Industrial Automation

(Automação de Processos Industriais)

Analysis of Discrete Event Systems

http://users.isr.ist.utl.pt/~jag/courses/api1213/api1213.html

Slides 2010/2011 Prof. Paulo Jorge Oliveira
Rev. 2011-2013 Prof. José Gaspar

Syllabus:

Chap. 6 – Discrete Event Systems [2 weeks]

…

Chap. 7 – Analysis of Discrete Event Systems [2 weeks]

Properties of DESs.

Methodologies to analyze DESs:

* The Reachability tree.
* The Method of Matrix Equations.

…

Chap. 8 – DESs and Industrial Automation [1 week]

Some pointers to Discrete Event Systems

History: http://prosys.changwon.ac.kr/docs/petrinet/1.htm

Tutorial: http://vita.bu.edu/cgc/MIDEDS/
http://www.daimi.au.dk/PetriNets/

Analyzers, and Simulators:
http://www.ppgia.pucpr.br/~maziero/petri/arp.html (in Portuguese)
http://wiki.daimi.au.dk:8000/cpntools/cpntools.wiki
http://www.informatik.hu-berlin.de/top/pnk/download.html

Bibliography:

* Cassandras, Christos G., **"Discrete Event Systems - Modeling and PerformanceAnalysis"**, Aksen Associates, 1993.
* Peterson, James L., **"Petri Net Theory and the Modeling of Systems"**, Prentice-Hall,1981
* **Petri Nets and GRAFCET: Tools for Modelling Discrete Event Systems** R. DAVID, H. ALLA, New York : PRENTICE HALL Editions, 1992

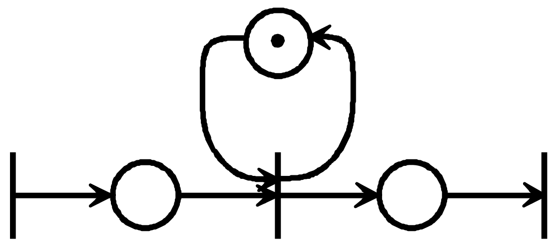
Properties of Discrete Event Systems

1. Reachability

Given a Petri net C=(P, T, I, O, μ_0) with initial marking μ_0 , the **set of all markings** that can be obtained is the **Reachable Set**, R(C, μ).

Note: **in general** R**(C, μ) is infinite!**

How to describe and compute R(C, μ)?



Reachability problem: Given a Petri net C with initial marking μ_0 , does the marking μ' belong to the set of all markings that can be obtained, i.e. $\mu' \in R(C, \mu)$?

Property usage: State μ **belongs / does not belong** to R(C, μ_0).
Net C has a **finite / infinite** Reachable Set.

Properties of Discrete Event Systems

2. Coverability

Given a Petri net C=(P, T, I, O, μ_0) with initial marking μ_0 , the state $\mu' \in R(C, \mu)$ **is covered** if

$$\mu'(i) \leq \mu(i), \text{ for all places } p_i \in P.$$

Property usage:

State μ **is / is not covered** by state μ'.

State μ **can / cannot be covered** by other reachable states.

Is it possible to use this property to help on the search for the reachable set?

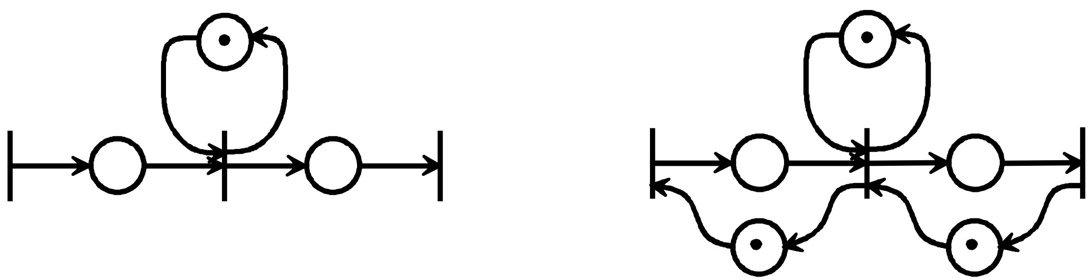
Yes! Details after some few slides.

Properties of Discrete Event Systems

3. Safeness

A place $p_i \in P$ of the Petri net $C=(P, T, I, O, \mu_0)$ **is safe** if
for all $\mu' \in R(C, \mu_0)$: $\mu_i' \leq 1$.

A Petri net **is safe** if all its places are safe.



Petri net not safe Petri net safe

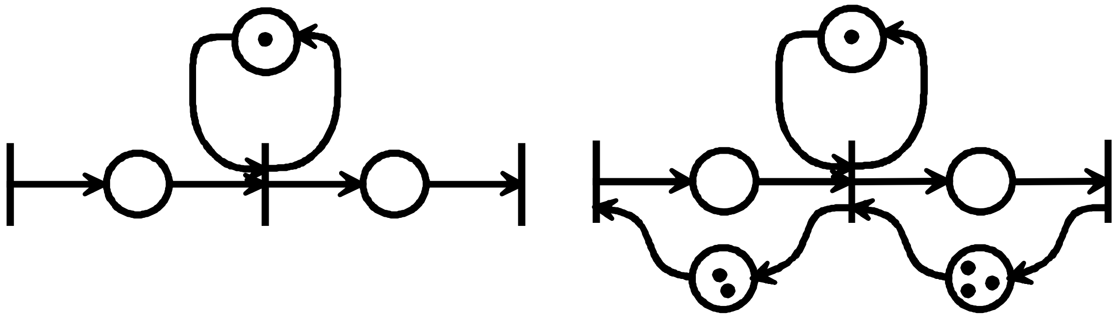
Property usage: Place p_i / Net C **is / is not** **safe**.

Properties of Discrete Event Systems

4. Boundedness

Given a Petri net C=(P, T, I, O, μ_0), a place $p_i \in P$ is **k-bounded** if $\mu_i' \leq k$ for all $\mu' = (\mu_1', \ldots, \mu_i', \ldots, \mu_N') \in R(C, \mu_0)$.

A Petri net is **k-bounded** if all places are k-bounded.



Petri net not bounded   Petri net 3-bounded

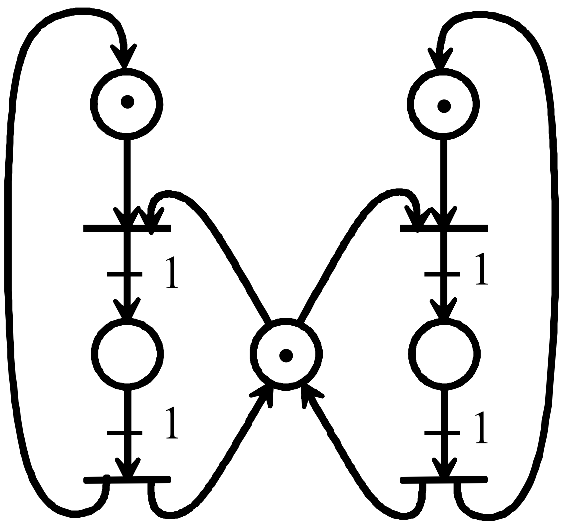
Property usage: Place p_i / Net C **is / is not** **k-bounded**.

Properties of Discrete Event Systems

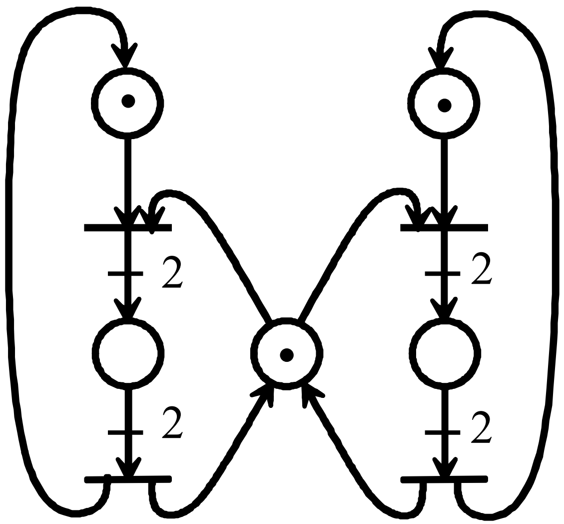
5. Conservation

A Petri net C=(P, T, I, O, μ_0) is **strictly conservative** if for all $\mu' \in R(C, \mu)$

$$\sum_{p_i \in P} \mu'(p_i) = \sum_{p_i \in P} \mu(p_i)$$



Petri net not conservative



Petri net strictly conservative

Property usage: Net C **is / is not (strictly) conservative**.

Properties of Discrete Event Systems

6. Liveness

A transition t_j is live of

Level 0 - if it can never be fired (transition is *Dead*).

Level 1 - if it is potentially firable, that is if there exists $\mu' \in R(C, \mu)$ such that t_j is enabled in μ'.

Level 2 - if for every integer *n*, there exists a firing sequence such that t_j occurs *n* times.

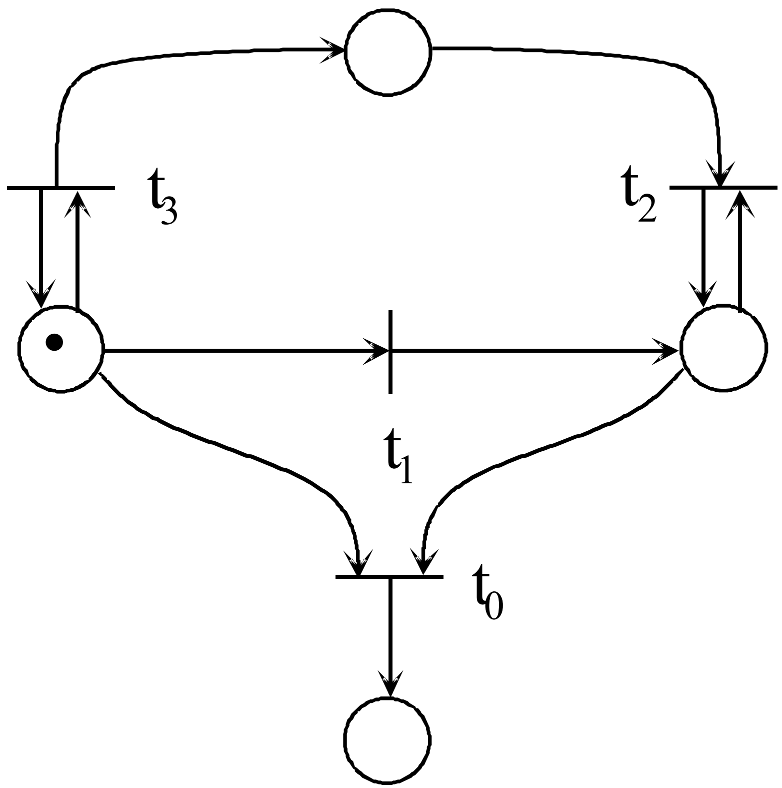
Level 3 - if there exists an infinite firing sequence such that t_j occurs infinite times.

Level 4 - if for each $\mu' \in R(C, \mu)$ there exist a sequence σ such that the transition t_j is enabled (transition is *Live*).

Properties of Discrete Event Systems

Example of liveness of transitions

- t_0 is of level 0.
- t_1 is of level 1.
- t_2 is of level 2.
- t_3 is of level 3.
- *this net does not have level 4 transitions.*



Properties of Discrete Event Systems

Reachability problem

Given a Petri net $C=(P, T, I, O, \mu_0)$ with initial marking μ_0 and a marking μ', is $\mu' \in R(C, \mu_0)$ reachable?

Analysis methods:

- Brute force...
- Reachability tree
- Matrix equations

Analysis Methods

Reachability Tree - construction *[Peterson81, §4.2.1]*

A reachability tree is a tree of reachable markings.
Tree nodes are states. The root node is the initial state (marking).
It is constituted by three types of nodes:

- **Terminal** no state changes after a terminal state
- **Interior** state can change after
- **Duplicated** state already found in the tree

The infinity marking symbol (ω) is introduced whenever a marking covers other. This symbol allows obtaining finite trees.

The reachability tree is useful to study properties previously introduced.
Some examples later.

Analysis Methods

Reachability Tree - construction *[Peterson81, §4.2.1]*

Algebra of the infinity symbol (ω):

For every positive integer *a* the following relations are verified:

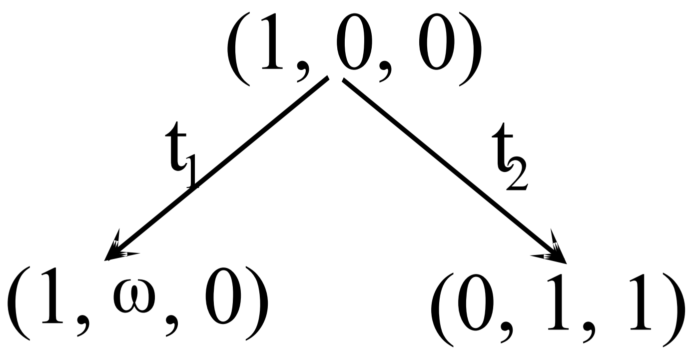
1. $\omega + a = \omega$
2. $\omega - a = \omega$
3. $a < \omega$
4. $\omega \leq \omega$

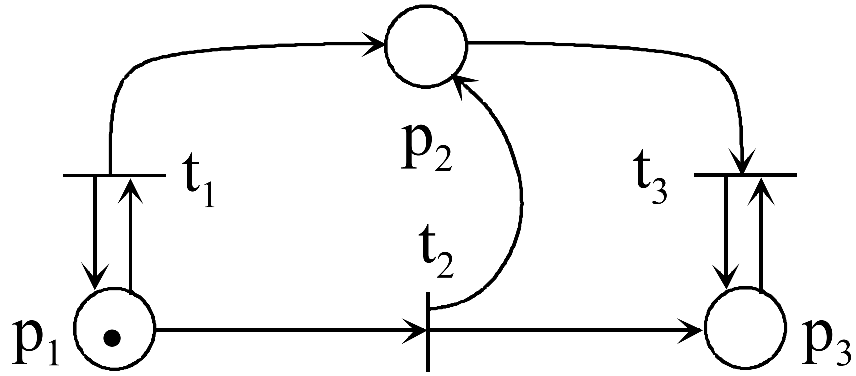
Reachability Tree and Deadlocks

Theorem - If there exist terminal nodes in the reachability tree then the corresponding Petri net has ***deadlocks***.

Analysis Methods

Example of reachability tree:

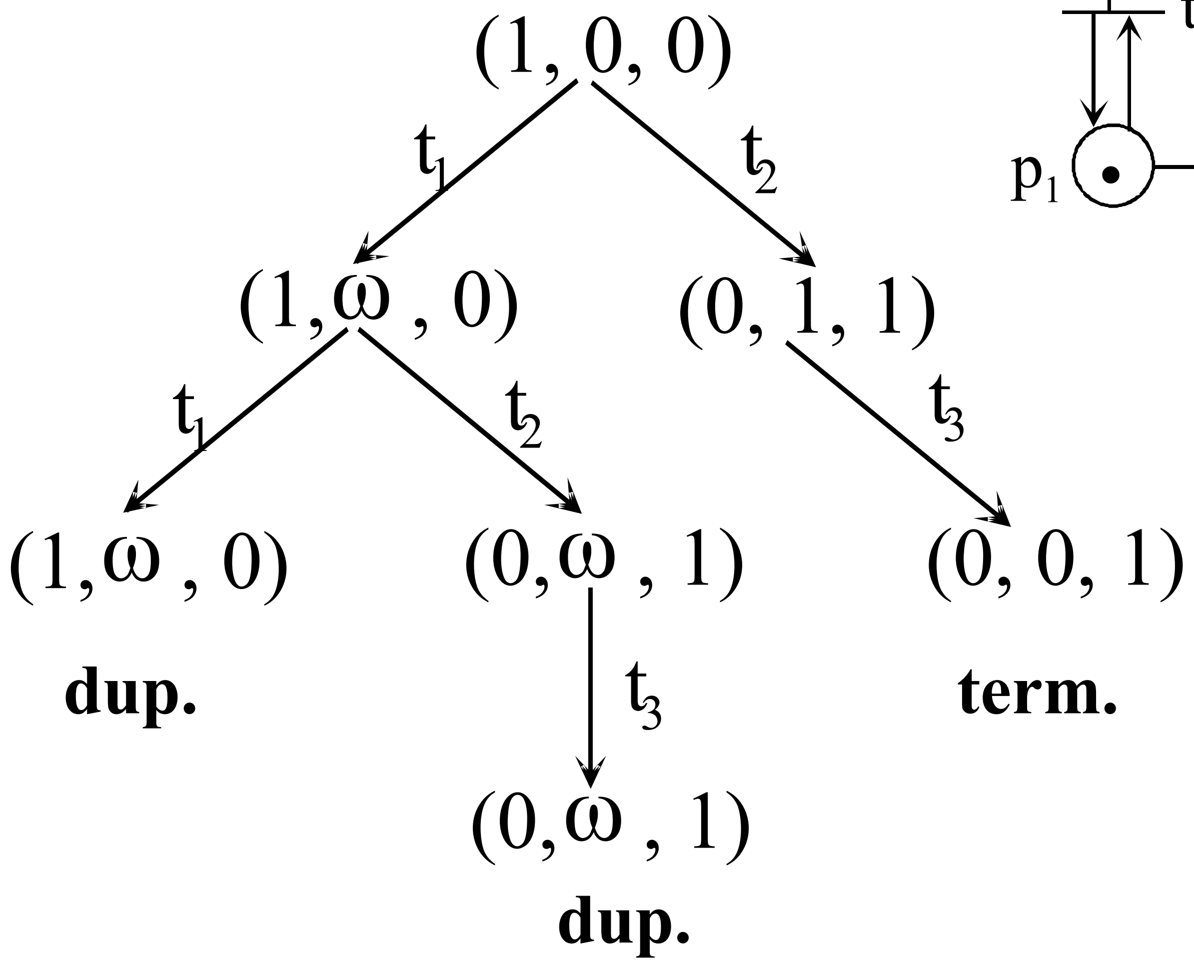


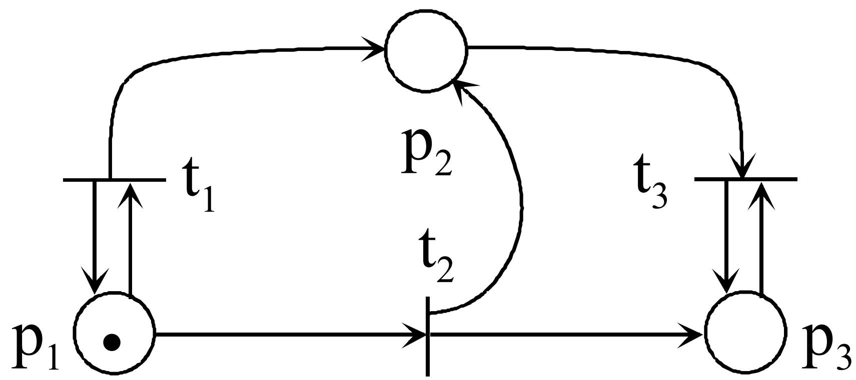


After t1 one obtains **(1, 0, 0)** which **is covered by (1, 1, 0).** Hence one introduces the **infinity symbol, ω** and writes the state as **(1, ω, 0)**.

Analysis Methods

Example of reachability tree:





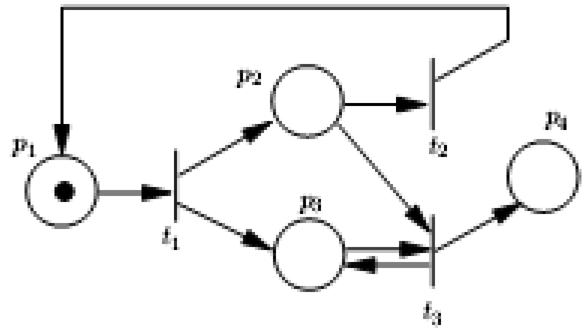
We can conclude immediately that there are

DEADLOCKS!

Analysis Methods

Reachability Tree vs Coverability Tree

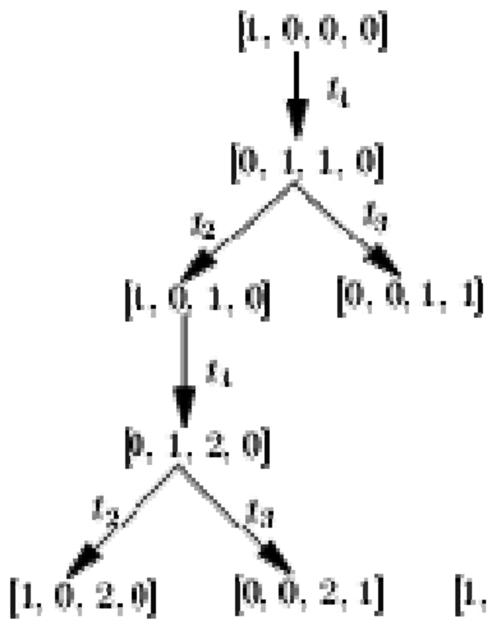
[Cassandras08, §4.4.2]



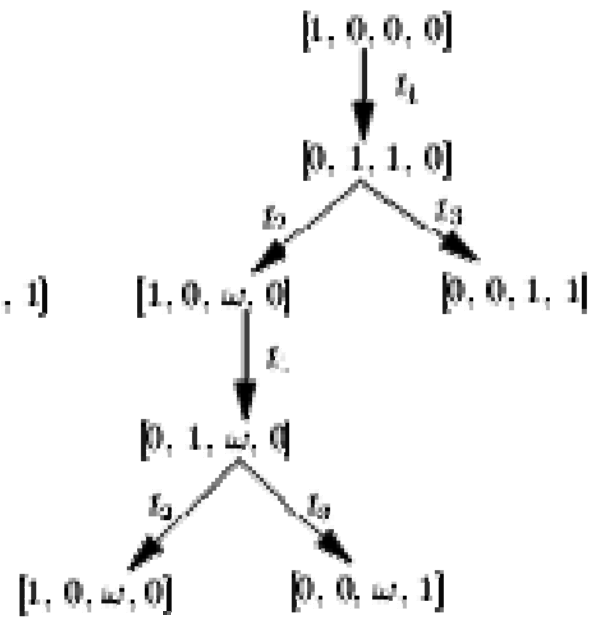
Considering a Petri net the **reachability tree** is "a tree whose root node is (...), then examine all transitions that can fire from this state, define new nodes in the tree, and repeat until all possible reachable states are identified."

"The reachability tree (...) may be infinite. A finite representation (...) is possible, but at the expense of losing some information. The finite version of an infinite reachability tree will be called a **coverability tree**."

Reachability tree ,



Coverability tree



(In this course we use Peterson's terminology, i.e. "reachability tree" in both cases)

Example1: simple Petri net, properties?

(P, T, A, w, x_0)

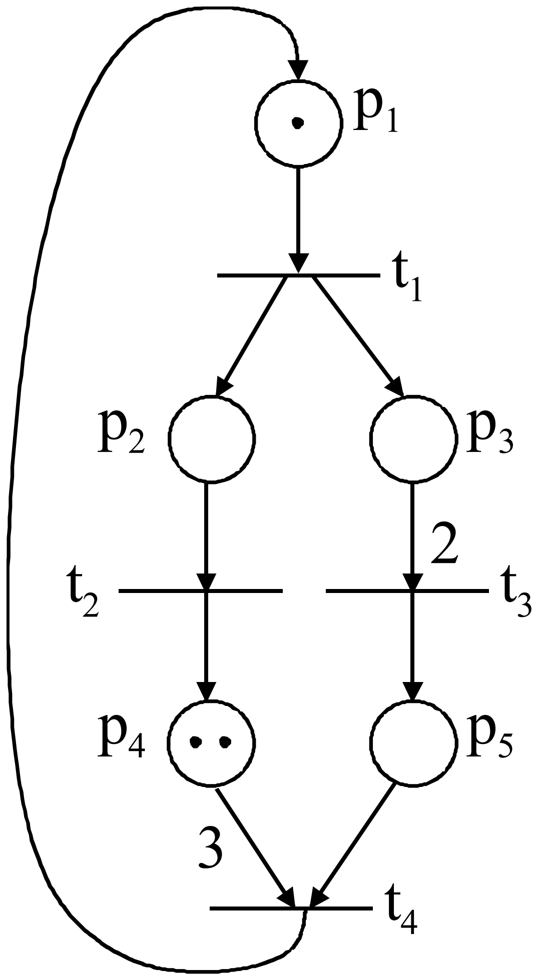
$P=\{p_1, p_2, p_3, p_4, p_5\}$

$T=\{t_1, t_2, t_3, t_4\}$

$A=\{(p_1, t_1), (t_1, p_2), (t_1, p_3), (p_2, t_2), (p_3, t_3),$
$(t_2, p_4), (t_3, p_5), (p_4, t_4), (p_5, t_4), (t_4, p_1)\}$

$w(p_1, t_1)=1, w(t_1, p_2)=1, w(t_1, p_3)=1, w(p_2, t_2)=1$
$w(p_3, t_3)=2, w(t_2, p_4)=1, w(t_3, p_5)=1, w(p_4, t_4)=3$
$w(p_5, t_4)=1, w(t_4, p_1)=1$

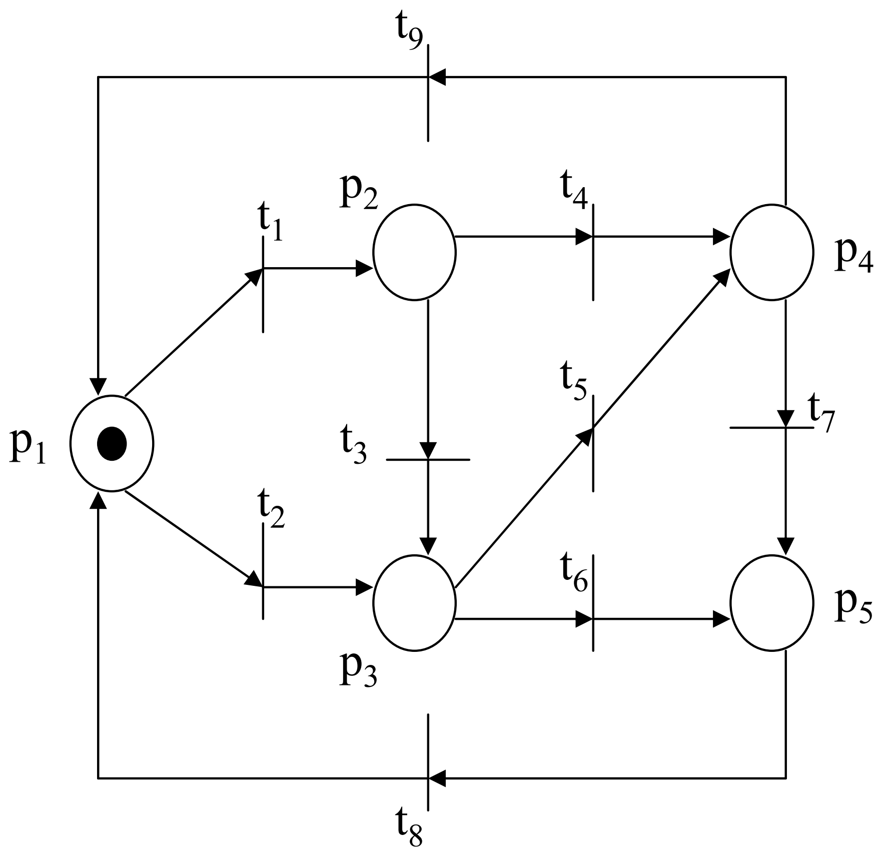
$x_0 = \{1, 0, 0, 2, 0\}$



Example2: simple automation system modeled using PNs, properties?

An automatic soda selling machine accepts
50c and $1 coins and
sells 2 types of products:
SODA A, that costs $1.50 and
SODA B, that costs $2.00.

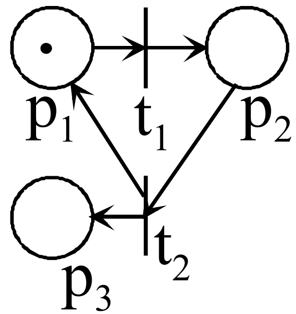
Assume that the money return operation is omitted.

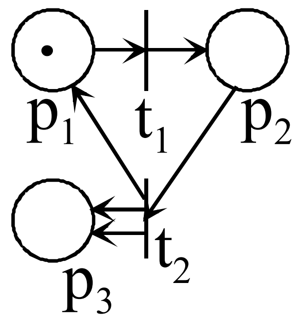


p_1: machine with $0.00;
t_1: coin of 50 c introduced;
t_8: SODA B sold.

Example3:

(counter-example)





$$
\begin{array}{c}
(1, 0, 0) \\
\downarrow t_1 \\
(0, 1, 0) \\
\downarrow t_2 \\
(1, 0, \omega) \\
\downarrow t_1 \\
(0, 1, \omega) \\
\downarrow t_2 \\
(1, 0, \omega) \\
\text{dup.}
\end{array}
\qquad\qquad
\begin{array}{c}
(1, 0, 0) \\
\downarrow t_1 \\
(0, 1, 0) \\
\downarrow t_2 \\
(1, 0, \omega) \\
\downarrow t_1 \\
(0, 1, \omega) \\
\downarrow t_2 \\
(1, 0, \omega) \\
\text{dup.}
\end{array}
$$

Different reachable sets
but the
same reachability tree

Decidability Problem:

Can one reach (1,0,1)? **Yes** in one net, **No** in the other one. Simple to answer in this net, but undecidable in general due to the symbol ω.

The reachability tree does not ensure decidability of state reachability.

Analysis Methods

Method of the Matrix Equations (of State Evolution)

The dynamics of the Petri net state can be written in compact form as:

$$\mu(k+1) = \mu(k) + Dq(k)$$

This methodology can also be used to study the other properties previously introduced. Requires some thought... ;)

where:

$\mu(k+1)$ - marking to be reached
$\mu(k)$ - initial marking
$q(k)$ - firing vector (transitions)
D - incidence matrix. Accounts the balance of tokens, giving the transitions fired.

Analysis Methods

How to build the Incidence Matrix, D ?

For a Petri net with *n* places and *m* transitions

$$\mu \in N_0^{\,n}$$

$$q \in N_0^{\,m}$$

$$D = D^+ - D^- \qquad \in Z^{n\times m}$$

The enabling firing rule is $\mu \geq D^- q.$

Can also be written in compact form as the inequality $\mu + Dq \geq 0,$
interpreted element-by-element.

Note: unless otherwise stated in this course all vector and matrix inequalities are read element-by-element.

Analysis Methods

Properties that can be studied immediately with the Method of Matrix Equations:

- Reachability (sufficient condition)

> **Theorem** – if the problem of finding the transition firing vector that drives the state of a Petri net from μ to state μ' has no solution, resorting to the method of matrix equations, then the problem of reachability of μ' does not have solution.

- Conservation – the firing vector is a by-product of the MME.

- Temporal invariance – cycles of operation can be found.

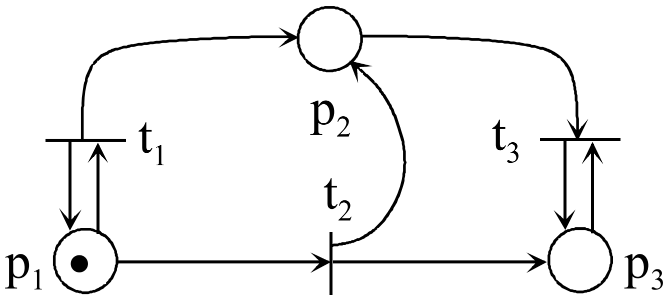
Analysis Methods 1. Reachability

Reachability problem: Given a Petri net C with initial marking μ_0 , does the marking μ' belong to the set of all markings that can be obtained, i.e. μ' ∈ *R*(C, μ)?

Example using the method of matrix equations

$$\mu(k+1)=\mu(k)+Dq(k)$$

Given the net:



$$D=\begin{bmatrix}0 & -1 & 0\\ 1 & 1 & -1\\ 0 & 1 & 0\end{bmatrix} \quad \mu(k)=\begin{bmatrix}1\\0\\0\end{bmatrix}$$

Problem:
is μ(k+1) reachable?
e.g.

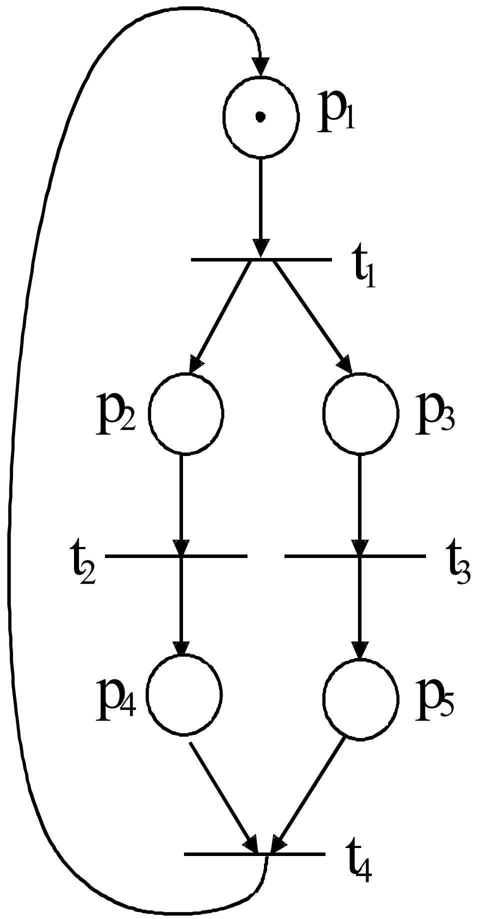
$$\mu(k+1)=\begin{bmatrix}1\\3\\0\end{bmatrix},$$

Solution, find q(k):

$$q(k)=\begin{bmatrix}\sigma_{t1}\\ \sigma_{t2}\\ \sigma_{t3}\end{bmatrix} \qquad \begin{cases}1=1-\sigma_{t2}\\ 3=\sigma_{t1}+\sigma_{t2}-\sigma_{t3}\\ 0=\sigma_{t2}\end{cases} \qquad \begin{cases}\sigma_{t2}=0\\ \sigma_{t1}-\sigma_{t3}=3 \quad \text{Verify!}\end{cases}$$

∃ q such that Dq(k)= μ(k+1)-μ(k) is a ***necessary*** *but* ***not sufficient*** *condition.*

Example of a Petri net



2. Conservation

To maintain the (weighted) number of tokens one writes:

$$x^T \mu' = x^T \mu + x^T Dq$$

and therefore:

$$\boxed{x^T D = 0}$$

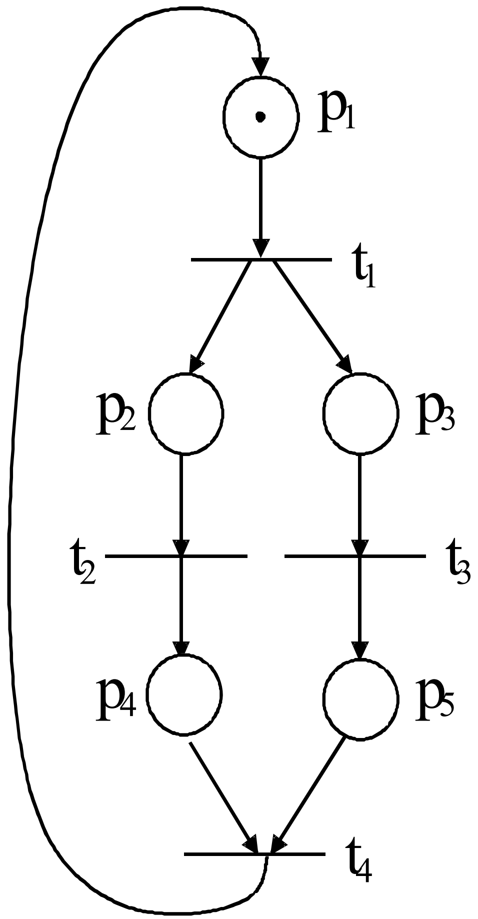
∃x is a ***necessary and sufficient*** *condition*

$$D = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \left\{ \begin{array}{c} -x_1 + x_2 + x_3 = 0 \\ -x_2 + x_4 = 0 \\ -x_3 + x_5 = 0 \\ x_1 - x_4 - x_5 = 0 \end{array} \right\} \begin{array}{c} x_1 = x_2 + x_3 \\ x_2 = x_4 \\ x_3 = x_5 \end{array}$$

This example has a solution in the form of an undetermined system of equations, where we can choose:

$$x^T = [2 \quad 1 \quad 1 \quad 1 \quad 1].$$

Example of a Petri net



3. Temporal invariance

To determine the transition firing vectors that make the Petri net return to the same state(s):

$$\boxed{Dq = 0}$$

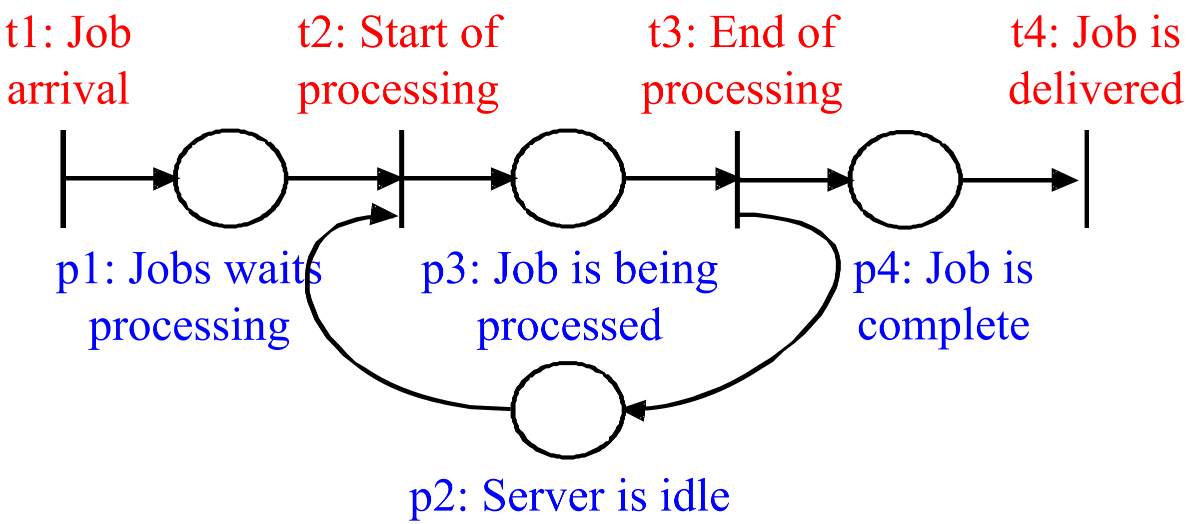
∃ q is a necessary (not sufficient) condition

$$D = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}, \quad q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \qquad \begin{cases} -q_1 + q_4 = 0 \\ q_1 - q_2 = 0 \\ q_1 - q_3 = 0 \\ q_2 - q_4 = 0 \\ q_3 - q_4 = 0 \end{cases}$$

This example has a solution in the form of an undetermined system of equations from which we can choose e.g.:

$$q = [1 \;\; 1 \;\; 1 \;\; 1]^T .$$

Example for the analysis of properties:



Q: How to have ***Conservation*** *?*

A: Find w such that $\mathbf{w}^T.\mathbf{D=0}$

$$D = \begin{bmatrix} 1 & -1 & & \\ & -1 & 1 & \\ & 1 & -1 & \\ & & 1 & -1 \end{bmatrix}$$

Event	Pre-conditions	Pos-conditions
t1	-	p1
t2	p1, p2	p3
t3	p3	p4, p2
t4	p4	-

$$w^T = [w_1\ w_2\ w_3\ w_4] = ?$$

Discrete Event Systems

Example of a simple automation system modeled using PNs

An automatic soda selling machine accepts

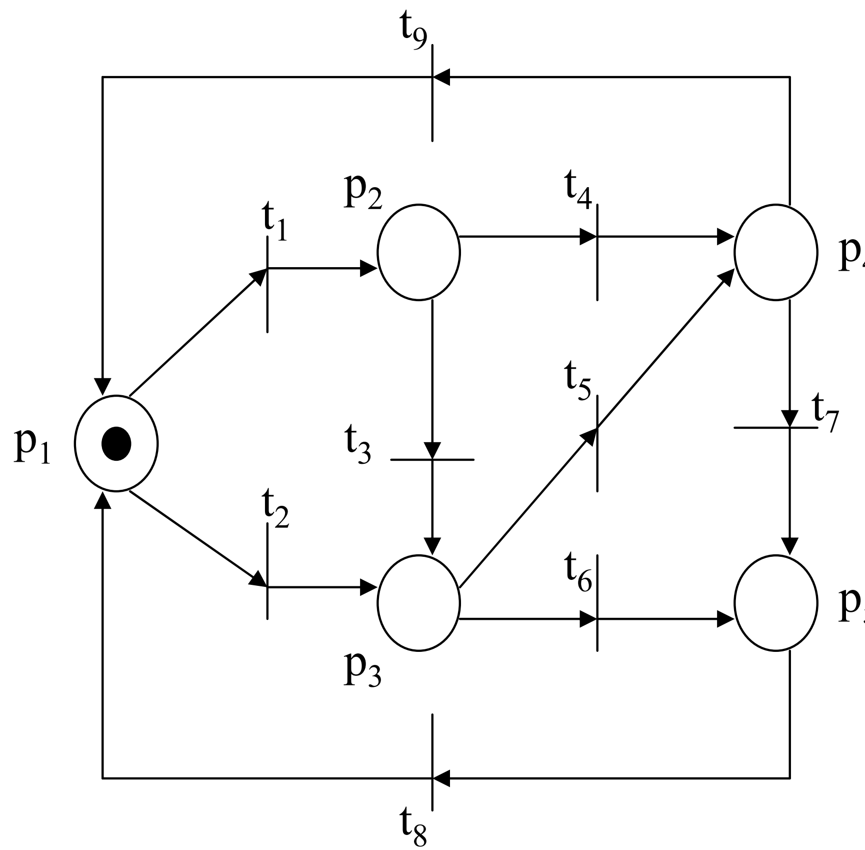
50c and \$1 coins and

sells 2 types of products:

SODA A, that costs \$1.50 and

SODA B, that costs \$2.00.

Assume that the money return operation is omitted.

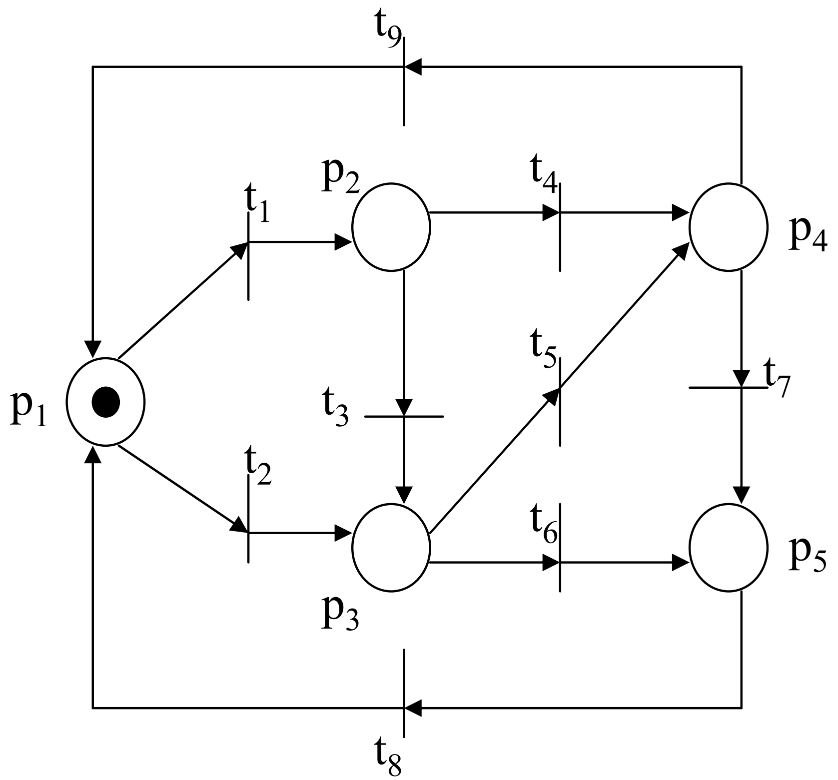


p_1: machine with \$0.00;
t_1: coin of 50 c introduced;
t_8: SODA B sold.

Q: Are there transition firing vectors that make the Petri net return to the same state?

Discrete Event Systems

Example of a simple automation system modeled using PNs




```
D= [ -1   -1    0    0    0    0    0    1    1
      1    0   -1   -1    0    0    0    0    0
      0    1    1    0   -1   -1    0    0    0
      0    0    0    1    1    0   -1    0   -1
      0    0    0    0    0    1    1   -1    0 ]
```

Time invariance ? Find q such that. D.q=0

```
>> q= null(D, 'r')
q =
    1   -1    1    0    1
   -1    1   -1    1    0
    1    0    0    0    0
    0   -1    1    0    1
    0    1    0    0    0
    0    0   -1    1    0
    0    0    1    0    0
    0    0    0    1    0
    0    0    0    0    1

>> q(:,1)= q(:,1)+q(:,4);
>> q(:,2)= q(:,2)+q(:,5);
>> q(:,3)= q(:,3)+q(:,4);
q =
    1    0    1    0    1
    0    1    0    1    0
    1    0    0    0    0
    0    0    1    0    1
    0    1    0    0    0
    1    0    0    1    0
    0    0    1    0    0
    1    0    1    1    0
    0    1    0    0    1
```

Complexity and Decidibility

The ***reachability tree*** *and* ***matrix equation*** *techniques allow properties of* ***safeness****,* ***boundedness****,* ***conservation****, and* ***coverability*** *to be determined for Petri nets. In particular, a necessary condition for reachability is established.*

However, these techniques are not sufficient to solve several other problems, especially ***liveness****,* ***reachability (sufficient condition)****, and* ***equivalence****.*

[Petersen 81, ch5]

In the following: we will discuss the complexity and decidability of the problems not solved.

Complexity and Decidibility

• Till the end of this chapter, *problem* is intended as a question with yes/no answer, e.g. "does $\mu' \in R(C,\mu)$?"

• A *problem* is *undecidable* if it is proven that no algorithm to solve it exists.

An example of a undecidable problem is the stop of a Turing machine (TM):

"Will the TM stop for the code n after using the number m?".

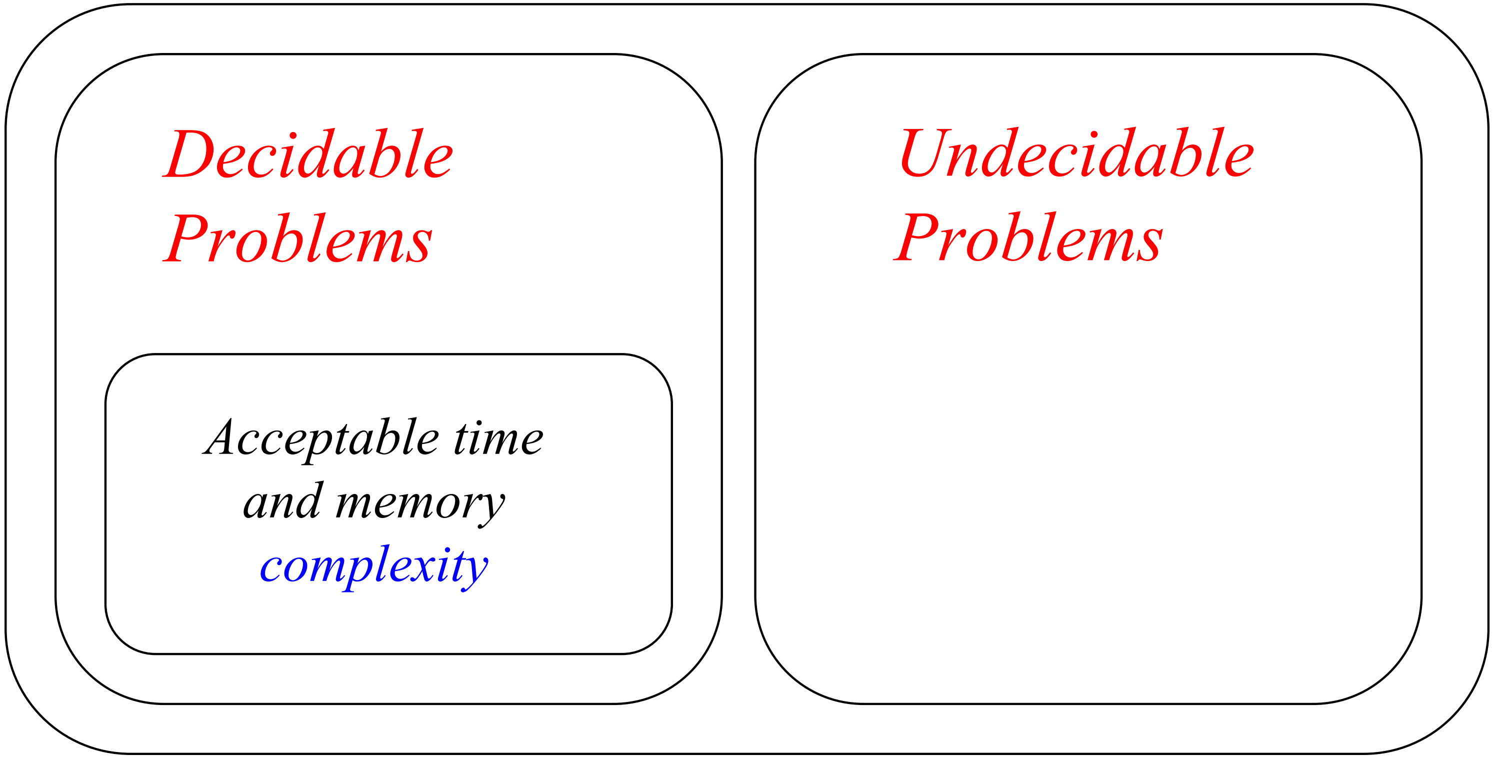
• For *decidable problems*, the *complexity* of the solutions have to be taken into account, that is, the computational cost in terms of memory and time.

Basic example: a multiplication of numbers has solution (algorithm taught

in the school), but the complexity was different in the arabic and latin civilizations...

Complexity and Decidibility

Problems



Reducibility

One *benefits of reducibility* when to solve a given problem it is *possible to* ***reduce*** *it to another problem with known solution.*

Theorem: Assume that the problem A is reducible to problem B,

then an instance of A can be transformed in an instance of B and:

- If B is decidable then A is decidable.
- If A is undecidable then B is undecidable.

Reducibility

Equality Problem: Given two marked Petri nets

$C_1=(P_1, T_1, I_1, O_1)$ and $C_2=(P_2, T_2, I_2, O_2)$, with markings μ_1 e μ_2,

respectively, is $R(C_1, \mu_1) = R(C_2, \mu_2)$?

Subset Problem: Given two marked Petri nets

$C_1 = (P_1, T_1, I_1, O_1)$ and $C_2 = (P_2, T_2, I_2, O_2)$, with markings μ_1 e μ_2,

respectively, is $R(C_1, \mu_1) \subseteq R(C_2, \mu 2)$?

The **equality** problem is **reducible** to the **subset** problem
(equality is obtained by proving that each set is a subset of the other)

Decidibility

If a problem is ≈ undecidable does it mean that it is not solvable?

No, while not proved to be undecidable there is hope it can be solved!

Classical example, Fermat Last Theorem:

Does $x^n + y^n = z^n$ have a solution for n>2 and nontrivial integers x, y e z?

Now, it is known that the problem is impossible, i.e. its answer is *No*. The problem remained ≈ undecidable for more than 2 centuries (solution proven in 1998).

The *Turing Machine (TM) Halting* problem is undecidable.

If it were decidable, for instance the Fermat last theorem would have been proven long time ago, i.e. there would be an algorithm (*TM* with code *n*) that computing all combinations of x,y,z and n>2 (number m) to find a solution verifying $x^n + y^n = z^n$.

Reachability Problems

Given a Petri net $C=(P,T,I,O)$ with initial marking μ

Reachability Problem:

Considering a marking μ', does $\mu' \in R(C,\mu)$?

Sub-marking Reachability Problem:

Given the marking μ' and a subset $P' \subseteq P$, exists $\mu'' \in R(C,\mu)$ such that $\mu''(p_i) = \mu' \; \forall p_i \in P'$?

Zero Reachability Problem:

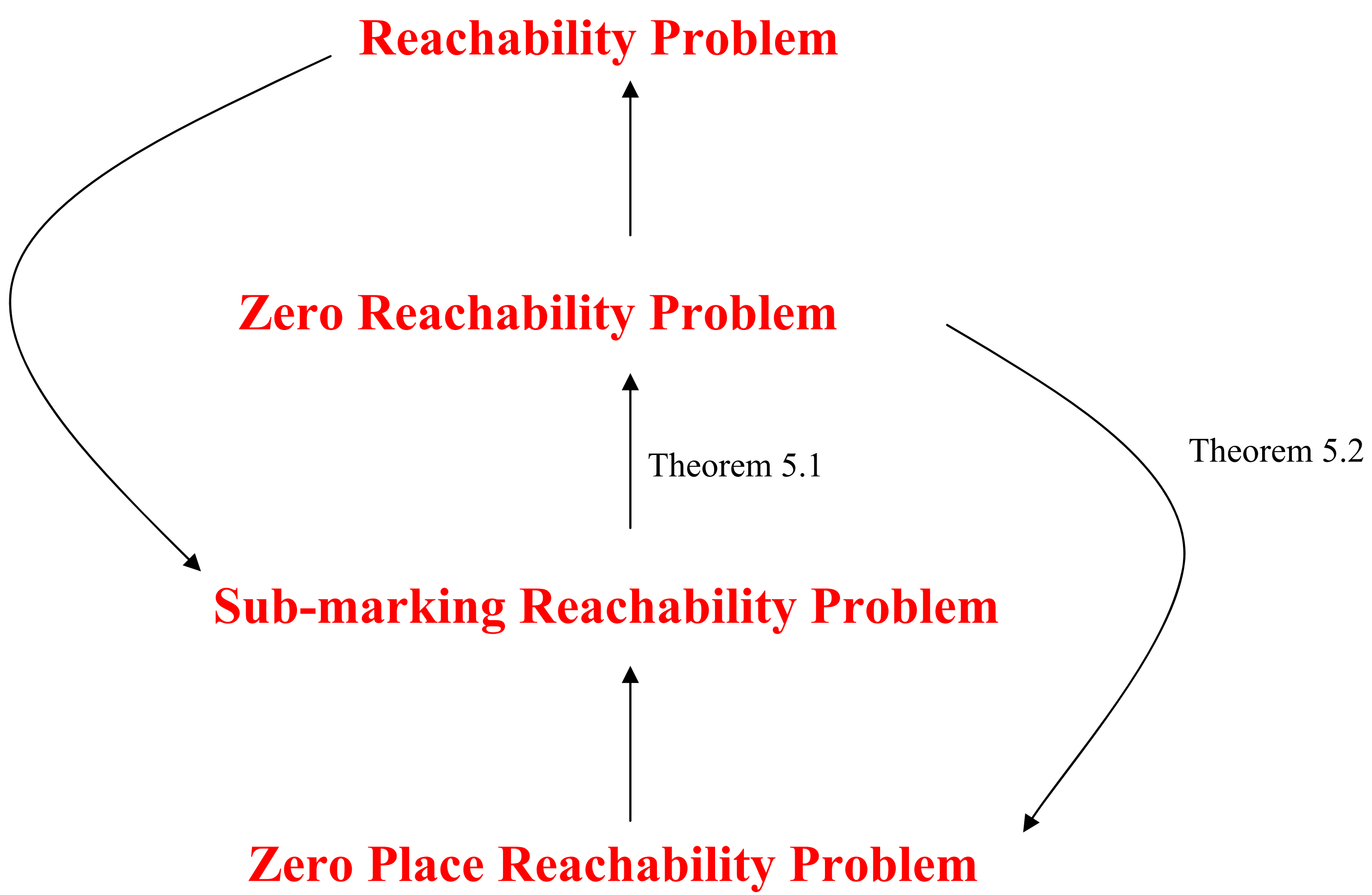
Given the marking $\mu'=(0\ 0\ \ldots\ 0)$, does $\mu' \in R(C,\mu)$?

Zero Place Reachability Problem:

Given the place $p_i \in P$, is $\mu' \in R(C,\mu)$ with $\mu'(p_i) = 0$?

Reachability Problems

Legend:
A→B means A is reducible to B



Reachability Problems

Theorem 5.3: The following reachability problems are equivalent:

- **Reachability Problem**;
- **Zero Reachability Problem**;
- **Sub-marking Reachability Problem**;
- **Zero Place Reachability Problem**.

Liveness and Reachability

(Given a Petri net C=(P,T,I,O) with initial marking μ)

Liveness Problem

Are all transitions t_j of T live?

Transition Liveness Problem

For the transition t_j of T, is t_j live?

The **liveness** problem is **reducible** to the **transition** liveness problem. To solve the first it remains only to solve the second for the *m* Petri net transitions (#T = *m*).

Liveness and Reachability

(Given a Petri net C=(P,T,I,O) with initial marking μ)

Theorem 5.5: The problem of reachability is reducible to the liveness problem.

Theorem 5.6: The problem of liveness is reducible to the reachability problem.

Theorem 5.7: The following problems are **equivalent**:

- **Reachability problem**
- **Liveness problem**

Decidibility results

Theorem 5.10: The sub-marking reachability problem is reducible to the reachable subsets of a Petri net.

Theorem 5.11: The following problem is undecidable:

- Subset problem for reachable sets of a Petri net

They are all reducible to the famous Hilbert's 10th problem:

The solution of the Diophantine equation of n variables, with integer coefficients $P(x_1, x_2, \ldots, x_n)=0$ is undecidable.

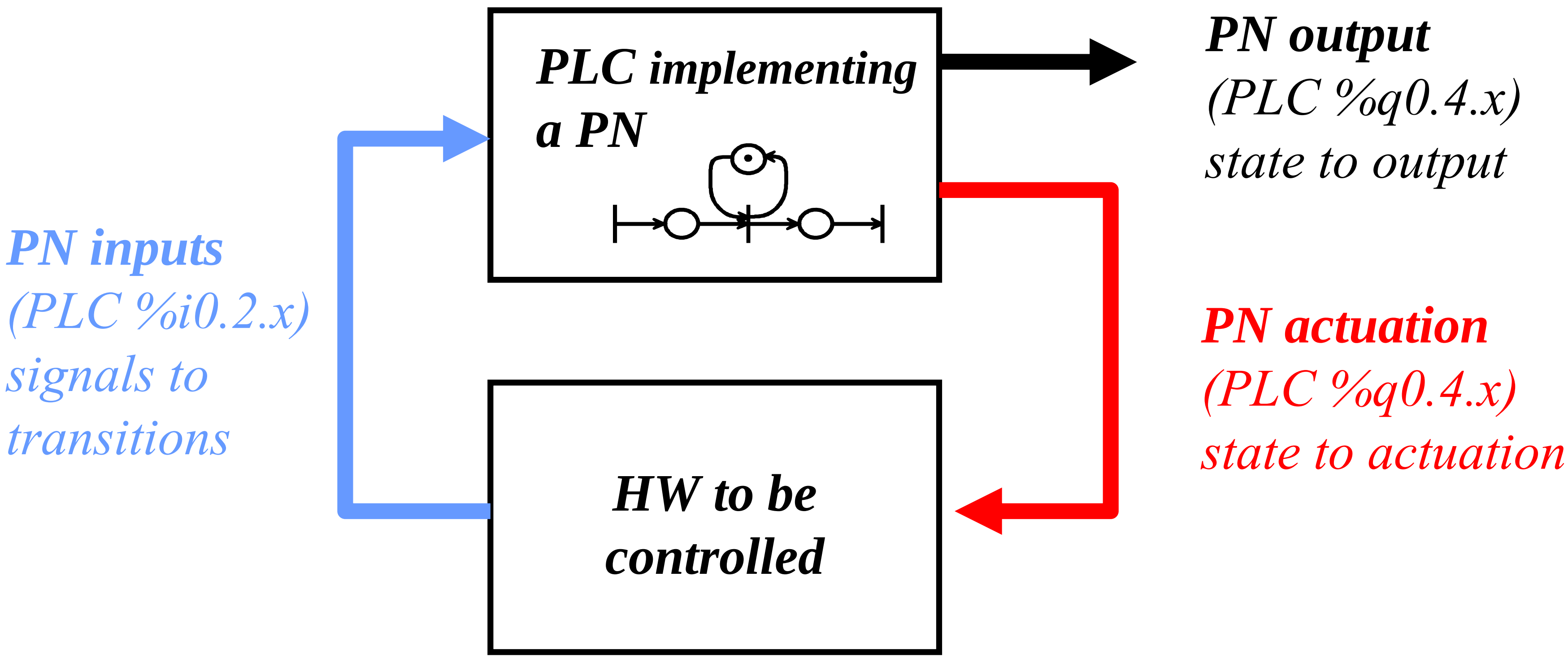
(proof by Matijasevic that it is undecidable in the late 1970s).

Decidibility

"... most decision problems involving finite-state automata can be solved algorithmically in finite time, i.e., they are ***decidable****. Unfortunately, many problems that are decidable for* ***finite state automata*** *are no longer decidable for* ***Petri nets****, reflecting a natural trade off between decidability and model-richness. (...) Overall, it is probably most helpful to think of Petri nets and automata as* ***complementary modeling approaches****, rather than competing ones.“*

[Cassandras 2008]

Simulating a Petri net with HW inputs and outputs

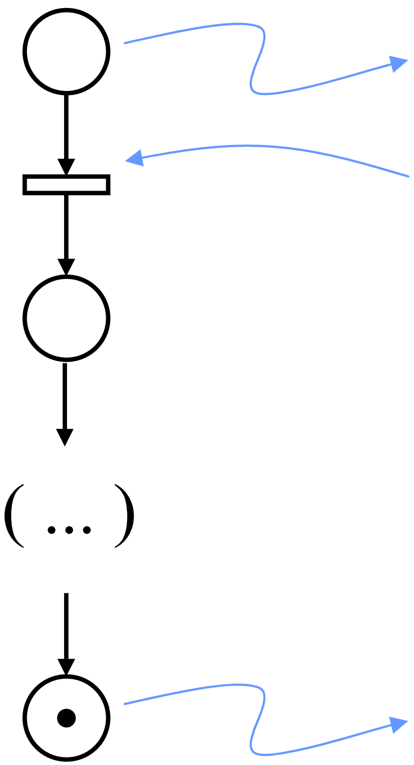


Summary of simulators: (a) simulation of the Petri net,
(b) simulation of the hardware to be controlled

Summary of functions: (1) state/places to actuation,
(2) signals to transitions,
(3) state/places to output

Simulating a Petri net with HW inputs and outputs

Example: keyboard reading



1. state to actuation:
   power kb columns
2. signals to transitions:
   wait signal on kb lines
3. state to output:
   key X is pressed

See example in Matlab:

a) PN_sim.m
b) PN_device_kb_IO.m

1) PN_s2act.m
2) PN_tfire.m
3) PN_s2yout.m

Simulation of a Petri net

```
function [tSav, MPSav, youtSav]= PN_sim(Pre, Post, M0, ti_tf)
%
% Simulating a Petri net, using a SFC/Grafcet simulation methodology.
% See book "Automating Manufacturing Systems", by Hugh Jack, 2008
% (ch20. Sequential Function Charts)
%
% Petri net model:
%  M(k+1) = M(k) +(Post-Pre)*q(k)
%  Pre and Post are NxM matrices, meaning N places and M transitions

% 0. Start PN at state M0
%
MP=M0;
ti=ti_tf(1); tf=ti_tf(2); tSav= (ti:5e-3:tf)';
MPSav= zeros( length(tSav), length(MP) );
youtSav= zeros( length(tSav), length(PN_s2yout(MP)) );

for i= 1:length(tSav)

    % 1. Check transitions (update state)
    tm= tSav(i);
    qk= PN_tfire(MP, tm);
    qk2= filter_possible_firings(MP, Pre, qk(:));
    MP= MP +(Post-Pre)*qk2;

    % 2. Do place activities
    yout= PN_s2yout(MP);

    % Log all results
    MPSav(i,:)= MP';
    qkSav(i,:)= qk2';
    youtSav(i,:)= yout;

end
```

```
function qk2= filter_possible_firings(M0, Pre, qk)
% verify Pre*q <= M
% try to fire all qk entries

M= M0;
mask= zeros(size(qk));
for i=1:length(qk)
    % try accepting qk(i)
    mask(i)= 1;
    if any(Pre*(mask.*qk) > M)
        % exceeds available markings
        mask(i)= 0;
    end
end
qk2= mask.*qk;
```

```
function lines= PN_device_kb_IO(act, t)

% Define 4x3-keyboard output line-values given actuation on the 3 columns
% and an (internal) time table of keys pressed
% Input:
%  act: 1x3 : column actuation values
%  t   : 1x1 : time
% Output:
%  lines: 1x4 : line outputs

global keys_pressed
if isempty(keys_pressed)
    % first column = time in seconds
    % next 12 columns = keys pressed at time t
    keys_pressed= [...
        0  mk_keys([]) ; 1  mk_keys(1)   ; ...
        2  mk_keys([]) ; 3  mk_keys(5)   ; ...
        4  mk_keys([]) ; 5  mk_keys(9)   ; ...
        6  mk_keys([]) ; 7  mk_keys([1 12]) ; ...
        8  mk_keys(12) ; 9  mk_keys([]) ; ...
        ];
end

% pressed keys yes/no
ind= find(t>=keys_pressed(:,1));
if isempty(ind)
    lines= [0 0 0 0]; % default lines output for t < 0
    return
end
keys_t= keys_pressed(ind(end), :);

% if actuated column and key pressed match, than activate line
lines= sum( repmat(act>0, 4,1) & reshape(keys_t(2:end), 3,4)', 2);
lines= (lines > 0)';
```

Keyboard simulator: generate line values given column values

```
function y= mk_keys(kid)
y= zeros(1,12);
for i=1:length(kid)
    y(kid(i))= 1;
end
```

Prototypes of the interfacing functions

```
function act= PN_s2act(MP)

% Create 4x3-keyboard column actuation
%
% MP:  1xN : marked places (integer values >= 0)
% act: 1x3 : column actuation values (0 or 1 per entry)
```

```
function qk= PN_tfire(MP, t)

% Possible-to-fire transitions given PN state (MP) and the time t
%
% MP: 1xN : marked places (integer values >= 0)
% t : 1x1 : time
% qk: 1xM : possible firing vector (to be filtered later with enabled
%             transitions)
```

```
function yout= PN_s2yout(MP)

% Show the detected/undetected key(s) given the Petri state
%
% MP: 1xN : marked places (integer values >= 0)
```

Top 10 Challenges in Logic Control for Manufacturing Systems

by Dawn Tilbury from University of Michigan

10. Distributed Control (General management of distributed control applications, Open/distributed control -- ethernet-based control)

9. Theory (No well-developed and accepted theory of discrete event control, in contrast to continuous control)

8. Languages (None of the programming languages do what we need but nobody wants a new programming language)

7. Control logic synthesis (automatically)

6. Standards (Machine-control standards -- every machine is different, Validated standards, Standardizing different types of control logic programming language)

5. Verification (Standards for validation, Simulation and verification of controllers)

4. Software (Software re-usability -- cut and paste, Sophisticated software for logic control, User-unfriendly software)

3. Theory/Practice Gap (Bridging the gap between industry and academia, Gap between commercial software and academic research)

2. Education (Educating students for various PLCs, Education and keeping current with evolution of new control technologies, Education of engineers in logic control, Lack of curriculum in discrete-event systems)

And the number one challenge in logic control for manufacturing systems is...

1. Diagnostics (Integrating diagnostic tools in logic control, Standardized methodologies for design, development, and implementation of diagnostics)