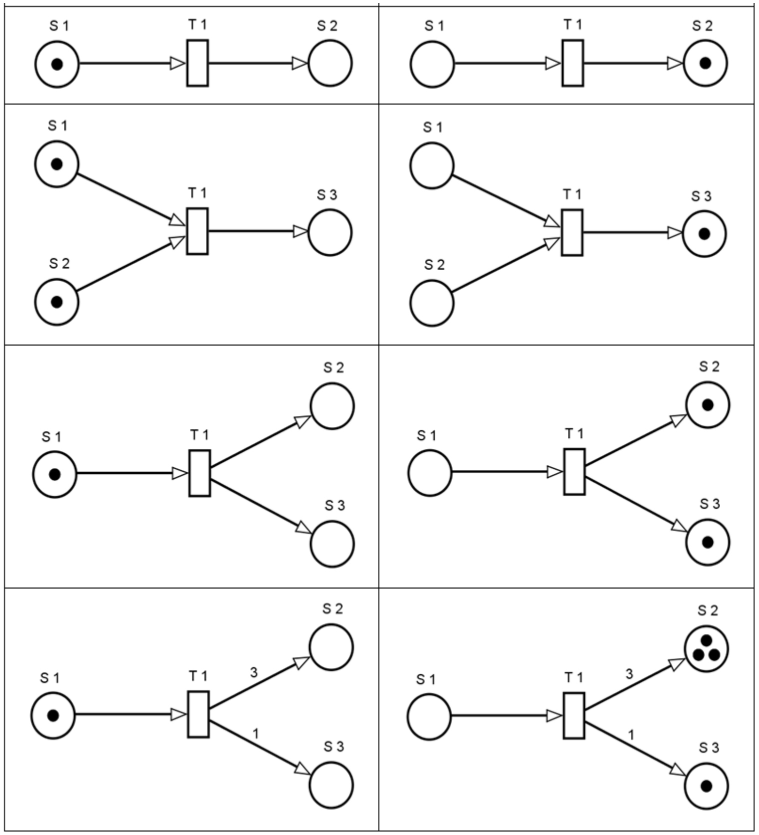
Rules



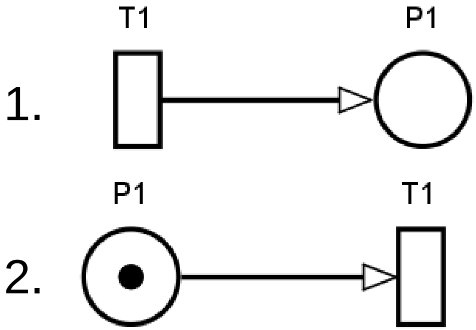
- Places can have 0 to ∞ tokens, but are limited to its defined capacity
- Transitions are only activated, if all of the input places are marked with at least one token
- Once activated, transitions firing randomly or to defined delays (fixed number or random distributed)
- Edges can only link transitions to places, not two places (or transitions) with each other

Rules II

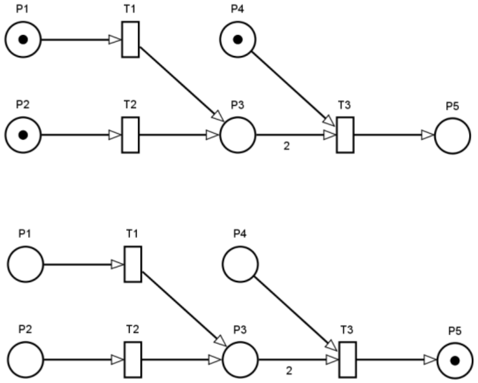
- Edges can be weighted:
  - An input arc with weight three requires three tokens in the adequate input place to activate the transition
  - An output arc with weight three requires insert of three tokens in the adequate place
  - No number equals a weight of 1
- When transition fires,the number of tokens corresponding to the weight of the input arc will be removed, and the number corresponding to the weight of the output arc will be produced in the output places



Special nets

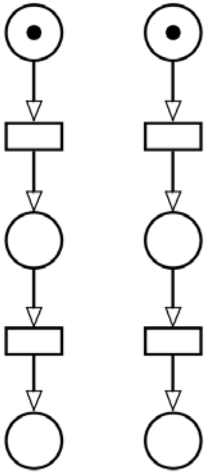


- In petri nets there is a way to:
  - 1. Produce new tokens with a source
  - 2. Delete tokens with a sink

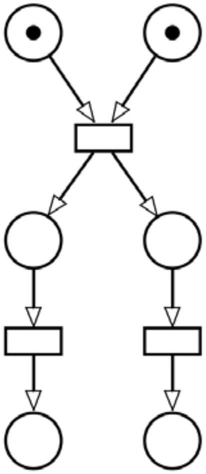


- A petri net is called dead, if all transitions are dead. This means no transition can never be activated again. After you wait long enough, the petri net at the right side will be dead, because all tokens moved to place 5

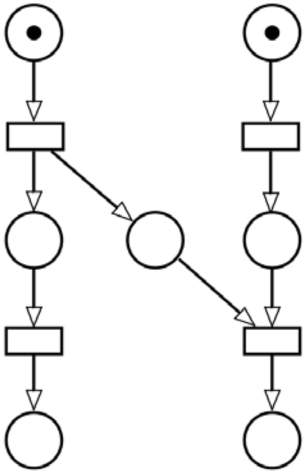
Special nets



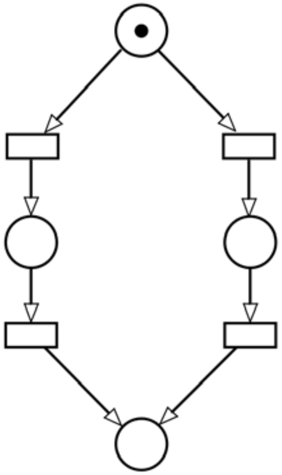
Concurrency



Synchronization

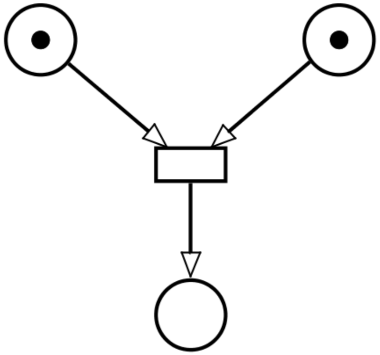


Communication

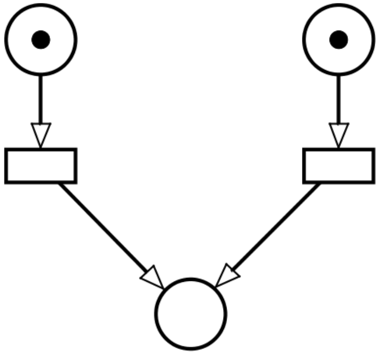


Conflict

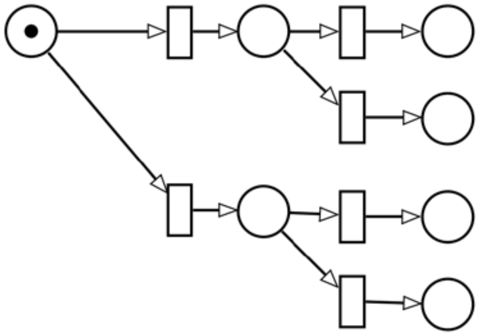
Petri nets as fault tree or event tree



And-Operator



Or-Operator



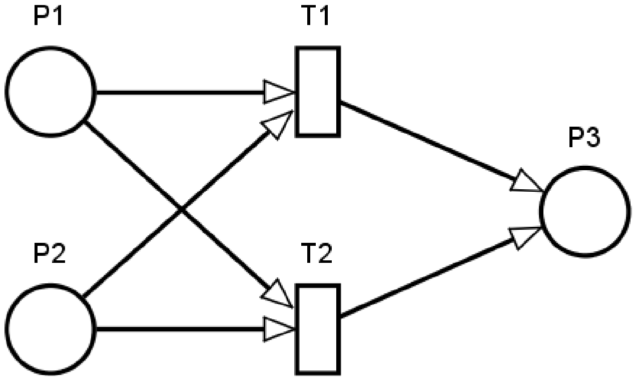
Event tree

Remember: Every fault tree/event tree can be modelled with a petri net, but not every petri net can be modelled as a fault tree/event tree

Formal notation

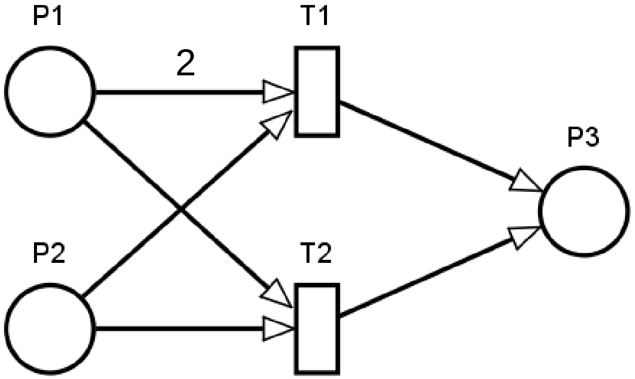
- Every net (*N*) is a triple: *N=(P,T,E)*
- *P*=Places, *T*=Transitions, *E*=Edges
- Intersection of *P* and *T* is zero: $P \cap S = 0$
- *E* is always part of the union of *PxT* and *TxP*: $E \subseteq (T \times P) \cup (P \times T)$

- Example:



$$P = \{P_1, P_2, P_3\}$$
$$T = \{T_1, T_2\}$$
$$E = \{(P_1, T_1), (P_1, T_2), (P_2, T_1), (P_2, T_2), (T_1, P_3), (T_2, P_3)\}$$

Matrix notation for *E*



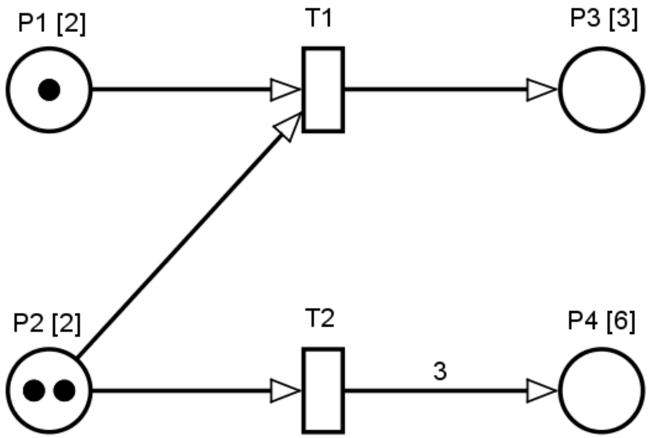
- *E* consist of an input and a output matrix: $E = E^{+} - E^{-}$

- Input matrix: $$E^{+} = \begin{array}{c c c} t_1 & t_2 & \\ \begin{pmatrix} 2 & 1 \\ 1 & 1 \\ 0 & 0 \end{pmatrix} & & \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array} \end{array}$$

- Output matrix: $$E^{-} = \begin{array}{c c c} t_1 & t_2 & \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{pmatrix} & & \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array} \end{array}$$

Complete description of a Petri Net

- To describe a Petri Net completely, you need also information about the initial marking and the capacity of the places



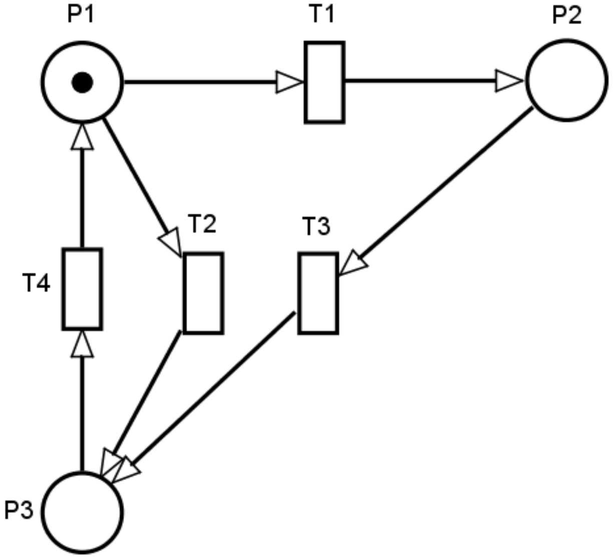
- Initial marking (m_0): $m_0 = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 0 \end{pmatrix}$

- Capacity (k): $k = \begin{pmatrix} 2 \\ 2 \\ 3 \\ 6 \end{pmatrix}$

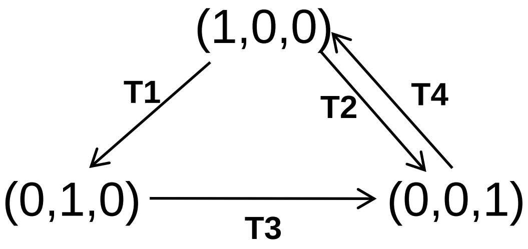
Reachability graph

- Shows all possible system states and their way to other states by fired transitions

Petri Net



Reachability Graph



Characteristics

Pros

- Simple design – easy to learn
- Graphical illustration
- Lot of tools available, in which changes of firing rates are easy to implement

Cons

- Nets grow very fast
- No guidelines for correct build up of a net – every net looks individual
- Big nets are difficult to understand

Extensions

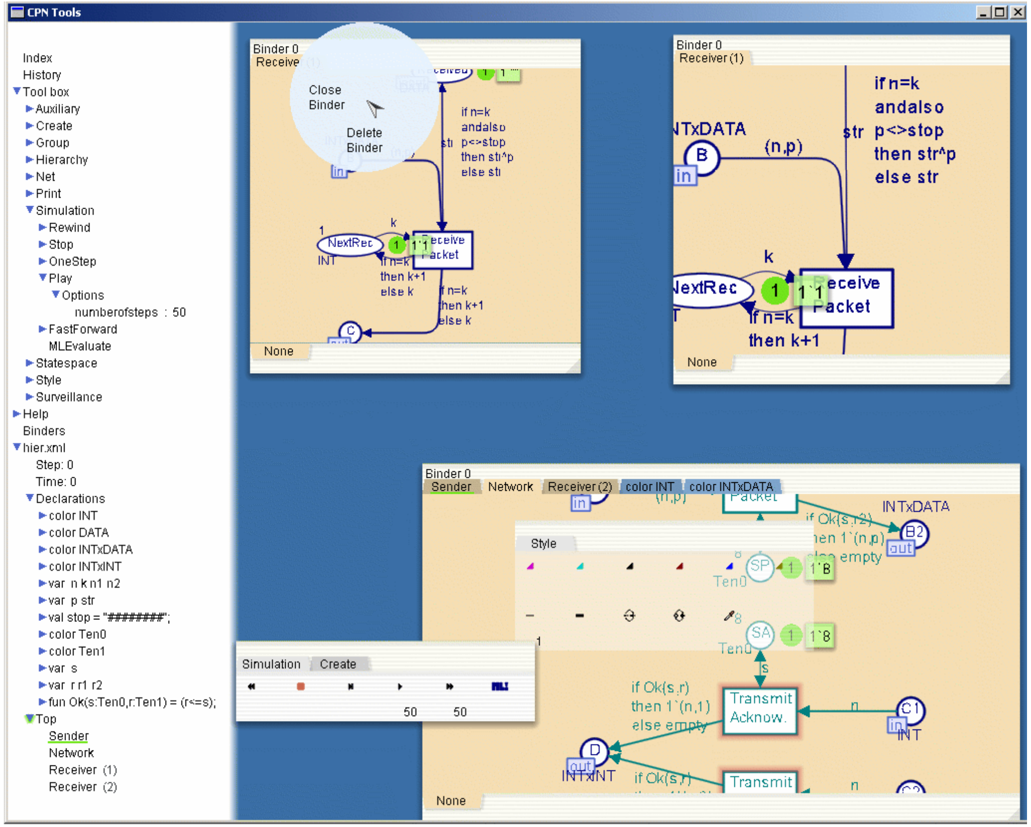
- Today simple Petri Nets can‘t fullfill all the needs from scientists, so there have been some extensions, which make the method more powerfull

- Coloured Petri Nets (CPN)
  - CPN combines simple Petri Nets with programming. With Coloured Petri Nets it‘s possible to give every token a colour, which allows it to differentiate between tokens.

- Timed Petri Nets
  - By add a specific firing delay to every transition, a Petri Net becomes a Timed Petri Net. This could be a determined time or a random distribution.

- There exist also a combination of these two extensions called Stochastic Coloured Petri Nets (SCPN)

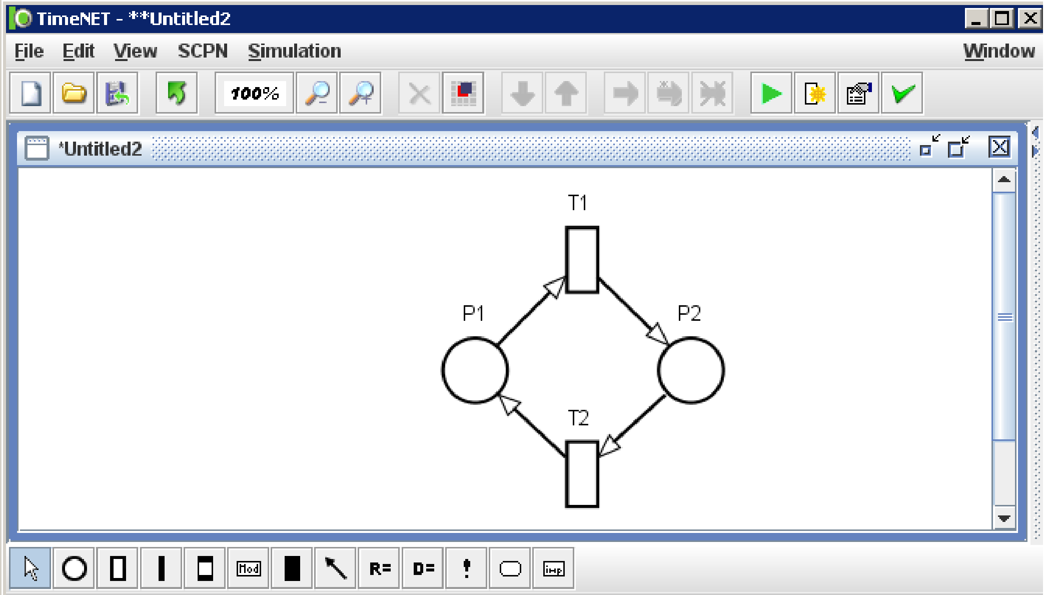
Simulation tools – CPN-Tools



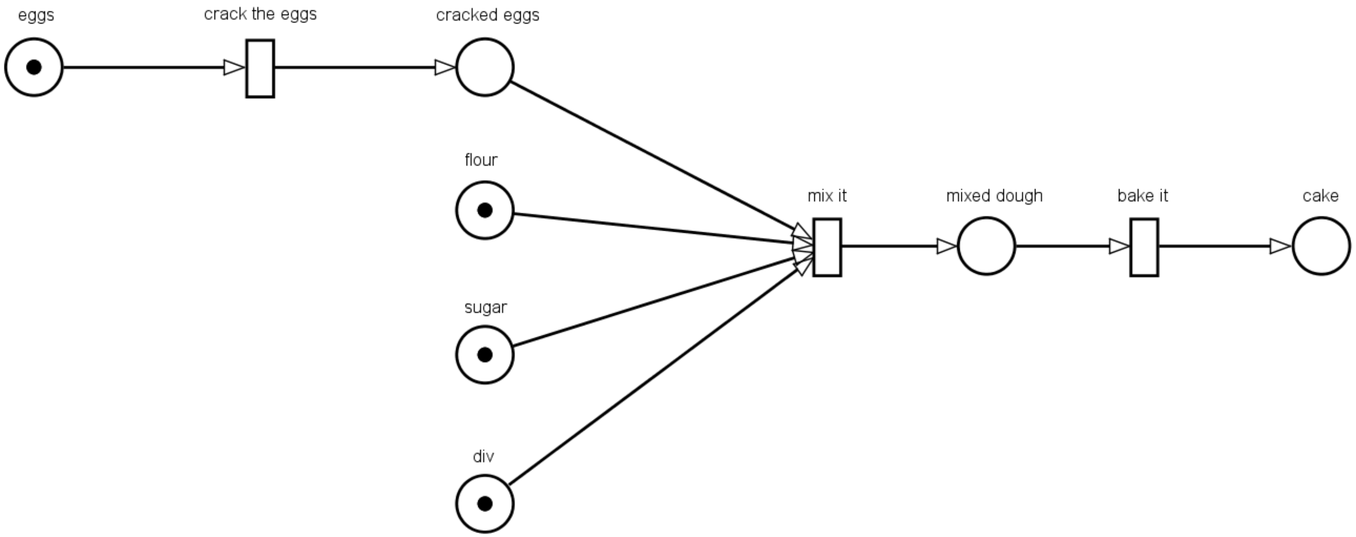
- Developed at Aarhus University (DK)
- Very Common, a lot of expirience
- High capacity allows huge nets
- Uncommon user interface
- No user guide, but tutorial on the web
- Freeware

Simulation tools - TimeNet

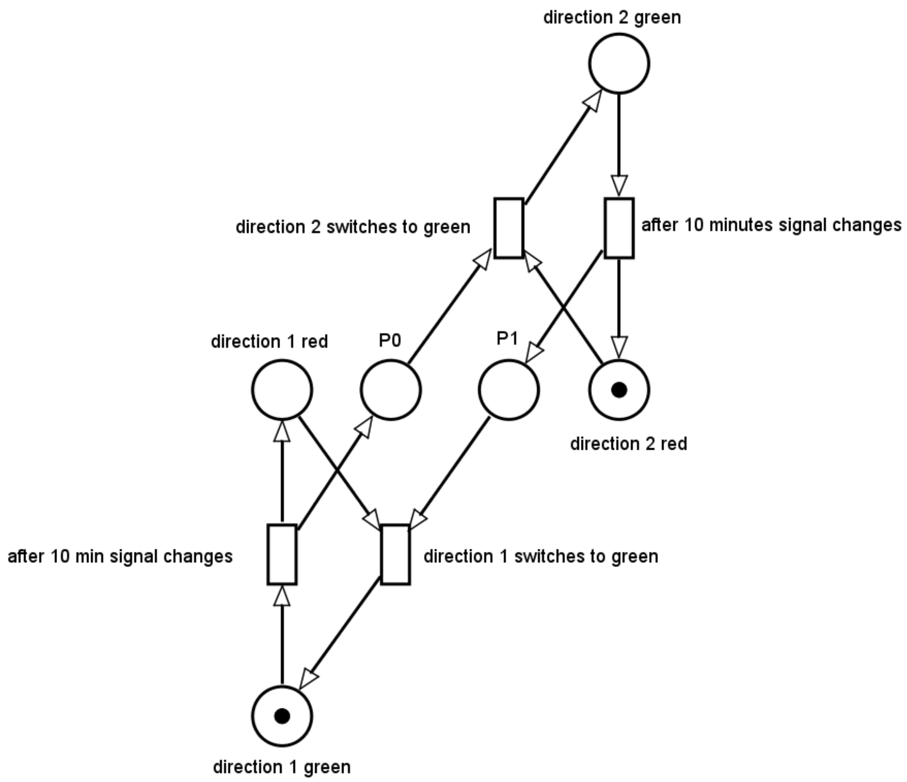
- From TU Ilmenau (DE)
- Newest version works with Java user interface
- Simple to understand
- Good Userguide
- Limited capacity
- Free for academic use



Example I – bake a cake



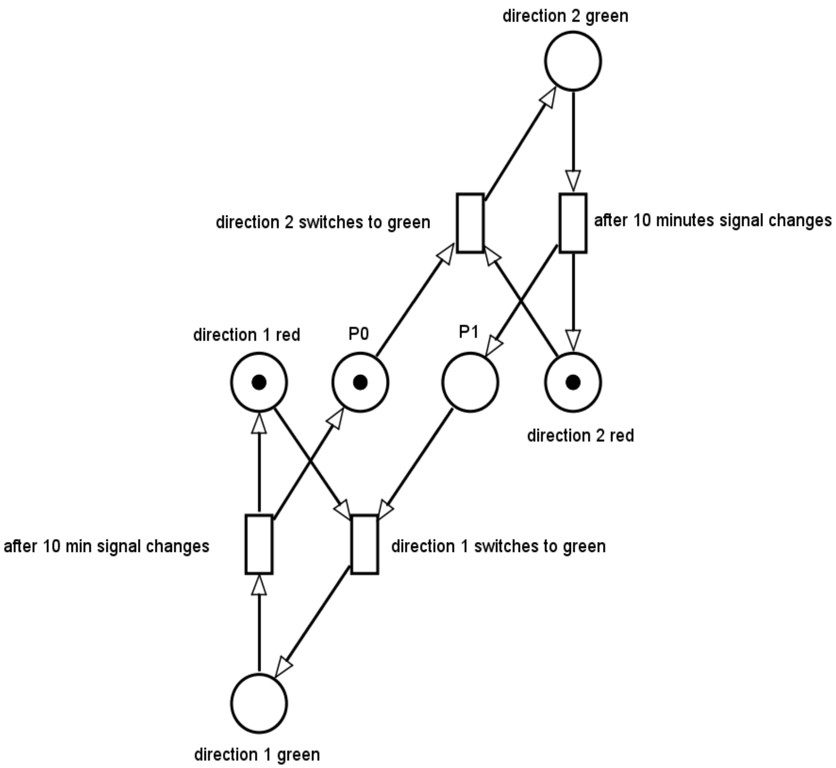
Example II – traffic signal

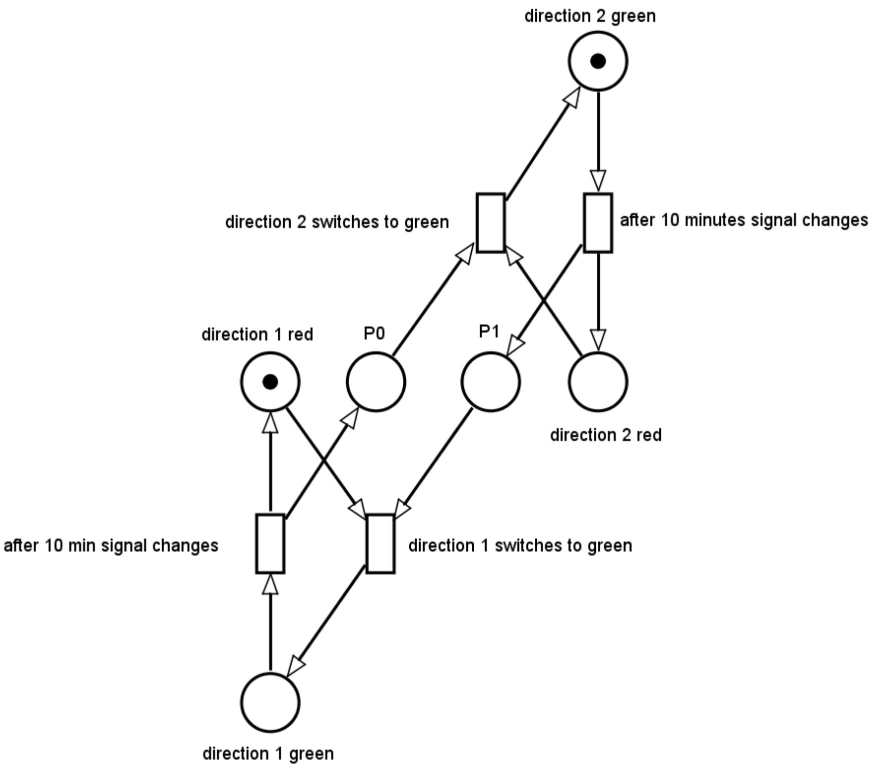

direction 2 green
direction 2 switches to green
after 10 minutes signal changes
direction 1 red
P0
P1
direction 2 red
after 10 min signal changes
direction 1 switches to green
direction 1 green

- Starting with one signal green, the other red
- 10 minutes one direction is allowed to drive
- After changing to red, there is some time to switch (every car has to pass the section)
- When switching is completed, direction 2 can drive

Example II – traffic signal

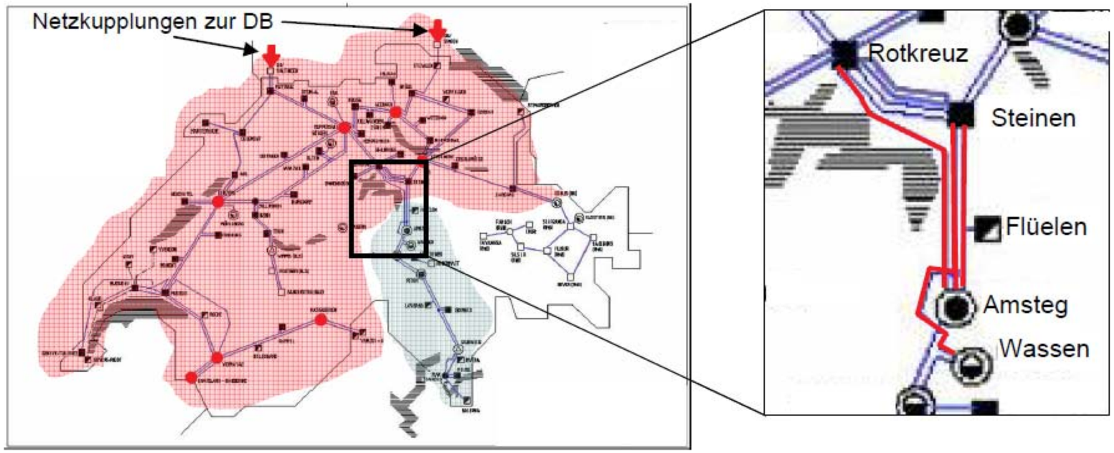
The signal after the first and the second switch. After two switches, the game starts again from the opposite side


direction 2 green
direction 2 switches to green
after 10 minutes signal changes
direction 1 red
P0
P1
direction 2 red
after 10 min signal changes
direction 1 switches to green
direction 1 green


direction 2 green
direction 2 switches to green
after 10 minutes signal changes
direction 1 red
P0
P1
direction 2 red
after 10 min signal changes
direction 1 switches to green
direction 1 green

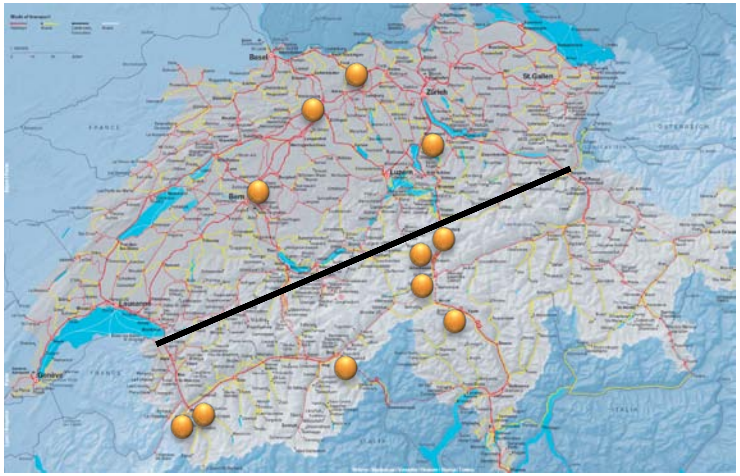
Example III – SBB Power supply

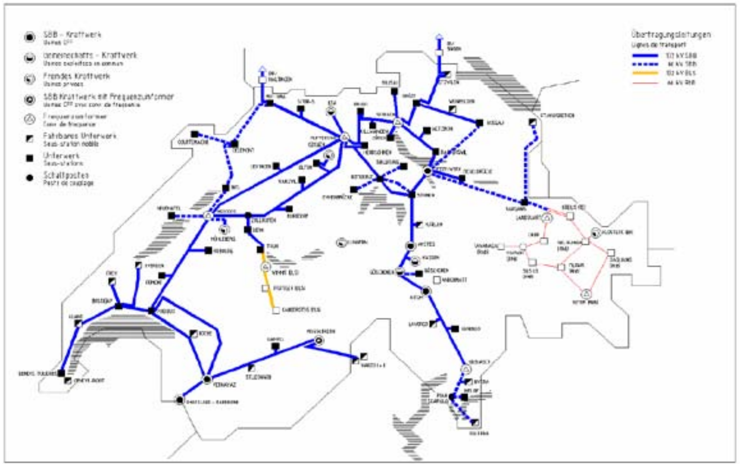
- Motivation
  - Blackout in June 2005 due to linebreak in Steinen
  - In the south (Uri/Ticino) over a third of the power production is located
  - In the north, the power from the south is needed, so north island broke down due to sudden power drop
  - South island got instable due to sudden load drop and finally also broke down



Example III – SBB Power supply

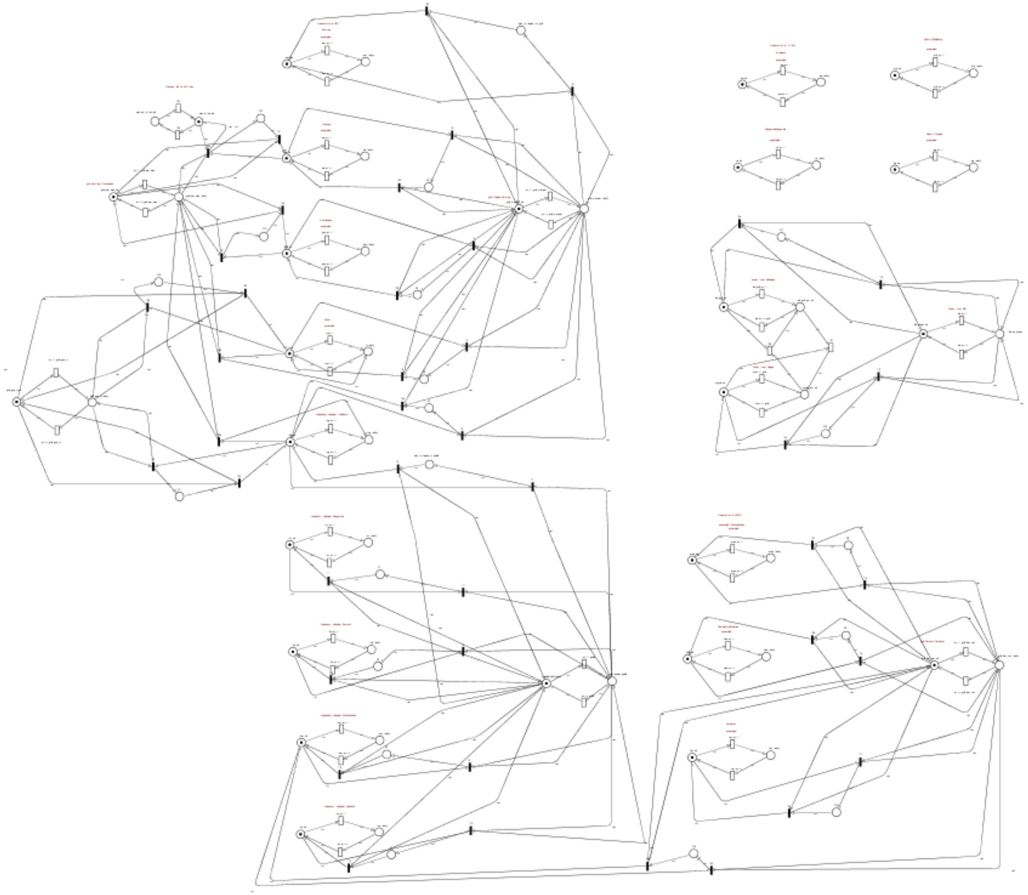
- Goal
  - Calculate the available power in the northern part of Switzerland (above black line)
  - Problem: 7 out of 11 power plants are below the line in the mountains
  - To bring the power in the North only one connection for the Ticino and one for the Valais (two with the Lötschberg) can be used
  - Fail of one of these connections could lead to blackout





Example III – SBB Power supply

- Implementation
  - Stochastic Petri Net
  - Power plants and power lines are modelled with two places (ok/fail)
  - If connection fails, all power plants of these region switch to fail mode
  - If plant is in ok mode, the power reaches the desired region



Bibliography

Books

- C. A. Petri. “Kommunikation mit Automaten“, Schriften des Rheinisch-Westfälischen Institutes für instrumentelle Mathematik an der Universität Bonn, (1962) – “Invention“ of Petri Nets
- W.G. Schneeweiss. Petri Nets for Reliability Modeling, LiLoLe Verlag (1999), ISBN 3-934447-00-7
- L. Priese, H. Wimmel. Petri-Netze, Springer Verlag (2008), ISBN 978-3-540-76970-5
- P.J.Haas. Stochastic Petri Nets, Springer Verlag (2002), ISBN 0-387-95445-7

Websites

- http://www.informatik.uni-hamburg.de/TGI/PetriNets/ - everything about Petri Nets
- http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/java/Braunl/ - simple Petri Net simulator