Discrete and continuous dynamic systems

Petri Nets
Definition and operation

Anna Ibolya Pózna

University of Pannonia
Faculty of Information Technology
Department of Electrical Engineering and Information Systems

pozna.anna@virt.uni-pannon.hu

April 2021

Lecture overview

1. Previous notions
   - Discrete event systems
   - Automata
   - Petri nets
2. Generalized Petri net models
   - Low level Petri nets
   - Hierachical Petri nets
   - Timed Petri nets
   - Coloured Petri nets
3. Reachability graph of Petri nets
   - Operation (dynamics) of Petri nets
   - Parallel and conflicting execution steps
   - Solution of Petri net models
   - The reachability graph
4. Analysis of discrete event system models

Discrete event systems

Characteristic properties:

- the *range space* of the signals (input, output, state) is **discrete**: $x(t) \in \mathbf{X} = \{x_0, x_1, ..., x_n\}$
- *event*: the occurrence of change in a discrete value
- *time is also* ***discrete***: $T = \{t_0, t_1, ..., t_n\} = \{0, 1, ..., n\}$

Only the **order of the events** is considered

- description of sequential and parallel events
- **application area**: scheduling, operational procedures, resource management

Automaton - abstract model: $\mathbf{G} = (X, U, Y, f, g, x_0)$

- **finite set of states**: $X = \{x_1, x_2, ...x_n\}$
- **finite set of input events**: $U = \{\varepsilon; u_1, u_2, ..., u_m\}$
- **finite set of output events**: $Y = \{\varepsilon; y_1, y_2, ..., y_n\}$
- **(partial) state transition function**: $f : X \times U \rightarrow X$ e.g. $f(x_1, u_3) = x_2$
- **output function**:
  $g : X \times U \rightarrow Y$ e.g. $g(x_1, u_3) = y_1$ (Mealy automaton)
  $g : X \rightarrow Y$ e.g. $g(x_1) = y_2$ (Moore automaton)
- *initial state*: x_0

Graphical description: weighted directed graph

- **Vertices**: states (X)
- **Edges**: state transitions (f)
- **Edge weights**: input/output symbols (Mealy), input symbols (Moore)

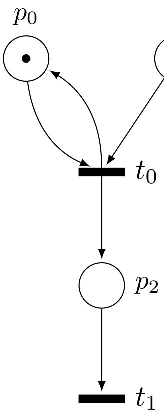
Automata - discrete event systems

	Automaton model	Discrete event state space model
State space	X	$\mathcal{X} \in \mathbb{Z}^n$
Input u	string from U	discrete time discrete valued signal
Output y	string from Y	discrete time discrete valued signal
State equation	$x(k+1) = f(x(k), u(k))$	$x(k+1) = \Psi(x(k), u(k))$
Output equation	$y(k) = g(x(k), u(k))$ (Mealy) $y(k) = g(x(k))$ (Moore)	$y(k) = h(x(k), u(k))$

(ordinary) Petri net - abstract description: $\mathbf{PN} = (P, T, I, O)$

Static description (structure)

- set of **places (conditions)**: P
- set of **transitions (events)**: T
- **Input (pre-condition) function**:
  $I : T \to P^{\infty}$
- **Output (consequence) function**:
  $O : T \to P^{\infty}$



Graphical description: bipartite directed graph

- **Vertices**: places (P) and transitions (T) (partitions)
- **Edges**: input and output functions (I, O)

Overview - Generalized Petri nets

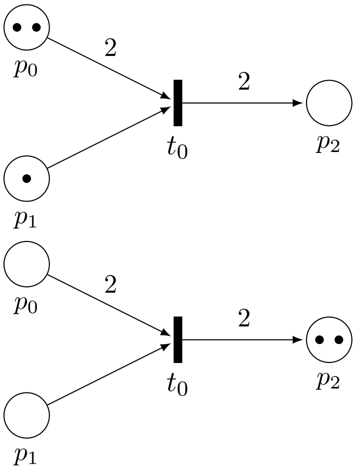
Generalized Petri net models

- **Low level Petri nets**
- **Hierarchical Petri nets**
- **Timed Petri nets**: using inscriptions
  - clock: built in (or special "source" place)
  - firing time to transitions
  - (waiting time for places)
- **Coloured Petri nets**: using inscriptions
  - tokens have discrete value ("colour")
  - colour set to places
  - discrete functions to the transitions and arcs

Low level Peeri nets

Definition: $PN = (P, T, F, W, M_0)$

- P set of places
- T set of transitions
- $F \subseteq (P \times T) \cup (T \times P)$ set of arcs
- $W : F \rightarrow \mathbb{N}$ arc weights
- $M_0 : P \rightarrow \mathbb{N}$ initial marking
- $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$



Enabling of a transition

$$\mu(p) \geq W(p, t), \forall p, \text{ where } p \text{ is the input place of } t$$

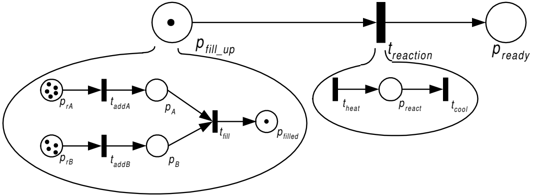
Firing of a transition: it consumes and produces tokens according to the weight function

Hierarchical Petri nets

Super net - subnets:

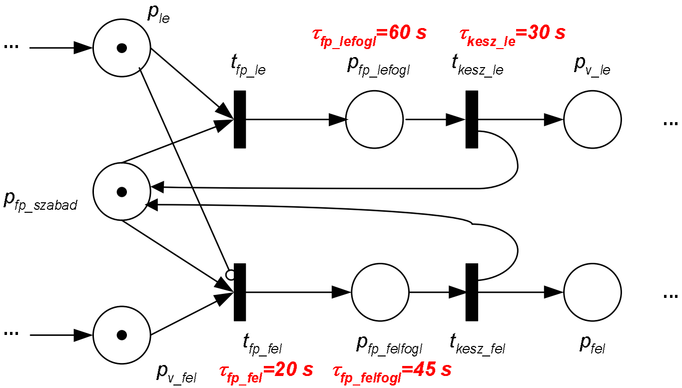
building in: to any place or transition

similar repetitive net-fragments



Petri net model of a runway – 3

Timed Peri net model



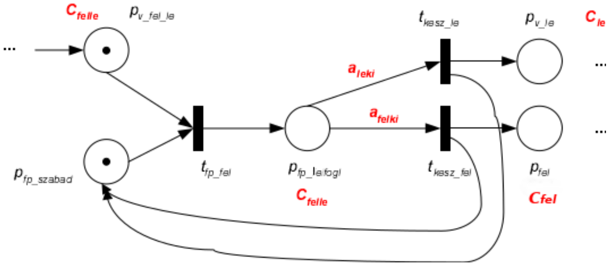
Petri net model of a runway – 4

Coloured Peri net model: "inscriptions"

Edge fucntion: a_{felki} : **if** $val(p_{fp\_lefogl}) = " \uparrow "$ **then** $"true"$

$a_{fel} = val(p_{fp\_lefogl})$, $val(p_{fel}) = a_{fel}$

Colour set: $C_{felle} = \{ \uparrow , \downarrow \}$



Overview - Petri nets: operation and reachability graph

Dynamics of Petri nets

Marking function: marking points (**tokens**)

$$\mu : \mathbf{P} \to \mathbb{N} \quad , \quad \mu(p_i) = \mu_i \geq 0$$
$$\underline{\mu}^T = [\mu_1, \mu_2, \ldots, \mu_n] \quad , \quad n = |\mathbf{P}|$$

A transition is **enabled** when its pre-conditions are "true" (there is at least one **token** on its input places)

$$\forall p \in I(t,p) : \mu(p) \geq 1$$

An enabled transition may **fire** (operates): it "consumes" tokens from all of its input places and produces tokens in each output places

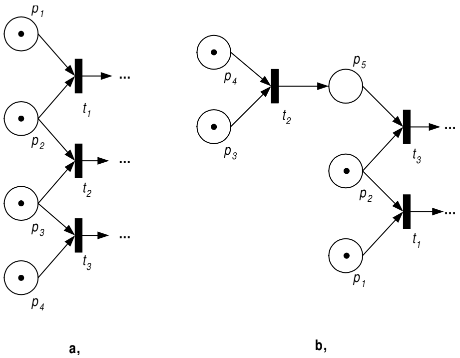
Notion: $\underline{\mu}^{(i)}[t_j > \underline{\mu}^{(i+1)}$

Firing (operation) sequence

$$\underline{\mu}^{(0)}[t_{j0} > \underline{\mu}^{(1)}[t_{j1} > ...[t_{jk} > \underline{\mu}^{(k+1)}$$

Parallel events

More than one enabled (fireable) transition:
concurrency (independent conditions), conflict, confusion



The solution problem

Abstract problem statement

Given:

- a *formal description* of a discrete event system model
- *initial state(s)*
- *external events*: system inputs

Compute:

- the sequence of *internal (state and output) events*

The solution is **algorithmic**! **The problem is NP-hard**!

Petri net models – reachability graph

Solution: marking (systems state) sequences
reachability graph (tree) (weighted directed graph)

- *vertices*: markings
- *edges*: if exists transition the firing of which connects them
- *edge weights*: the transition and the external events

Construction:

1. *start*: at the given initial state (marking)
2. *adding a new vertex*: by firing an enabled transition (with the effect of inputs!)

May be NP-hard (in conflict situation or non-finite operation)

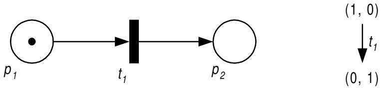
Construction of the reachibility graph

$\underline{\mu}^{(0)}$ is the root. L is the list of new nodes.

1. Add $\underline{\mu}^{(0)}$ to L.
2. If L is empty then stop, the reachability graph is ready. Otherwise choose the first node x from L with the associated marking. Remove x from L.
3. If another node y exists in the graph with the same associated marking, then x is a duplicate.
4. If no transition enabled in the marking of x then x is a terminal node (deadlock).
5. For all transitions enabled in the marking of node x:
   - Create a new node and connect it to x with an edge, labelled by the fire transition. Add this node to the end of L.
   - Determine the marking associated with the new node.
6. Continue with step 2.

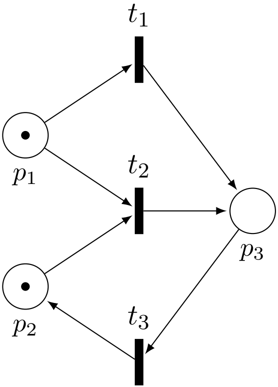
Reachability graphs

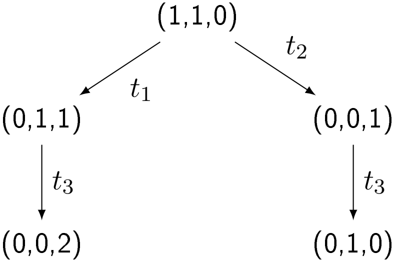
Finite case



a, b,

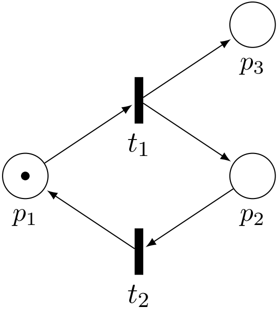
Branches





Reachability graphs

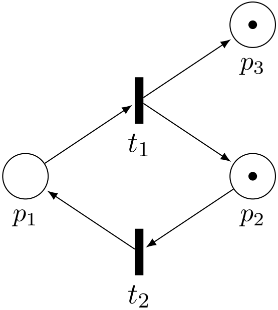
Non-finite case



(1,0,0)

Reachability graphs

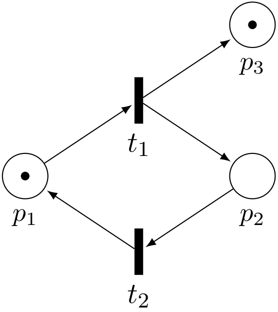
Non-finite case

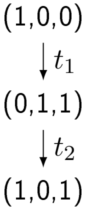


$$(1,0,0) \xrightarrow{t_1} (0,1,1)$$

Reachability graphs

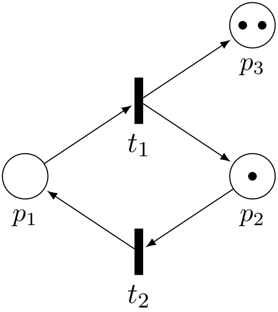
Non-finite case

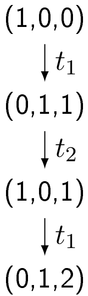




Reachability graphs

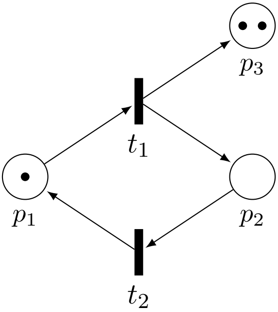
Non-finite case





Reachability graphs

Non-finite case

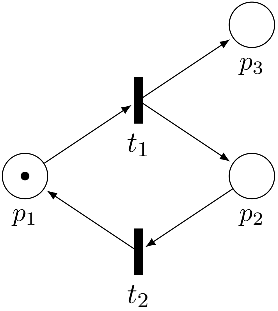


$$
\begin{array}{c}
(1,0,0) \\
\downarrow t_1 \\
(0,1,1) \\
\downarrow t_2 \\
(1,0,1) \\
\downarrow t_1 \\
(0,1,2) \\
\downarrow t_2 \\
\ldots
\end{array}
$$

Non-finite reachability graph

Reduction: using the ω symbol

- a marking $\underline{\mu}$' "dominates" an other node $\underline{\mu}$, if $\underline{\mu}$ is on the path from the root to $\underline{\mu}$' and
  - $\forall p \in P\ \mu'(p) \geq \mu(p)$ (might be equal for all places!)
  - $\exists p \in P\ \mu'(p) > \mu(p)$ (at least one place has more tokens on it)
- the diverging number of tokens can be denoted by ω
- if the parent marking contains ω, then all of its children will have ω in the same place

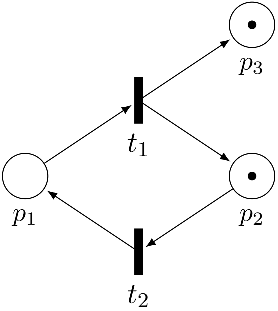


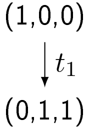
(1,0,0)

Non-finite reachability graph

Reduction: using the ω symbol

- a marking $\underline{\mu}$' "dominates" an other node $\underline{\mu}$, if $\underline{\mu}$ is on the path from the root to $\underline{\mu}$' and
  - $\forall p \in P \ \mu'(p) \geq \mu(p)$ (might be equal for all places!)
  - $\exists p \in P \ \mu'(p) > \mu(p)$ (at least one place has more tokens on it)
- the diverging number of tokens can be denoted by ω
- if the parent marking contains ω, then all of its children will have ω in the same place

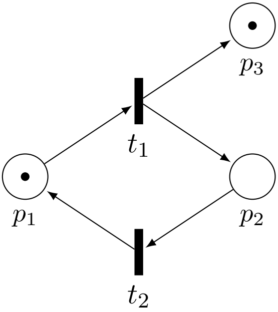


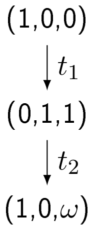


Non-finite reachability graph

Reduction: using the ω symbol

- a marking $\underline{\mu}$' "dominates" an other node $\underline{\mu}$, if $\underline{\mu}$ is on the path from the root to $\underline{\mu}$' and
  - $\forall p \in P\ \mu'(p) \geq \mu(p)$ (might be equal for all places!)
  - $\exists p \in P\ \mu'(p) > \mu(p)$ (at least one place has more tokens on it)
- the diverging number of tokens can be denoted by ω
- if the parent marking contains ω, then all of its children will have ω in the same place

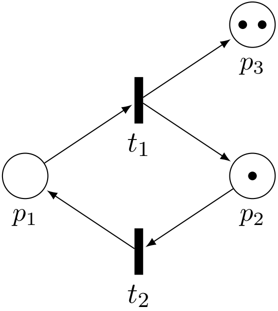


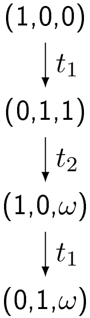


Non-finite reachability graph

Reduction: using the ω symbol

- a marking $\underline{\mu}'$ "dominates" an other node $\underline{\mu}$, if $\underline{\mu}$ is on the path from the root to $\underline{\mu}'$ and
  - $\forall p \in P\ \mu'(p) \geq \mu(p)$ (might be equal for all places!)
  - $\exists p \in P\ \mu'(p) > \mu(p)$ (at least one place has more tokens on it)
- the diverging number of tokens can be denoted by ω
- if the parent marking contains ω, then all of its children will have ω in the same place

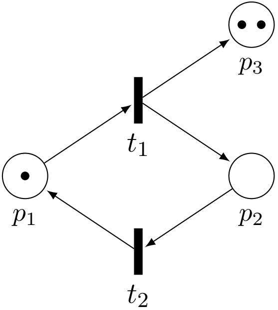


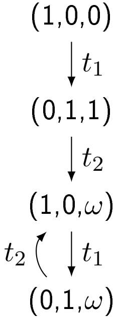


Non-finite reachability graph

Reduction: using the ω symbol

- a marking $\underline{\mu}$' "dominates" an other node $\underline{\mu}$, if $\underline{\mu}$ is on the path from the root to $\underline{\mu}$' and
  - $\forall p \in P \ \mu'(p) \geq \mu(p)$ (might be equal for all places!)
  - $\exists p \in P \ \mu'(p) > \mu(p)$ (at least one place has more tokens on it)
- the diverging number of tokens can be denoted by ω
- if the parent marking contains ω, then all of its children will have ω in the same place





Analysis of Petri net models

Dynamic properties

- *behavioural* (initial state dependent)
- *structural* (only depends on the structure graph)

Behavioural properties

- *reachabiliy* (coverability, controllability)
- *deadlocks*, liveness
- *boundedness*, safeness
- (token) conservation

Structural properties

- *state and transition invariant*: cyclic behaviour

Reachability of Petri net models

The notion of **reachability**: whether there exists

- to a given *[initial state ($\underline{\mu}^{(I)}$), final state ($\underline{\mu}^{(F)}$)]* pair
- *a firing sequence*, such that

$$\underline{\mu}^{(I)}[t_{j0} > \underline{\mu}^{(1)}[t_{j1} > \dots[t_{jk} > \underline{\mu}^{(F)}$$

The notion of **coverability**:

$$\underline{\mu}'' \geq \underline{\mu}' \quad \Leftrightarrow \quad \forall i: \; \mu_i'' \geq \mu_i'$$

The same as the usual controllability

Boundedness of Petri nets

Related properties to **boundedness**

- *finiteness (boundedness)*: Is the number of tokens finite for every initial state?
- *Safeness*: the bound is 1 for each place

Can be defined (examined) for the **whole net** or only for **a given set of places**

Conservative Petri net: the number of tokens is constant (resource-conservation)

Liveness of Petri nets

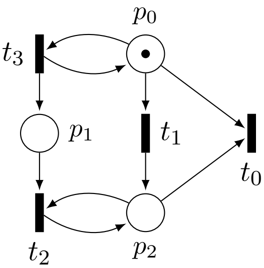
The notion of **liveness**: from a given initial state

- for a *transition*: is there a firing sequence when the transition is active?
- for a *set of transition*, for the whole net
- different levels of liveness for a transition t:
  - L0-live or dead: t can never fire in any firing sequence
  - L1-live or potentially fireable: t can fire at least once in some firing sequence
  - L2-live: t can fire at least k times in some firing sequence
  - L3-live: t can fire infinitely often in some firing sequence
  - L4-live: t is L1 live-for every marking

Deadlock: a non-final state from where there is no enabled (fireable) transition

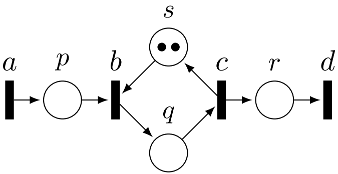
Liveness examples

- t_0 is L0-live (dead)
- t_1 is L1-live
- t_2 is L2-live
- t_3 is L3-live



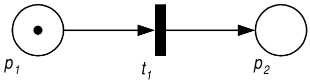
Resource allocation (queue with two servers)

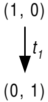
- p client to be served
- q client is being served
- r client has been served
- s free servers
- a arrival of client
- d departure of client
- b start of service
- c end of service
- **All transitions are L4-live**



Simple Petri net examples

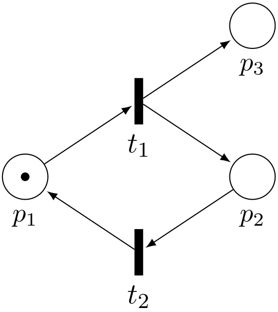
Deadlock: the marking $(0, 1)$

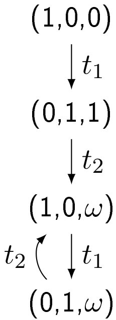




a, b,

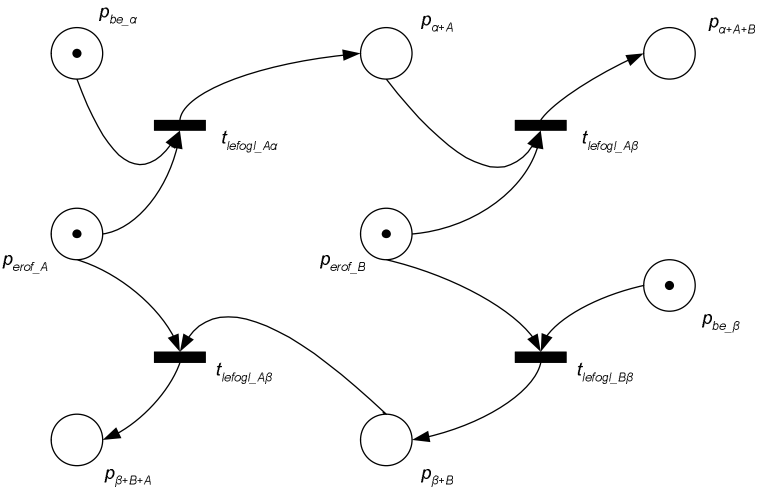
Non-bounded place: p_3



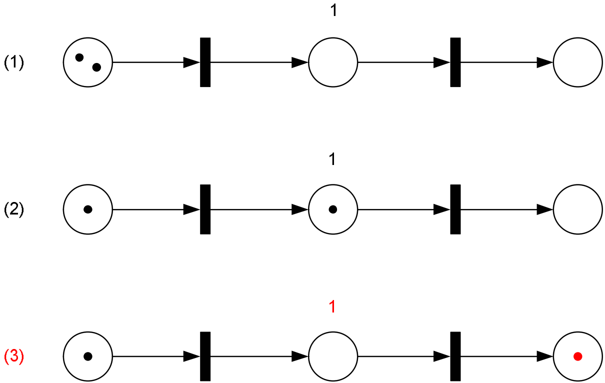


Resource allocation deadlock

Conflict situations



A safe place example



The capacity of the places changes the enabling of the transitions

Dynamic analysis methods of Petri net models – 1

Analysis of **behavioural properties**

- by constructing the *reachability graph*
- and *searching* on the vertices of the graph
- may be *NP-hard*

Problems:

- cyclic behaviour
- non-bounded places

Dynamic analysis methods of Petri net models – 2

Structural properties

- by constructing the *occurrence matrix* of the Petri net graph

$$H \in \mathbb{R}^{|P| \times |T|}$$

- and solving *linear set of equations*
- *polynomial time*, restricted importance

The elements of the occurrence matrix (for nets without loops)

$$h_{ij} = w(p_i, t_j) = \begin{cases} < 0 & \text{if} \quad p_i \quad \text{precondition} \\ > 0 & \text{if} \quad p_i \quad \text{consequence} \end{cases}$$

Place and transition invariants

Place invariant: set of conservation places $P_{INV} \subseteq P$
by solving the equation

$$z^T H = \underline{0}^T \quad , \quad z \in \mathbb{R}^{|P|}$$

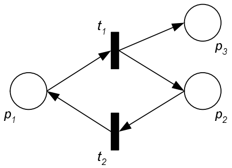
for its non-trivial solutions (z is the indicator vector)

Transition invariant: a set of transitions $T_{INV} \subseteq T$ that brings the system back to the initial state
by solving the equation

$$Hv = \underline{0} \quad , \quad v \in \mathbb{R}^{|T|}$$

for its non-trivial solutions (v is the indicator vector)

Place and transition invariants – Example



Place invariant:

$$\begin{bmatrix} z_1 & z_2 & z_3 \end{bmatrix} \cdot \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix} \quad \Rightarrow \quad z_1 = z_2$$

Transition invariant: without p_3 !!

$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Rightarrow \quad v_1 = v_2$$