GRAFCET and Petri Nets

Prof. J.-D. Decotignie
CSEM Centre Suisse d'Electronique et de Microtechnique SA
Jaquet-Droz 1, 2007 Neuchâtel
jean-dominique.decotignie@epfl.ch

Outline

- introduction
- GRAFCET
- GEMMA
- Petri nets

csem

Introduction

- Description of process evolution
- Description of process interaction
  - ☞ Petri nets
- Functional specifications
- Requirements for automata
  - ☞ GRAFCET

csem

GRAFCET

- Graphe de Commande Etape Transition (Step Transition Control Graph)
- 2 levels
  - Functional specification
  - Operational specification
- Described in the international standard IEC 848 under the name of function charts
- [Dav90] R. David, A. Alla, « Petri nets and Grafcet", Prentice Hall, 1992.
- R. David, Grafcet: a powerful tool for specification of logic controllers, IEEE Trans. on Control Systems Technology, vol. 3, Issue 3, Sept. 1995, pp.253 - 268

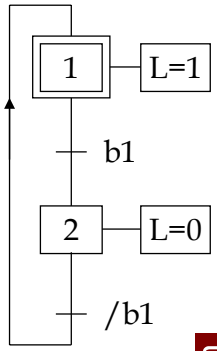
csem

GRAFCET

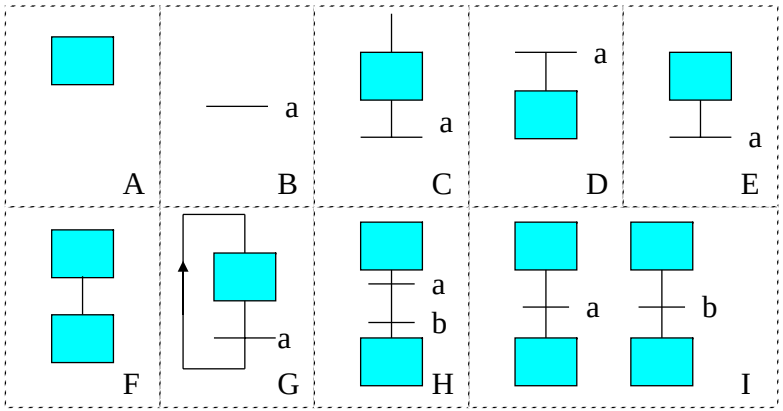
- Definition
- Evolution rules
- Defining actions
- Taking time into account
- Defining transition conditions
- Execution algorithm
- Macrostep and macroactions

GRAFCET - definition

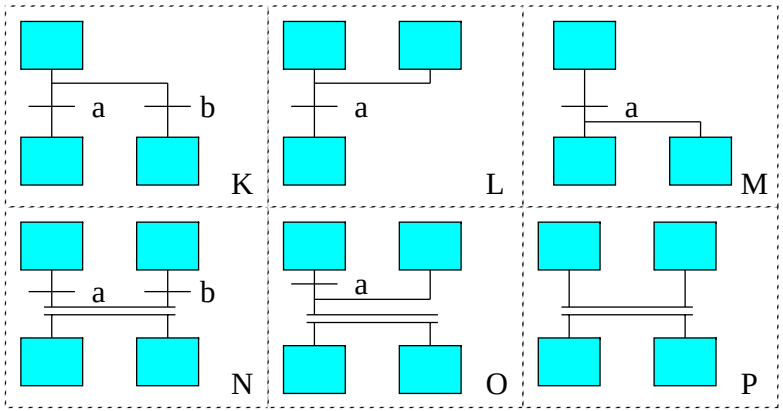
- Directed graph derived from PN
  - Quadruple C = {S, TR, A, M_0}
  - N steps $s_i \in S$; each step s_i may be active (X_i= true) or not (X_i=false). M_0 denotes the set of steps active at startup
  - L transitions $tr_i \in TR$; to each transition is associated a boolean condition (receptivity)
  - Steps and transitions are linked by arcs $a_i \in A$
- + EVOLUTION CONDITIONS



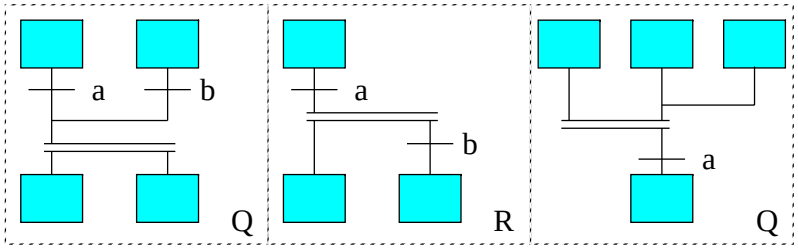
Exercise - syntax



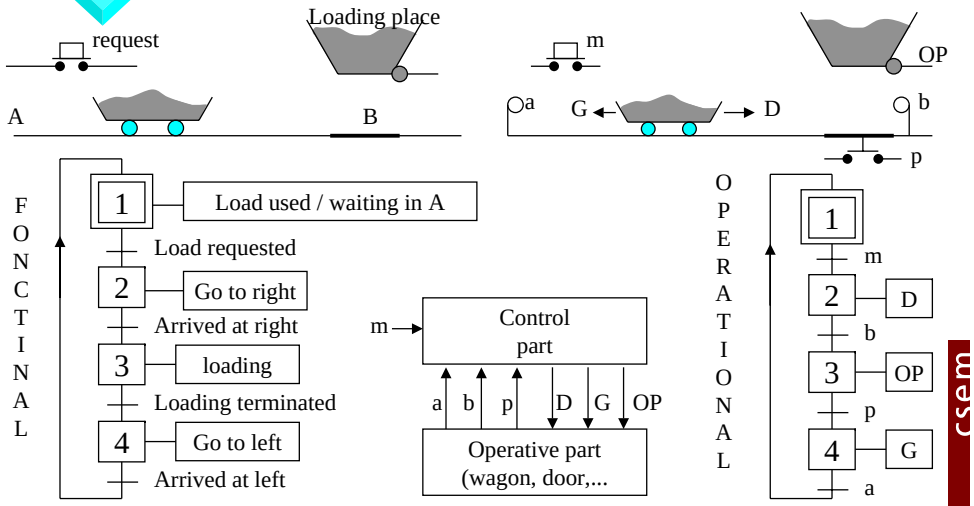
Exercise - syntax (2)



Exercise - syntax (3)



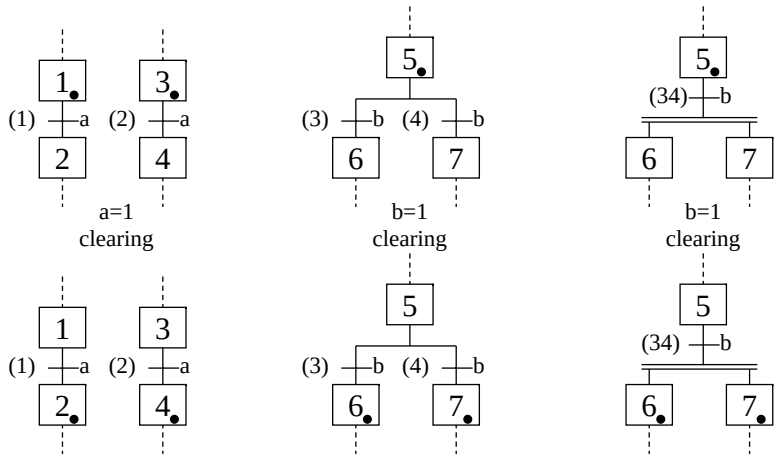
GRAFCET – an example



Evolution Conditions

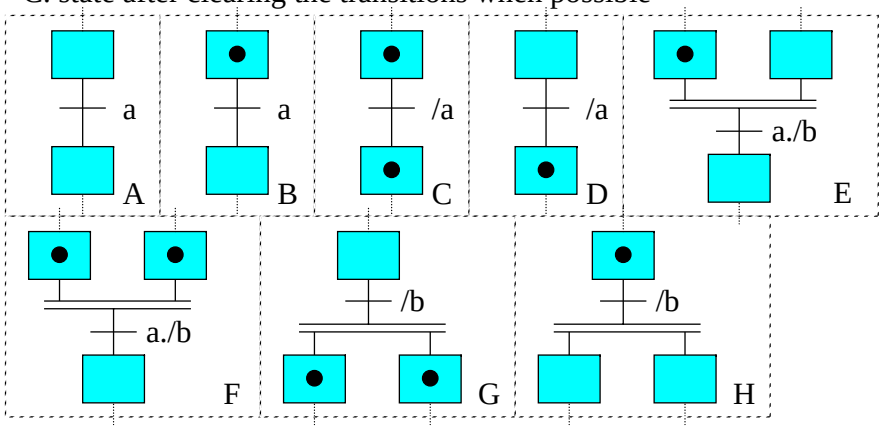
- Evolution is performed from the initial state defined as M_0 by clearing transitions according to 5 rules:
  - all steps immediately preceding the transition must be active, the transition is then enabled
  - then, if receptivity is true, the transition may be cleared
  - all transitions that may be cleared are cleared simultaneously
  - clearing a transition leads to the activation of all immediately following steps and deactivation of all immediately preceding steps
  - a step that is activated and deactivated remains active

Evolution Conditions (2)



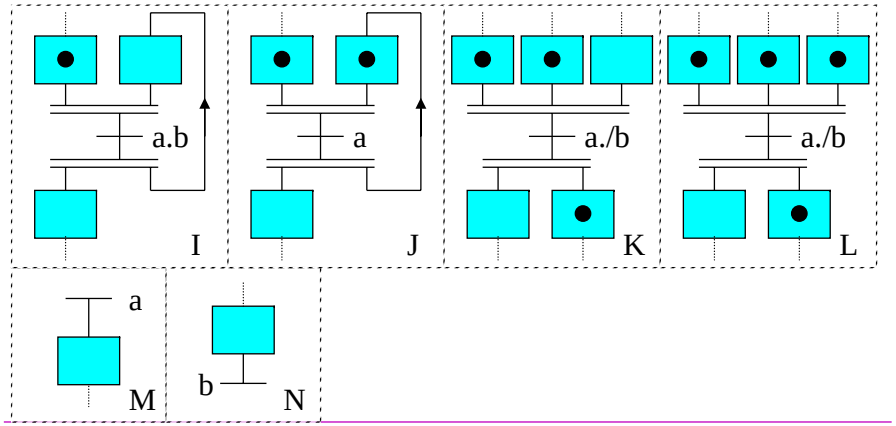
Exercise - transitions

A. enabled transitions; B. firable transitions if a=1 and b=0;
C. state after clearing the transitions when possible

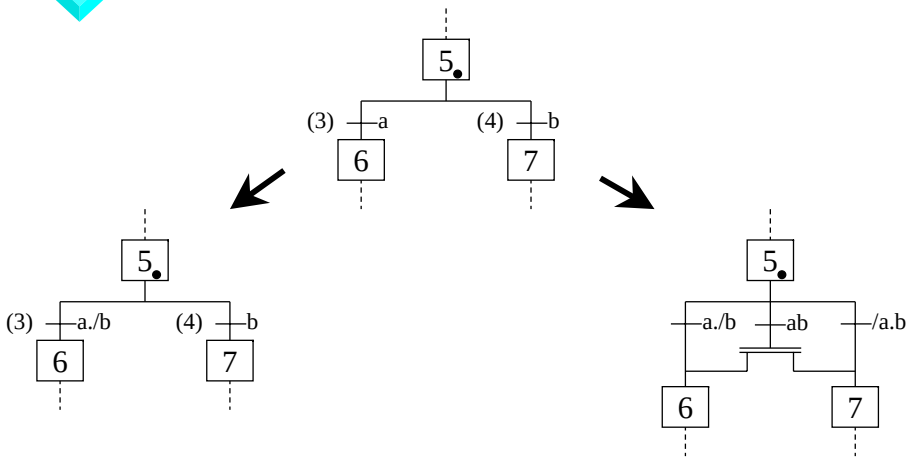


Exercise - transitions (2)

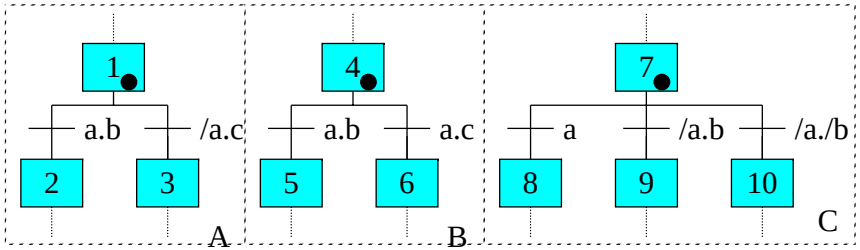
A. enabled transitions; B. firable transitions if a=1 and b=0;
C. state after transition when possible



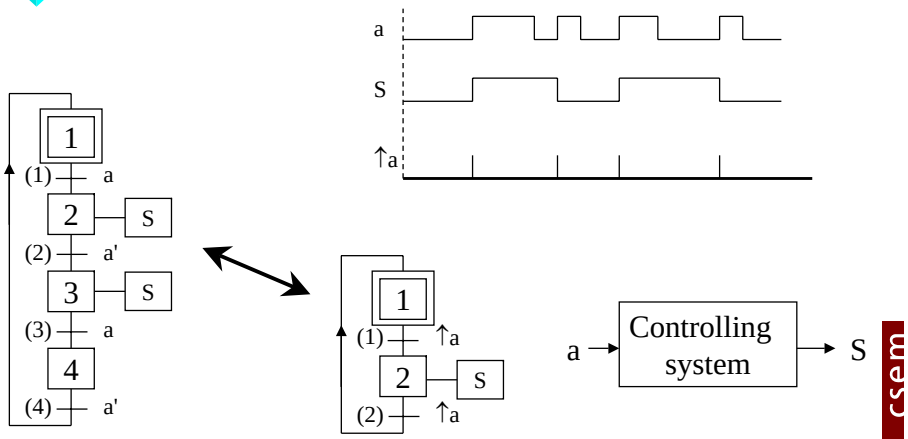
Conflicts



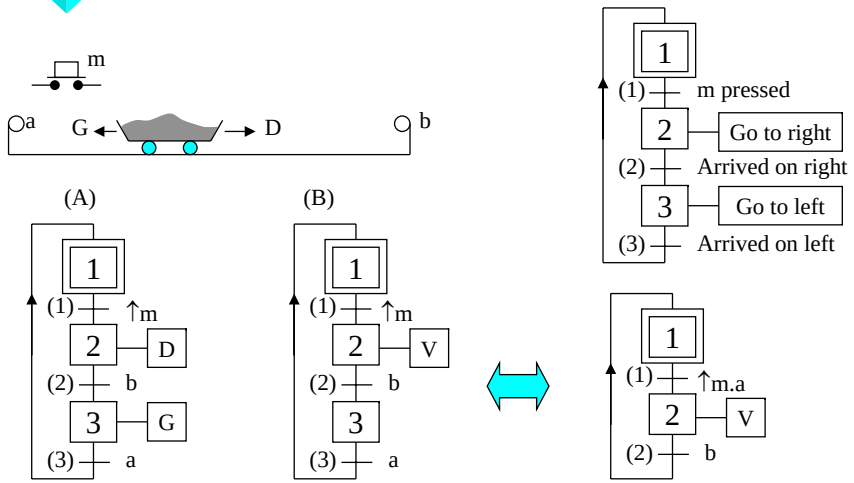
Exercise: is there a conflict ?



Divider by 2

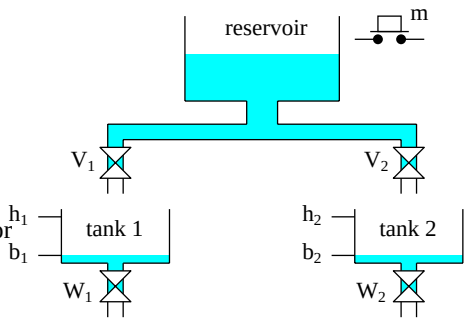


Examples of logical graphs

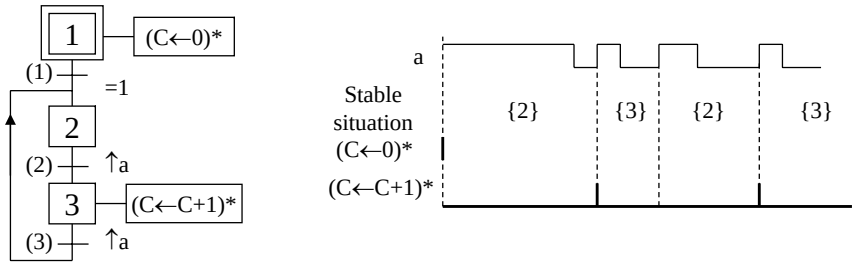


Example

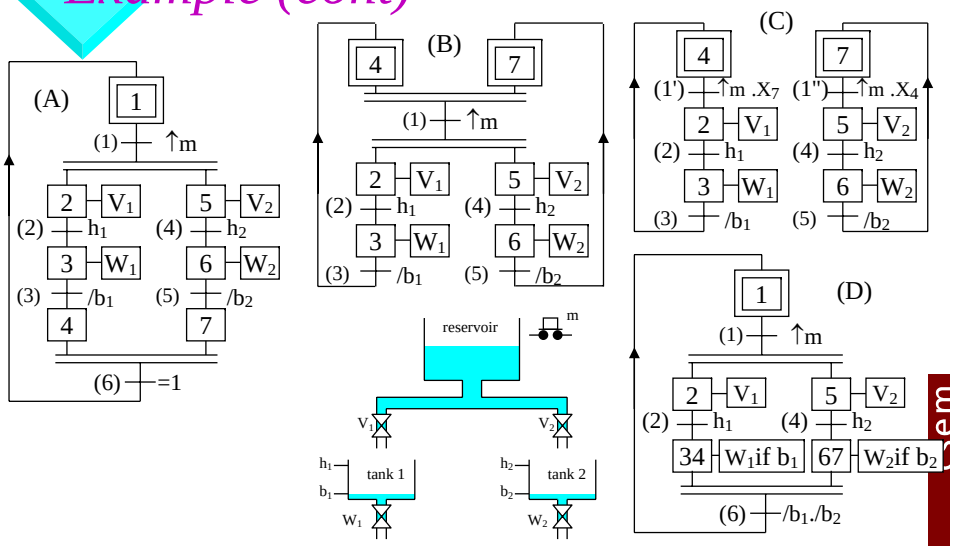
- Initial conditions
  - Tanks empty, valves closed
- Sensors and actuators
  - V_i, W_i = 1 if open
  - h_i, b_i = 1 if level above sensor
- operations:
  - Fill each tank until above h_i, close valve V_i and open W_i until level below b_i. Procees cannot be repeated before both tanks are empty



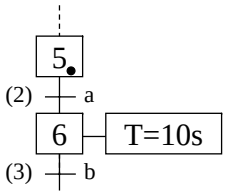
Increment a counter

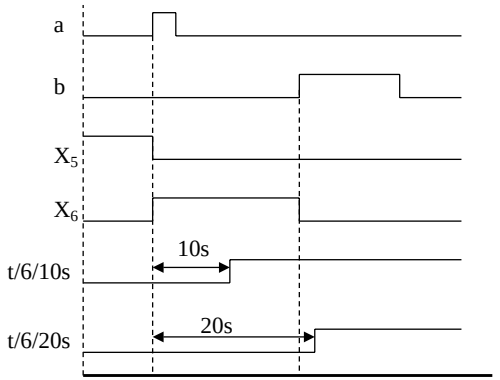


Example (cont)

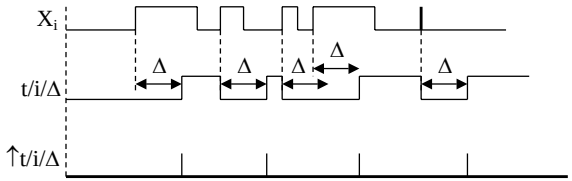


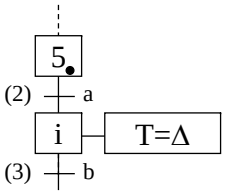
Taking time into account



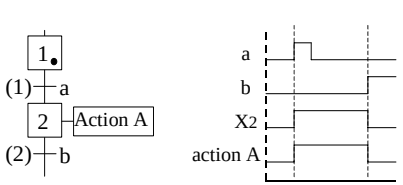


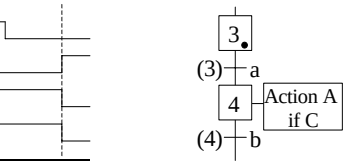
Timer behavior

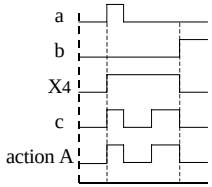


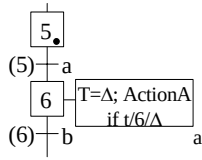


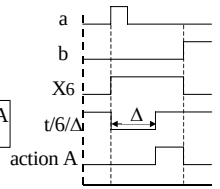
Continuous actions (à niveau)

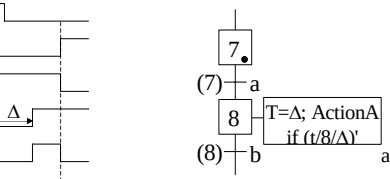


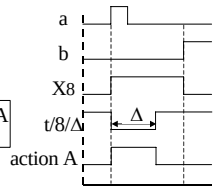






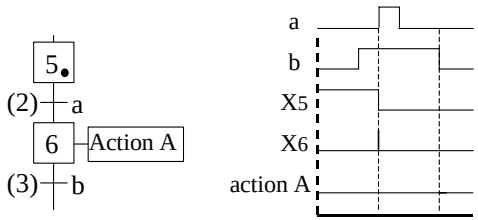




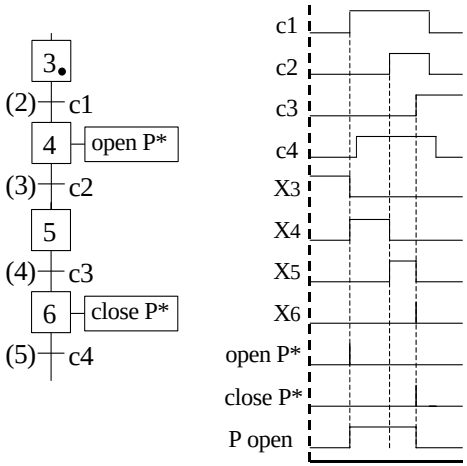


Duration of a continuous action

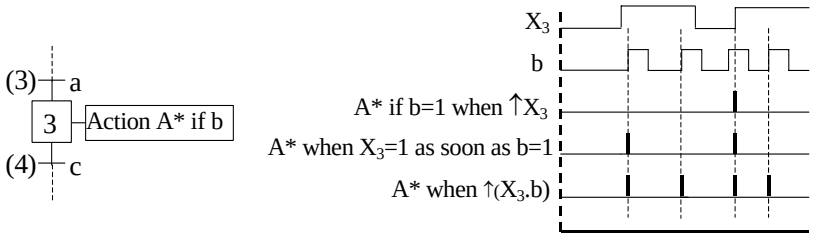
- Continuous actions are only defined for stable situations



Pulse shaped actions



Conditional Pulse Shaped Actions

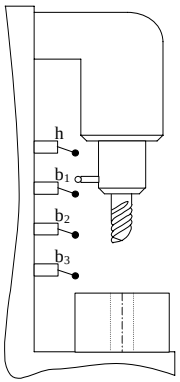


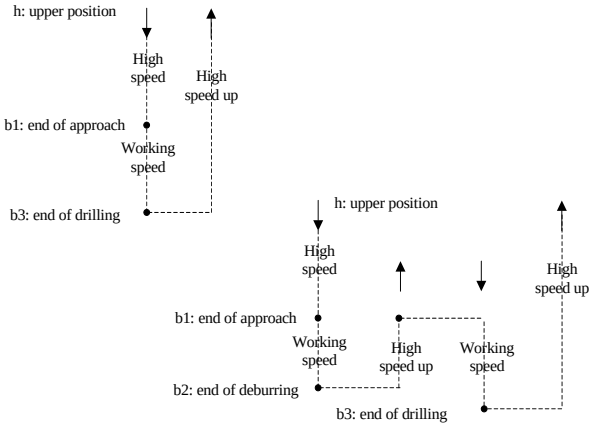
- A* if b is ambiguous. Unsufficient notation !!
- We shall not keep this capability

Actions types

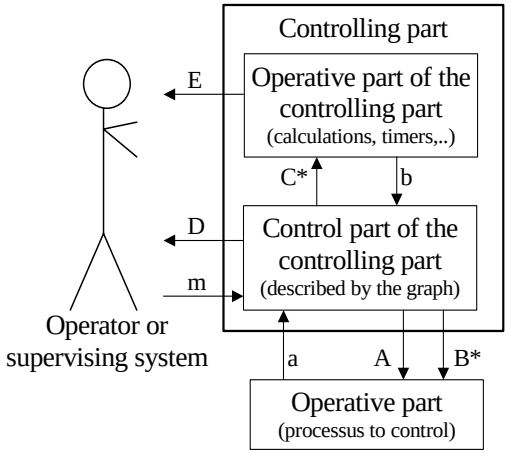
- Continuous actions (à niveau)
  - unconditional
  - conditional
    - Condition is a boolean variable
    - Condition is based on a timer
- Pulse shaped actions
  - Always unconditional

Exercise: drilling machine





Actions and outputs



inputs: a,b,m
actions: A, B*, D, C*
outputs: A, B*, D, E

Transition conditions

- variables
  - External boolean variables
    - Coming from the process or from the external world
    - relative to time (t/i/Δ)
  - Internal boolean variables
    - Relative to the state of a step (Xi)
    - State of the operative part (of the controlling part)
- receptivity $R_i = C_i E_i$
  - condition C_i = boolean combination of variables
  - event E_i = rising (falling) edge of an external variable (e always occurring event)

Events

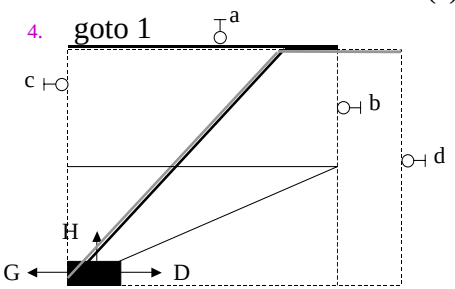
- $\forall a \in \{0,1\}$ si a(t)=1 when $t \in [t1, t2[$ et $t \in [t3, t4[$
  - definition 1: $\uparrow a$ occurs in t1 and in t3
  - definition 2: $\downarrow a$ occurs in t2 and in t4
  - definition 3: $\uparrow a.b$ occurs at the same instant as $\uparrow a$ each time b=1 at this instant
  - definition 4: $\uparrow a.\uparrow b$ occurs when $\uparrow a$ and $\uparrow b$ occur at the same instant
- Show that
  - $\uparrow a = \downarrow a'$ $\quad \uparrow a.a = \uparrow a$ $\quad \uparrow a.a' = 0$ $\quad \downarrow a.a = 0$
  - $\uparrow a.\uparrow a = \uparrow a$ $\quad \uparrow a.\uparrow a' = 0$ $\quad \uparrow a.e = \uparrow a$

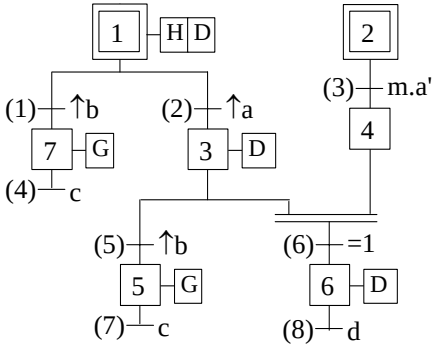
Graph interpretation

- 2 persons in front of the same graph and assuming the same sequence of inputs should deduce the same sequence of outputs.
- Asumptions:
  - 2 uncorelated events may not occur at the same time
  - The graph has enough time to reach a stable state before the occurrence of the next event

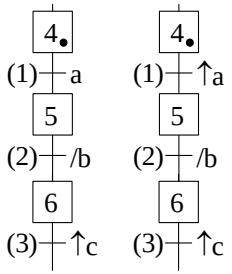
Interpretation Algorithm (without stability search)

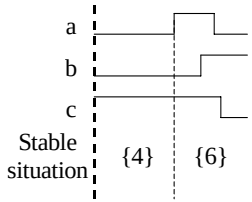
1. Read the state of the inputs
2. evolve (one or more simultaneous clearings)
3. Execute actions
4. goto 1





Iterated Clearing





Interpretation Algorithm (with stability search)

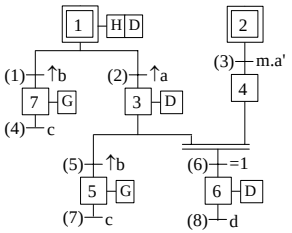
1. initialization: activation initial step(s), execution of associated pulse shaped actions; go to 5
2. If an external event Ei occurs, determine the set T_1 of transitions that can be cleared on Ei; if $T_1 \neq \varnothing$ goto 3, else modify conditional actions and goto 2
3. Clear transitions that can be. If, after clearing, the state is unchanged goto 6
4. Execute the pulse shaped actions that are associated to the step that were activated at step 3 (incl. timers)
5. Determine the set T2 of transitions that can be cleared on occurrence of e. If $T_2 \neq \varnothing$, goto 3

Interpretation Algorithm (with stability search)(2)

6. A stable situation has been reached
   1. Determine the set A_0 of continuous actions that must be deactivated (incl. conditional actions)
   2. Determine the set A_1 of continuous actions that must be activated (incl. conditional actions)
   3. Set to 0 all the actions that belong to A_0 and not to A_1. Set to 1 all the actions that belong to A_1
   4. go to 2

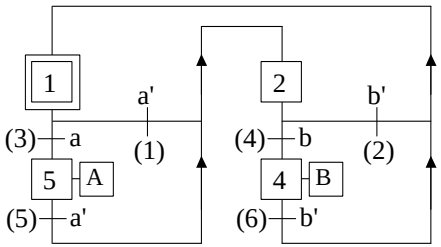
Interpretation Algorithm (with stability search)(3)

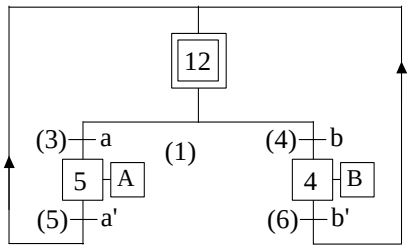
- stage 1: situation {1,2}
- stage 5: (1) (2) et (3) enabled, cannot be cleared on e, $T_2=\varnothing$
- stage 6: stable situation {1,2}, $A_0=\varnothing$, A_1= {H,D}
- stage 2: m=1, on e, T_1= {3}
- stage 3: clearing (3)
- stage 4: no pulse shaped action
- stage 5: $T_2=\varnothing$
- stage 6: $A_0=\varnothing=A_1$
- stage 2: on ↑a, T_1= {2}
- stage 3: clearing (2)



- stage 4: no pulse shaped action
- stage 5: T_2= {(6)}
- stage 3: clearing (6)
- stage 4: no pulse shaped action
- stage 5: $T_2=\varnothing$
- stage 6: A_0= {H,D}, A_1= {D}
- .....

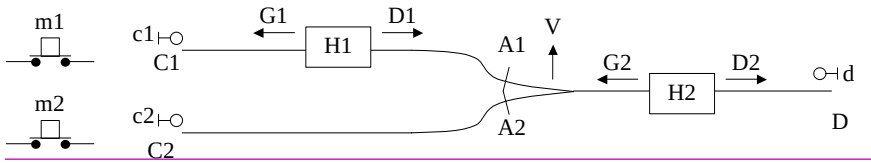
Non stable cycle



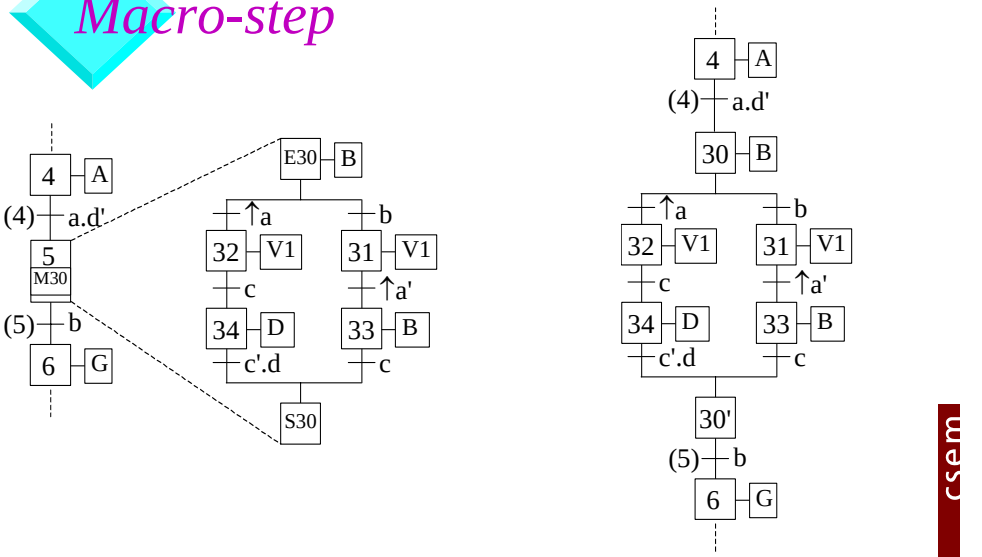


Exercise [Dav90]

Let H1 and H2 be two wagons carrying goods from loading points C1 and C2 respectively to an unloaded point D. Variables c1,c2 and d correspond to end of track sensors. They turn to 1 when a wagon is present at the given point. Variable a1 turns to 1 when the front wheels of wagon H1 are on the tracks between A1 and D (same for a2 if wagon H2 is between A2 and D).
If wagon H1 is in C1 and if button m1 is pressed, a cycle C1 → D → C1 starts with a possible wait in A1 until the track common to both wagons is free, then a wait in D during 100s. Wagon H2 performs the same on C2 → D → C2. The path C1-D is set when V equals 1, otherwise path C2-D is set.



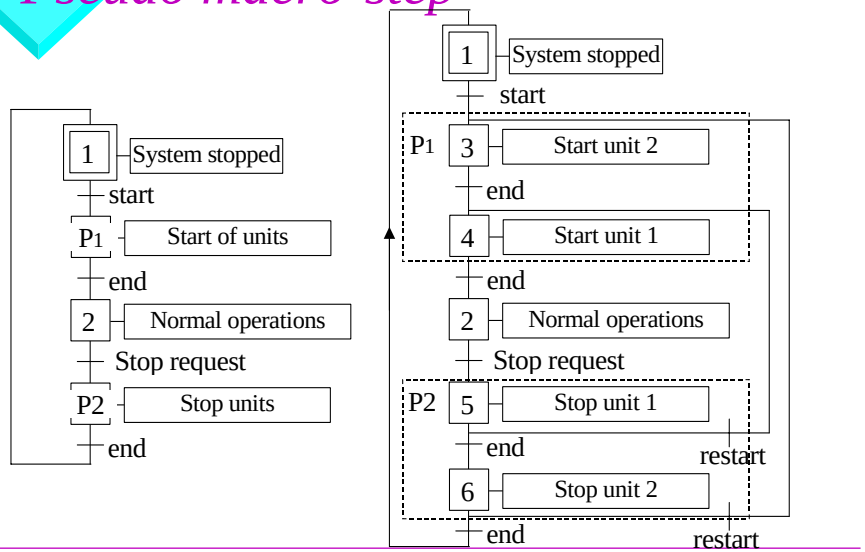
Macro-step



Pseudo macro-step

- Represents a set of steps
- All arcs do not necessarily go to the same step (no need for a unique entry step)
- All arcs do not necessarily go to the same step (no need for a unique exit step)
- All arcs entering or leaving the pseudo macro-step need be represented

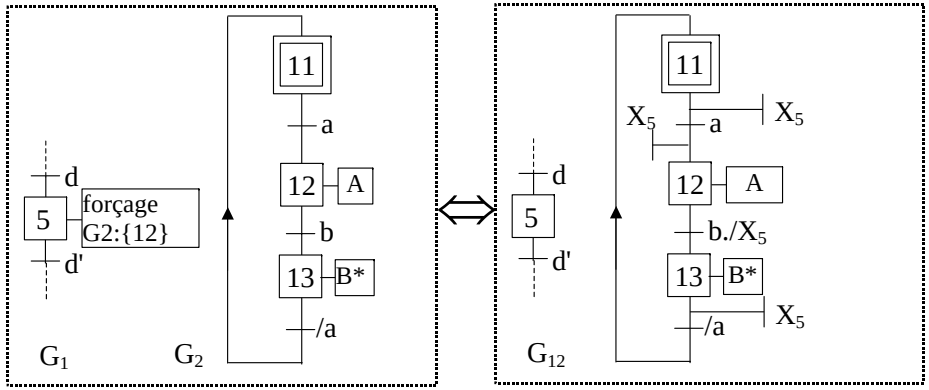
Pseudo macro-step



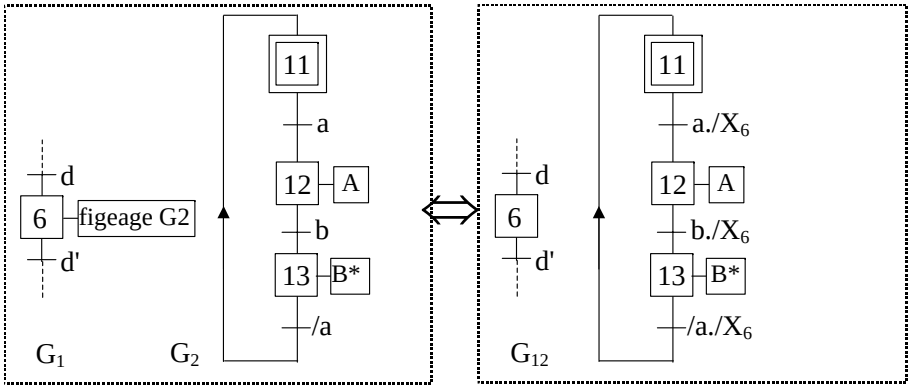
Macro actions

- Globaly act on a GRAFCET
- Do not increase the expression power
- But ease specification
- Exhibit same properties as actions
  - Continuous
    - Forçage(forcing), figeage(freezing), masquage(masking)
  - Pulse shaped
    - Forcer (force), masquer(mask), démasquer(unmask), figer(freeze), libérer(release)

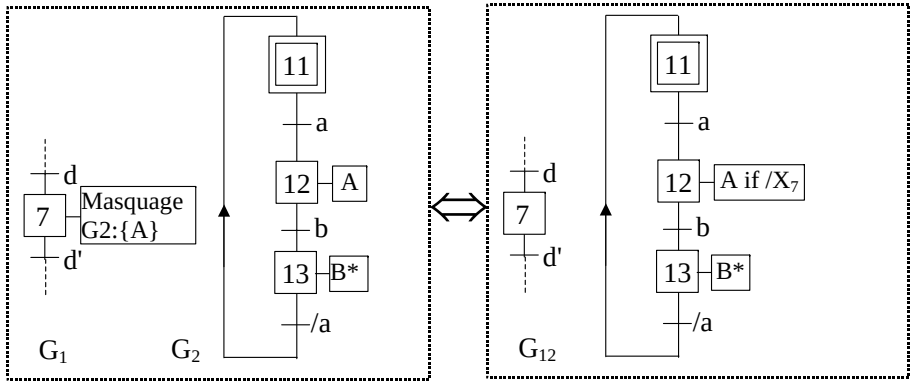
Macroaction: Forcing (Forçage)



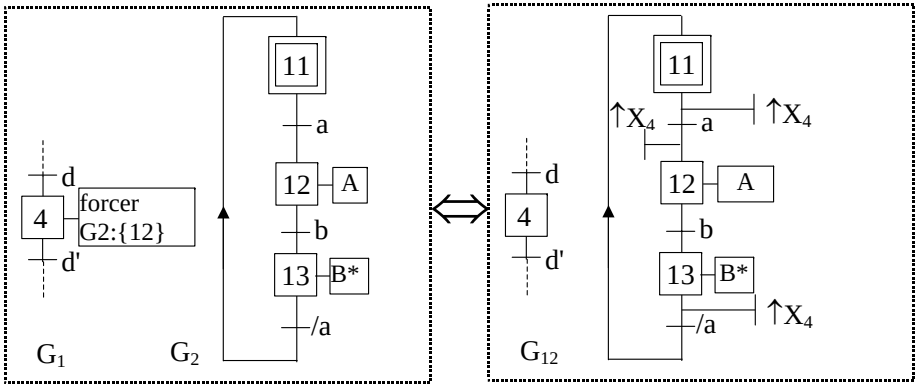
Macroaction: Freezing (Figeage)



Macroaction: Masking (Masquage)



Macrocaction : Force (Forcer)



Outline

- introduction
- GRAFCET
- GEMMA
- Petri nets

GEMMA

- Guide d'Etude des Modes de Marche et d'Arrêt

- Problems with GRAFCET
- Objectives
- Concepts
- How to use GEMMA
- Advantages
- Drawbacks

Problems with GRAFCET

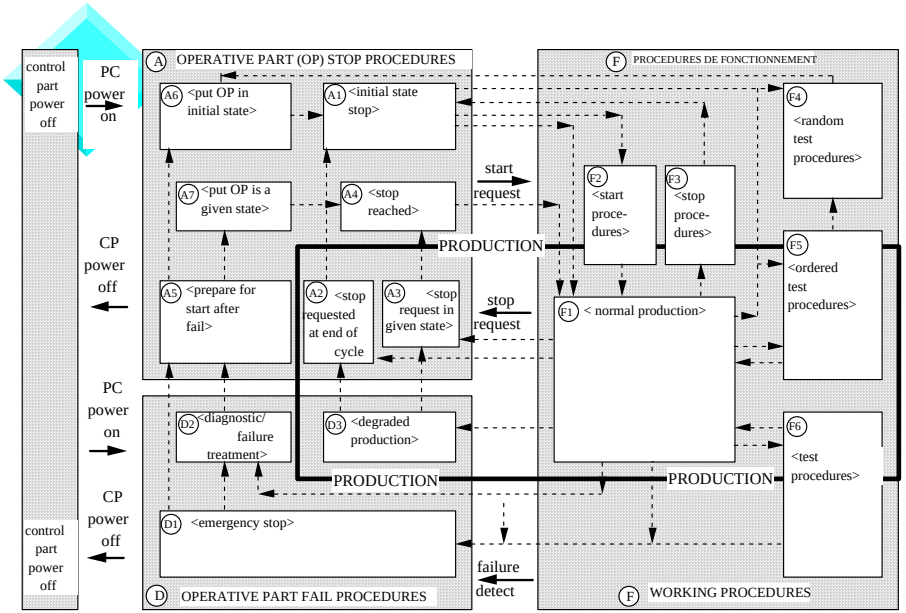
- GRAFCET is good to describe normal operation but:
  - Not well suited to describe
    - Emergency cases
    - Manual modes
    - Degraded modes
  - Does not favor a good separation of operating modes
  - Does not give a clear vocabulary concerning modes
  - Does not include any security aspect

Objectives of GEMMA inventors

- Create a graphical tool that can deal with:
  - Emergency cases
  - Manual modes
  - Degraded modes
- Estblish a precise vocabulary
- Tool show present itself as a guide (checklist)

GEMMA concepts

- Assumes that controlling part is operational
- Separate production states from other states
- Distinguishes 3 categories of operating modes
  - Working procedures
  - Stopping procedures
  - Failure procedures

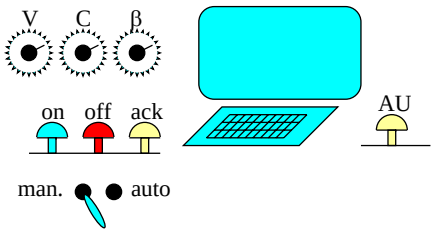


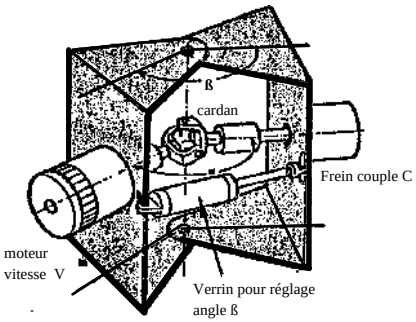
How to use GEMMA

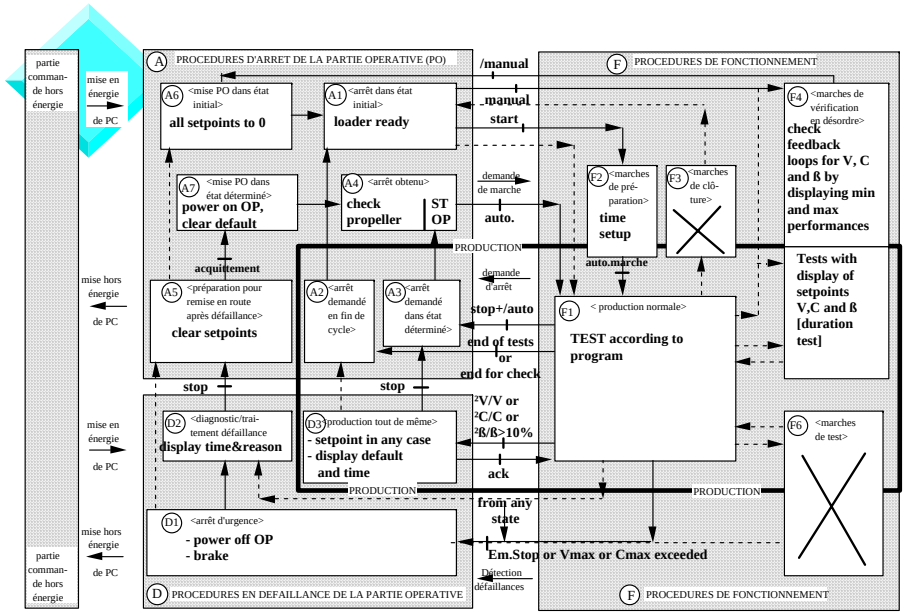
- Selection of modes
- Showing relationships
- Search for evolution conditions

Example – joint test bed

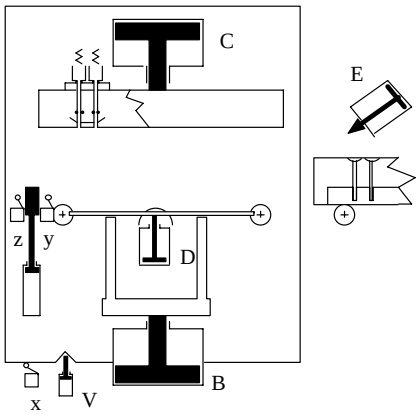
- 3 tested values V, C, β
- Long tests on a typical run
- Periodic request for withdraw and check



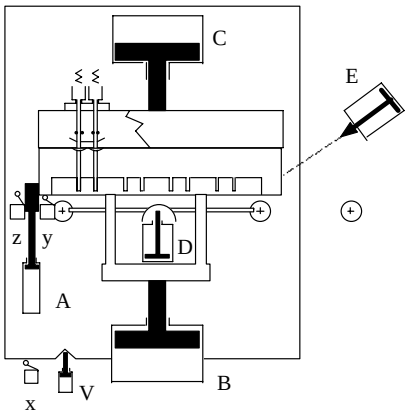




Example – valve test

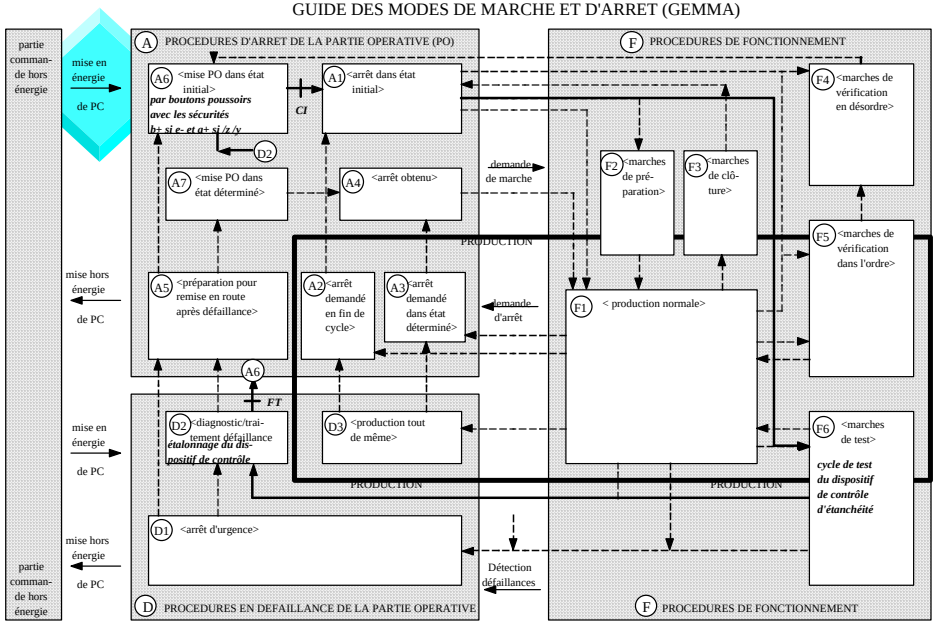


Before feeding



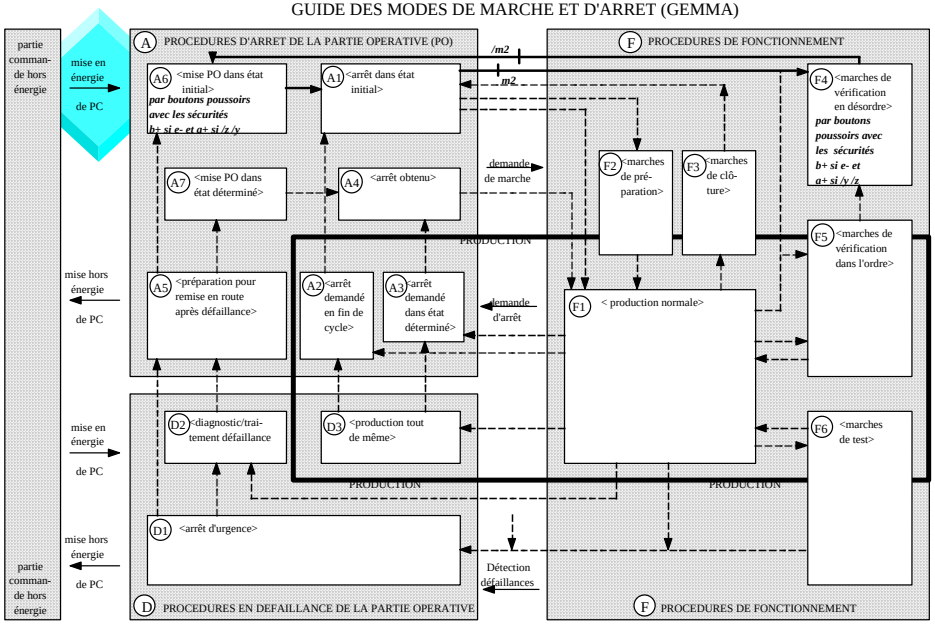
After feeding

GUIDE DES MODES DE MARCHE ET D'ARRET (GEMMA)

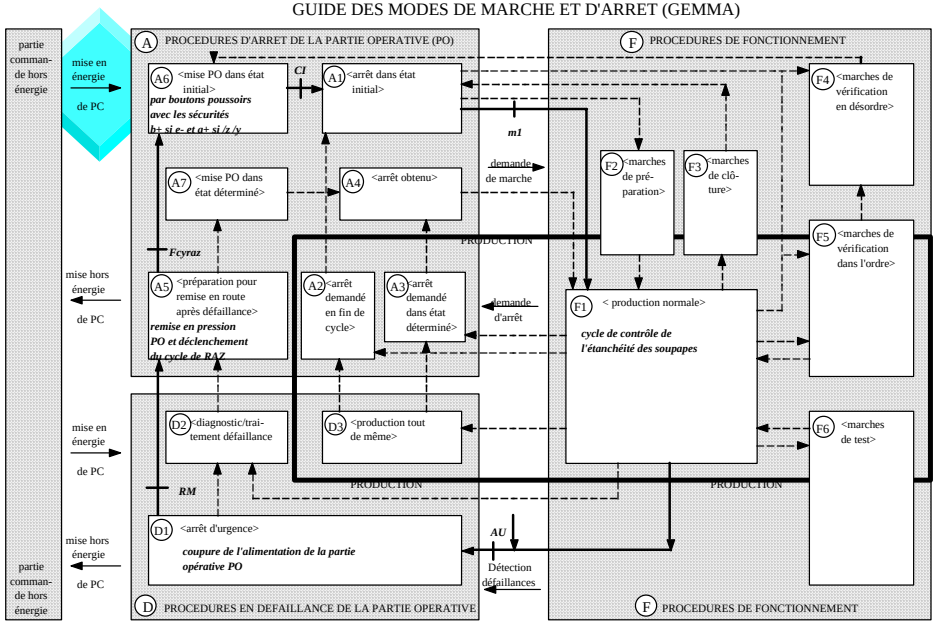


Système de test d'étanchéité de soupapes: boucle de test

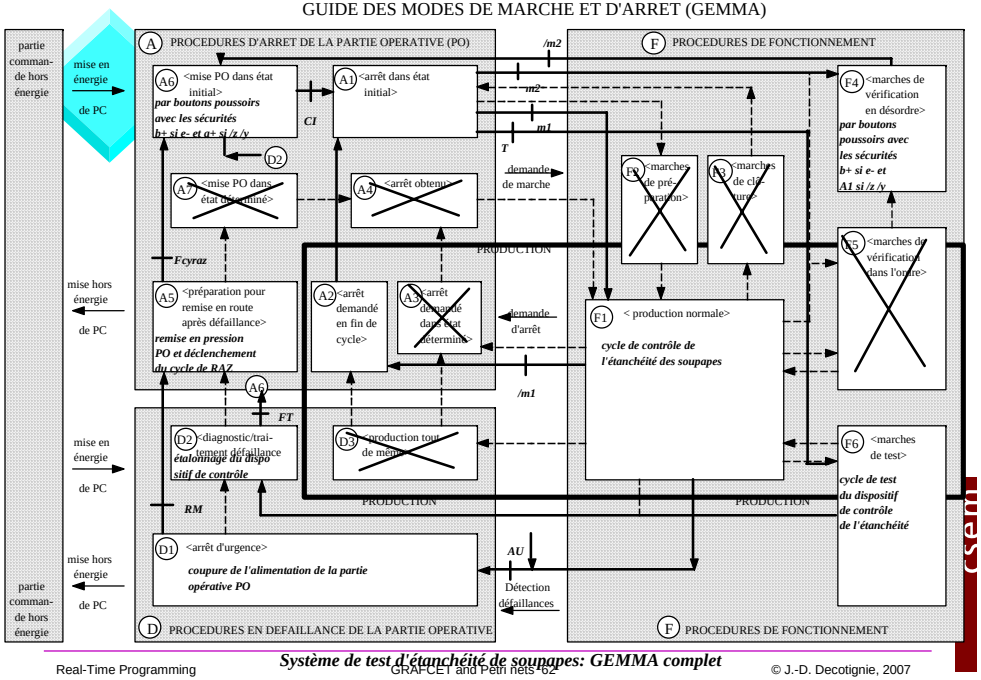
GUIDE DES MODES DE MARCHE ET D'ARRET (GEMMA)



Système de test d'étanchéité de soupapes: marche manuelle

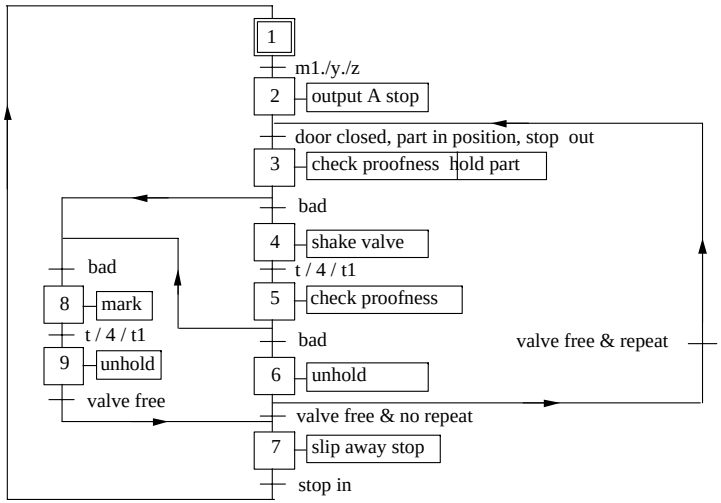


Système de test d'étanchéité de soupapes: arrêt d'urgence



Système de test d'étanchéité de soupapes: GEMMA complet

Valve test - GRAFCET



GEMMA - analysis

- Advantages
  - concise and synthetic
  - graphic
  - Non ambiguous vocabulary
  - Clearly shows manual parts
  - Helps defining securities and operator commands
  - May be used as user manual
- Drawbacks
  - Difficult to use of complex cases
  - Some of the limitations of GRAFCET

How to manage an automation project

- Phases according to GEMMA tenants
  - Define requirements
  - Use GEMMA for start/stop operations
  - Create GRAFCET
  - Select control technology
  - Technological realization
  - Trial and commissioning

Outline

- introduction
- GRAFCET
- GEMMA
- Petri nets

Petri nets

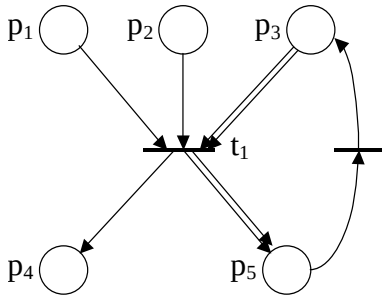
- Invented by C. Petri in 1962 in his PhD thesis ("Kommunikation mit Automaten)
- J. Peterson, "Petri Net Theory and the Modelling of Systems", Prentice Hall, 1981.
- R. David, A. Alla, "Du Grafcet aux réseaux de Petri", Hermes, 1990.
- T. Murata, Petri nets: Properties, analysis and applications, Proc. of the IEEE, Vol. 77 (4), April 1989, pp.541 - 580
- 2 view points
  - Theoretical
    - Definitions and evolution rules
    - Properties
  - Application to functional specifications

Petri Nets

- Definition
- Marking
- PN types
- Modelling with Petri nets
- Modelling synchronization and cooperation between tasks
- Design
- Analysis and validation

Petri nets - definition

- Directed graph
  - tuple C={P, T, I, O}
  - N places $p_i \in P$
  - L transitions $t_i \in T$
  - Input matrix I [LxN] places → transitions
  - Output matrix O [LxN] transitions → places

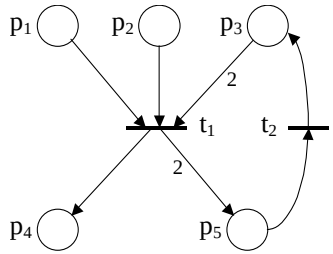


Petri net example

- $P = \{p_1, p_2, p_3, p_4, p_5\}$
- $T = \{t_1, t_2\}$

$$I = \begin{array}{c|ccccc} & p_1 & p_2 & p_3 & p_4 & p_5 \\ \hline t_1 & 1 & 1 & 2 & 0 & 0 \\ t_2 & 0 & 0 & 0 & 0 & 1 \end{array}$$

$$O = \begin{array}{c|ccccc} & p_1 & p_2 & p_3 & p_4 & p_5 \\ \hline t_1 & 0 & 0 & 0 & 1 & 2 \\ t_2 & 0 & 0 & 1 & 0 & 0 \end{array}$$

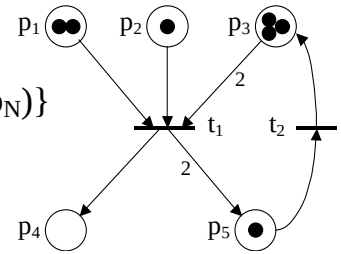


$\#(p_i, I(t_j))$ weight of arc $p_i \rightarrow t_j$

$\#(p_i, O(t_j))$ weight of arc $t_j \rightarrow p_i$

Petri nets - marking

- marking
  $M = \{m(p_1),....., m(p_i), ..., m(p_N)\}$
  with $m(p_i)$ = number of tokens in place p_i
- Initial marking M_0,
  example $M_0=\{2,1,3,0,1\}$
- Evolution by firing transitions
- Can then represent behavior (dynamic)

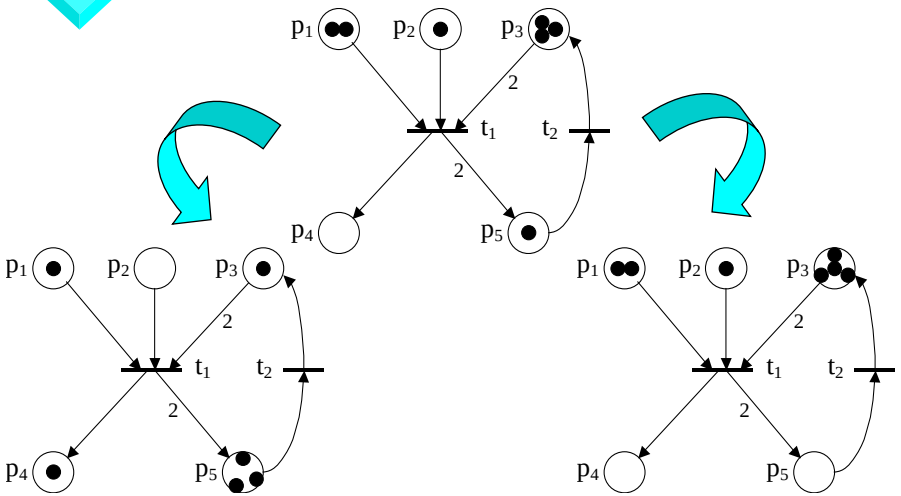


PN – evolution rules

- a transition t_i may be fired <u>iif</u> for all i
  $$m(p_i) \geq \#(p_i, I(t_j))$$
- Only one transition may fired at a time
- A transition is fired instantaneously by:
  - Removing in each place that immediately preceeds the transition a number of tokens equal the weight of arc that links them
  - Adding in each place that immediately follows the transition a number of tokens equal the weight of arc that links them
    $$m(p_i) := m(p_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j))$$

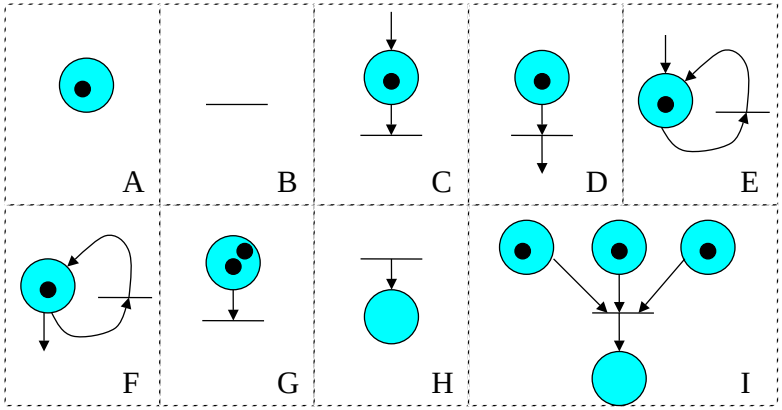
EVOLUTION IS NOT DETERMINISTIC

Evolution rules



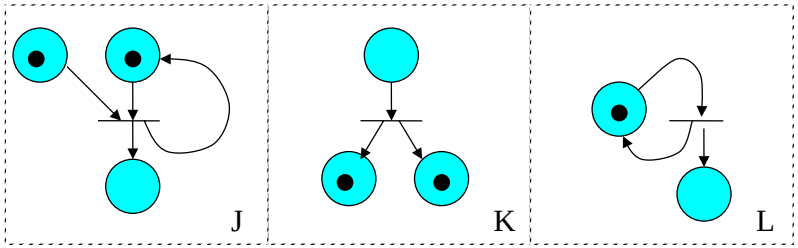
Exercise - syntax

For each of the graphs below, answer the following questions:
1. Is it a PN ? 2. What are the validated transitions ?
3. What will be the marking after firing ? 4. What are the validated transitions



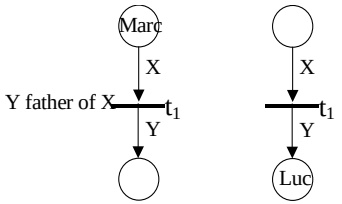
Exercise – syntax (2)

For each of the graphs below, answer the following questions:
1. Is it a PN ? 2. What are the validated transitions ?
3. What will be the marking after firing ? 4. What are the validated transitions



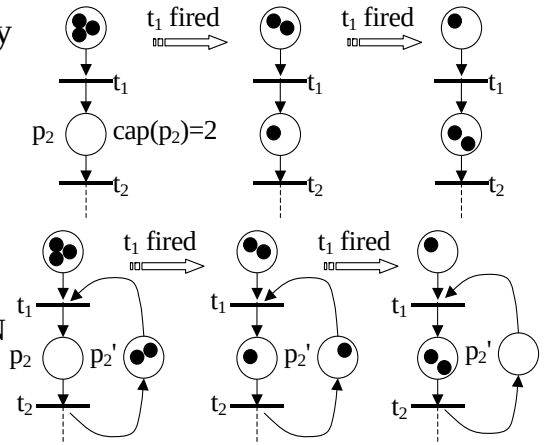
PN types

- Generalized PNs
  - There a weight on each arc
  - Can be transformed into ordinary PN
- PNs with predicates
  - Cannot be transformed into ordinary PN



PN types (2)

- PNs with capacity
  - Can be transformed into ordinary PN

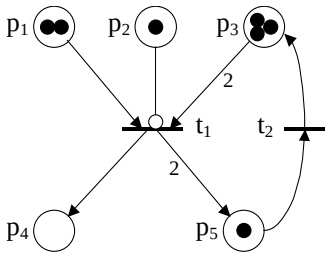


PN types (3)

- Colored PNs
  - Tokens have colors
  - Any colored PN with a finite number of colors may be transformed into an ordinary PN
- PNs with priorities
  - Cannot be transformed into ordinary PN
- Continuous PNs
  - The number of tokens can be a real number
  - Cannot be transformed into ordinary PN

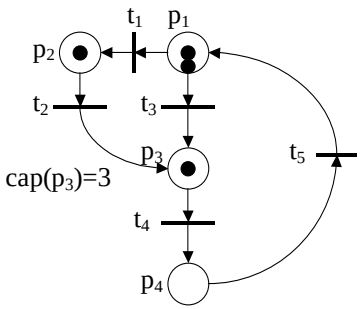
PN types (4)

- PNs with inhibiting arcs
  - Cannot be transformed into ordinary PNs
- No autononous PNs
  - Several types
    - Synchronized PNs
    - Time PNs
    - Stochastic PNs
    - Stochastic Time PNs
    - ....
  - Cannot be transformed into ordinary PNs



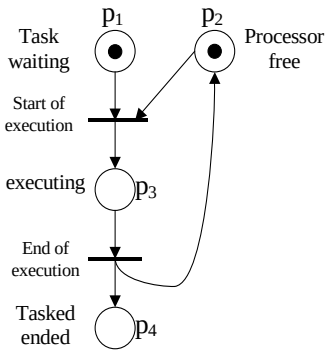
Exercise

Transform this PN into an ordinary PN



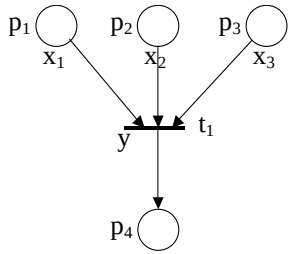
Modelling with PNs

- actions associated to places
  - Activated by token presence
- actions associated to transitions (short duration)
  - Activated by transition firing

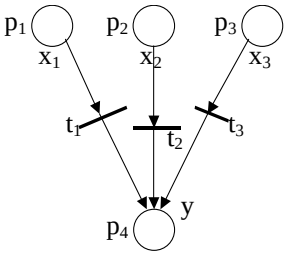


Conditions « and" & « or"

- type "AND"

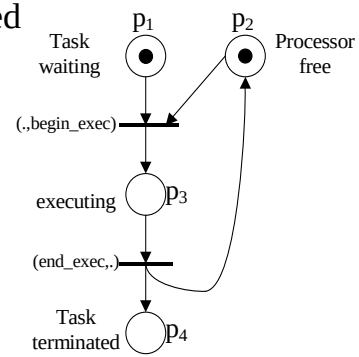


- type "OR"

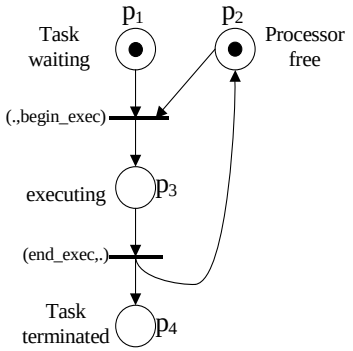


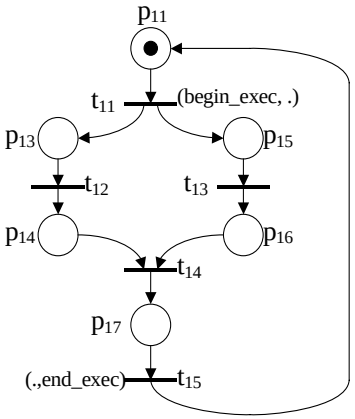
Introduction of external conditions

- When synchronisation is required controlled system → controlling system
- Label on the transition (if condition, action)
- Transition may be fired iif external condition satisfied
- Timers (extern or minimal sejourn duration in a place)

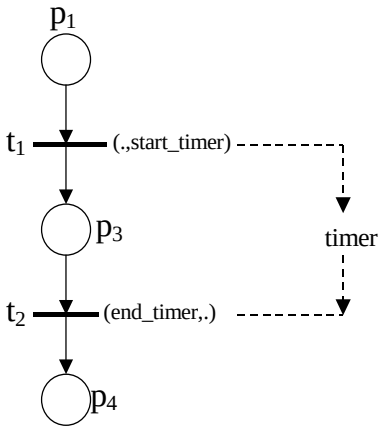


Introduction of external conditions (2)



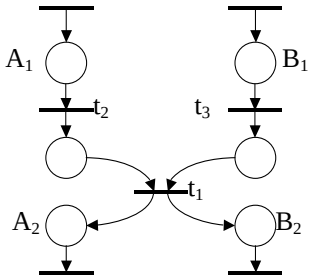


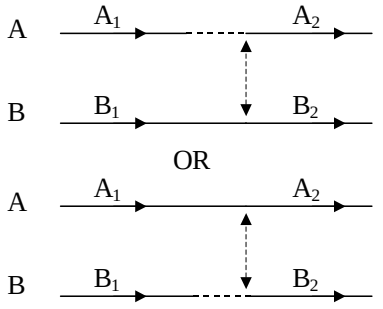
Introduction of external conditions (3)



Modelling task synchronisation mechanisms

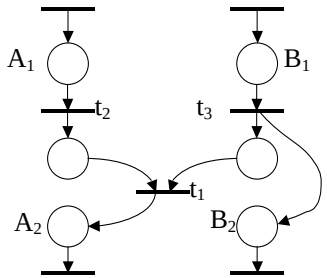
- Mutual synchronisation (rendez-vous)

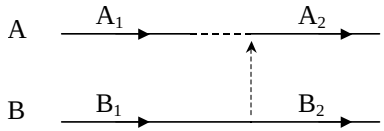




Modelling task synchronisation mechanisms(2)

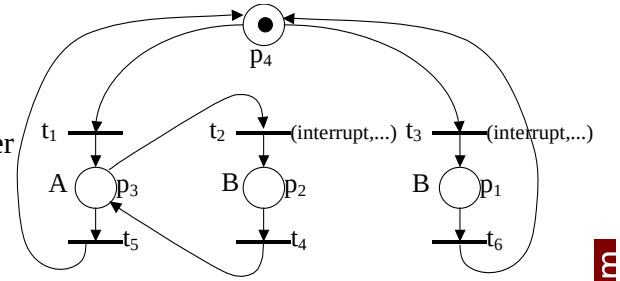
- mailbox





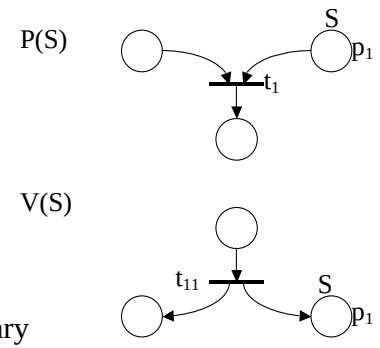
Modelling task synchronisation mechanisms(3)

- Interrupt driven task
  - As an external condition on a transition
  - Interrupt handler modeled using 2 places

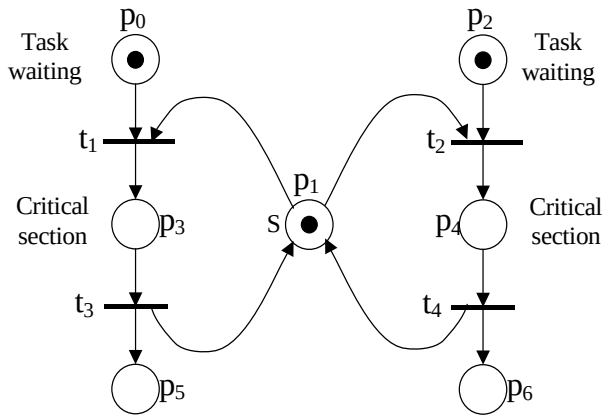


Modelling task synchronisation mechanisms (4)

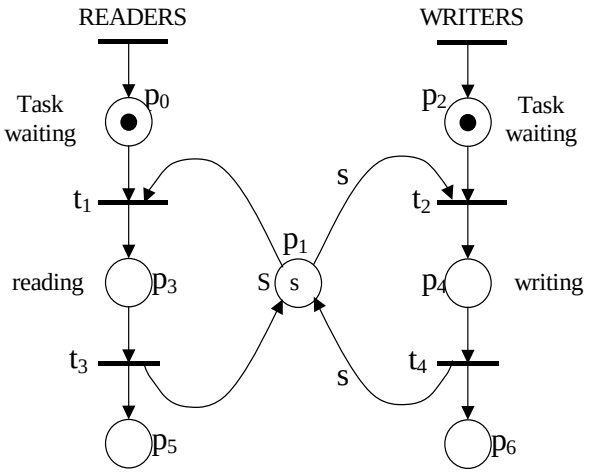
- Mutual exclusion
  - Modeled by a single token
  - P(S): AND condition on a transition
  - V(S): OR condition on a transition
  - Critical section synchronized by an auxiliary place



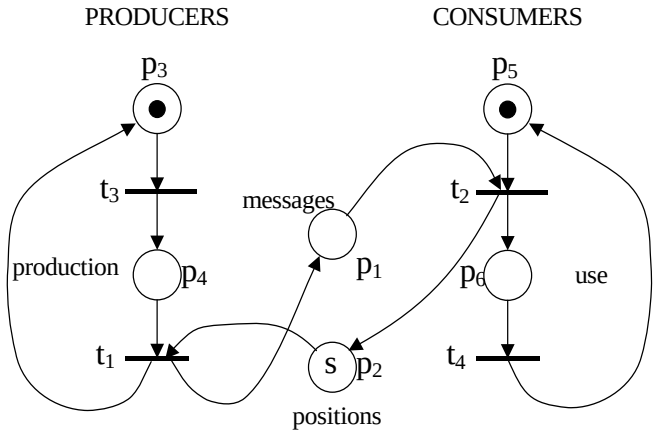
Mutual exclusion through semaphore



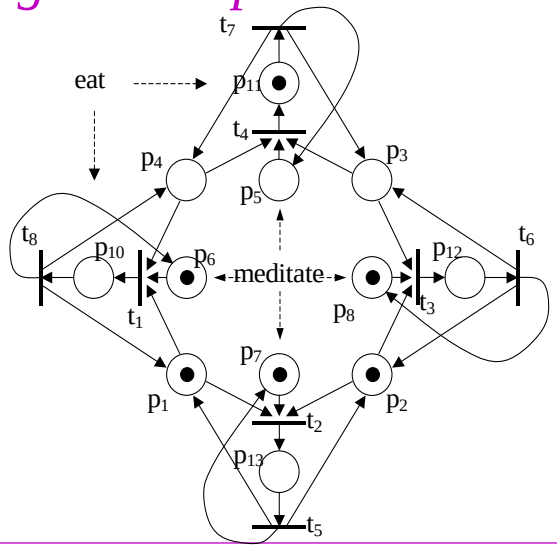
Readers – writers problem



Producers - consumers



Dining Philosophers



Exercise

- A consulate let its clients enter then the entrance door is closed and the client may be served. After being served, each client leaves the consulate through another door. The entrance door cannot be opened again before all the clients have left.
  - Design a PN with inhibitor arcs that represents this specification
  - Modify the first solution so that only one client may enter at a time
  - Describe the same case using an ordinary PN
  - A maximum of 4 clients may enter before the door is closed. Model this case using an ordinary PN

Design and validation

- Petri nets properties
- Design by successive refinements
- Analysis by reduction

Petri nets properties

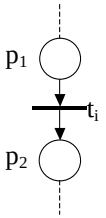
- Structural properties
  - Are only dependent on the topology
- Behavioral properties
  - Depend on the initial marking

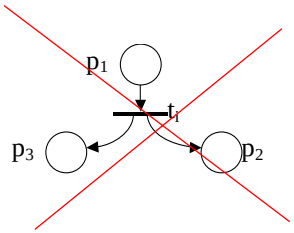
Structural properties

- State graph
- Event graph
- Without conflict
- Free choice
- Extended free choice
- simple
- pure
- Without loop

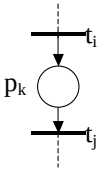
Structural properties (2)

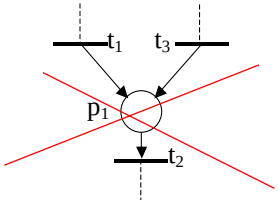
- State graph
  - iif each transition has exactly one input and one output place





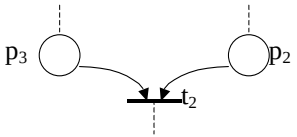
- Event graph
  - iif each place has exactly one input and one output transition

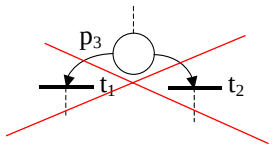




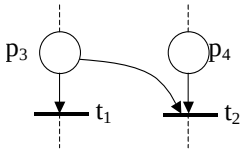
Structural properties (3)

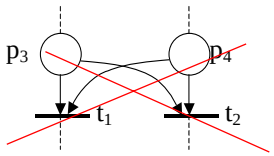
- Without conflit
  - Each place has at most one output transition





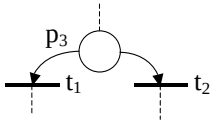
- simple
  - Each transition is involved in at most one conflict

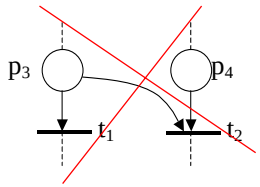




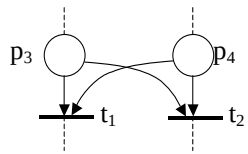
Structural properties (4)

- Free choice
  - In case of conflict, $< p_j, \{t_1, t_2, ..\}>$ none of the transitions has another input place than p_j



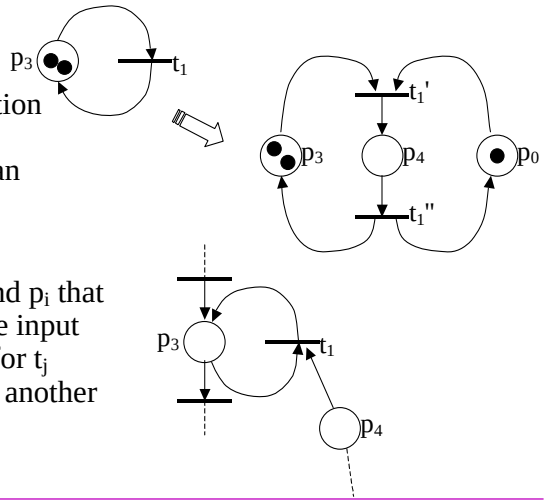


- Extended free choice
  - In case of conflict, all the involved transitions have the same input places



Structural properties (5)

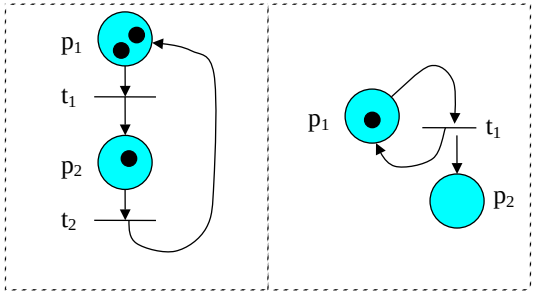
- Pure PN
  - There is no transition that has an input place that is also an output place
- Without loop
  - If there exists t_j and p_i that is at the same time input and output place for t_j then t_j has at least another input place



Exercise - properties

For each PN below, respond to the following questions:

1. Is it a state graph ?
2. Is it an event graph ?
3. Is it without conflict ?
4. It is simple ?
5. Is it pure ?
6. Is it without loop ?



Behavioral properties

- Reachability
- Boundedness
- Conservativeness
- Liveness
- Reversibility, home state
- Coverability
- Persistence
- Synchronic distance
- Fairness

Reachability

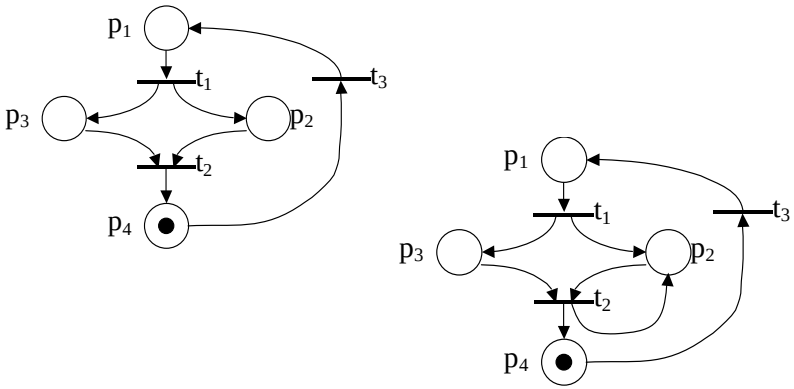
- Can the system reach a given state or exhibit a given behavior (correct or not) ?
  - To answer this question, one has to find a sequence of firing that brings from M_0 to M_i
  - A marking M_i is said reachable from M_0, if there exists a sequence of firings that transforms M_0 in M_i
  - M_i is said immediately reachable from M_0, if there is a unique firing that transforms M_0 en M_i
  - $R(M_0)$: set of all reachable markings from M_0
  - $L(M_0)$: set of all firing sequences from M_0

Bounded and safe

- Places are often used to represent:
  - Information storage in communication systems and computers
  - Products and tools in production systems
- The boundedness property is useful to detect overflows in resource usage
- A PN is said k-bounded if the number of tokens whichever the place does not exceed k for all markings in $R(M_0)$
- A PN is said safe if k=1 (1-bounded)

csem

Bounded and safe (2)



csem

Exercise

- 2 tasks T1 and T2 execute on the same processeur in time sharing (part of T1 is executed, then part of T2 is, then part of T1, …)
- Model the same behavior with 4 tasks
- Is the resulting PN bounded ?

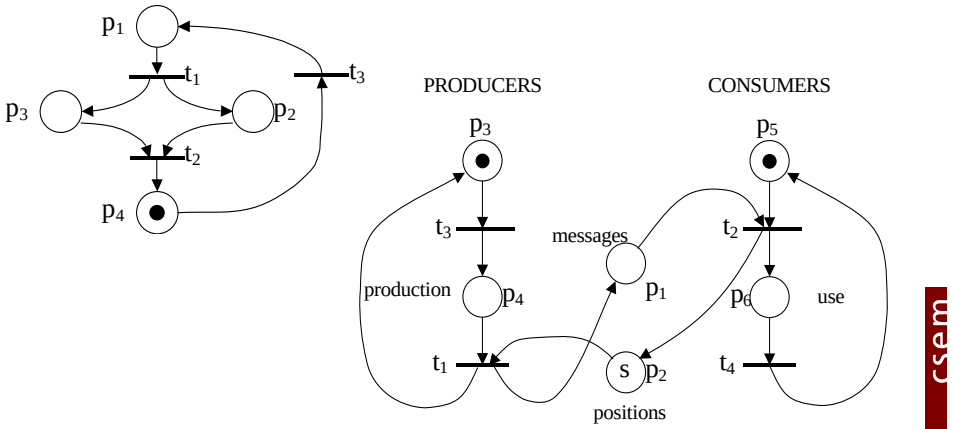
csem

Conservativeness

- In a real system, the number of resources is limited. If tokens represent resources, it may be a desired properties that the number of tokens remains constant whichever the marking in $R(M_0)$
  - A PN is said strictly conservative if the number of tokens is constant
  - A PN is said conservative if there exists a weighting vector $w=[w_1,....., w_i, ..., w_N]$ such that for each marking in $R(M_0)$, $\sum w_i\, m(p_i)$ remains constant.

csem

Conservativeness - example



Liveness

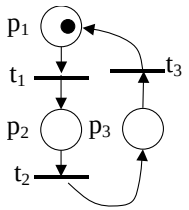
- The idea of liveness is closely related to blocking and interblocking cases
- Necessary conditions (all 4 together)
  - Limited access
    - Resources can only be shared by a finite number of tasks
  - No preemption
    - Once allocated, a resource cannot be withdrawn from the task
  - Multiple requests
    - Tasks already own resources when they additional resource requests
  - The request graph has circular paths

Liveness (2)

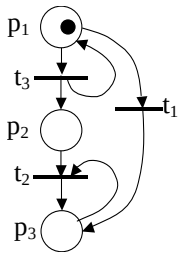
- A transition t of a PN is said :
  - L0-live (dead) if t does not belong to any sequence of $L(M_0)$
  - L1- live (potentialy firable) if t may fired at least once in a sequence of $L(M_0)$
  - L2- live if t may be fired k times (k integer >1) in a sequence of $L(M_0)$
  - L3- live if t may be fired an infinite number of times in a sequence of $L(M_0)$
  - L4- live (live) if t is L1- live for any marking in $R(M_0)$

Liveness - example

- live



- t_2, t_3 are L1, L2 et L3 live
- t_1 is L1 live



Exercise

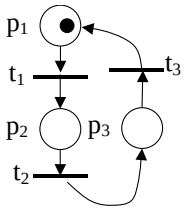
- 2 computers share a common memory. It is assumed that each computer may be in one of the following states:
  - Does not need the shared memory
  - Needs the shared memory, but it has not yet been assigned
  - Uses the shared memory
- Your assignment
  - Model this behavior
  - Is the result conservative
  - Show that the resulting PN is live

Reversibility and home state

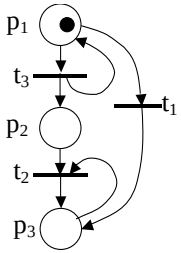
- For a real-time system, it is of prime importance to recover in case of error and then return to a correct state
  - A PN is said reversible if the initial marking M_0 may be reached from any marking in $R(M_0)$
  - A given state M_i of a PN is said « home state » if for any marking M of $R(M_0)$, M_i can be reached from M

Reversibility and home state - example

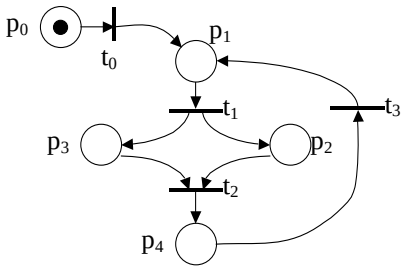
- reversible



- Not reversible

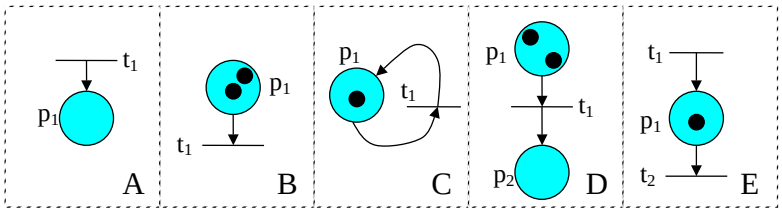


- Home state



Exercise - properties

For each PN below, please indicate if it is:
1. bounded ? 2. live ? 3. Without blocking? 4. conservative ?



PN Properties

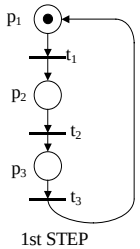
- Reachable marking
  - A marking that can reached from M_0
- K-bounded (if K=1, safe)
  - Maximum number of tokens ∀place =K
- reversible
  - M_0 may be reached from any marking
- conservative (same weights, strictly conservative)
  - Weighted sum of tokens in places = Constant

PN Properties (2)

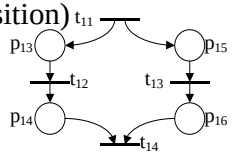
- live
  - All transitions are live
- Well shaped
  - live + bounded + reversible

Design by refinement

- First step
  - Simple net
  - Complex tasks associated to transitions
  - validation
- Second step
  - Transitions are replace by a block (a PN that begins with an initial transition and ends with a final transition)
  - validation
- Following steps
  - Repeat step 2

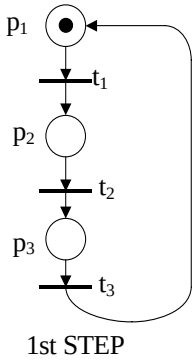


1st STEP

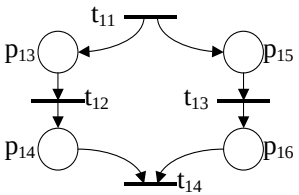


BLOCK

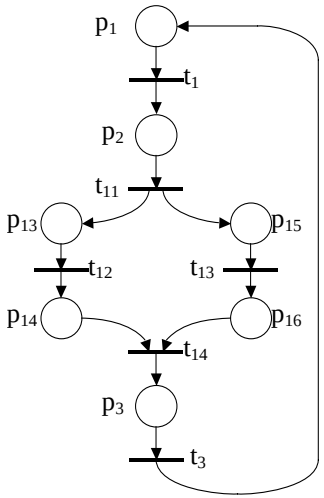
Design by refinement (2)



1st STEP

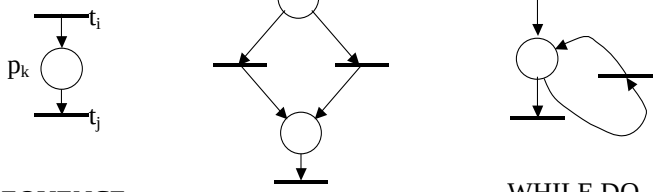


BLOCK



Structure approach

- We only use blocks that have good properties



SEQUENCE

IF THEN ELSE

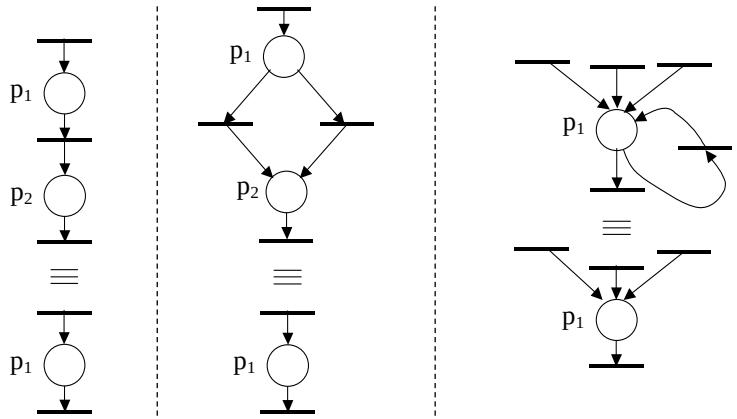
WHILE DO

Analysis by reduction

- Sequence simplification
  - Eliminate sequences of places
- Concatenate parallel paths
- Suppress identity transitions

ORIGINAL PROPERTIES MUST BE PRESERVED

Analysis by reduction (2)



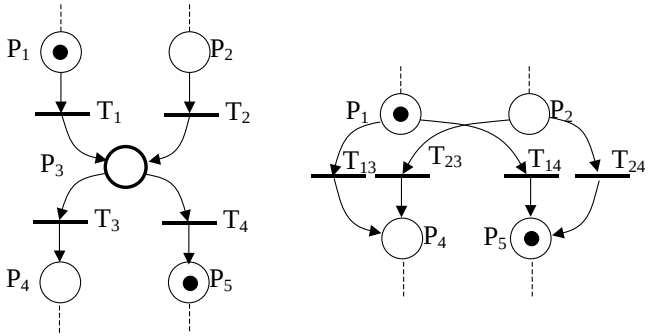
Reduction techniques

- Preserving boundedness and liveness properties
  - R1: place substitution
  - R2: implicit place
  - R3: neutral transition
  - R4: identical transitions
- Preserving invariants
  - Ra: non pure transition for ordinary PN
  - Rb: pure transition for ordinary PN

Reduction R1: place substitution

- Place Pi can be removed if it complies with the 3 following conditions
  - Pi output transitions have no other input place than Pi
  - There is no transition Tj that is at the same time input and output transition of Pi
  - At least one output transition of Pi is not a sink transition

Reduction R1- example



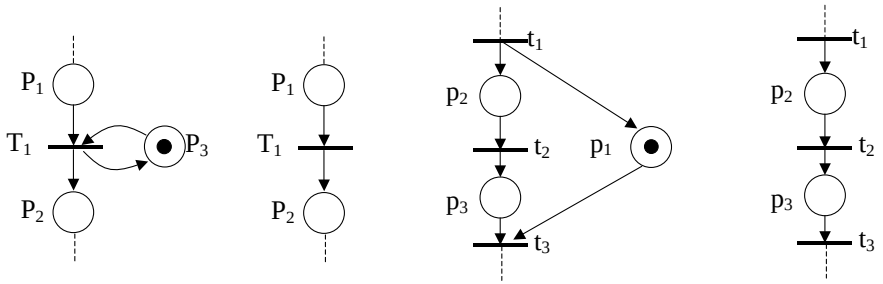
Keeps: bounded, safe, live, without blocking, home state, conservative. However, bound and home state are not always known

Reduction R2: implicit place

- Place Pi is implicit if it fulfils the following 2 conditions
  - Marking of this place may never block the firing of its output transitions
  - Its marking may be deducted from the marking of other places according to

$$M(P_i) = \sum_{k \neq i} \alpha_k M(P_k) + \beta$$

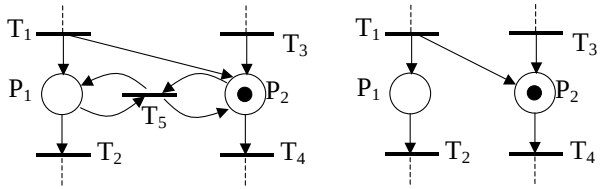
Reduction R2- example



Keeps: bound, live, without blocking, home state, conservative. May be safe after reduction even if original is not. It is not always possible to know the home state and the bound.

Reduction R3: neutral transition

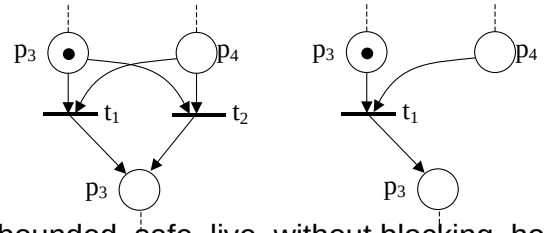
- Transition T is neutral iif the set of all its input places is identical to the set of all its output places



Keeps: bounded, safe, live, without blocking, home state, conservative. It is not always possible to know the home state and the bound.

Reduction R4 – identical transitions

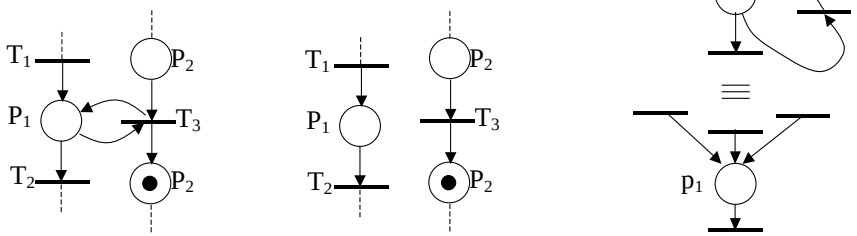
- 2 transtions are identical if they have the same set of input places and the same set of output places



Keeps: bounded, safe, live, without blocking, home state, conservative. It is not always possible to find the home state and the bound.

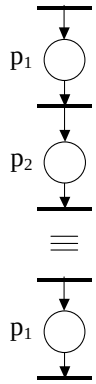
Reduction Ra – non pure transition

- Transition Tj with place Pi and arcs Tj->Pi and Pi->Tj
- Reduction
  - Suppress arcs Tj->Pi and Pi->Tj
  - Suppress transition Tj if it is isolated

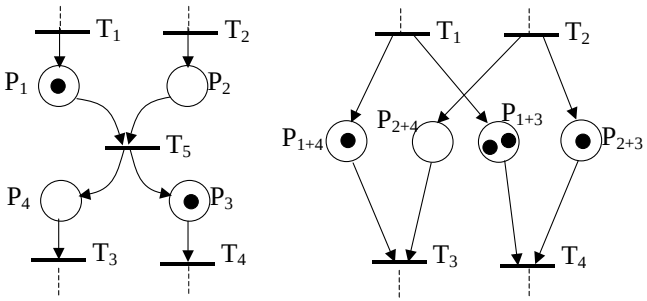


Reduction Rb – pure transition

- The transition must possess at least one input and one output place
- Reduction
  - Suppress transition
  - Each couple of places Pi,Pk such that Pi is an input place and Pk is an output place, is replaced by a place Pi+Pk (union of places)

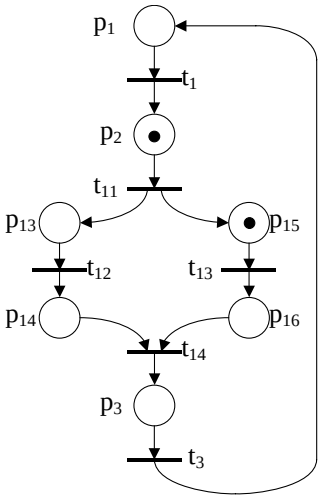


Reduction Rb- example



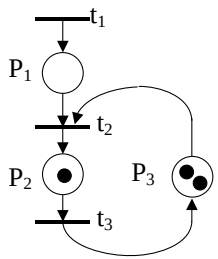
Exercise

- Reduce the following PN



Exercise

- Reduce the PN below and find its place invariants

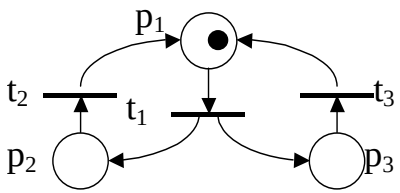


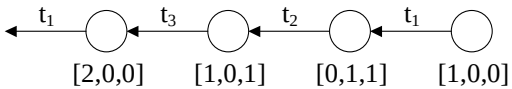
PN Formal analysis

- Systematic evolution from the initial marking
  - Gives the set of reachable markings
    - From which the properties are derived
- 3 techniques
  - Reachability graph
  - Reachability tree
  - Matrix analysis
    - Place invariant
    - Transition invariant

Reachability graph

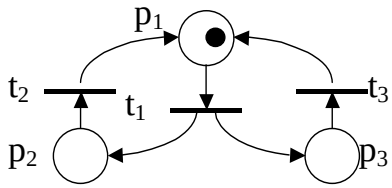
- We create a graph of reachable markings
- Exhaustive search
  - Risk of combinatorial explosion

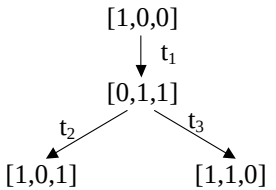




Reachability tree

- If we want to show that the PN is not bounded, it might be useless to continue
- This is the case if M_1 that has been reached from M, $M_1 \geq M$ ($\forall i: m_1(p_i) \geq m(p_i)$)
- if $M_1=M$, we are back to the same marking
- if $M_1>M$, it is possible to repeat the same firing sequence and increment again the number of tokens





Analyis by reachability tree

- The tree is always finite
- The tree permits to check the following properties
  - Not bounded if there exsists M1>M
  - Conservative if the number of tokens is constant
  - L1-Live is each transitions appears at least once in the tree
  - Reversibility and home state
- Complex to use

Matrix analysis

- Uses I and O matrices
- Look for a non trivial solution to
  $| O-I |.\underline{x}^T=0$ avec $x= | x_1, ..., x_i, ..., x_N |$ (1)
- Let M et M_1 be two successive markings
  $M_1 = M + O[t_j] – I[t_j]$ after firing t_j (2)
- Let $\rho = | \rho_1, ..., \rho_i, ..., \rho_N |$ be a solution of (1)
- Let multiply each term of (2) by ρ_i and sum up on i

$$\sum_{i=1}^{N} m_1(p_i)\rho_i = \sum_{i=1}^{N} \rho_i m(p_i) + \sum_{i=1}^{N} \rho_i \left[O(t_j, p_i) - I(t_j, p_i)\right]$$

Place invariant

- result:

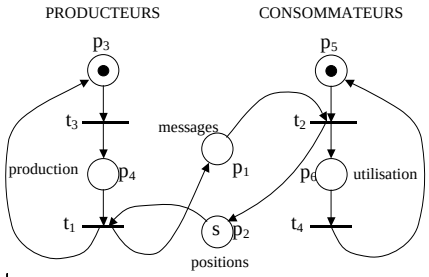
$$\sum_{i=1}^{N} m_1(p_i)\rho_i = \sum_{i=1}^{N} \rho_i m(p_i)$$

- If all ρ_i have the same sign and are non zero
  - The PN is conservative (strictly is ρ_i are all equal)
  - The PN is bounded
- If all ρ_i have the same sign and some are zero
  - A subset of the PN is conservative

Place invariant - example

$$I=\begin{vmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

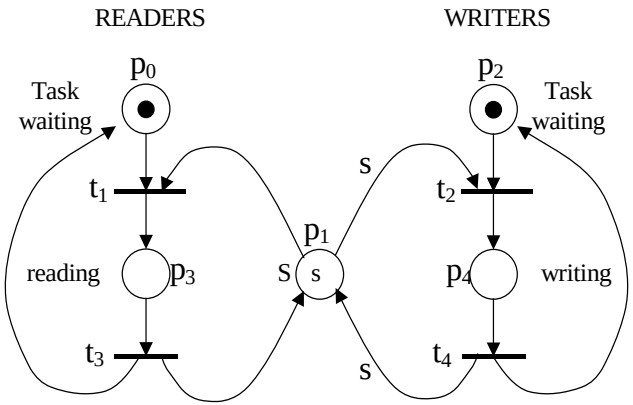
$$O=\begin{vmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$



$$|O-I|.\underline{x}^T=\begin{vmatrix} 1 & -1 & 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{vmatrix}.\underline{x}^T=0$$

- Possible solution $x_i=1$, invariant = s+2; bounded, strictly conservative

Exercise – readers-writers



Transition invariant

- Look for a non trivial solution to
  $|O-I|^T.\underline{x}^T=0$ avec $x=|x_1, ..., x_i, ..., x_M|$ (1)
- M is the number of transitions
- Gives an indication of liveness
  - Sequence of transitions that can be fired repeatedly

Interesting information

- http://www.informatik.uni-hamburg.de/TGI/PetriNets/
- http://home.arcor.de/henryk.a/petrinet/e/hpsim_e.htm

csem