

COMP4100 Lectures in 2007

by Malcolm Newey

Introduction to Petri Nets

Lecture FM-15 May 2 2007

References:

Fundamentals of Software Engineering,

Ghezzi, Jazayeri & Mandrioli, pp 174-199

Software Engineering Fundamentals, Behforooz & Hudson, pp 168-176

Petri Net World: <http://www.informatik.uni-hamburg.de/TGI/PetriNets/>

Purpose and Scope

What are Petri Nets are used for?

- The specification of asynchronous systems;
- A generalization of finite state machine;
- Manufacturing systems;
- As a general theory of discrete parallel systems (the first);

Like FSMs, Petri nets are expressible both graphically and symbolically.

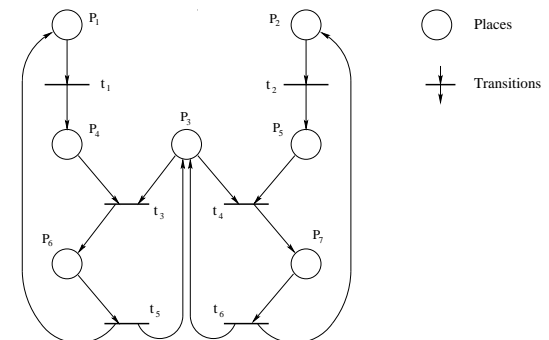
- Pictures for people and mathematics for machines.

Background

- Carl Adam Petri's 1962 PhD thesis was *Kommunikation mit Automaten*
- There are thousands of papers on PNs
- Many varieties - algebraic PNs, product nets, coloured PNs
- Many industrial examples of CPNs reported
http://www.daimi.au.dk/CPnets/intro/example_indu.html

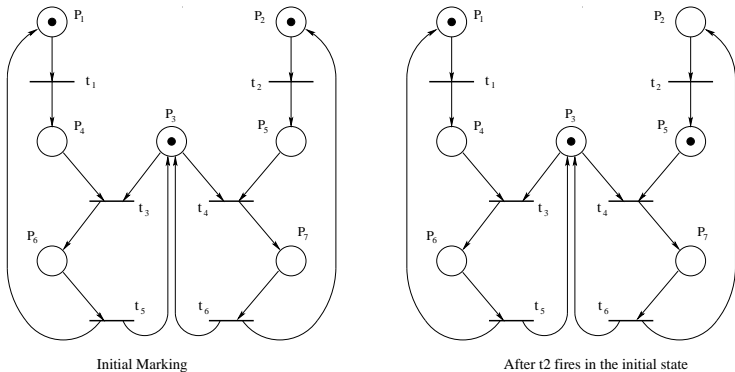
Example Petri Net

A simple example showing the connection of *places* and *transitions*. Each transition has some number of input places and output places.



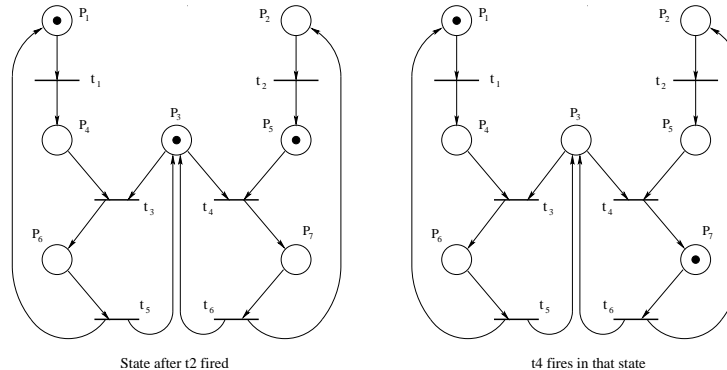
Dynamics of a Petri Net

A transition can fire if all its input places are marked. As a consequence all its output places become marked.



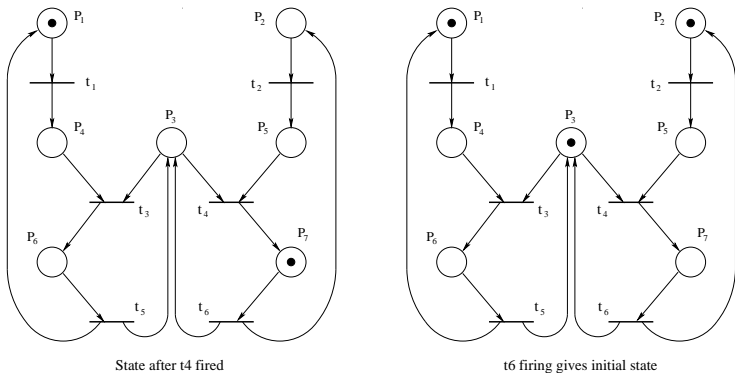
Dynamics of a Petri Net - II

Both the t_1 and the t_4 transitions could fire at this point; no others can.



Dynamics of a Petri Net - III

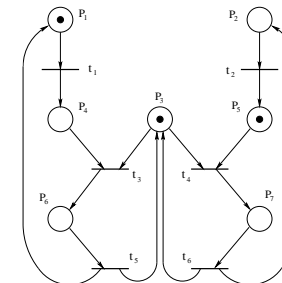
The example continues in one of just two possible ways.



Possible Interpretation

Our running example could be an abstraction of many different systems but one comes to mind. It is that of a system of two processes (A and B) which (each) repeatedly request and exclusively use a shared resource, R .

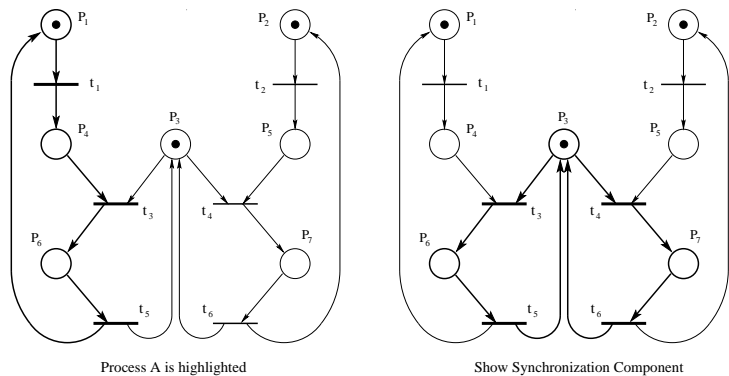
- P_4 has a mark if A is waiting for R
- P_5 has a mark if B is waiting for R
- P_1 has a mark if A is not waiting for R
- P_2 has a mark if B is not waiting for R
- P_3 has a mark if the resource is free
- P_6 has a mark if R is allocated to A
- P_7 has a mark if R is allocated to B



Identifying the 'Sequential' Components

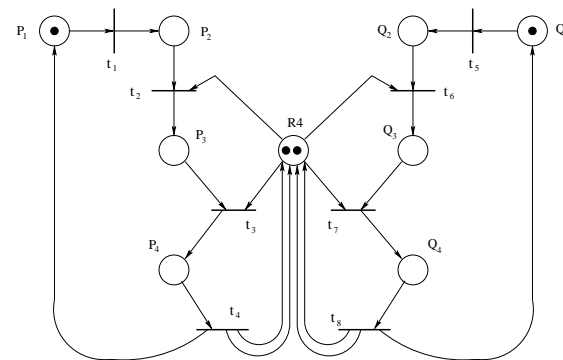
The component on the left captures the behaviour of process A.

The middle component captures the possible behaviours of the synchronizing agent.



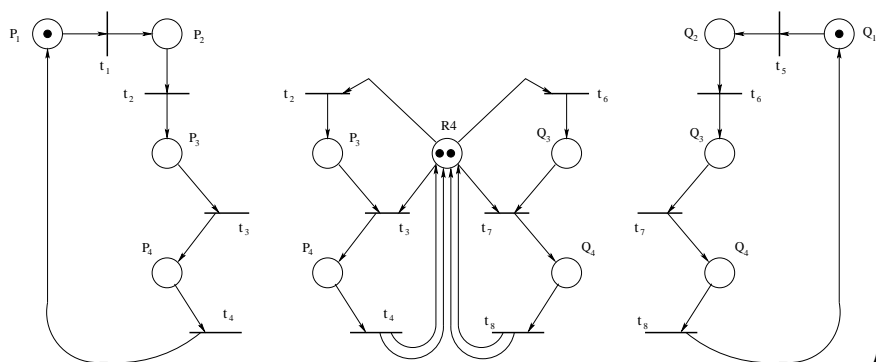
A System with Deadlock

This Petri net models 2 interacting processes that each repeatedly requests both of the 2 available units of a shared resource.



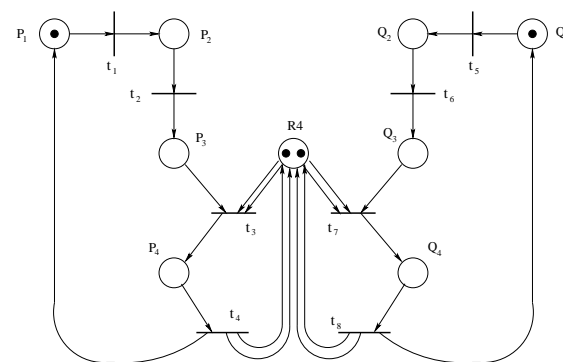
Showing the components

Note that the responsibility for causing deadlock rests with the resource allocator component.



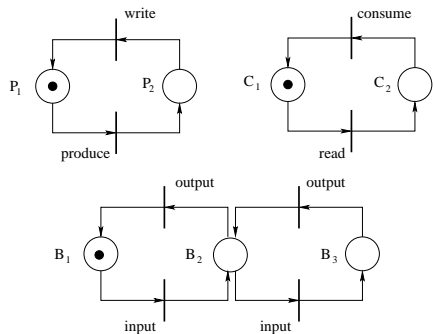
Getting Rid of Deadlock

This shows the standard way of avoiding the deadlock in the last system. It is done by insisting that each process allocate both units at once, rather than piecemeal.



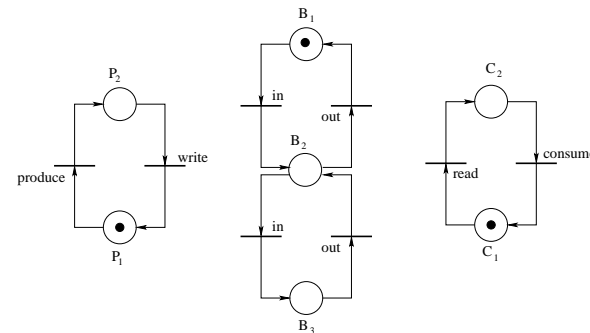
Producer-Consumer Systems

The three PNs below model, respectively, a producer, a consumer and a double buffer. However there is no synchronization.



Coordinating Production and Consumption

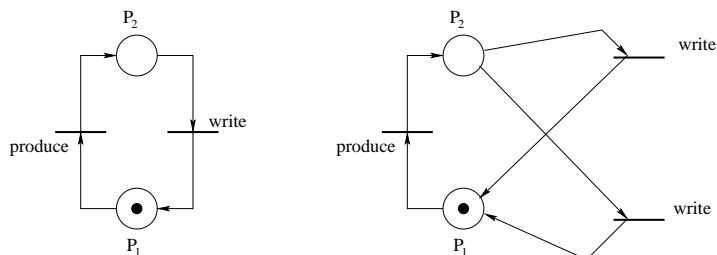
The write transition of the producer should *match* either one of the input transitions of the buffer. Similarly, the read transition of the consumer should match either one of the output transitions of the buffer.



Preparing for Integration

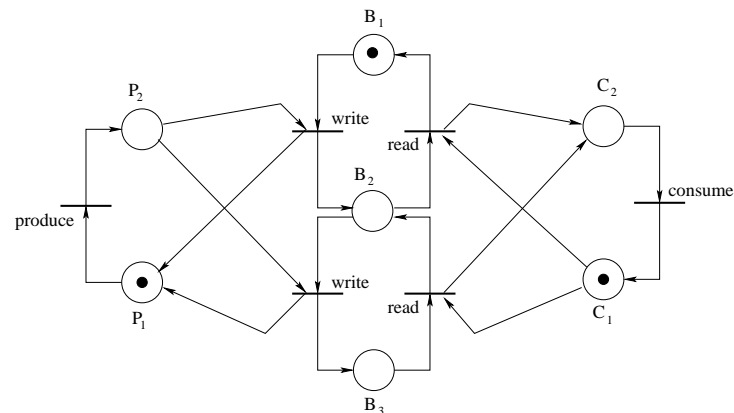
The behaviours of these two petri nets are the same.

The second is obtained from the first by introducing nondeterministic choice between two instances of the write transition.



Integrated PN for producer-consumer Instance

So here it is!!



Petri Net Properties

There are standard issues for systems being modeled.

- Liveness - are there future transitions?
- Boundedness - is max no. of tokens at a place limited?
- Reachability - is some state possible? (reachability tree is computable)
- Safety - is some state impossible?
- Fairness - does a transition get its proper share (whatever that is)?

Adding Priority

Compare the use of the standard conditional with guarded commands.

- Dijkstra guards (as in Ada, CSP) allow execution to be nondeterministic.
- The standard if-then-else enforces priority between available options.

Can add a priority function to Petri Nets.

The type of the function will be a mapping from transitions to natural numbers.

The extra semantics to describe the behaviour is captured by the rule "No transition can fire if there is an eligible transition of higher priority."

Tokens with Values

A big limitation of standard PNs is the fact that they model control flow without consideration of how it is affected by data.

A standard PN extension is to define tokens to have a numeric value that can be used in the computation of whether a transition is eligible for firing.

- The set of input tuples for a transition is the Cartesian product of the sets of tokens that are at the input places for that transition.
- Eligibility for firing is a predicate on input tuples. A tuple that satisfies this predicate is called a ready tuple.
- The value of the output token(s) is a function over ready tuples.

Tokens with Values - an Example

- In the following PN fragment, we will take the *predicates* for the transitions to be as follows:

$$t_1: p_1 < 3$$

$$t_2: p_1 \geq p_2$$

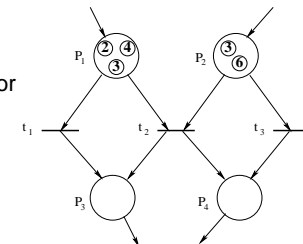
$$t_3: p_2 > 3$$

- We also take the *transformation function* for the transitions to be:

$$t_1: p_1 + 1$$

$$t_2: p_1 - p_2$$

$$t_3: p_2 - 1$$



The Net Changes State

- All transitions are eligible to fire.
- The ready tuples for t_2 , for example, are $\{(4, 3), (3, 3)\}$. The sets of ready tuples for t_1 and t_3 are just the singleton sets $\{(2)\}$ and $\{(6)\}$.
- This is what happens when transition t_2 fires using the ready tuple $(4, 3)$.

