

Petri Nets

1. Basic Petri Net Model
2. Properties and Analysis of Petri Nets
3. Extended Petri Net Models

Petri Nets

⇒ Systems are specified as a directed bipartite graph.

The two kinds of nodes in the graph:

1. *Places*: they hold the distributed state of the system expressed by the presence/absence of tokens in the places.
2. *Transitions*: denote the activity in the system

⇒ *The state of the system*: captured by the marking of the places (number of tokens in each place)

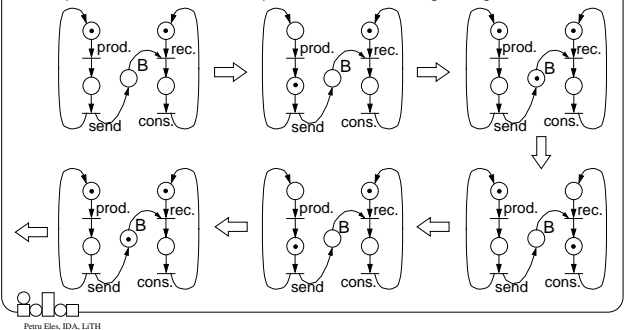
Petri Nets (cont'd)

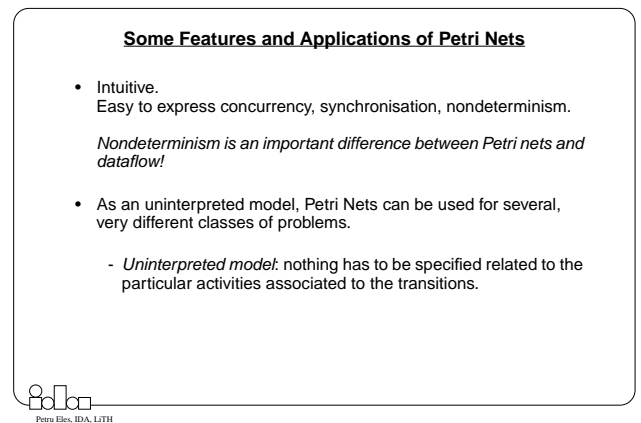
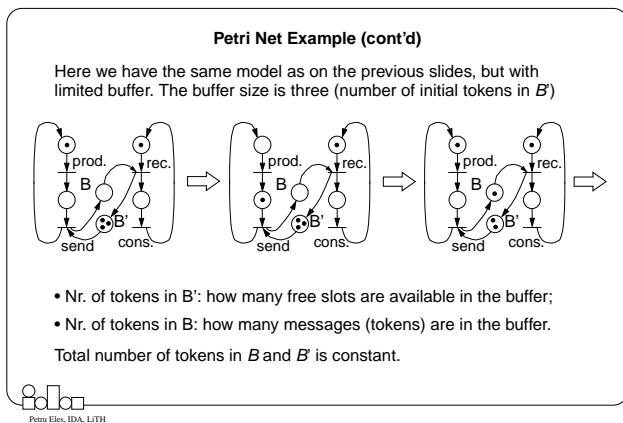
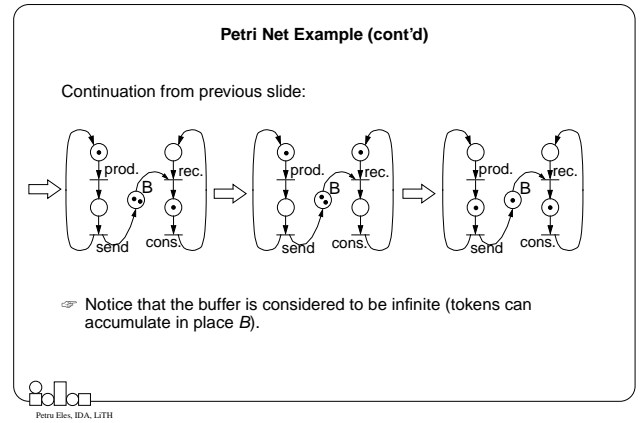
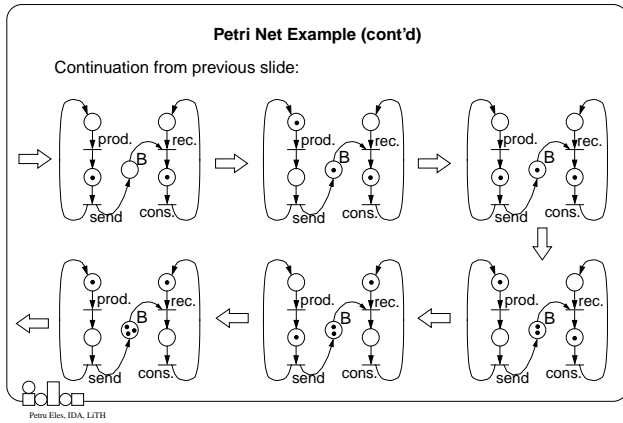
⇒ The dynamic evolution of the system: determined by the firing process of transitions.

- A transition may fire whenever all its predecessor places are marked.
- If a transition fires it removes a token from each predecessor place and adds a token to each successor place.

Petri Net Example

A producer and a consumer process communicating through a buffer:





Some Features and Applications of Petri Nets (cont'd)

- Petri Nets have been intensively used for modeling and analysis of industrial production systems, information systems, but also
 - Computer architectures
 - Operating systems
 - Concurrent programs
 - Distributed systems
 - Hardware systems



Properties and Analysis of Petri Nets

☞ Several properties of the system can be analysed using Petri nets:

- **Boundedness:** the number of tokens in a certain place does not exceed a given limit.
If this limit is 1, the property is sometimes called *safeness*.
 - You can check that available resources are not exceeded.



Properties and Analysis of Petri Nets (cont'd)

- **Liveness:**
 - A transition t is called live if for every possible marking there exists a chance for that transition to become enabled.

The whole net is live, if all its transitions are live.

 - liveness is important in order to check that a system is not deadlocked.
- **Reachability:** given a current marking M of the net, and another marking M' , does there exist a sequence of transitions by which M' can be obtained?
 - You can check that a certain desired state (marking) is reached.
 - You can check that a certain undesired state is never reached.



Properties and Analysis of Petri Nets (cont'd)

Mathematical tools are available for analysis of Petri Nets.



The properties discussed above can be formally verified.

☞ Petri nets (like dataflow systems) are *asynchronous concurrent*.

- Events can happen at any time.
- There exists a partial order of events:



Extended Petri Net Models

Basic Petri Net models have a limited expressive power.

☞ Timed Petri Nets

- Transitions have associated times (time intervals)
- Tokens are carrying time stamps.

- With timed Petri nets we can model the timing aspects

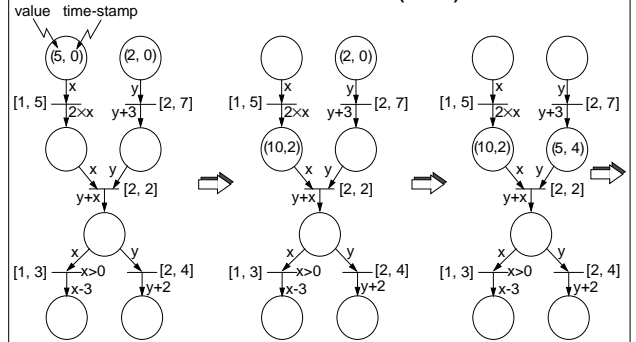
☞ Coloured Petri Nets

- Tokens have associated values
- Transitions have associated functions

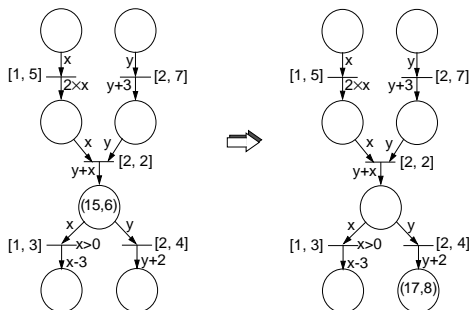
- Coloured Petri Nets are similar to dataflow models (but also capture nondeterminism!).



Extended Petri Net Models (cont'd)



Extended Petri Net Models (cont'd)



Extended Petri Net Models (cont'd)

- Extended Petri Nets have a larger expressive power than classical Petri Nets.



Analysis is more complex; the formal analysis of properties can take unacceptably large amounts of time (memory).

- Simulation of the Petri Net can be used in order to verify the system and to estimate performance



Summary

- Petri Nets are a mixture of dataflow and state-based model. *Places* hold the distributed state of the system (represented by the marking); *transitions* denote the activity of the system.
- Petri nets elegantly capture concurrency, synchronisation, and nondeterminism.
- A large class of problems can be solved using Petri Net modelling; system properties like boundedness, liveness, and reachability can be formally analysed.
- Petri nets, like dataflow, are asynchronous, concurrent models. Events can happen at any time; there exists a partial order of events.



Summary (cont'd)

- Basic Petri Nets are limited in their expressive power
 - in timed Petri Nets an explicit notion of time has been introduced;
 - in coloured Petri nets tokens have associated values.
- Formal reasoning about extended Petri Net models is very difficult because of complexity issues. Simulation of the models is often used for system validation.



Discrete Event Models

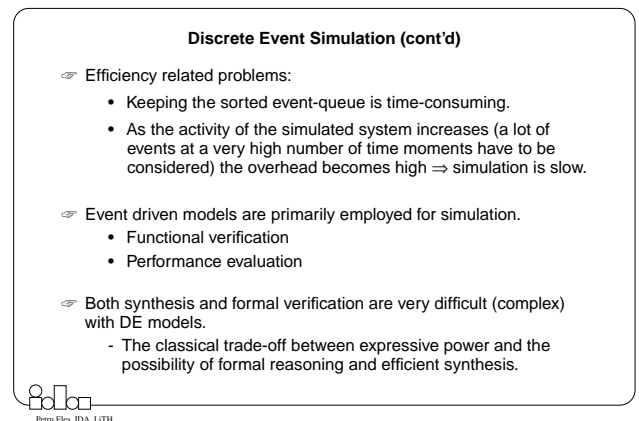
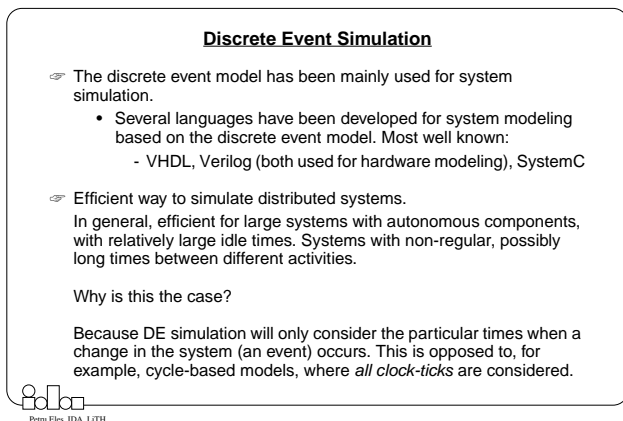
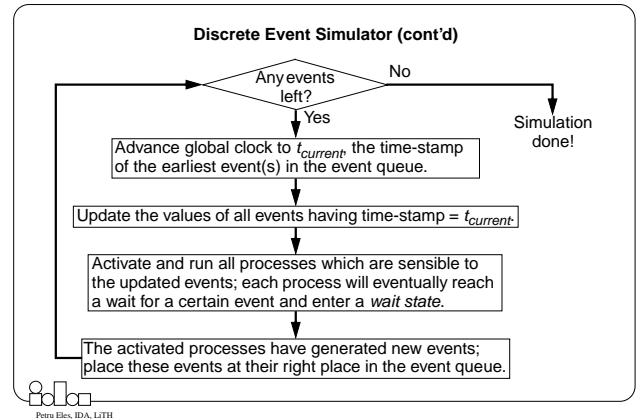
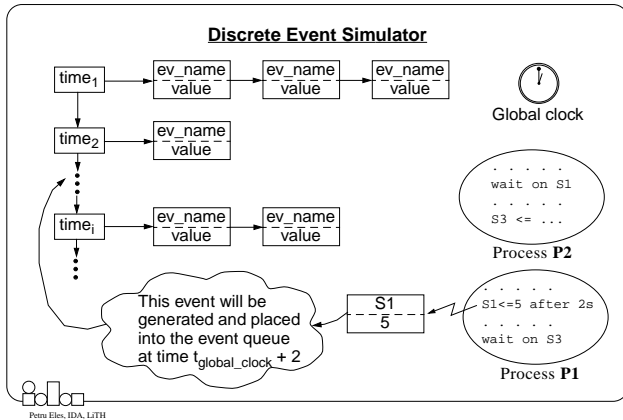
1. What Is a Discrete Event Model?
2. Discrete Event Simulation
3. Efficiency of Discrete Event Simulation
4. Potential Ambiguities in Discrete Event Simulation



Discrete Event Models

- The system is a collection of processes that respond to events.
- *Each event carries a time-stamp indicating the time at which the event occurs.*
- *Time-stamps are totally ordered.*
- A Discrete Event (DE) simulator maintains a *global event queue sorted by the time-stamps*. The simulator also keeps a single global time.





Summary

- Discrete Event is a very powerful modelling technique, in terms of expressive power.
Models are concurrent and asynchronous.
The timing model is also very general. Delays on computation or communication can be arbitrary.
- We pay for the large expressive power by the reduced potential of formal reasoning and efficient synthesis.
Discrete event models are mainly used for simulation.
- Simulation is based on maintaining a unique, sorted event queue.
This can create problems with simulation efficiency for large system models.
- In order to avoid problems with zero delay computations, “delta delays” are used (e.g. in VHDL).