Applications of Petri Nets in Manufacturing

Computational Models for Complex Systems

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa
http://pages.di.unipi.it/milazzo
milazzo@di.unipi.it

Laurea Magistrale in Informatica
A.Y. 2019/2020

Introduction

As an example of application of Petri nets in the context of manufacturing, we describe the method proposed in
J. Ezpeleta, J.M. Colom, and J. Martinez. "A Petri Net Based Deadlock Prevention Policy for Flexible Manufacturing Systems". IEEE Trans. on Robotics and Automation, vol. 11, n. 2, 1995

More recent application domains for Petri nets include:

- Performance evaluation
- Business processes
- Biology

In these application domains, Petri nets are usually extended with stochastic time and simulation approaches are used to perform analises

Flexible Manufacturing Systems

Flexible Manufacturing Systems (FMS) are composed by:

- A set of workstations (or machines) where products must be processed
- A flexible transport system (e.g. robot arms) which load and unload the workstations

Flexible Manufacturing Systems

The sequence of operations performed in order to manufacture a product is called working process (WP)

A system resource is an element of the manufacturing system that is able to hold a product (for storage, operation, transport,...)

- Examples of resource: machines, robot arms, ...

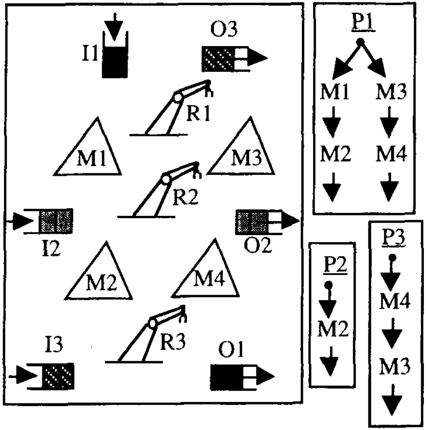
A FMS can execute more than one WPs at the same time, for the production of different products

- different WPs are executed concurrently
- resources are shared by the concurrent WPs

Flexible Manufacturing Systems

Example of FMS:

- I*i* and O*i* are the input and output buffers of product *i*
- P*i* describes the WP of product *i*
- R1,R2,R3 load and unload the buffers and the machines close to them



Deadlocks

Competition for resources by concurrent WP can cause deadlocks.

A deadlock situation is due to a wrong resource allocation policy:

- deadlocks are caused by circular wait situations for a set of resources

Deadlock situations have to be characterized in order to avoid the system to reach them (deadlock prevention/avoidance) or to recover from such a situation (deadlock recovery)

Deadlocks

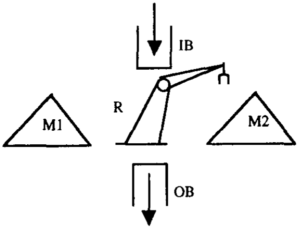
Deadlock prevention: a control policy is established in a static way in order to guarantee that deadlocks cannot be reached

Deadlock avoidance: the system execution is monitored and at every time the control policy determines (on-line) how to proceed in order to avoid deadlock

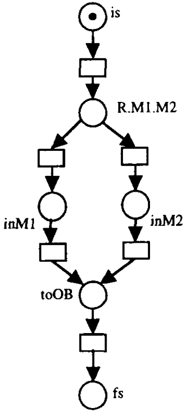
Deadlock recovery: the system is free to reach a deadlock state, but a control policy exists which can recognize the situation and restore a non-deadlock state

The proposed method falls in the deadlock prevention category

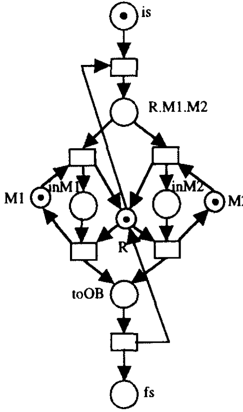
Modelling Flexible Manufacturing Systems



A robotized cell: products from the input buffer are processed either in machine M1 or M2



Petri net modelling the processing of the product



Final model with the resource capacity constraints (M1, M2 and R can hold one product each)

Modelling Flexible Manufacturing Systems

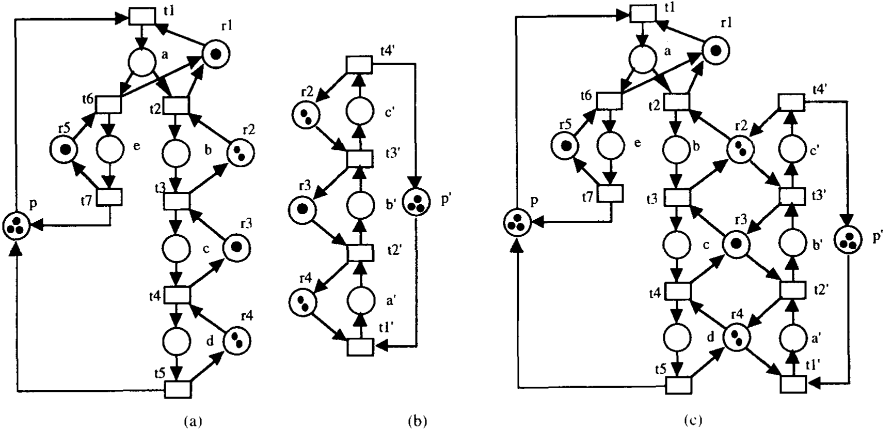
Constraints for the modeling of WPs:

- A WP has an initial and a final state (can be merged to represent continuous productions)
- Choices are allowed in a WP, but iterations are not
- Only one shared resource is allowed to be used at each state of the WP (i.e. a product can be held by only one resource at a time)
- Initial and final states do not use resources

Concurrency:

- several instances of the same WP (multiple tokens)
- different WPs sharing some of the resources (composed nets)

Modelling Flexible Manufacturing Systems



Two (separate) Petri nets modeling different WPs (with shared resources r2, r3,r4)

Composition of the two nets

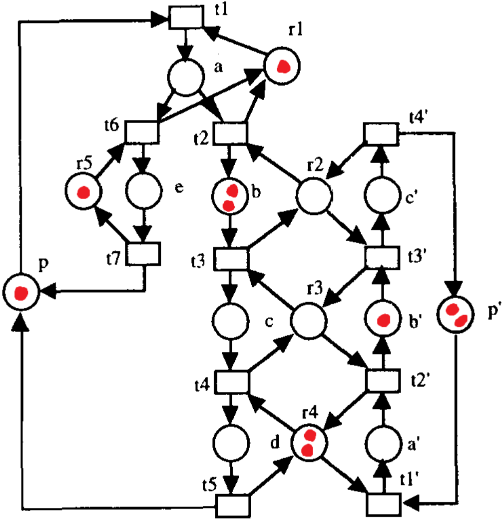
Modelling Flexible Manufacturing Systems

Each WP is deadlock free

The composition of the WPs can reach a deadlock

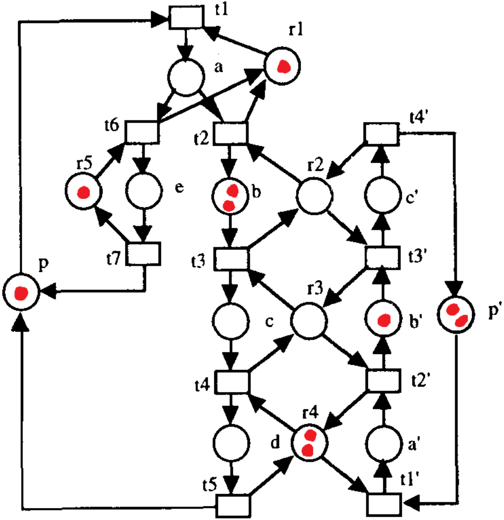
Modelling Flexible Manufacturing Systems

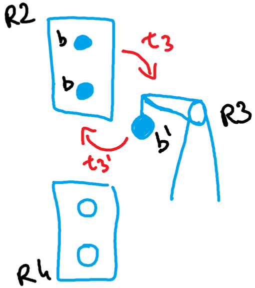
Deadlock marking: $2b$ are using $r2$ and wait for $r3$, b' is using $r3$ and waits for $r2$



Modelling Flexible Manufacturing Systems

Intuitively...





Siphons

The proposed deadlock prevention method is based on the identification of siphons

Given a place p, let

- $Pre(p)$ be the transitions having p has an output place
- $Post(p)$ be the transitions having p as an input place

Given a set of places $S = p_1, p_2, \ldots$, let

- $Pre(S) = Pre(p_1) \cup Pre(p_2) \cup \ldots$
- $Post(S) = Post(p_1) \cup Post(p_2) \cup \ldots$

A siphon is a set of places S such that $Pre(S) \subseteq Post(S)$

Siphons and deadlocks

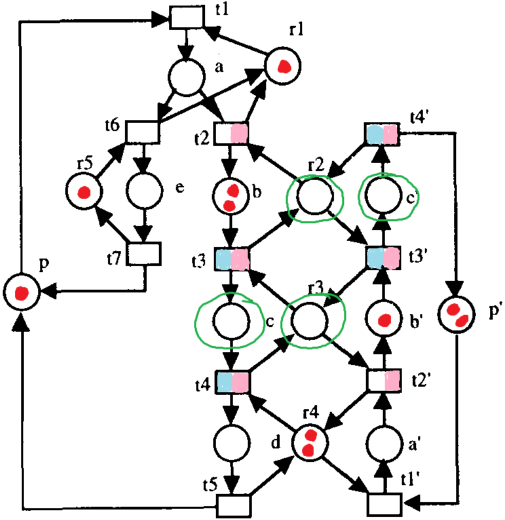
Theorem. A Petri net modeling a FMS is live (i.e. deadlock free) if and only if for every reachable marking m and for every (minimal) siphon S, it holds $m(S) \neq 0$.

$m(S)$ denotes the overall number of tokens in the set of places S

Note that this results does not hold in general for Petri nets, but only for nets constructed as we have seen in FMSs modelling

Siphons and deadlocks

The green places constitute a siphon ($Pre(S)$ are blue, $Post(S)$ are pink). They are actually all empty in the case of deadlock



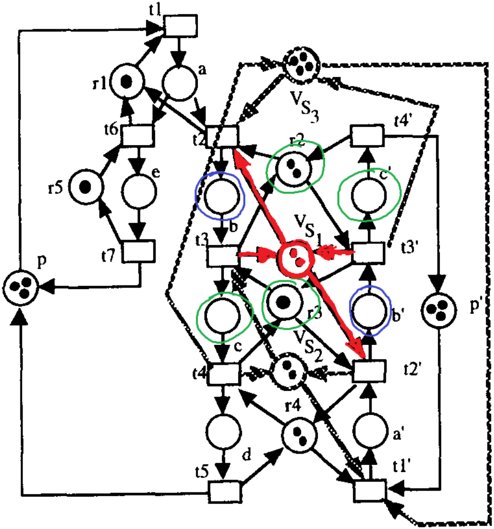
Siphons and deadlocks

Roughly speaking, siphons represent shared resources in which each transition releasing one of them is also a transition requiring another of them

- If all of the places of a siphon are empty, then all of the resources they represent are acquired
- In order for one of the resources to be released, one of the transitions in $Pre(S)$ has to be fired
- But the transitions in $Pre(S)$ are also in $Post(S)$, so they require tokens from the places in S... which are all empty!

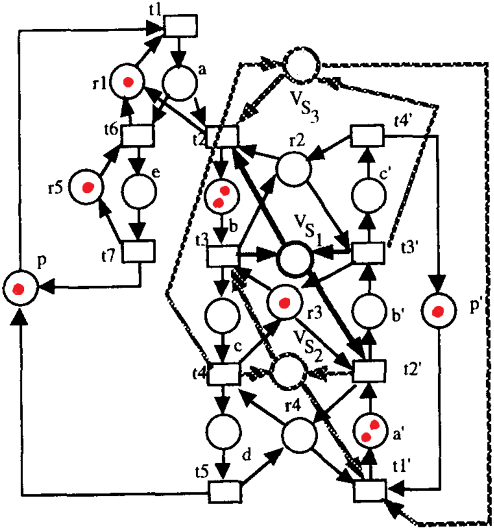
Deadlock prevention

Let's add additional places and tokens to constraint the use of resources involved in a siphon (at most 2 of b and b' can be present)



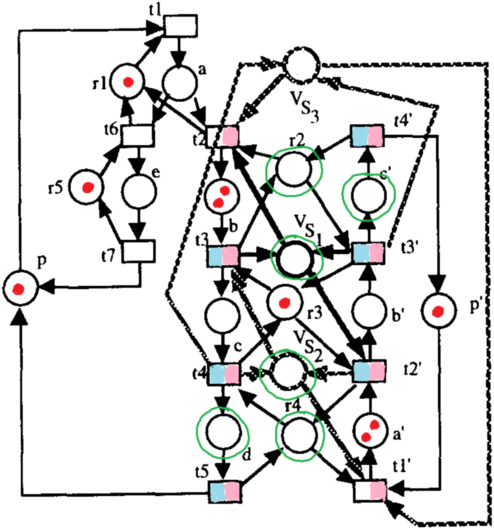
Deadlock prevention

Problem, the new places can create new deadlocks!!



Deadlock prevention

Indeed, we have created a new siphon...



Deadlock prevention

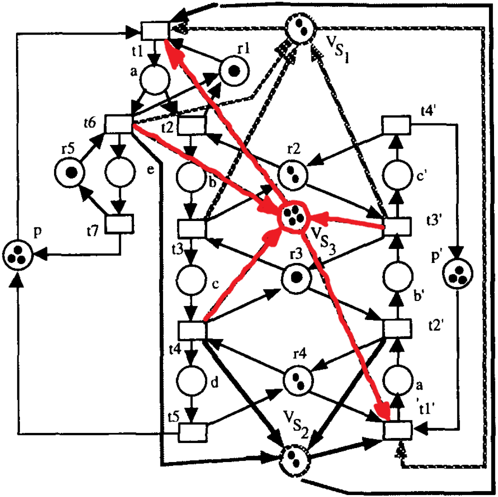
The deadlock prevention policy has to take into account all the possibile siphons

- not one by one...

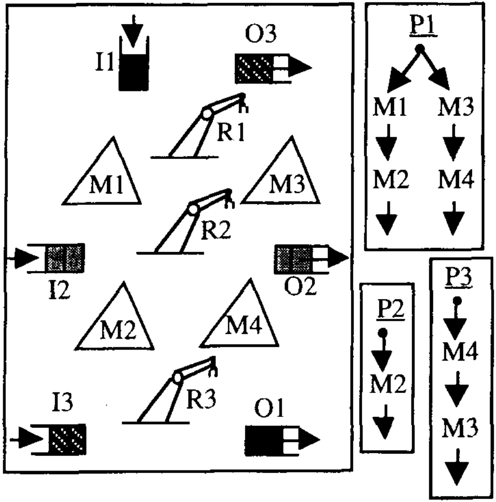
The idea:

- One place is added for each siphon in the net with a small enough number of tokens
- Every time the production of a new product starts, one token is removed from each of such places
  - from the very beginning product reserves its right to acquire the resources involved in the siphons
- As soon as the product reaches a place from where one of the siphons cannot be reached, it releases the corresponding token

Deadlock prevention



Case study



Siphons can be automatically computed

- for each subset of places S, check if $Pre(S) \subseteq Post(S)$

Siphons that can become empty can be computed

- for each siphon S, check if they do not support a place invariant

On the right, the list of possibly empty siphons of the case study

i	S_i	m_0
1	{P3M4,P1R3,R3,M4}	3
2	{P2R2,P2R2',P1R2,P1R2',P3M3,R2,M3}	3
3	{P2R2,P2R2',P1R2,P1R2',P3R1,M1,M3,R1,R2}	6
4	{P2R2',P1M2,P1R2',P3R2,M2,R2}	3
5	{P2R2',P1M2,P1R2',P3M3,M2,R2,M3}	5
6	{P2R2',P1M2,P1R2',P3R1,M1,M2,M3,R1,R2}	8
7	{P2R2,P2R2',P1R2,P1M4,P3R2,R2,M4}	3
8	{P2R2,P2R2',P1R2,P1M4,P3M3,M3,M4,R2}	5
9	{P2R2,P2R2',P1R2,P1M4,P3R1,M1,M3,M4, R1,R2}	8
10	{P2R2',P1M2,P1M4,P3R2,M2,M4,R2}	5
11	{P2R2',P1M2,P1M4,P3M3,M2,M3,M4,R2}	7
12	{P2R2',P1M2,P1M4,P3R1,M1,M2,M3,M4, R1,R2}	10
13	{P2R2,P2R2',P1R2,P1R3,P3R2,M4,R2,R3}	4
14	{P2R2,P2R2',P1R2,P1R3,P3M3,R2,R3,M3,M4}	6
15	{P2R2,P2R2',P1R2,P1R3,P3R1,M1,M3, M4,R1,R2,R3}	9
16	{P2R2',P1R3,P3R2,M2,M4,R2,R3}	6
17	{P2R2',P1R3,P3M3,M2,M3,M4,R2,R3}	8
18	{P2R2',P1R3,P3R1,M1,M2,M3,M4,R1,R2,R3}	11

Case study

The obtained deadlock-free Petri net!