UNIVERSITÀ DEGLI STUDI FIRENZE

DINFO
Dipartimento di Ingegneria dell'Informazione

Software Engineering for Embedded Systems

Advanced topics on schedulability analysis

Laura Carnevali

Software Technologies Lab
Department of Information Engineering
University of Florence
`http://stlab.dinfo.unifi.it/carnevali`
laura.carnevali@unifi.it

Outline

Introduction

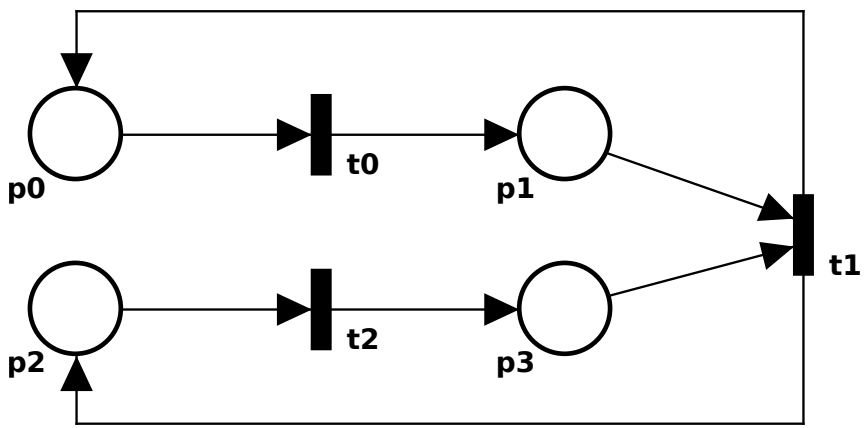
Analytical approaches vs state-space based methods

- Analytical approaches
  - Assume tasks with deterministic Worst Case Execution Time (WCET)
  - Provide exact results for task-sets made of periodic and independent tasks
  - Provide pessimistic results for task-sets including sporadic tasks and inter-task dependencies (e.g., semaphore synchronizations, data-flow precedences)
  - Provide efficient schedulability tests
- State-space based methods
  - Encompass temporal parameters varying within a min-max interval
  - Support modeling and analysis of a wide class of real-time systems (e.g., real-time task-sets, communication protocols, . . .)
  - Provide exact results for any task-set that can be modeled and analyzed
  - Require larger computational complexity than analytical approaches

Petri Nets

PN syntax (1/2)

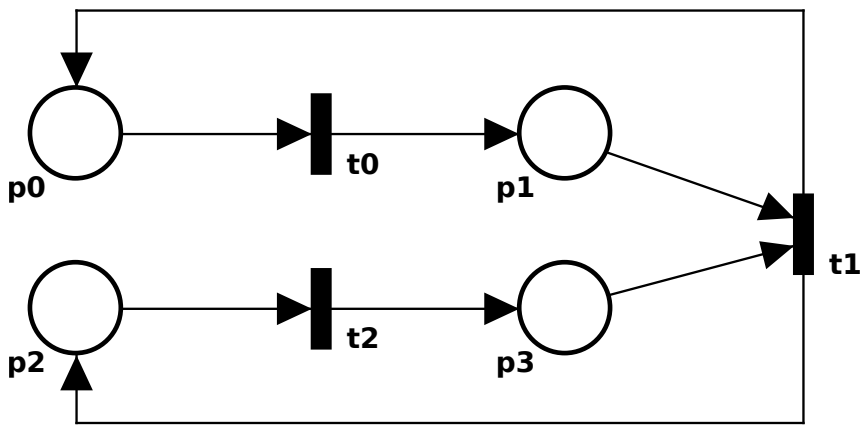
- Petri Nets (PNs) support the representation of **concurrent** systems
- A PN is a tuple $\langle P, T, A^-, A^+ \rangle$ where:
  - P is the set of **places** (circles), T is the set of **transitions** (bars), $P \cap T = \varnothing$
  - $A^- \subseteq P \times T$ is the set of **preconditions** (directed arcs from circles to bars)
  - $A^+ \subseteq T \times P$ is the set of **postconditions** (directed arcs from bars to circles)
- A PN is a directed bipartite graph where:
  - the set of vertices is $P \cup T$
  - the set of arcs is $A^- \cup A^+$



C.A. Petri, "Kommunication mit Automaten," Ph.D. thesis, Technischen Hoschule Darmstadt, 1962.

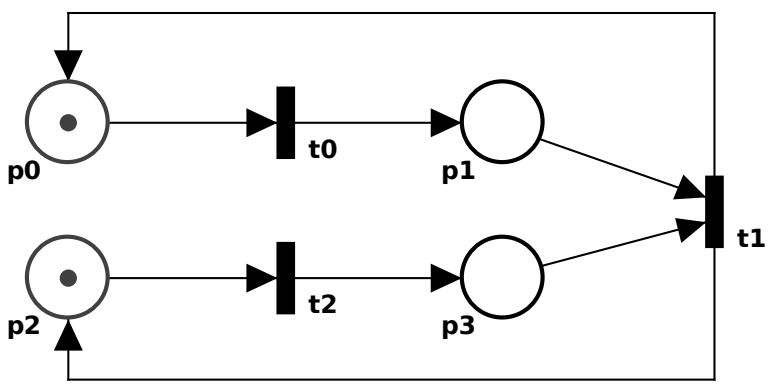
PN syntax (2/2)

- A place p is said to be an **input place** for a transition t if $(p, t) \in A^-$ (i.e., if there is a directed arc from p to t)
  - e.g., `p0` is an input place for `t0`
- A place p is said to be an **output place** for a transition t if $(t, p) \in A^+$ (i.e., if there is a directed arc from t to p)
  - e.g., `p1` is an output place for `t0`



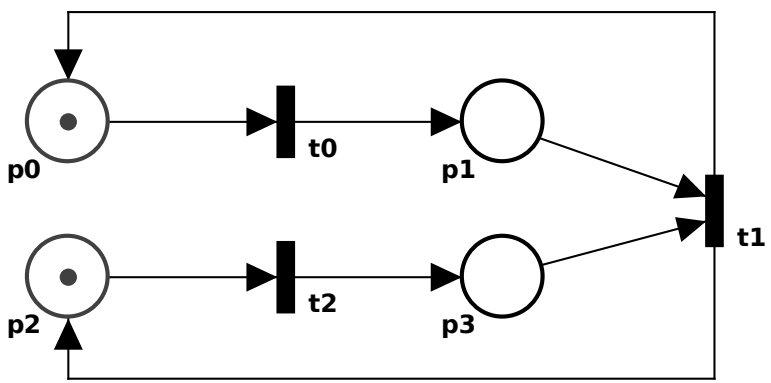
State of a PN

- A **marking** $m : P \to \mathbb{N}$ assigns a natural number of **tokens** to each place
  - Tokens do not have identity
  - e.g., `p0` and `p1` contain one token each
- A transition t is **enabled** by a marking m iff $m(p) > 0 \; \forall \, p \in P \mid (p, t) \in A^-$ (i.e., iff m assigns at least one token to each input place of t)
  - Enabled transition represent **concurrent events**
  - e.g., `t0` and `t2` are enabled, `t1` is not enabled
- The **state** of a PN is a singleton $s = \langle m \rangle$ where m is a marking
  - A transition is enabled in a state $s = \langle m \rangle$ iff it is enabled by marking m
  - An enabled transition eventually fires or is disabled
  - e.g., the current state of the example PN is $\langle \mathtt{p0}, \mathtt{p2} \rangle$



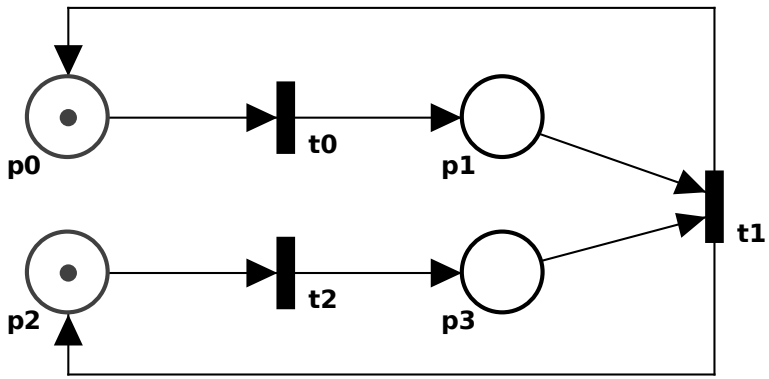
PN semantics (1/2)

- An **execution** of a PN is a path $\omega = s_0 \xrightarrow{\gamma_1} s_1 \xrightarrow{\gamma_2} s_2 \xrightarrow{\gamma_3} \ldots$ such that:
  - $s_0 = \langle m_0 \rangle$ is the initial state and m_0 is the initial marking
  - γ_i is the i-th fired transition
  - $s_i = \langle m_i \rangle$ is the state reached after the firing of γ_i
  - γ_i is selected among the transitions **enabled** in state $s_{i-1} = \langle m_{i-1} \rangle$
  - after the firing of γ_i, the new marking m_i is derived from m_{i-1} by
    1. removing a token from each input place of γ_i
       (i.e., $m_{\text{tmp}} = m_{i-1}(p) - 1 \; \forall \, p \,|\, (p, \gamma_i) \in A^-$)
    2. adding a token to each output place of γ_i
       (i.e., $m_i = m_{\text{tmp}}(p) + 1 \; \forall \, p \,|\, (\gamma_i, p) \in A^+$)
  - ω is a finite or infinite path
- E.g., $\langle \text{p0 p2} \rangle \xrightarrow{\text{t0}} \langle \text{p1 p2} \rangle \xrightarrow{\text{t2}} \langle \text{p1 p3} \rangle \xrightarrow{\text{t1}} \langle \text{p0 p2} \rangle$ is a finite execution

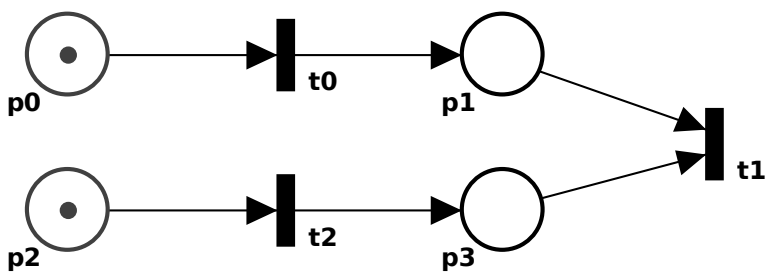


PN semantics (2/2)

- Example 1: a PN that allows infinite executions
  - e.g., $\omega = \langle \mathrm{p0\,p2} \rangle \xrightarrow{\mathrm{t0}} \langle \mathrm{p1\,p2} \rangle \xrightarrow{\mathrm{t2}} \langle \mathrm{p1\,p3} \rangle \xrightarrow{\mathrm{t1}} \langle \mathrm{p0\,p2} \rangle$ is a finite execution
  - e.g., the infinite repetition of ω is an infinite execution

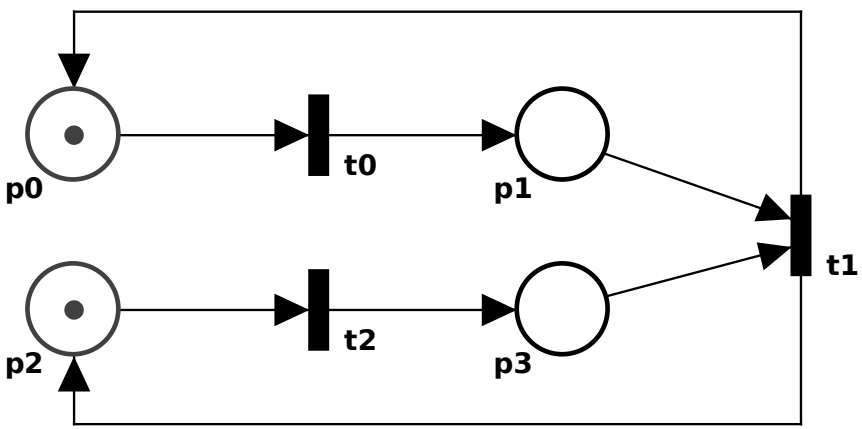


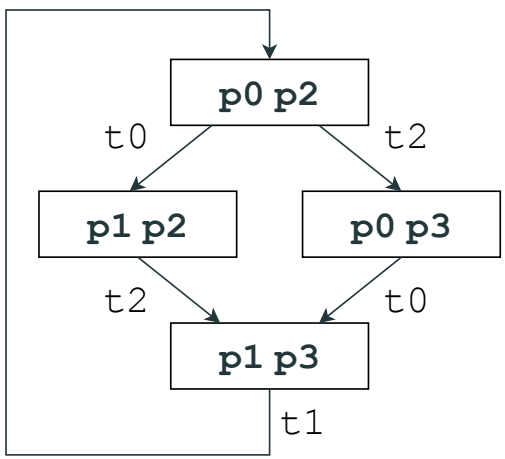
- Example 2: a PN that does not allow infinite executions
  - e.g., $\langle \mathrm{p0\,p2} \rangle \xrightarrow{\mathrm{t0}} \langle \mathrm{p1\,p2} \rangle \xrightarrow{\mathrm{t2}} \langle \mathrm{p1\,p3} \rangle \xrightarrow{\mathrm{t1}} \langle \rangle$
  - e.g., $\langle \mathrm{p0\,p2} \rangle \xrightarrow{\mathrm{t2}} \langle \mathrm{p0\,p3} \rangle \xrightarrow{\mathrm{t0}} \langle \mathrm{p1\,p3} \rangle \xrightarrow{\mathrm{t1}} \langle \rangle$



Reachability graph of a PN

- The set $\mathcal{R}(m_0)$ of markings that can be reached from the initial marking m_0 is the minimum set of markings such that:
  - $m_0 \in \mathcal{R}(m_0)$
  - if $m \in \mathcal{R}(m_0) \wedge \langle m \rangle \xrightarrow{t} \langle m' \rangle$ with $t \in T$ then $m' \in \mathcal{R}(m_0)$
- The **reachability graph** is a directed graph such that:
  - the set of vertices is $\mathcal{R}(m_0)$
  - the set of arcs includes an arc from m_i to m_j iff $\exists\, t \in T \mid \langle m_i \rangle \xrightarrow{t} \langle m_j \rangle$
- A path in the reachability graph identifies an execution of the PN





Enumeration of the reachability graph

Algorithm 1 State-space exploration from the initial marking m_0

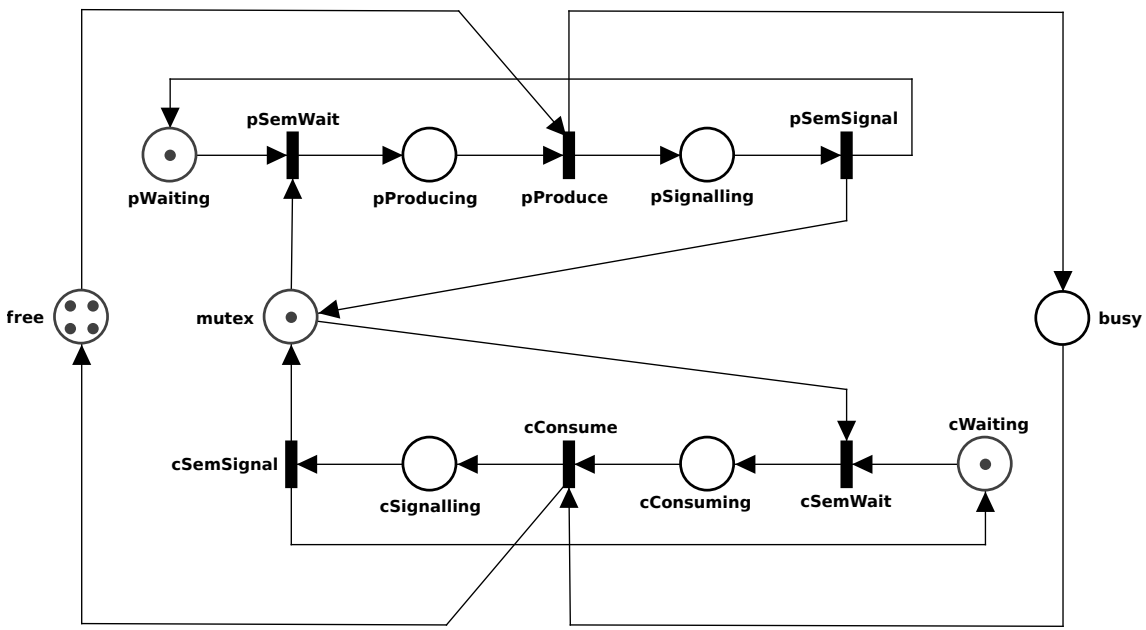
1: $Q \leftarrow \{\cdot \dot{\rightarrow} m_0\}$ // Q is the set of successions to explore
2: $G \leftarrow \varnothing$ // G is the reachability graph
3: **while** $Q \neq \varnothing$ **do**
4: select and remove a succession $m \xrightarrow{t} m'$ from Q
5: **if** IsNewSuccession$(m \xrightarrow{t} m')$ **then**
6: $G = G \cup \{m \xrightarrow{t} m'\}$
7: **for** each $t' \in$ EnabledTransitions(m') **do**
8: $m' \xrightarrow{t'} m'' =$ SuccessionEvaluator(m', t')
9: $Q = Q \cup \{(m, t, m')\}$
10: **end for**
11: **end if**
12: **end while**

The producer/consumer problem (1/4)

- Problem definition
  - A producer produces objects and puts them into a buffer
  - A consumer consumes objects and removes them from the buffer
  - The buffer may have a limited capacity
  - Production and consumption are in mutual exclusion (**safety** requirement)
  - Production and consumption will eventually occur (**liveness** requirement)
- Problem solution
  - Use a binary semaphore to guarantee mutual exclusion

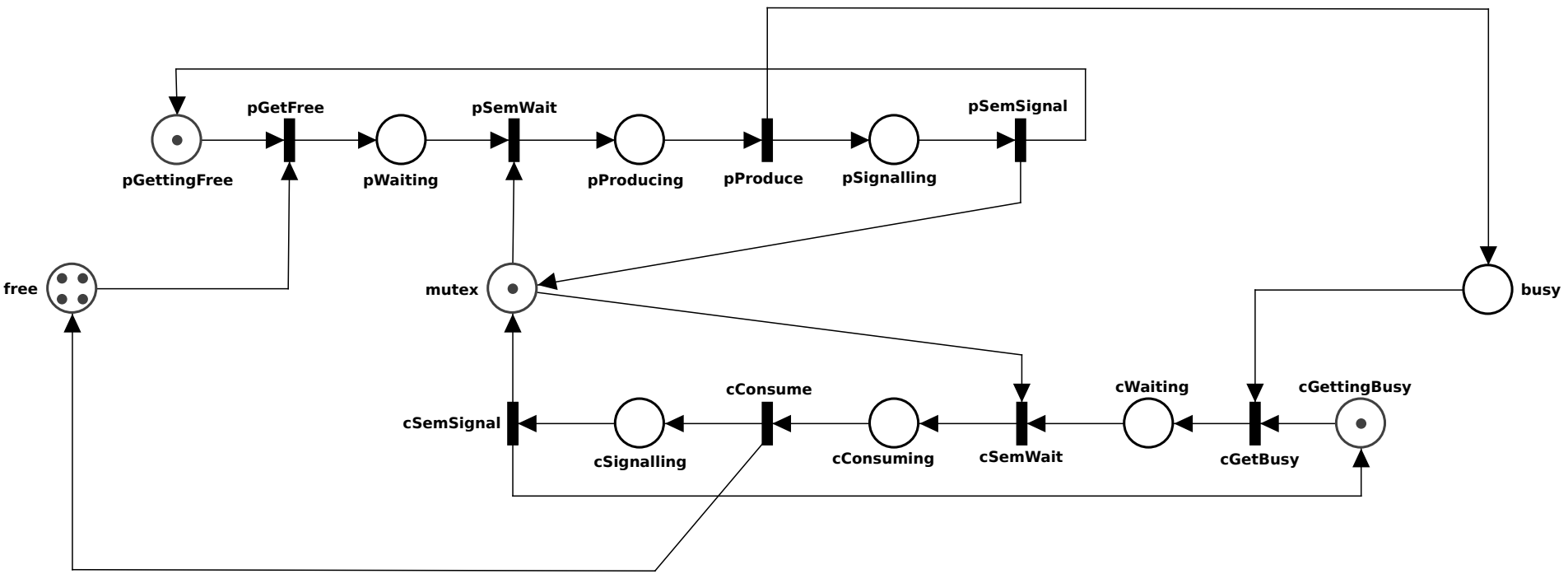
The producer/consumer problem (2/4)

- Place invariants
  - $m(\texttt{pWaiting}) + m(\texttt{pProducing}) + m(\texttt{pSignalling}) = 1 \; \forall \, m \in \mathcal{M}$
  - $m(\texttt{cWaiting}) + m(\texttt{cProducing}) + m(\texttt{cSignalling}) = 1 \; \forall \, m \in \mathcal{M}$
  - $m(\texttt{free}) + m(\texttt{busy}) = 4 \; \forall \, m \in \mathcal{M}$ (4 is the **buffer capacity**)
- The model guarantees mutual exclusion (safety requirement satisfied)
- The model is prone to **deadlock** (liveness requirement not satisfied)
  - No enabled transition in any marking $m \,|\, m(\texttt{pProducing}) = 1 \wedge m(\texttt{busy}) = 4$
  - No enabled transition in any marking $m \,|\, m(\texttt{cConsuming}) = 1 \wedge m(\texttt{free}) = 4$



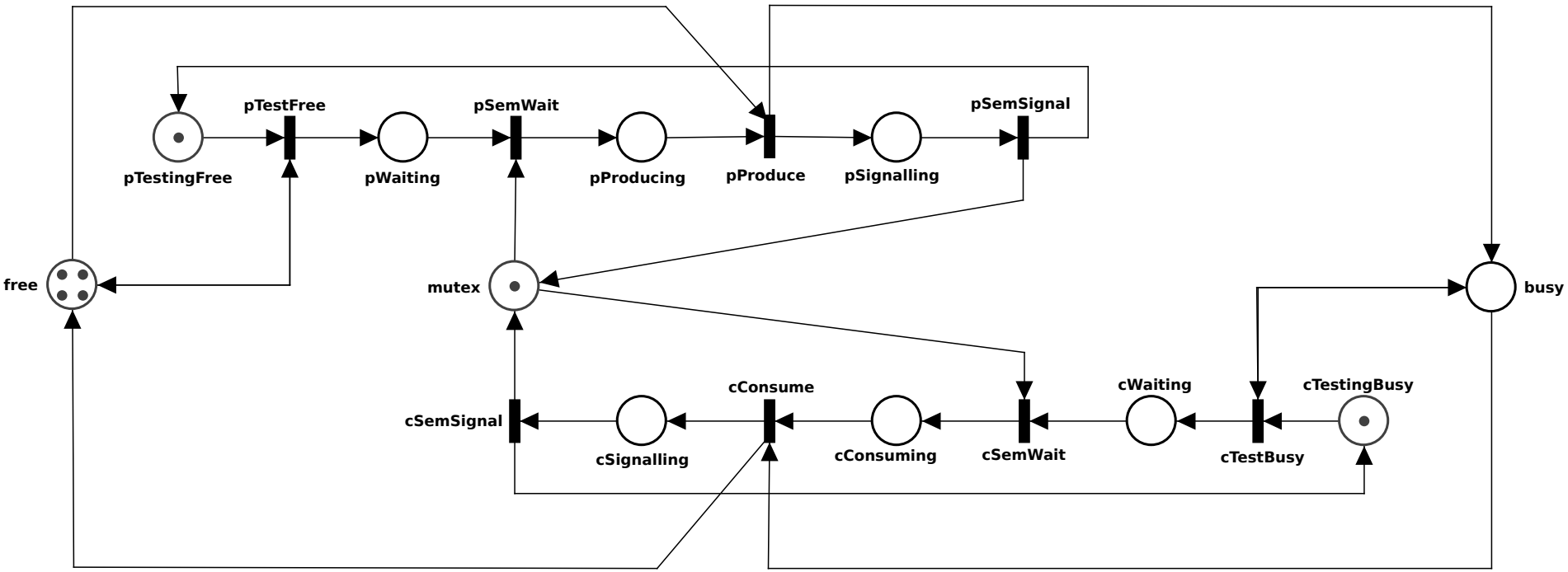
The producer/consumer problem (3/4)

- The model guarantees mutual exclusion (safety requirement satisfied)
- The model prevents deadlock (liveness requirement satisfied)
- The model cannot be translated into real-time code
  - The number of current products could be represented by a shared variable `sv`
  - There may be a context switch (due to the fact that the semaphore is busy) after the producer/consumer has read the variable and before it modifies it



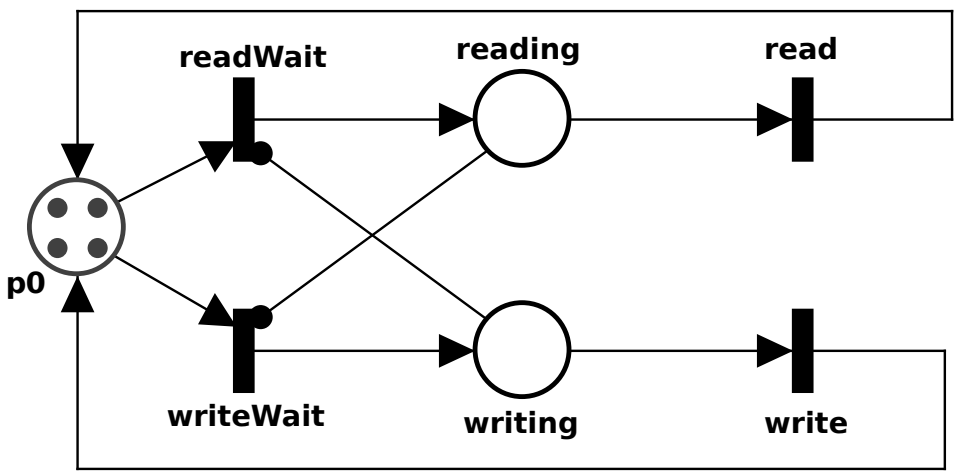
The producer/consumer problem (4/4)

- The model guarantees mutual exclusion (safety requirement satisfied)
- The model prevents deadlock (liveness requirement satisfied)
- The model can be translated into real-time code
  - The shared variable is first tested by the producer/consumer and then modified



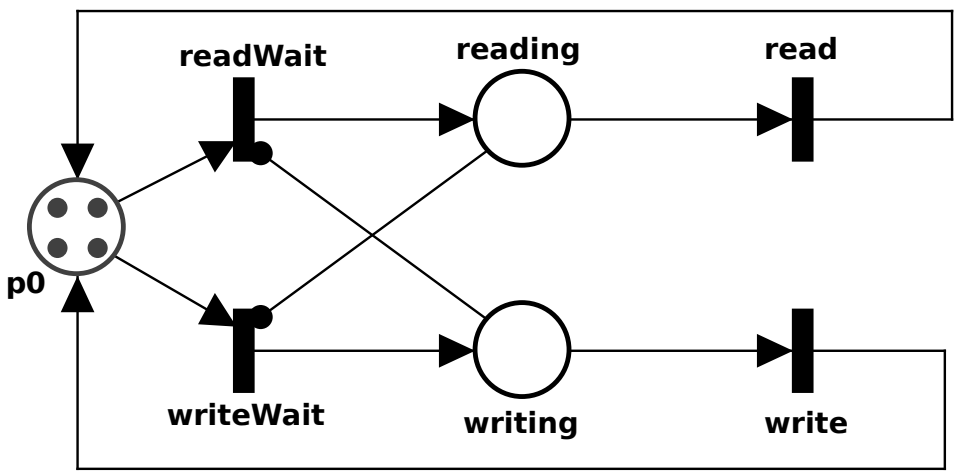
Petri nets with inhibitor arcs (1/2)

- PNs with inhibitor arcs have the Turing machine expressivity (PNs do not)
- A PN with inhibitor arcs is a tuple $\langle P, T, A^-, A^+, A^\bullet \rangle$ where:
  - P, T, A^-, A^+ are the elements of a PN
  - $A^\bullet \subseteq P \times T$ is the set of **inhibitor arcs**
    (directed bullet-headed arcs from circles to bars)
- A place p is said to be an **inhibitor place** for a transition t if $(p, t) \in A^\bullet$ (i.e., if there is a bullet-headed arc from p to t)
  - e.g., `reading` is an inhibitor place for transition `writeWait`
  - e.g., `writing` is an inhibitor place for transition `readWait`



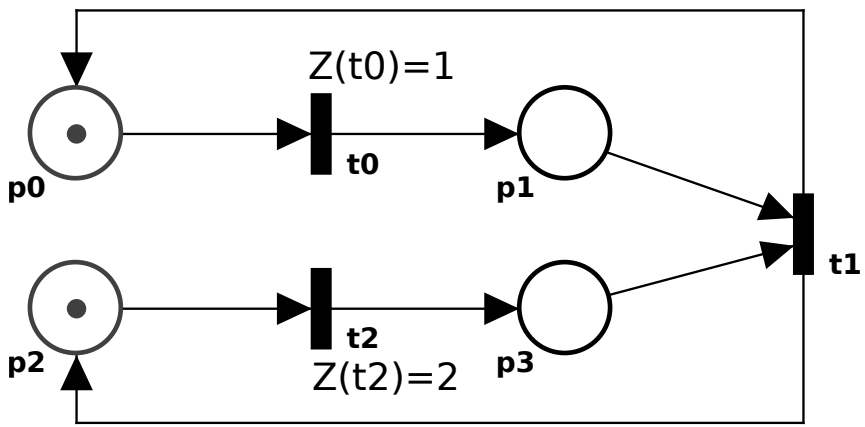
Petri nets with inhibitor arcs (2/2)

- A transition t is **enabled** by a marking m iff:
  - $m(p) > 0 \; \forall \, p \in P \,|\, (p, t) \in A^-$
    (i.e., m assigns at least one token to each input place of t)
  - $m(p) = 0 \; \forall \, p \in P \,|\, (p, t) \in A^\bullet$
    (i.e., m assigns no token to each inhibitor place of t)
- Example: read-write synchronization
  - Transition `readWait` cannot fire if place `writing` contains a token
    (i.e., a process is not allowed to read if another process is writing)
  - Transition `writeWait` cannot fire if place `reading` contains a token
    (i.e., a process is not allowed to write if another process is reading)



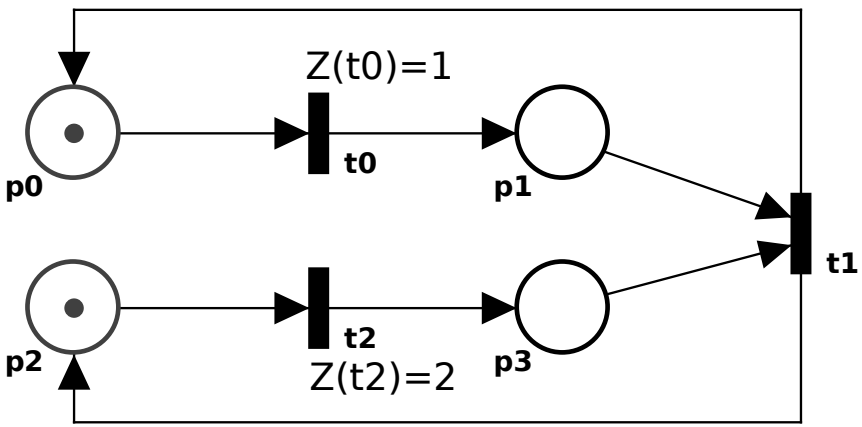
Petri nets with priorities (1/2)

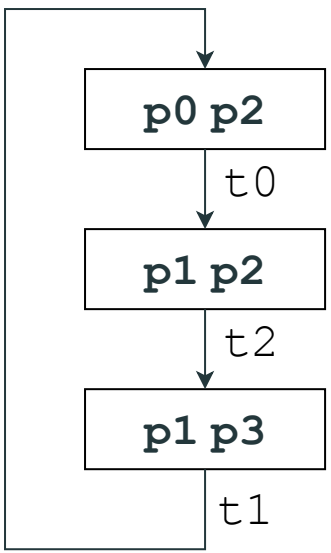
- PNs with priorities have the Turing machine expressivity (PNs do not)
- A PN with priorities is a tuple $\langle P, T, A^-, A^+, Z \rangle$ where:
  - P, T, A^-, A^+ are the elements of a PN
  - $Z : T \to \mathbb{N}$ associates each transition with a **priority** (possibly annotated next to the bar representing the transition)
  - The lowest the priority number is, the largest the priority level is
- Example: `t0` has larger priority level than `t2`



Petri nets with priorities (2/2)

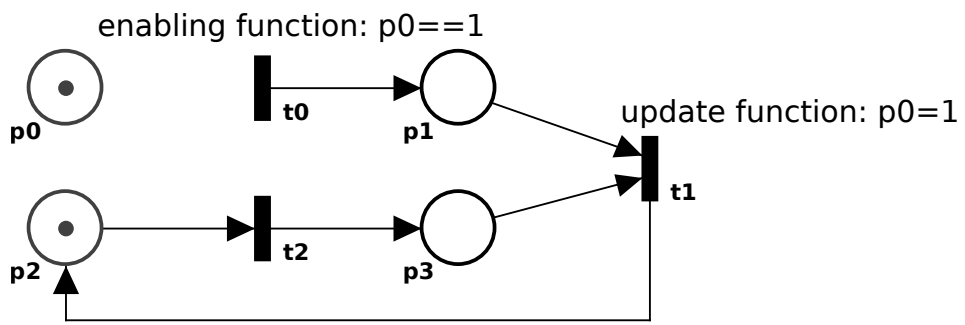
- A transition t is **enabled** by a marking m iff:
  - $m(p) > 0 \; \forall \, p \in P \mid (p, t) \in A^-$
    (i.e., m assigns at least one token to each input place of t)
  - $\forall \, t' \in T \mid t \neq t'$, if $m(p) > 0 \; \forall \, p \in P \mid (p, t') \in A^-$ then $Z(t) < Z(t')$
    (i.e., t has larger priority level than any transition with non-empty input places)
- Example: a single transition is enabled in each reachable marking





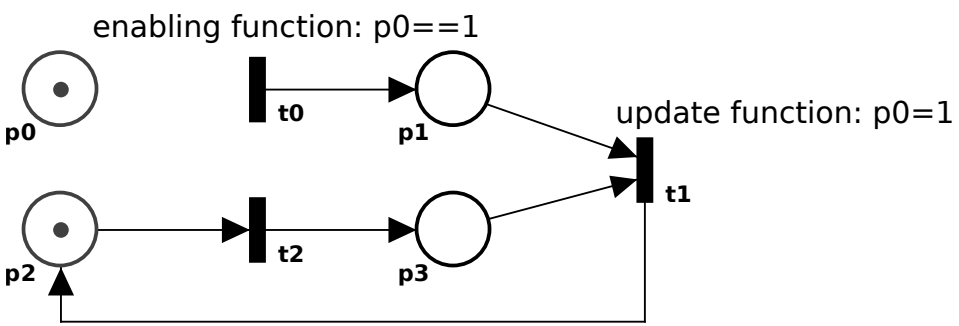
Petri nets with enabling and update functions (1/2)

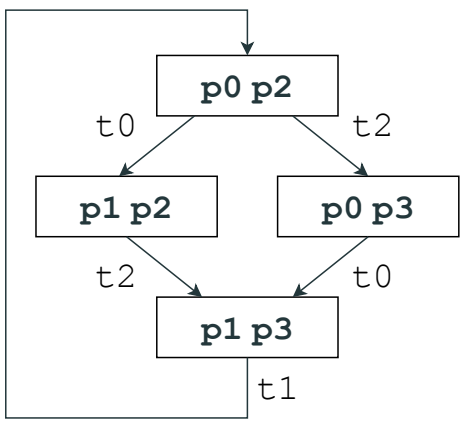
- Improve the modeling convenience wrt PNs with inhibitor arcs
- Do not increase the model expressivity wrt PNs with inhibitor arcs
- A PN with enabling and update functions is a tuple $\langle P, T, A^-, A^+, E, U \rangle$:
  - P, T, A^-, A^+ are the elements of a PN
  - E associates each transition $t \in T$ with an **enabling function** $E(t) : \mathcal{M} \to \{\text{TRUE}, \text{FALSE}\}$, with $\mathcal{M}$ being the set of reachable markings
  - U associates each transition $t \in T$ with an **update function** $U(t) : \mathcal{M} \to \mathcal{M}$
  - $E(t)$ and $U(t)$ may be annotated next to the bar representing the transition
- Example
  - `t0` has an enabling function that evaluates to TRUE if `p0` contains a token
  - `t1` has an update function that assigns to a marking m another marking m' obtained from m by assigning one token to place `p0`



Petri nets with enabling and update functions (2/2)

- A transition t is **enabled** by a marking m iff:
  - $m(p) > 0 \; \forall \, p \in P \,|\, (p, t) \in A^-$
    (i.e., m assigns at least one token to each input place of t)
  - $E(t)(m) = \text{TRUE}$ (i.e., the enabling function of t evaluates to true in m)
- Given a marking m_{i-1} and a transition γ_i enabled by m_{i-1}, marking m_i reached after the firing of γ_i is derived from m_{i-1} by
  1. removing a token from each input place of γ_i
     (i.e., $m_{\text{tmp}} = m_{i-1}(p) - 1 \; \forall \, p \,|\, (p, \gamma_i) \in A^-$)
  2. adding a token to each output place of γ_i
     (i.e., $m_{\text{tmp2}} = m_{\text{tmp}}(p) + 1 \; \forall \, p \,|\, (\gamma_i, p) \in A^+$)
  3. applying the update function of γ_i (i.e., $m_i = U(\gamma_i)(m_{\text{tmp2}})$)

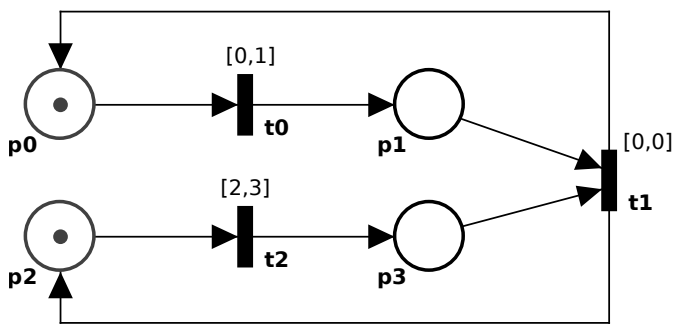




Time Petri Nets

TPN syntax

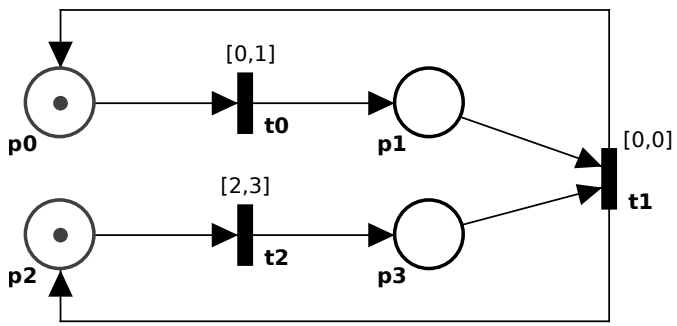
- Time Petri Nets (TPNs) extend PNs (with inhibitor arcs) with **time**
- A TPN is a tuple $\langle P, T, A^-, A^+, A^\cdot, EFT, LFT \rangle$ where:
  - P, T, A^-, A^+, $A^\cdot$ are the elements of a PN with inhibitor arcs
  - EFT associates each transition t with an **earliest firing time** $EFT(t) \in \mathbb{Q}_{\geq 0}$
  - LFT associates each transition t with a **latest firing time** $LFT(t) \in \mathbb{Q}_{\geq 0}$
  - $EFT(t) \leq LFT(t)\ \forall\, t \in T$
- Example
  - `t0` has earliest firing time equal to 0 and latest firing time equal to 1
  - `t1` has earliest firing time equal to 2 and latest firing time equal to 3
  - `t2` has earliest and latest firing time equal to 0



E. Vicario, "Static Analysis and Dynamic Steering of Time-Dependent Systems Using Time Petri Nets," IEEE Transactions on Software Engineering, 2001.

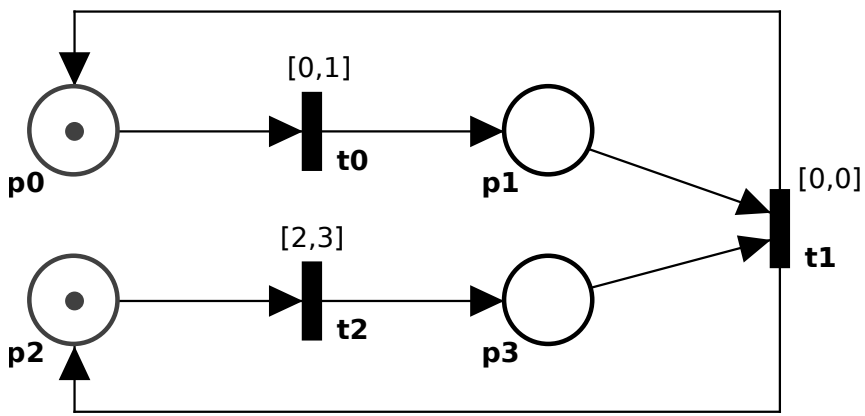
State of a TPN

- The **state** of a TPN is a pair $\langle m, \tau \rangle$ where
  - m is a marking (representing the logical location of the system)
  - τ associates each enabled transition t with a **time to fire** $\tau(t) \in \mathbb{R}_{\geq 0}$
- Example
  - `t0` can be associated with a time to fire $\tau(\texttt{t0}) \in [0, 1]$
  - `t1` can be associated with a time to fire $\tau(\texttt{t1}) \in [2, 3]$



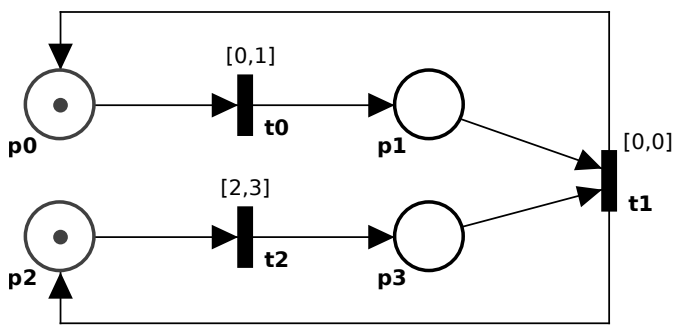
TPN semantics (1/5)

- **Strong** semantics of time constraints
  - A transition t **cannot fire before** it has been enabled with continuity for a time $T_{\min}$ not lower than its earliest firing time (i.e., $T_{\min} \geq EFT(t)$)
  - A transition t **cannot avoid to fire when** it has been enabled with continuity for a time $T_{\max}$ equal to its latest firing time (i.e., $T_{\max} = LFT(t)$, meaning that time cannot advance before t fires or is disabled by another firing)
  - Transition firings occur in zero-time
- **Weak** semantics of time constraints (not implemented by TPNs)
  - A transition that has not fired within its latest firing time could not be able to fire unless it is newly enabled again



TPN semantics (2/5)

- As in PNs with inhibitor arcs, a transition t is **enabled** by a marking m iff:
  - $m(p) > 0 \; \forall \, p \in P \mid (p, t) \in A^-$
    (i.e., m assigns at least one token to each input place of t)
  - $m(p) = 0 \; \forall \, p \in P \mid (p, t) \in A^\bullet$
    (i.e., m assigns no token to each inhibitor place of t)
- A transition t is **firable** in a state $s = \langle m, \tau \rangle$ iff:
  - t is enabled by marking m
  - $\tau(t) \leq \tau(t') \; \forall \, t' \in T \mid t'$ is enabled by m (i.e., the time to fire of t is not larger than the time to fire of any other enabled transition)
- In any possible initial state of the example TPN:
  - `t0` and `t2` are both enabled
  - `t0` is firable while `t2` is not firable

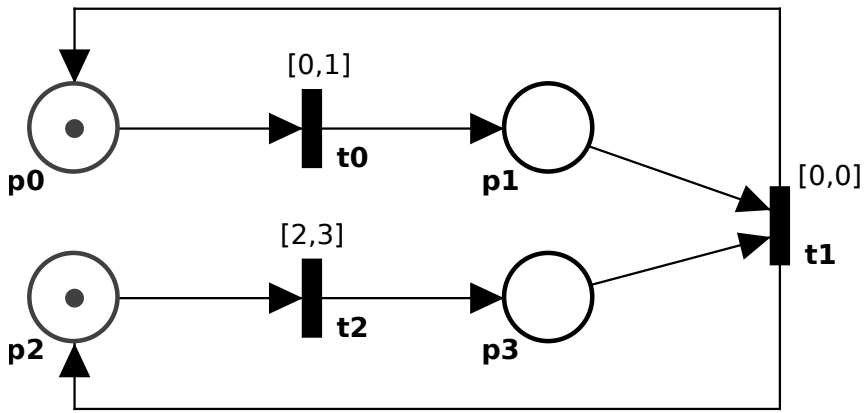


TPN semantics (3/5)

- The firing of a transition t replaces $s_0 = \langle m_0, \tau_0 \rangle$ by $s_1 = \langle m_1, \tau_1 \rangle$
  - As in PNs, m_1 is derived from m_0 by:
    1. removing a token from each input place of t
       (i.e., $m_{\text{tmp}} = m_0(p) - 1 \; \forall \, p \,|\, (p, t) \in A^-$)
    2. adding a token to each output place of t
       (i.e., $m_1 = m_{\text{tmp}}(p) + 1 \; \forall \, p \,|\, (t, p) \in A^+$)
  - A transition t' enabled by m_1 is termed:
    - **persistent** if it is not t and it is enabled both by m_0 and m_{tmp}
    - **newly enabled** if it is not enabled by m_0 or by m_{tmp},
      or if it is the fired transition t and it is enabled by m_1
      (i.e., no multiple enabledness, single server semantics)
  - A transition is termed **disabled** if it is enabled by m_0 but not by m_1
  - τ_1 is derived from τ by:
    1. reducing the time to fire of each **persistent** transition by the value of $\tau_0(t)$
       (i.e., $\tau_1(t') = \tau_0(t') - \tau_0(t) \; \forall \, t' \in T \,|\, t'$ is persistent in m_1)
    2. eliminating the time to fire of each transition **disabled** by the firing of t
    3. sampling the time to fire of each **newly enabled** transition t' in its firing
       interval (i.e., $\tau_1(t') \in [EFT(t'), LFT(t')] \; \forall \, t' \in T \,|\, t'$ is newly enabled in m_1)

TPN semantics (4/5)

- Example TPN
  - t0 is the only firable transition in any initial state with marking p0 p2
  - The firing of t0 in $\langle \mathtt{p0}\,\mathtt{p2}, \tau_0 \rangle$ yields $\langle \mathtt{p1}\,\mathtt{p2}, \tau_1 \rangle$ with $\tau_1(\mathtt{t2}) = \tau_0(\mathtt{t2}) - \tau_0(\mathtt{t0})$

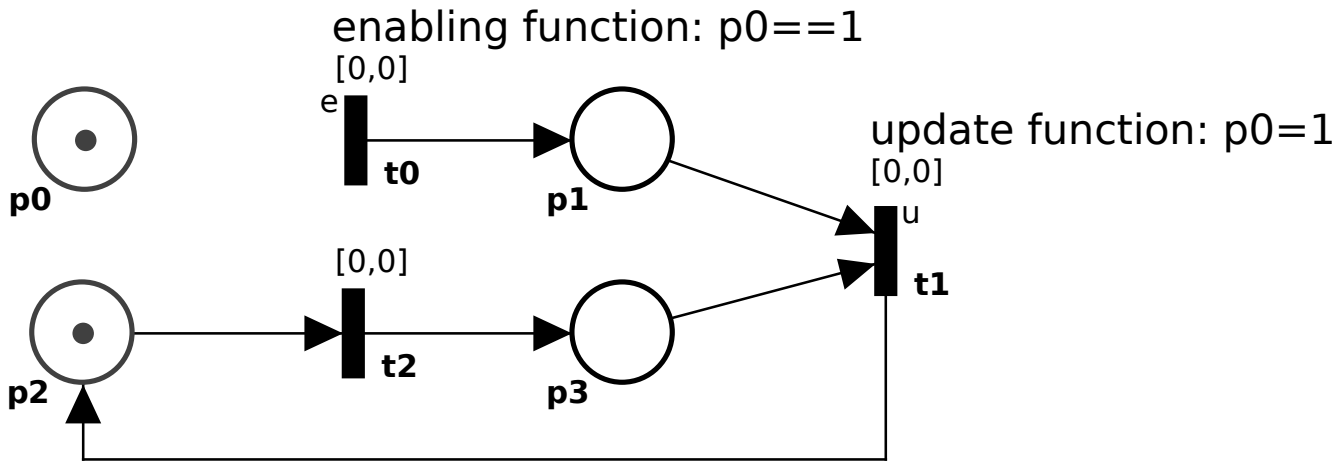


TPN semantics (5/5)

- An **execution** of a TPN is a path $\omega = s_0 \xrightarrow{\gamma_1} s_1 \xrightarrow{\gamma_2} s_2 \xrightarrow{\gamma_3} \dots$ such that:
  - $s_0 = \langle m_0, \tau_0 \rangle$ is the initial state
  - γ_i is the i-th fired transition
  - $s_i = \langle m_i, \tau_i \rangle$ is the state reached after the firing of γ_i
  - γ_i is selected among the transitions **firable** in state $s_{i-1} = \langle m_{i-1}, \tau_{i-1} \rangle$
  - after the firing of γ_i, the new marking m_i is derived from m_{i-1} by
    1. removing a token from each input place of γ_i
       (i.e., $m_{\text{tmp}} = m_{i-1}(p) - 1 \; \forall \, p \,|\, (p, \gamma_i) \in A^-$)
    2. adding a token to each output place of γ_i
       (i.e., $m_i = m_{\text{tmp}}(p) + 1 \; \forall \, p \,|\, (\gamma_i, p) \in A^+$)
  - τ_i is derived from τ_{i-1} by:
    1. reducing the time to fire of each persistent transition by the value of $\tau_{i-1}(\gamma_i)$
       (i.e., $\tau_i(t') = \tau_{i-1}(t') - \tau_{i-1}(\gamma_i) \; \forall \, t' \in T \,|\, t'$ is persistent in m_i)
    2. eliminating the time to fire of each transition disabled by the firing of γ_i
    3. sampling the time to fire of each newly enabled transition in its firing interval
       (i.e., $\tau_i(t') \in [EFT(t), LFT(t)] \; \forall \, t' \in T \,|\, t$ is newly enabled in m_i)
  - ω is a finite or infinite path

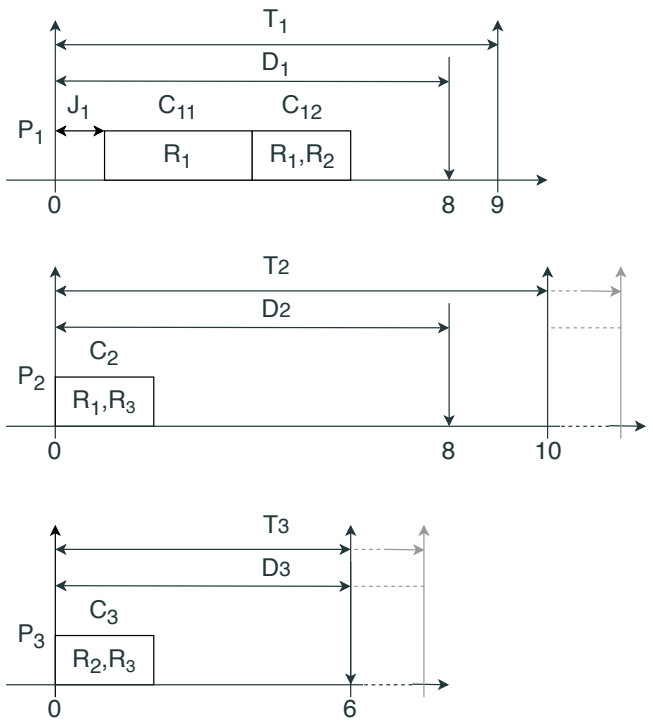
Extension of syntax and semantics of TPNs

- TPNs could be extended with enabling functions, update functions, priorities
  - These features would improve the modeling convenience wrt TPNs
  - These features would not increase the model expressivity wrt TPNs



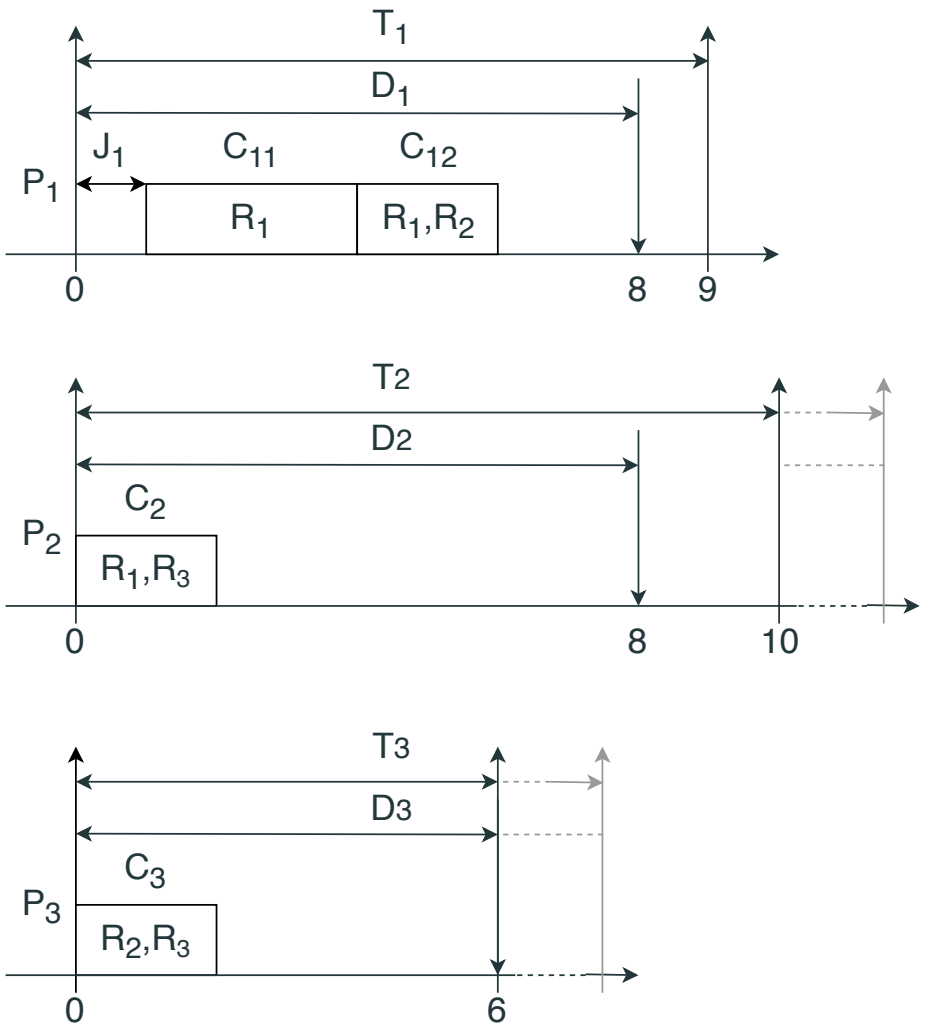
Modeling real-time task-sets with TPNs: an example (1/4)

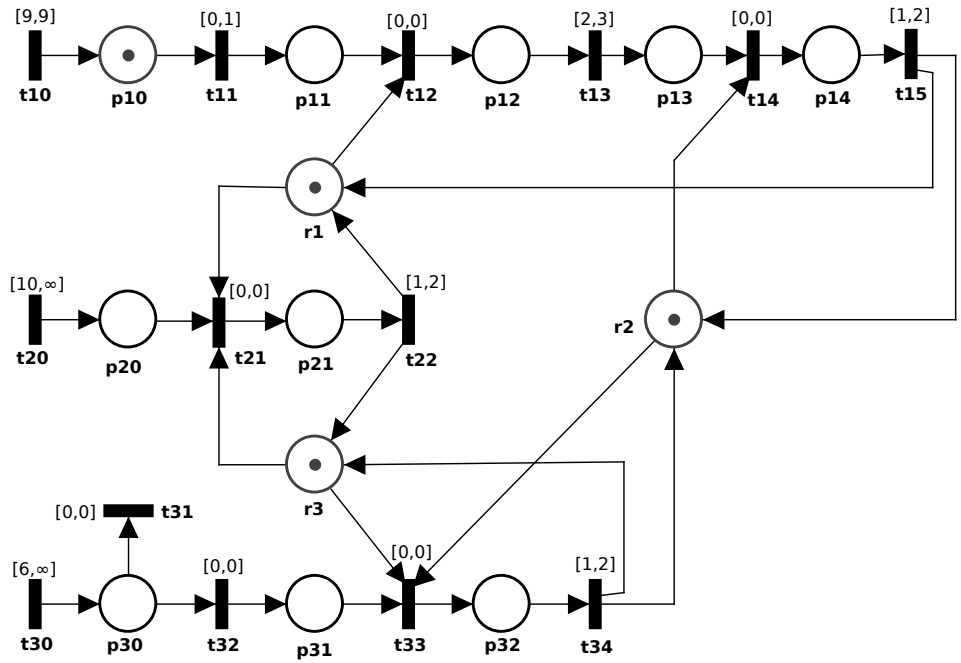
- 3 tasks P_1, P_2, P_3 contending for 3 **non-preemptable** resources R_1, R_2, R_3
  - P_1 and P_2 are hard real-time (mandatory), P_3 is soft real-time (optional)
- P_1 is periodic
  - period $T_1 = 9$
  - relative deadline $D_1 = 8$
  - absolute start time jitter $J_1 \leq 1$
  - computation 1: WCET $C_{11} \in [2, 3]$, uses R_1
  - computation 2: WCET $C_{12} \in [1, 2]$, uses R_1, R_2
- P_2 is sporadic
  - inter-arrival time $T_2 \geq 8$
  - relative deadline $D_2 = 8$
  - computation: WCET $C_{21} \in [1, 2]$, uses R_1, R_3
- P_3 is sporadic
  - inter-arrival time $T_3 \geq 6$
  - relative deadline $D_3 = 6$
  - computation: WCET $C_{31} \in [1, 2]$, uses R_2, R_3



Modeling real-time task-sets with TPNs: an example (2/4)

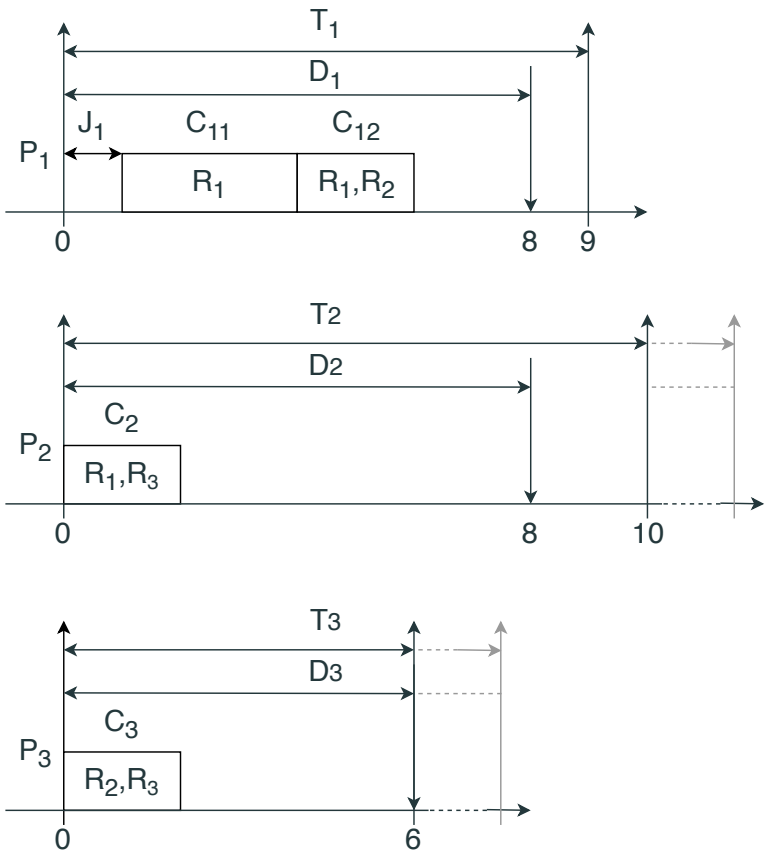
- The TPN could be automatically derived from the timeline

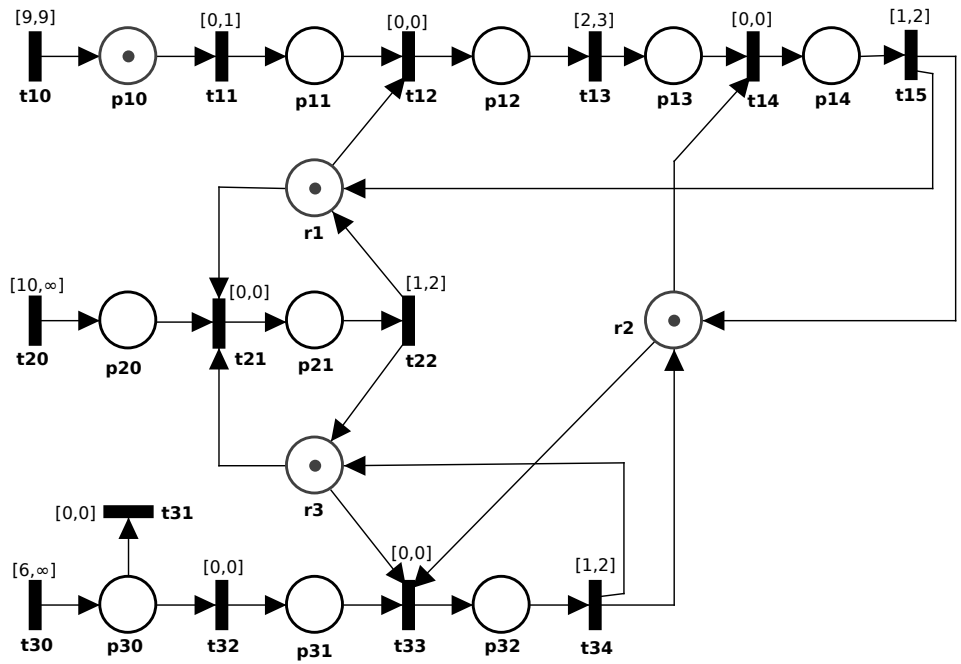




Modeling real-time task-sets with TPNs: an example (3/4)

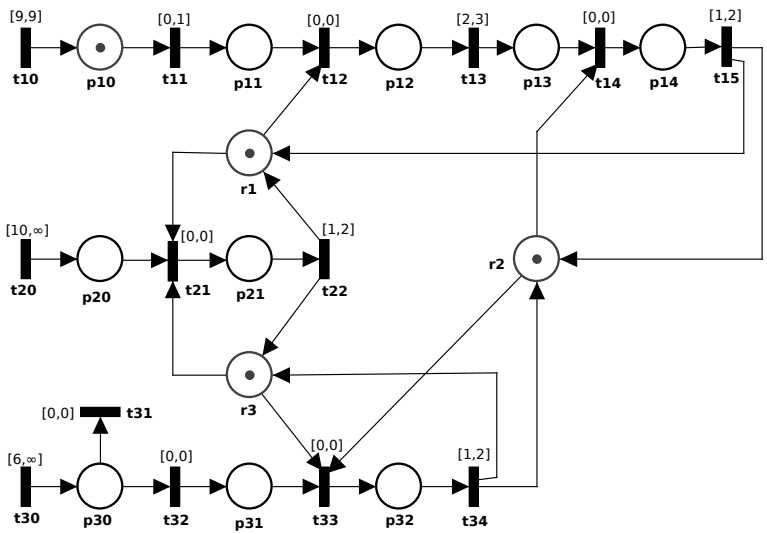
- Remarks
  - The TPN does not capture relative deadlines (which are properties to verify)
  - The model includes two types of non-determinism:
    - in the selection of the times to fire (not under the control of the designer)
    - in the acceptance rejection of the jobs of P_3 (under the control of the designer)





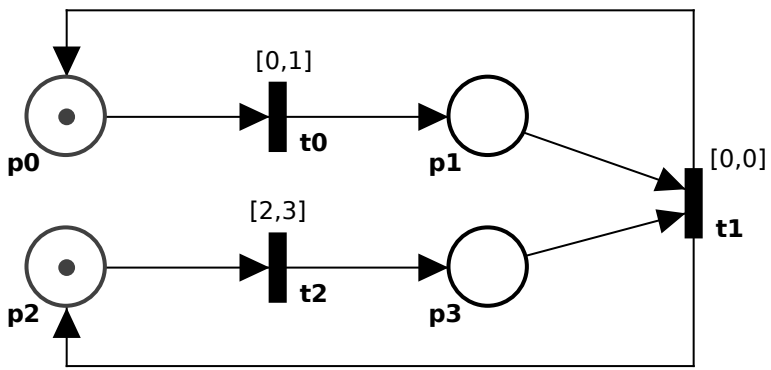
Modeling real-time task-sets with TPNs: an example (4/4)

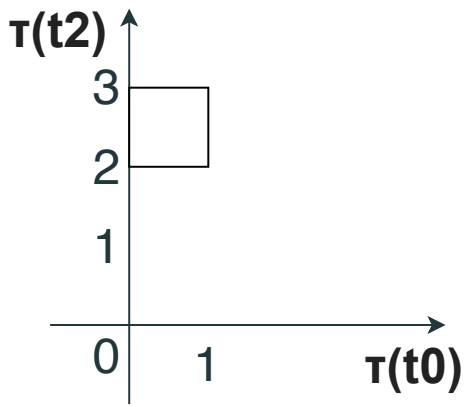
- Which questions could the analysis of the TPN model answer?
  - If P_3 jobs are always rejected, do P_1 and P_2 jobs always meet their deadline?
  - If P_3 jobs are always accepted, do P_1 and P_2 jobs always meet their deadline?
  - Is there an acceptance/rejection strategy guaranteeing deadlines of P_1 and P_2?
- Which questions could the TPN analysis answer in general?
  - **Timed reachability**: which are the feasible orderings among transitions under the constraints imposed by the temporal parameters of the model?
  - **Min/max duration**: what is the minimum and maximum time at which some firing sequence (i.e., sequence of transition firings) can be completed?



TPN analysis: preliminaries (1/3)

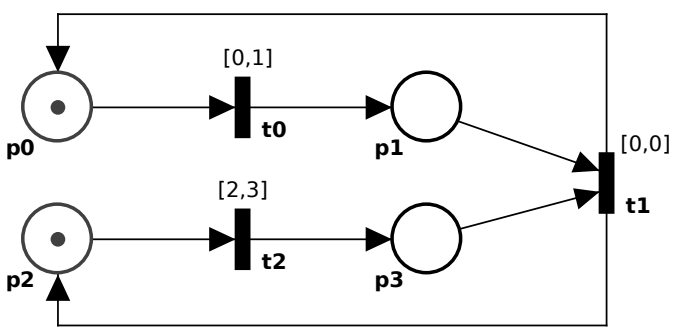
- Times-to-fire take values in dense sets ⇒
  ⇒ the reachability relation between states is not explicitly enumerable ⇒
  ⇒ the state space is covered through **equivalence classes**, each collecting a continuous variety of states (same marking, different times to fire vector)

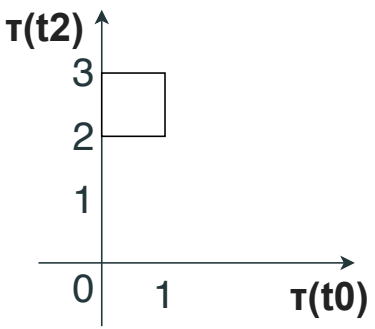




TPN analysis: preliminaries (2/3)

- A **state class** is a pair $S = \langle m, D \rangle$ where:
  - m is a marking (representing the logical location of the system)
  - D is a continuous set of values for the times to fire of the enabled transitions (D is usually termed **firing domain**)
- A state $s = \langle m_s, \vec{\tau_s} \rangle$ is included in a state class $S = \langle m, D \rangle$ if:
  - they have the same marking (i.e., $m_s = m$)
  - the times to fire vector $\vec{\tau_s}$ satisfies the constraints of D (i.e., $\vec{\tau_s} \in D$)
- The form of the constraints of D depend on:
  - the way the times to fire are advanced in the TPN semantics
  - the semantics of the reachability relation to establish on state classes
  - the form of the constraints in the initial state class



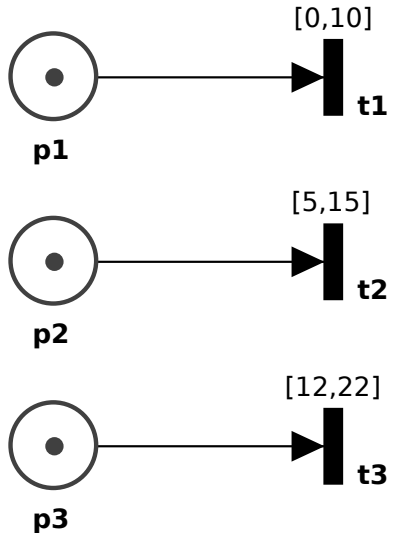


TPN analysis: preliminaries (3/3)

- A state class S' is **reachable** from class S through transition t ($S \xrightarrow{t} S'$) if and only if it contains all and only the states that are reachable from some state collected in S through some feasible firing of t, i.e., S' is reachable from S through t if and only if:
  - $\forall\, s' \in S'\ \exists\, s \in S \mid s \xrightarrow{t} s'$
  - $s \in S \wedge s \xrightarrow{t} s' \Rightarrow s' \in S'$
- Weak reachability relation termed **AE reachability**
  - Does not guarantee that every state $s' \in S'$ is reachable from every state $s \in S$
  - Guarantees that for any state $s' \in S'$ there exists at least one state $s \in S$ and a time τ such that t can be fired from s with firing time τ yielding state s'
- Strong enough to decide reachability properties under timing constraints
  - A state s' is reachable from some state in the initial class S_0 iff s' is contained in some class S which is reachable from S_0
  - A firing sequence ρ is firable iff there exists a class S' reachable from S_0 from which there exists a path reproducing the same events of ρ

TPN analysis: an intuitive introduction (1/5)

- A TPN model with 3 concurrent transitions



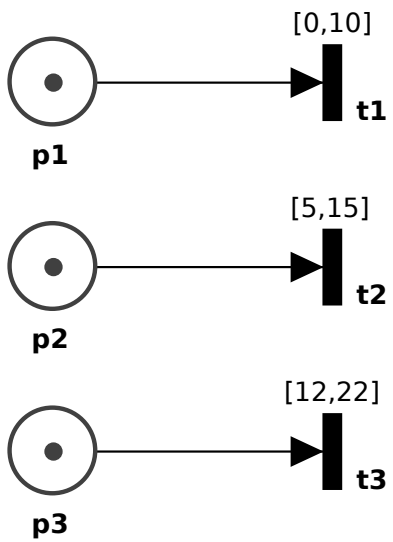
- Times to fire of the enabled transitions initially range within a hyper-rectangle (to simplify notation, the time to fire of transition t_i is denoted as $\tau_i \; \forall \, t_i \in T$)

$$D = \begin{cases} 0 \leq \tau_1 \leq 10 \\ 5 \leq \tau_2 \leq 15 \\ 12 \leq \tau_3 \leq 22 \end{cases}$$

TPN analysis: an intuitive introduction (2/5)

- The assumption that t_2 fires first restricts the firing domain:

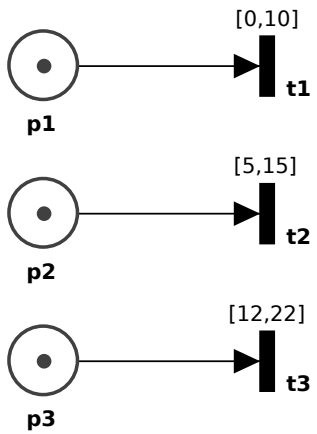
$$D_{t_2} = \begin{cases} D \\ \tau_2 \leq \tau_1 \\ \tau_2 \leq \tau_3 \end{cases} = \begin{cases} 0 \leq \tau_1 \leq 10 \\ 5 \leq \tau_2 \leq 15 \\ 12 \leq \tau_3 \leq 22 \\ \tau_2 - \tau_1 \leq 0 \\ \tau_2 - \tau_3 \leq 0 \end{cases}$$



TPN analysis: an intuitive introduction (3/5)

- Both t_1 and t_3 are persistent so their times to fire are shifted by τ_2:
  - $\tau_1' = \tau_1 - \tau_2 \Rightarrow \tau_1 = \tau_1' + \tau_2$
  - $\tau_3' = \tau_3 - \tau_2 \Rightarrow \tau_3 = \tau_3' + \tau_2$
- By replacing variables, D_{t_2} can be equivalently written as:

$$D_{t_2} = \begin{cases} 0 \leq \tau_1' + \tau_2 \leq 10 & (a) \\ 5 \leq \tau_2 \leq 15 & (b) \\ 12 \leq \tau_3' + \tau_2 \leq 22 & (c) \\ 0 \leq \tau_1' & (d) \\ 0 \leq \tau_3' & (e) \end{cases}$$



TPN analysis: an intuitive introduction (4/5)

- τ_2 is eliminated to obtain a domain D' depending only on the times to fire of the enabled transitions, i.e., $\langle \tau'_1, \tau'_3 \rangle \in D'$ iff $\exists\, \tau_2$ s.t. $\langle \tau'_1, \tau_2, \tau'_3 \rangle \in D_{t_2}$:

$$D'_{t_2} = \begin{cases} -15 \leq \tau'_1 \leq 5 & (a+(-b)) \\ -3 \leq \tau'_3 \leq 17 & (c+(-b)) \\ -22 \leq \tau'_1 - \tau'_3 \leq -2 & (a+(-c)) \\ 0 \leq \tau'_1 & (d) \\ 0 \leq \tau'_3 & (e) \\ \\ \mathbf{5 \leq \tau_2 \leq 15} & \mathbf{(b)} \end{cases}$$

$$D' = \begin{cases} -15 \leq \tau'_1 \leq 5 & (a+(-b)) \\ -3 \leq \tau'_3 \leq 17 & (c+(-b)) \\ -22 \leq \tau'_1 - \tau'_3 \leq -2 & (a+(-c)) \\ 0 \leq \tau'_1 & (d) \\ 0 \leq \tau'_3 & (e) \end{cases}$$

TPN analysis: an intuitive introduction (5/5)

- What is the meaning of the set D'?
  - The successor class contains all and only the state reachable from the initial class: $\forall \langle \tau_1', \tau_2' \rangle \in D' \; \exists \langle \tau_1, \tau_2, \tau_3 \rangle \in D$ such that $\tau_1' = \tau_1 - \tau_2 \wedge \tau_3' = \tau_3 - \tau_2$
  - This is the AE reachability relation!
- What about the form of the constraints of D'?
  - The firing domain D of the initial class is an hyper-rectangle
  - D' is a set of inequalities constraining the difference between times to fire
  - The form of D' is termed **Difference Bounds Matrix (DBM)**
  - Does repetitive derivation yield a general linear convex polyhedron? No!

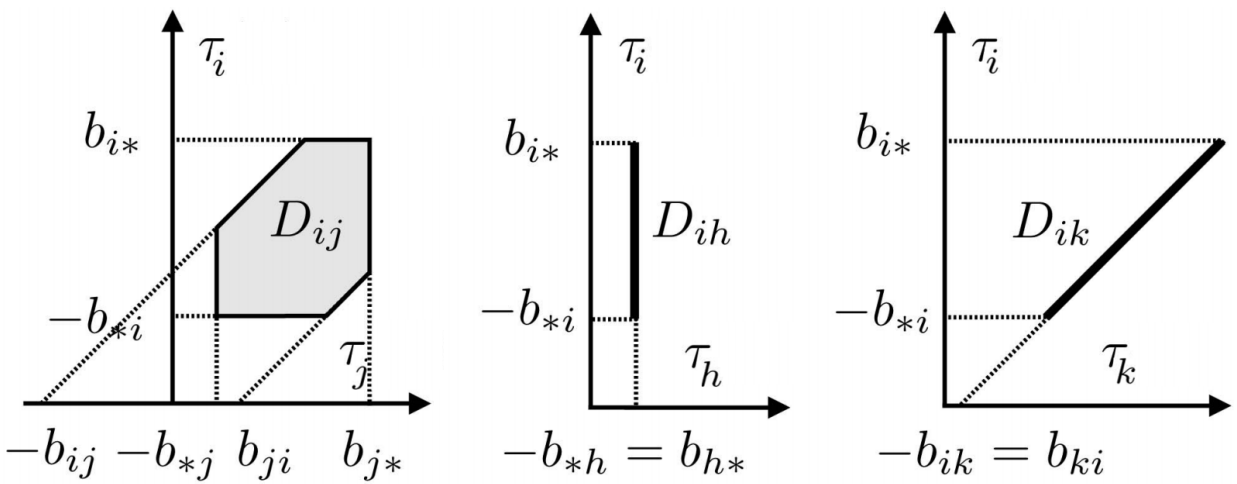
$$D = \begin{cases} 0 \leq \tau_1 \leq 10 \\ 5 \leq \tau_2 \leq 15 \\ 12 \leq \tau_3 \leq 22 \end{cases} \qquad D' = \begin{cases} -15 \leq \tau_1' \leq 5 & (a+(-b)) \\ -3 \leq \tau_3' \leq 17 & (c+(-b)) \\ -22 \leq \tau_1' - \tau_3' \leq -2 & (a+(-c)) \\ 0 \leq \tau_1' & (d) \\ 0 \leq \tau_3' & (e) \end{cases}$$

Difference Bounds Matrix: definition

- A DBM D in the space of a vector of unknown values $\boldsymbol{\tau} = \langle \tau_0, \tau_1, \ldots, \tau_N \rangle$ is a set of linear inequalities constraining the difference between elements of $\boldsymbol{\tau}$:

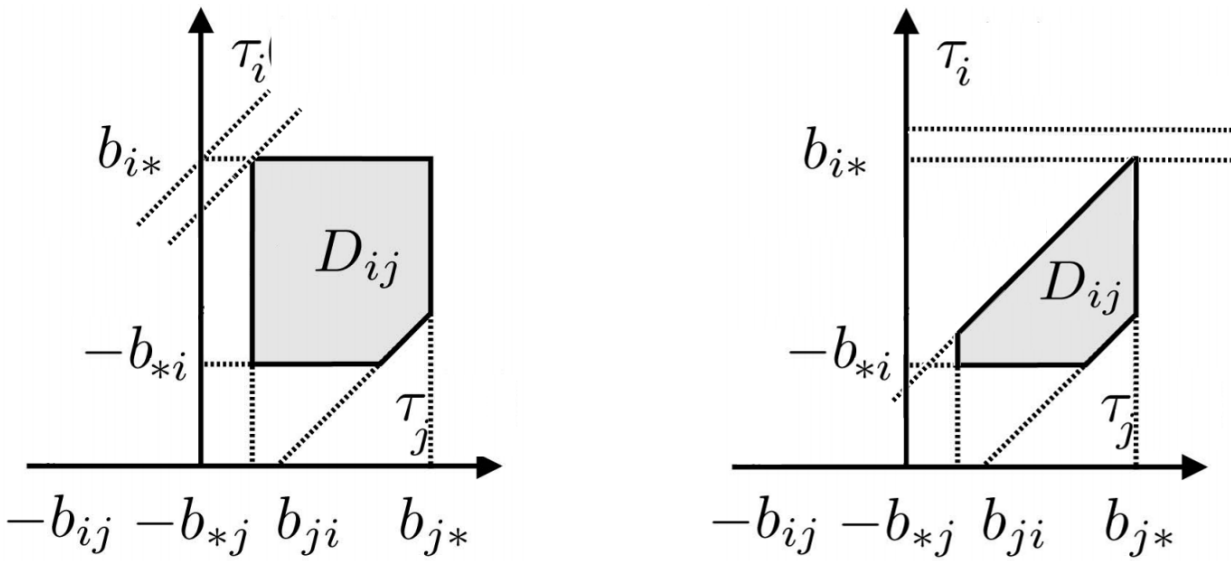
$$D = \begin{cases} \tau_i - \tau_j \leq b_{ij} & \forall\, i, j \in \{0, 1, \ldots, N-1\} \cup \{\star\} \text{ with } i \neq j,\ b_{ij} \in \mathbb{R} \cup \{\infty\} \\ \tau_\star = 0 \end{cases}$$

- $\tau_\star$ is a fictitious variable, also termed **ground**, used to:
  - cast constraints of the form $\tau_i \leq b_{i\star}$ into diagonal constraints $\tau_i - \tau_\star \leq b_{i\star}$
  - cast constraints of the form $\tau_i \geq -b_{\star i}$ into diagonal constraints $\tau_\star - \tau_i \leq b_{\star i}$
- The set of solution of a set of inequalities in DBM form is termed **DBM zone**



Difference Bounds Matrix: normal form (1/2)

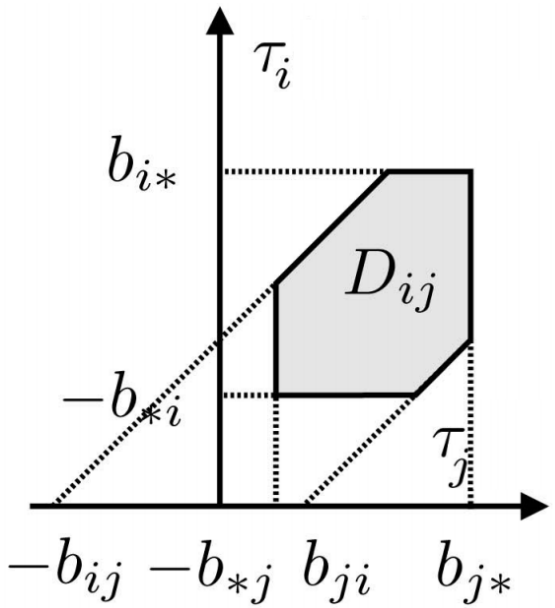
- May different sets of inequalities represent the same set of solutions?
  - Some constraints may be implied by the combination of other constraints
  - The same set of solutions can be represented by different coefficients b_{ij}



- A **normal form** is introduced where each coefficient b_{ij} is maximally tight (i.e., it cannot be reduced without changing the set of solutions)

Difference Bounds Matrix: normal form (2/2)

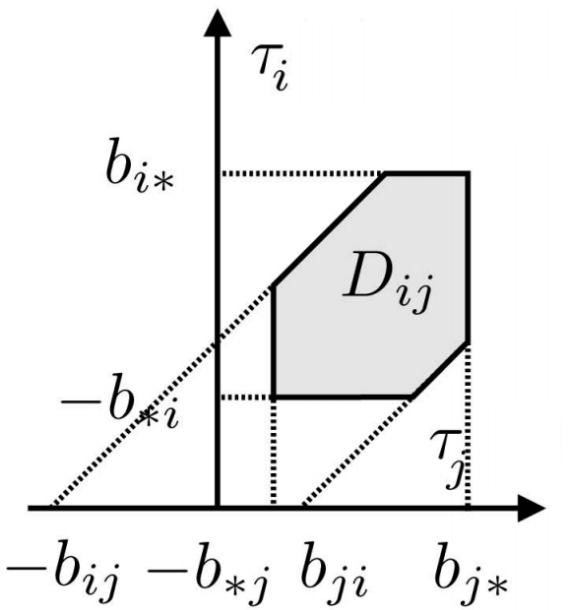
- A DBM D is in **normal form** if b_{ij} is the maximum value of $\tau_i - \tau_j$ that can be attained by any solution of the set $\forall\, i, j \in \{0, 1, \ldots, N-1\} \cup \{\star\} \mid i \neq j$

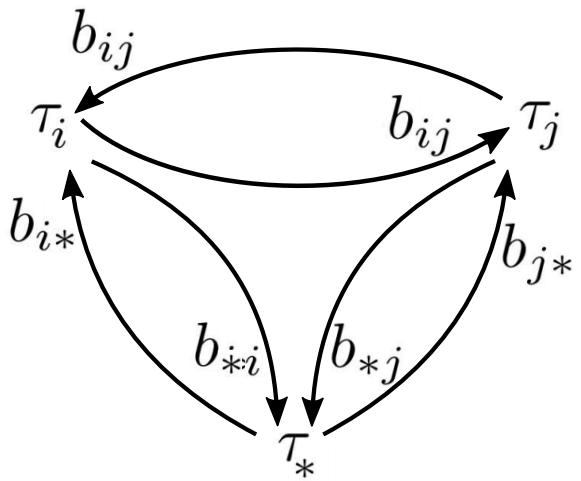


- A non-empty DBM is represented as a **graph** to prove that the normal form:
  - exists and is unique
  - is computed as the solution of an **all shortest path** problem in time $O(N^3)$

Difference Bounds Matrix: graph representation

- The DBM identified by the matrix of coefficients b_{ij} is biunivocally associated with a **complete directed labeled graph** $\langle V, E, b \rangle$ where:
  - the set of vertices V consists of the unknown values τ_i
  - a directed edge exists between any two edges (i.e., $E = V \times V$)
  - the edge from vertex τ_j to vertex τ_i is labeled by b_{ij}

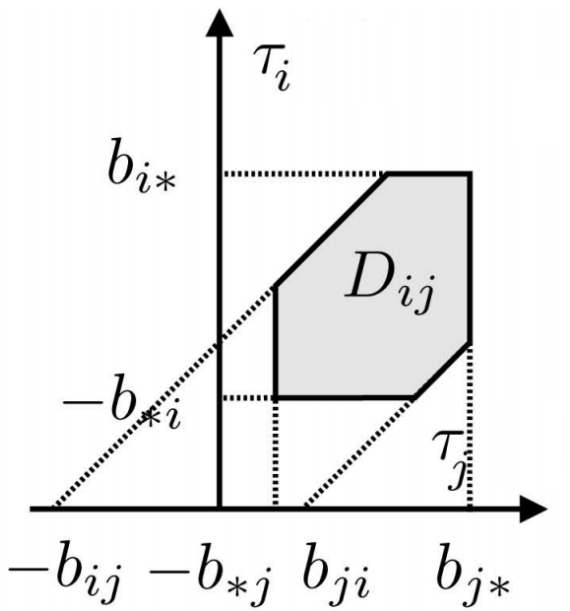


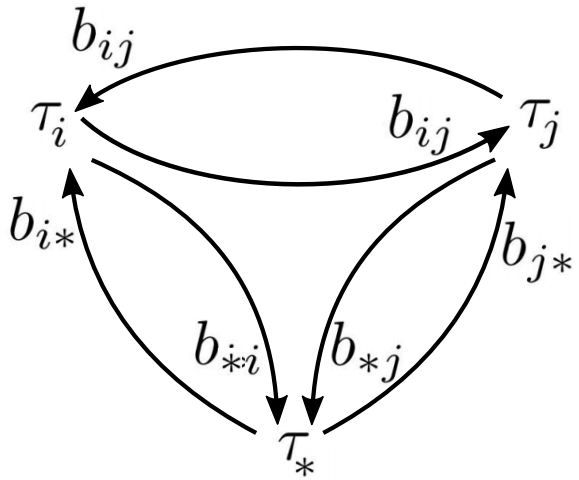


Difference Bounds Matrix: properties (1/4)

Lemma 1

*If a DBM zone is not empty, then the corresponding graph does not include any cycle where the sum of labels on traversed edges is negative (***negative cycle***).*

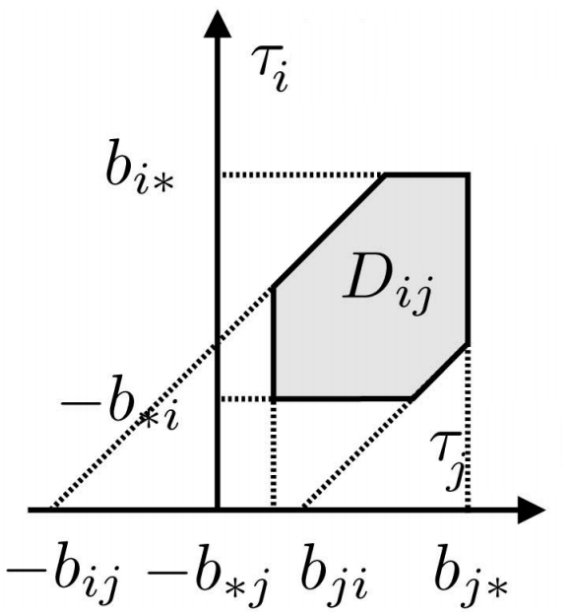


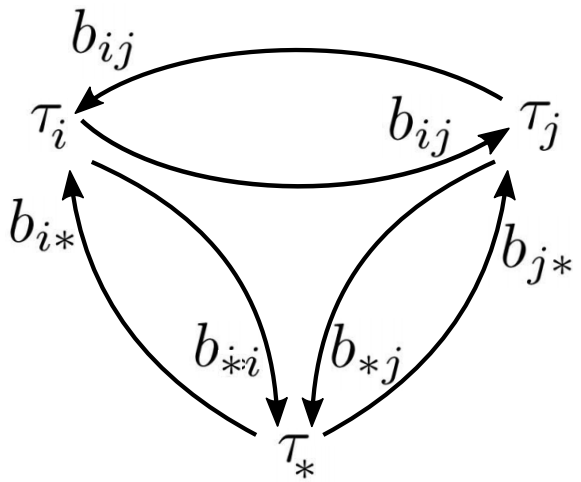


Difference Bounds Matrix: properties (2/4)

Lemma 2

A DBM is in normal form if and only if b_{ij} is the shortest path in the graph from vertex τ_j to vertex τ_i $\forall\, i, j \in \{0, 1, \ldots, N-1\} \cup \{\star\}$ with $i \neq j$.

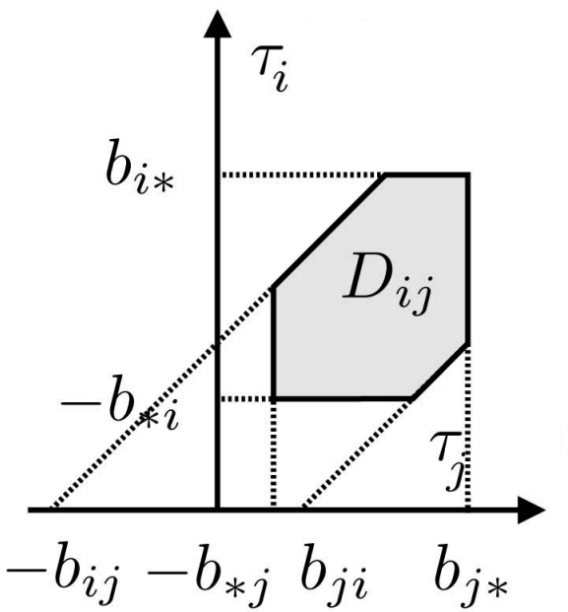


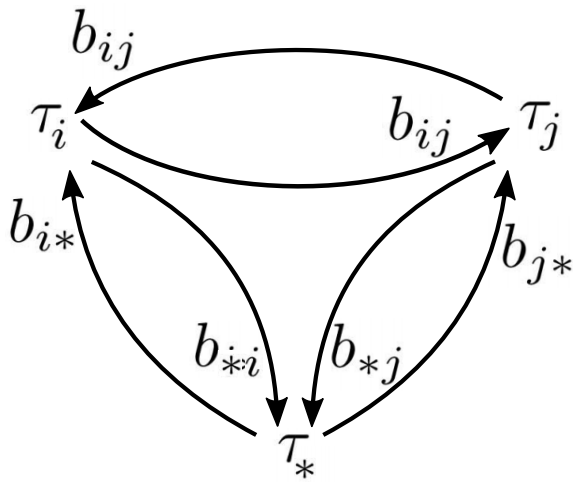


Difference Bounds Matrix: properties (3/4)

Lemma 3

Coefficients b_{ij} are the shortest paths between any two vertices τ_j and τ_i if and only if $b_{ij} \leq b_{ik} + b_{kj}$ $\forall\, i, j, k \in \{0, 1, \ldots, N-1\} \cup \{\star\}$ with $i \neq j$, $i \neq k$, $j \neq k$.

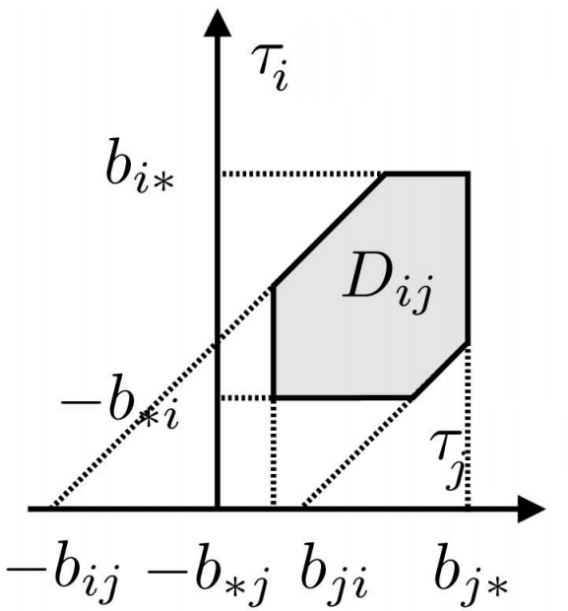


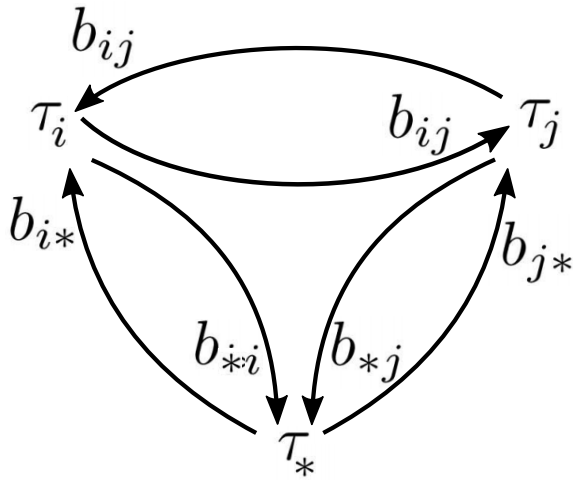


Difference Bounds Matrix: properties (4/4)

Theorem 4

A DBM is in normal form if and only if $b_{ij} \leq b_{ik} + b_{kj}$
$\forall\, i, j, k \in \{0, 1, \ldots, N-1\} \cup \{\star\}$ *with* $i \neq j$, $i \neq k$, $j \neq k$.





Difference Bounds Matrix: how to compute the normal form?

- The normal form can be computed by the **Floyd-Warshall algorithm**
  - N iterations indexed by $k = 0, 1, \ldots, N-1$
  - Iteration k derives b_{ij}^{k+1} defined as the shortest path from τ_j to τ_i that does not visit any intermediate node with index larger than k
    - **Initialization**: b_{ij}^0 is the initial value of coefficient b_{ij}
    - **First iteration** ($k = 0$): b_{ij}^1 is the shortest path from τ_j to τ_i that does not visit any intermediate node with index larger than 0
    - **Last iteration** ($k = N-1$): b_{ij}^N is the shortest path from τ_j to τ_i

Algorithm 2 Floyd-Warshall algorithm

1: **for** each $i, j \in \{0, 1, \ldots, N-1\} \cup \{\star\}$ with $i \neq j$ **do**
2: $\quad b_{ij}^0 = b_{ij}$
3: **end for**
4: **for** each $k = 0, 1, \ldots, N-1$ **do**
5: $\quad$ **for** each $i \in \{0, 1, \ldots, N-1\} \cup \{\star\}$ **do**
6: $\quad\quad$ **for** each $j \in \{0, 1, \ldots, N-1\} \cup \{\star\}$ with $i \neq j$ **do**
7: $\quad\quad\quad b_{ij}^{k+1} = \min\{b_{ij}^k, b_{ik}^k + b_{kj}^k\}$
8: $\quad\quad$ **end for**
9: $\quad$ **end for**
10: **end for**

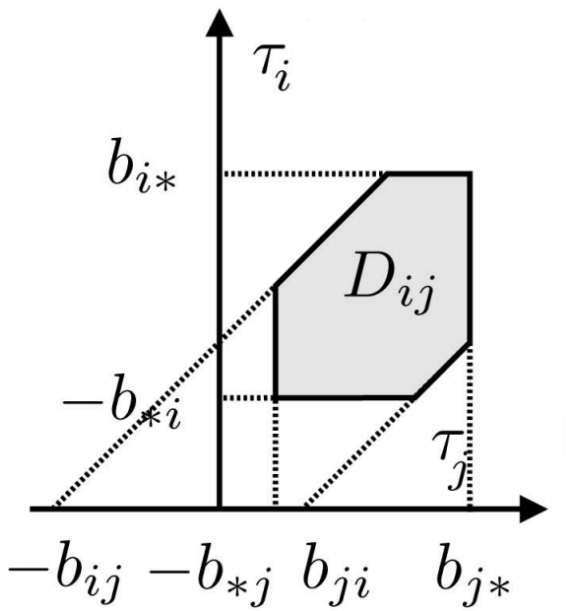
Difference Bounds Matrix: deciding emptiness

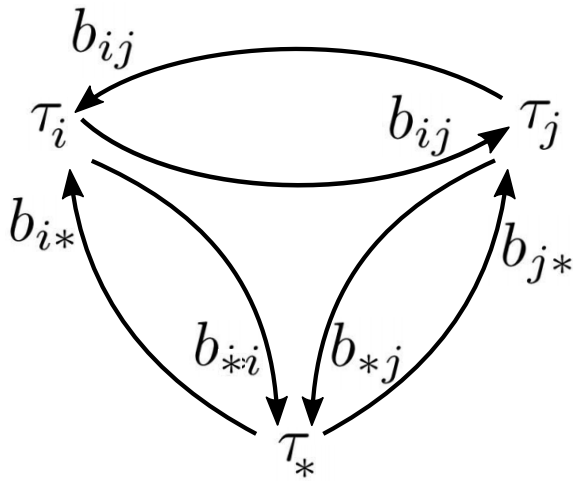
Lemma 5

If the graph corresponding to a DBM does not include any negative cycle, then the DBM zone is not empty.

Lemma 6

A DBM zone is not empty if and only if the corresponding graph does not include any negative cycle.





Difference Bounds Matrix: projection

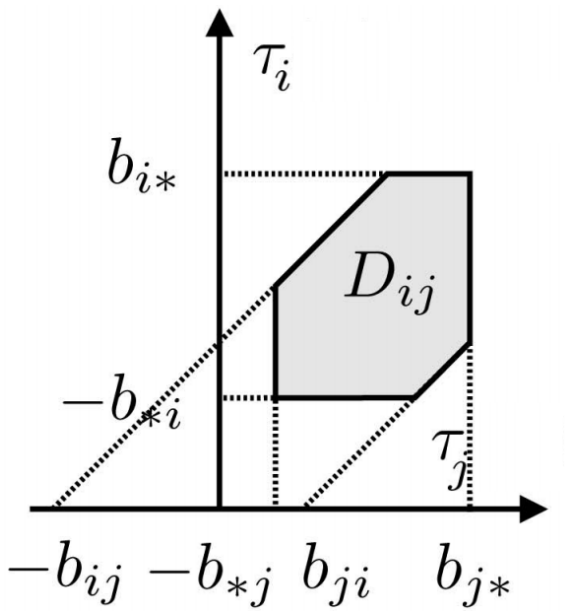
- Given a DBM D in the space of unknown values $\boldsymbol{\tau} = \langle \tau_0, \tau_1, \ldots, \tau_{N-1} \rangle$, the **projection** of D over the space of unknown values $\boldsymbol{\tau}_{\downarrow 0} = \langle \tau_1, \ldots, \tau_{N-1} \rangle$ is $D_{\downarrow 0} := \{ \langle \tau_1, \ldots, \tau_{N-1} \rangle \mid \exists\ \tau_0 \in \mathbb{R} \cup \{\infty\}$ such that $\langle \tau_0, \tau_1, \ldots, \tau_{N-1} \rangle \in D \}$
- Projection is used in the enumeration of the reachability relation between state classes
- Projection is efficiently implemented over a DBM zone in normal form

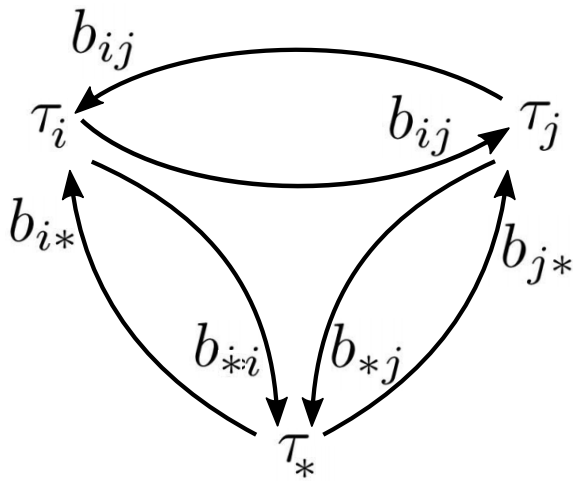
Lemma 7

If a DBM D in the space of unknown values $\boldsymbol{\tau} = \langle \tau_0, \tau_1, \ldots, \tau_{N-1} \rangle$ is in normal form, then the matrix of coefficients of the projection $D_{\downarrow 0}$ is obtained from the matrix of coefficients of D by discarding row 0 and column 0, i.e., the graph of $D_{\downarrow 0}$ is obtained from the graph of D by eliminating node τ_0 (without changing the constraint between any other two nodes).

TPN analysis: initial state class

- We assume that the firing domain D of the initial state class $S = \langle m, D \rangle$ is represented as a set of linear inequalities in DBM normal form, i.e., $D = \{\tau(t_i) - \tau(t_j) \leq b_{ij} \; \forall \, t_i, t_j \in T_e(m) \cup \{t_\star\}$ with $i \neq j$, where:
  - $b_{ij} \in \mathbb{R} \cup \{\infty\}$
  - $T_e(m)$ is the set of transitions enabled by marking m
  - $\tau(t_i)$ is the time to fire of transition $t_i \; \forall \, t_i \in T_e(m)$
  - $t_\star$ is the fictitious event of entrance into the class
  - $\tau(t_\star)$ is the **ground** time at which the class is entered





TPN analysis: successor existence

- A state class $S = \langle m, D \rangle$ with D in DBM normal form has a **successor** through transition t_0 if and only if t_0 is enabled by m and D accepts solutions where $\tau(t_0)$ is not larger than the time to fire of any other enabled transition, i.e., the **restricted firing domain** D_{t_0} has a non-empty set of solutions:

$$D_{t_0} = \begin{cases} \tau(t_i) - \tau(t_j) \leq b_{ij} \\ \tau(t_0) - \tau(t_h) \leq 0 \\ \forall\, t_i, t_j \in T_e(m) \cup \{t_\star\} \text{ with } t_i \neq t_j,\ \forall\, t_h \in T_e(m) \setminus \{t_0\} \end{cases}$$

Lemma 8

*A state class $S = \langle m, D \rangle$ with D in DBM normal form has a **successor** through transition t_0 if and only if $t_0 \in T_e(m)$ and $b_{h0} \geq 0\ \forall\, t_h \in T_e(m)$.*

- Successor detection: linear complexity wrt the number of enabled transitions
- How to compute the firing domain D' of the successor S' of S through t_0?

TPN analysis: successor computation - conditioning

- Condition D to the fact that t_0 fires first:

$$D_{t_0} = \begin{cases} D \\ \tau(t_0) - \tau(t_h) \leq 0 \\ \forall\, t_h \in T_e(m) \setminus \{t_0\} \end{cases} = \begin{cases} \tau(t_i) - \tau(t_j) \leq b_{ij} \\ \tau(t_0) - \tau(t_h) \leq 0 \\ \forall\, t_i, t_j \in T_e(m) \cup \{t_\star\} \,|\, t_i \neq t_j \\ \forall\, t_h \in T_e(m) \setminus \{t_0\} \end{cases}$$

$$= \begin{cases} \tau(t_i) - \tau(t_j) \leq b_{ij} \\ \tau(t_0) - \tau(t_h) \leq \min\{0, b_{0h}\} \\ \forall\, t_i, t_j \in T_e(m) \cup \{t_\star\} \,|\, t_i \neq t_j,\ \forall\, t_h \in T_e(m) \setminus \{t_0\} \end{cases}$$

- Represent D_{t_0} in normal form and let B_{ij} be its coefficients (quadratic complexity wrt the number of enabled transitions):

$$D_{t_0} = \begin{cases} \tau(t_i) - \tau(t_j) \leq B_{ij} \\ \tau(t_0) - \tau(t_i) \leq B_{0i} \\ \forall\, t_i, t_j \in T_e(m) \cup \{t_\star\} \,|\, t_i \neq t_j,\ t_i \neq t_0 \end{cases}$$

TPN analysis: successor computation - time advancement

- Reduce times to fire of persistent transitions in S' by the time elapsed in S, i.e., $\tau'(t_i) = \tau(t_i) - \tau(t_0)\ \forall\, t_i \in T_{\mathrm{e}}(m) \cap T_{\mathrm{p}}(S') \setminus \{t_0\}$ (where $T_{\mathrm{p}}(S')$ is the set of transitions that are persistent in S'):

$$D_{t_0} = \begin{cases} \tau(t_i) - \tau(t_j) \leq B_{ij} \\ \tau(t_i) - \tau(t_0) \leq B_{i0} \\ \tau(t_0) - \tau(t_i) \leq B_{0i} \\ \\ \tau(t_i) - \tau(t_\star) \leq B_{i\star} \\ \tau(t_\star) - \tau(t_i) \leq B_{\star i} \\ \tau(t_0) - \tau(t_\star) \leq B_{0\star} \\ \tau(t_\star) - \tau(t_0) \leq B_{\star 0} \\ \\ \forall\, t_i, t_j \in T_{\mathrm{e}}(m) \cap T_{\mathrm{p}}(S') \cup \{t_\star\} \mid \\ t_i \neq t_j,\ t_i \neq t_0,\ t_i \neq t_\star \end{cases} \qquad D'_{t_0} = \begin{cases} \tau'(t_i) - \tau'(t_j) \leq B_{ij} \\ \tau'(t_i) \leq B_{i0} \\ -\tau'(t_i) \leq B_{0i} \\ \\ \tau'(t_i) + \tau(t_0) \leq B_{i\star} \\ -\tau(t_0) - \tau'(t_i) \leq B_{\star i} \\ \tau(t_0) - \tau(t_i) \leq B_{0i} \\ \tau(t_0) - \tau(t_\star) \leq B_{0\star} \\ \\ \forall\, t_i, t_j \in T_{\mathrm{e}}(m) \cap T_{\mathrm{p}}(S') \cup \{t_\star\} \mid \\ t_i \neq t_j,\ t_i \neq t_0,\ t_j \neq t_\star \end{cases}$$

TPN analysis: successor computation - projection

- Eliminate the time to fire of the fired transition through a projection:

$$D''_{t_0} = \begin{cases} \tau'(t_i) - \tau'(t_j) \leq B_{ij} \\ \tau'(t_i) \leq B_{i0} \\ -\tau'(t_i) \leq B_{0i} \\ \forall\, t_i, t_j \in T_{\mathrm{e}}(m) \cup \{t_\star\} \,|\, t_i \neq t_j,\ t_i \neq t_0 \end{cases}$$

- Let C_{ij} be the normal form coefficients (i.e., $C_{ij} = B_{ij}$, $C_{i\star} = B_{i0}$, $C_{\star i} = B_{0i}$):

$$D''_{t_0} = \begin{cases} \tau'(t_i) - \tau'(t_j) \leq C_{ij} \\ \tau'(t_i) - \tau'(t_\star) \leq C_{i\star} \\ \tau'(t_\star) - \tau'(t_i) \leq C_{\star i} \\ \forall\, t_i, t_j \in T_{\mathrm{e}}(m) \cap T_{\mathrm{p}}(S') \cup \{t_\star\} \,|\, t_i \neq t_j,\ t_i \neq t_\star \end{cases}$$

- Times-to-fire of transitions not enabled by m' can be similarly eliminated

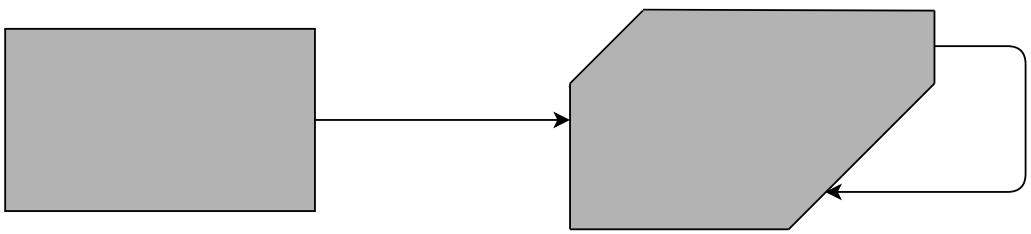
TPN analysis: successor computation - newly enabling

- Add the time to fire of each newly-enabled transition:

$$D' = \begin{cases} D''_{t_0} \\ \tau'(t_j) - \tau'(t_\star) \leq LFT(t_j) \\ \tau'(t_\star) - \tau'(t_j) \leq -EFT(t_j) \\ \forall\, t_j \in T_{\mathrm{n}}(S') \end{cases}$$

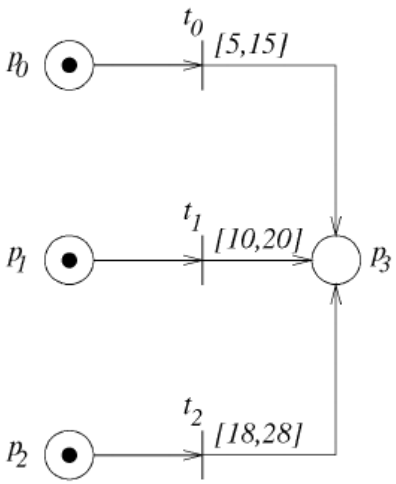
  where $T_{\mathrm{n}}(S')$ is the set of transitions newly-enabled in S'

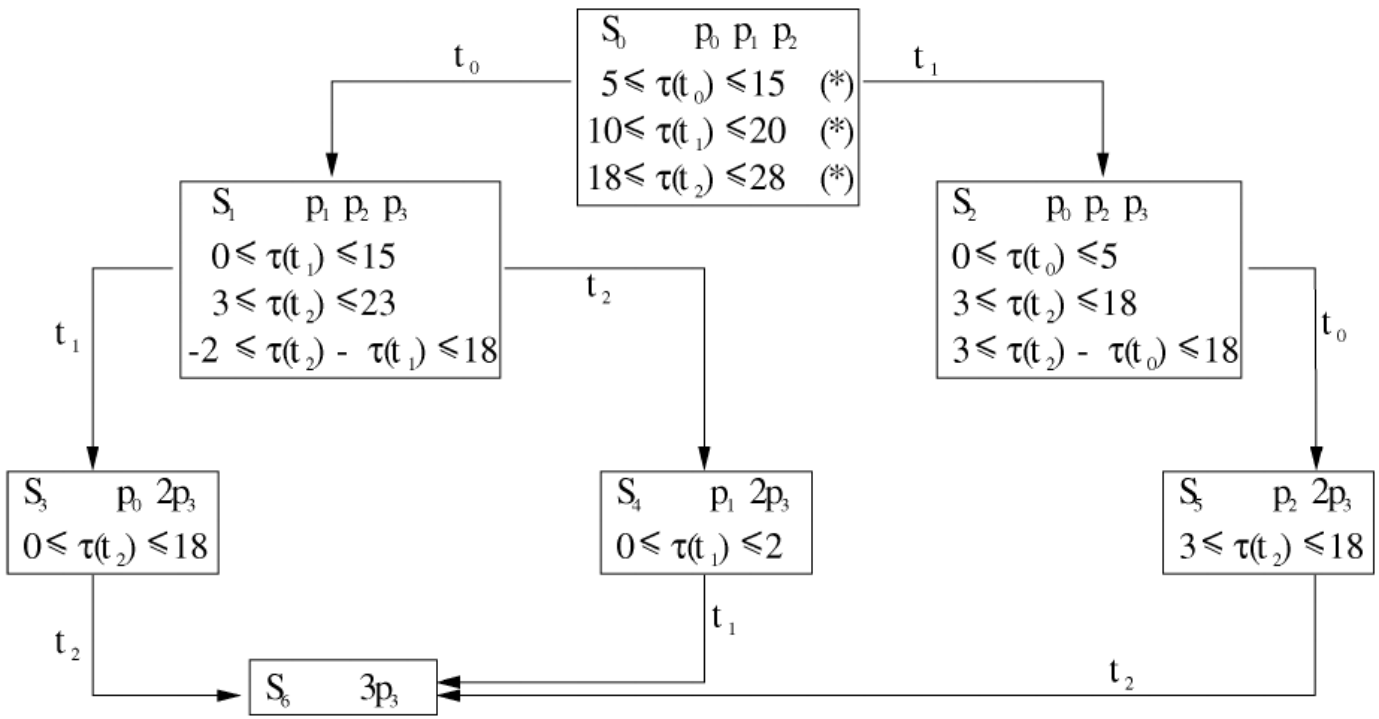
- The firing domain D' of the successor class S' is still in DBM form! $\Rightarrow$
  $\Rightarrow$ The DBM form is closed wrt the steps of successor computation!
  $\Rightarrow$ The reachability relation among state classes can be enumerated!



TPN analysis: state space enumeration

- Enumeration yields the **state class graph**
- A **symbolic run** is a path in the state class graph
  - Represents the dense variety of runs firing a given set of transitions in a given qualitative order with a dense variety of timings between subsequent firings
  - e.g., $S_0 \xrightarrow{t_0} S_1 \xrightarrow{t_2} S_4 \xrightarrow{t_1} S_6$





Domain of timings of a symbolic run

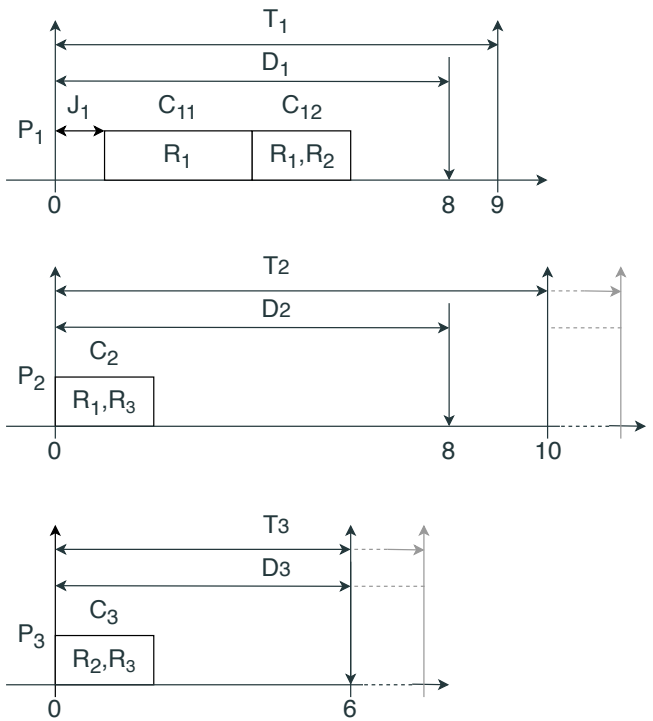
- Consider a symbolic run $\rho = S_0 \overset{t^0_{f(0)}}{\rightarrow} S_1 \overset{t^1_{f(1)}}{\rightarrow} S_2 \dots S_{N-1} \overset{t^{N-1}_{f(N-1)}}{\rightarrow} S_N$
  - $S_0 = \langle m_0, D_0 \rangle$ is the initial state class
  - $S_n = \langle m_n, D_n \rangle$ is the n-th state class visited by $\rho \; \forall \, n \in \{1, \dots, N\}$
  - $f(n)$ is the index of the transition fired in state class $S_n \; \forall \, n \in \{0, 1, \dots, N-1\}$
  - t^k_h is the time to fire of transition t_h in state class S_k
- The domain of timings of ρ is in DBM form:

$$D_\rho = \begin{cases} D_n \\ \tau(t^n_{f(n)}) - \tau(t^n_i) \leq 0 \\ \tau(t^n_{f(n)}) = \tau(t^{n+1}_\star) \\ \forall \, n \in \{0, 1, \dots, N-1\}, \; \forall \, t^n_i \in T_{\mathrm{e}}(m) \end{cases}$$

- Sojourn times in classes are **not independent** $\Rightarrow$ The worst case completion of ρ may not be the sum of the worst case sojourn times in $S_0, \dots, S_{N-1}$
- Computation of D_ρ: complexity $O(N^3)$
  - Note that N is the number of transitions **fired** along ρ
  - The number of transitions **enabled** along ρ can be $>> N$

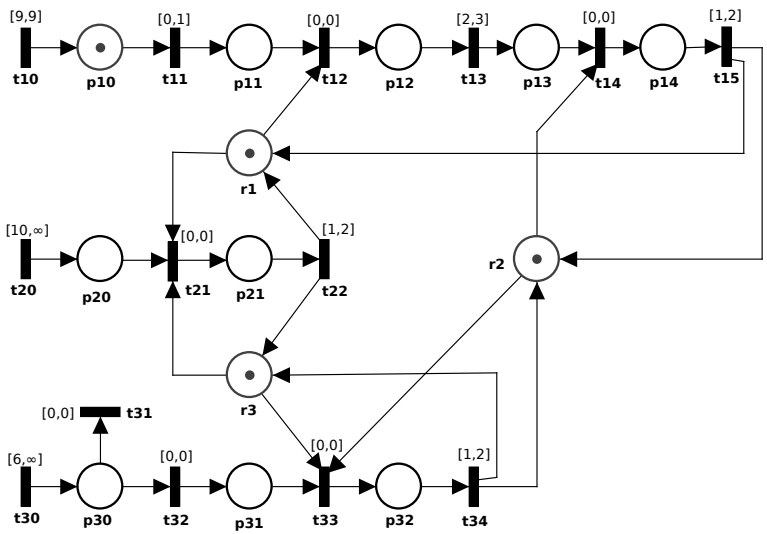
Modeling real-time task-sets with TPNs: an example (1/3)

- 3 tasks P_1, P_2, P_3 contending for 3 non-preemptable resources R_1, R_2, R_3
  - P_1 and P_2 are hard real-time (mandatory), P_3 is soft real-time (optional)
- P_1 is periodic
  - period $T_1 = 9$
  - relative deadline $D_1 = 8$
  - absolute start time jitter $J_1 \leq 1$
  - computation 1: WCET $C_{11} \in [2, 3]$, uses R_1
  - computation 2: WCET $C_{12} \in [1, 2]$, uses R_1, R_2
- P_2 is sporadic
  - inter-arrival time $T_2 \geq 8$
  - relative deadline $D_2 = 8$
  - computation: WCET $C_{21} \in [1, 2]$, uses R_1, R_3
- P_3 is sporadic
  - inter-arrival time $T_3 \geq 6$
  - relative deadline $D_3 = 6$
  - computation: WCET $C_{31} \in [1, 2]$, uses R_2, R_3



Modeling real-time task-sets with TPNs: an example (2/3)

- Which questions could the analysis of the TPN model answer?
  - If P_3 jobs are always rejected, do P_1 and P_2 jobs always meet their deadline?
  - If P_3 jobs are always accepted, do P_1 and P_2 jobs always meet their deadline?
  - Is there an acceptance/rejection strategy guaranteeing deadlines of P_1 and P_2?
- Which questions could the TPN analysis answer in general?
  - **Timed reachability**: which are the feasible orderings among transitions under the constraints imposed by the temporal parameters of the model?
  - **Min/max duration**: what is the minimum and maximum time at which some firing sequence (i.e., sequence of transition firings) can be completed?

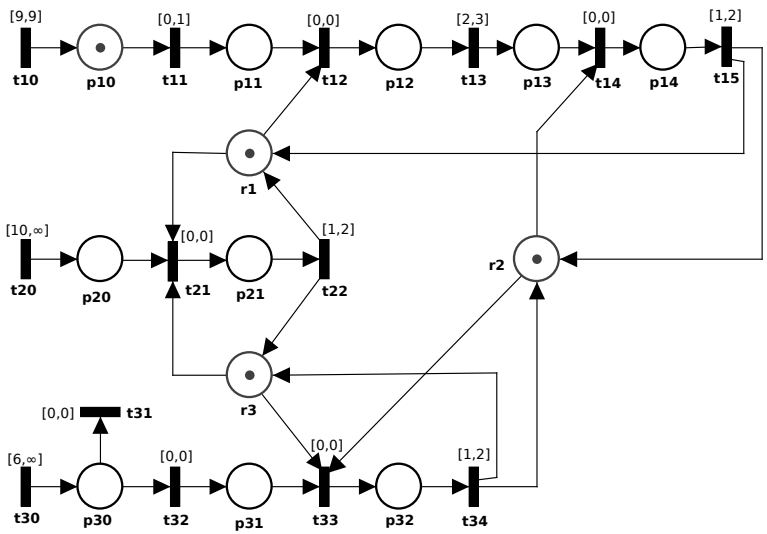


Modeling real-time task-sets with TPNs: an example (3/3)

- Both hard real-time tasks can miss their deadline:

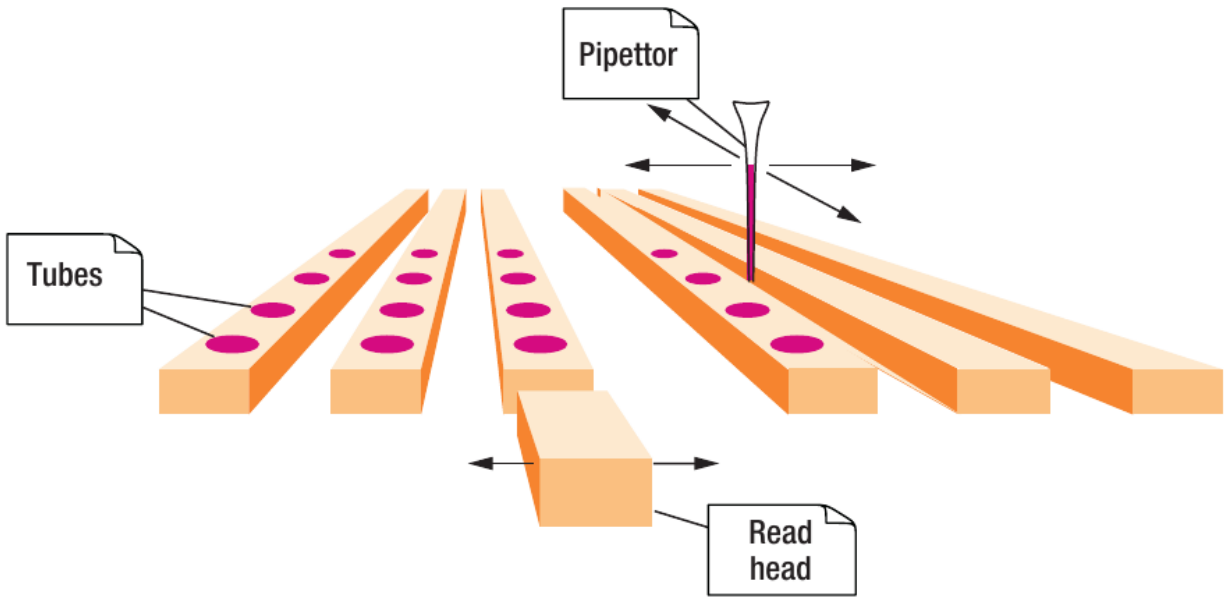
Task	Number of symbolic runs	Possible failures
P_1	3594	103
P_2	1871	352
P_3	1327	0

- How to decide acceptance/rejection of jobs of P_3 at runtime ? Using the **actual** execution timings to determine whether failing symbolic runs are feasible! A job of P_3 is discarded if a job of P_1/P_2 may miss its deadline!



Online scheduling in a biological analysis system (1/5)

- An electromechanical system for immunoenzymatic analysis by BioMérieux
  - Each slot comprises several test tubes
  - A pipettor and a read head can act simultaneously on test tubes

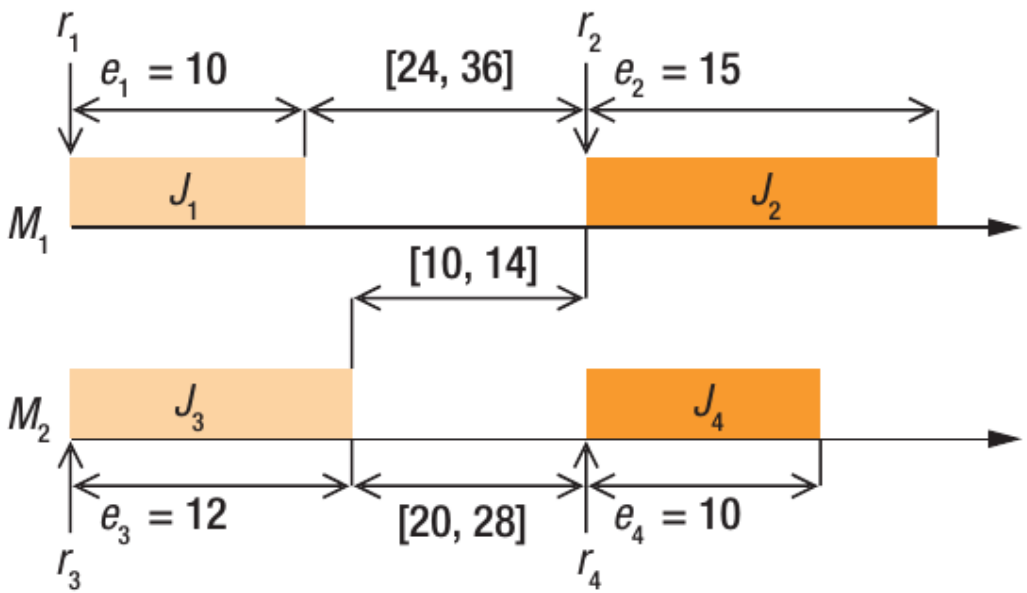


L. Ridi, J. Torrini, E. Vicario, "Developing a Scheduler with Difference-Bound Matrices and the Floyd-Warshall Algorithm," IEEE Software, 2012.

E. Vicario, L. Ridi, A. Carignano, J. Torrini, "Job scheduler for electromechanical system for biological analyses," Patent WO2013098088A1, 2013.

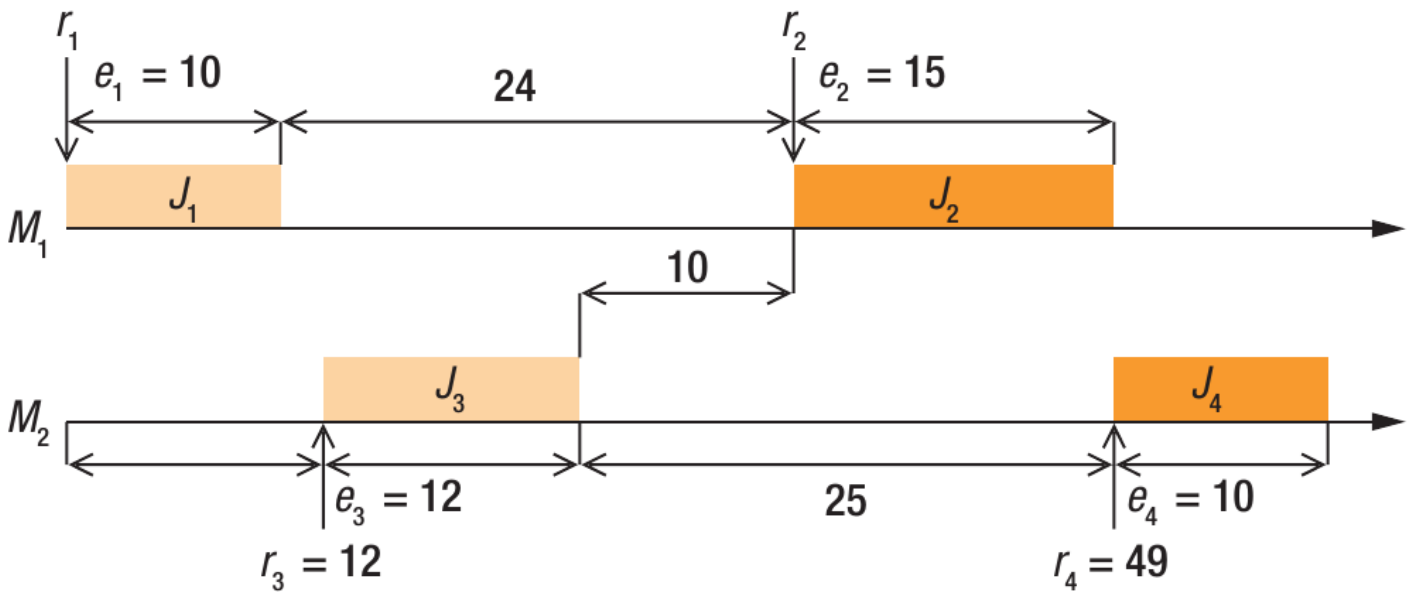
Online scheduling in a biological analysis system (2/5)

- An example of the scheduling problem
  - 4 jobs J_1, J_2, J_3, J_4 allocated on two machines M_1, M_2
  - e indicates an execution time, r indicates a release time
  - J_2 has a mutual exclusion with J_4
  - J_2 has a precedence constraints wrt the completion of J_1 and J_3
  - J_4 has a precedence constraints wrt the completion of J_3



Online scheduling in a biological analysis system (3/5)

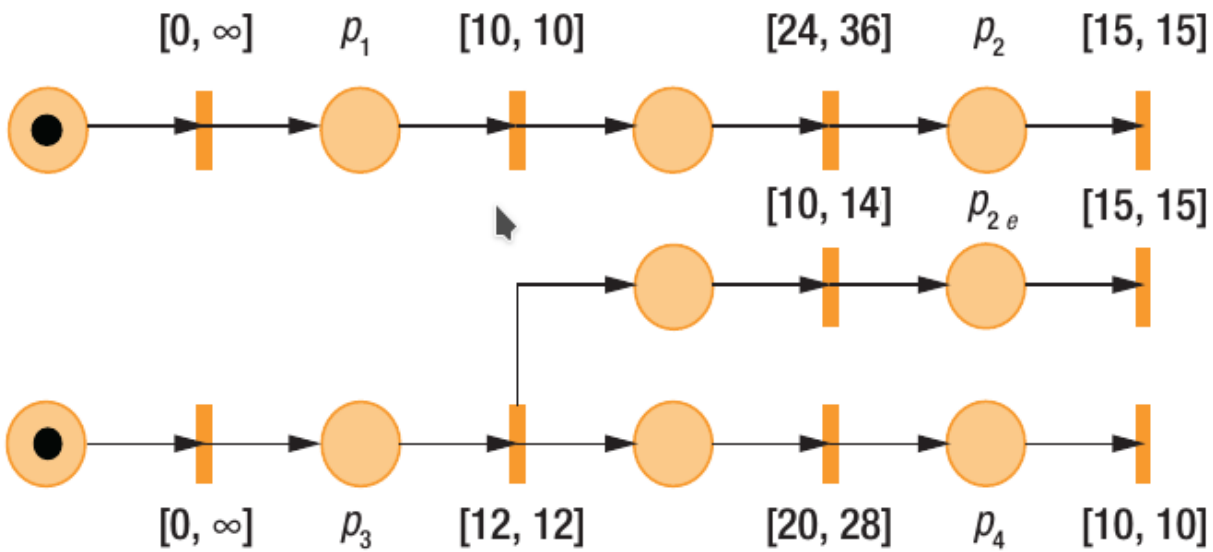
- A schedule that minimizes the overall completion time



- How to compute such a schedule?

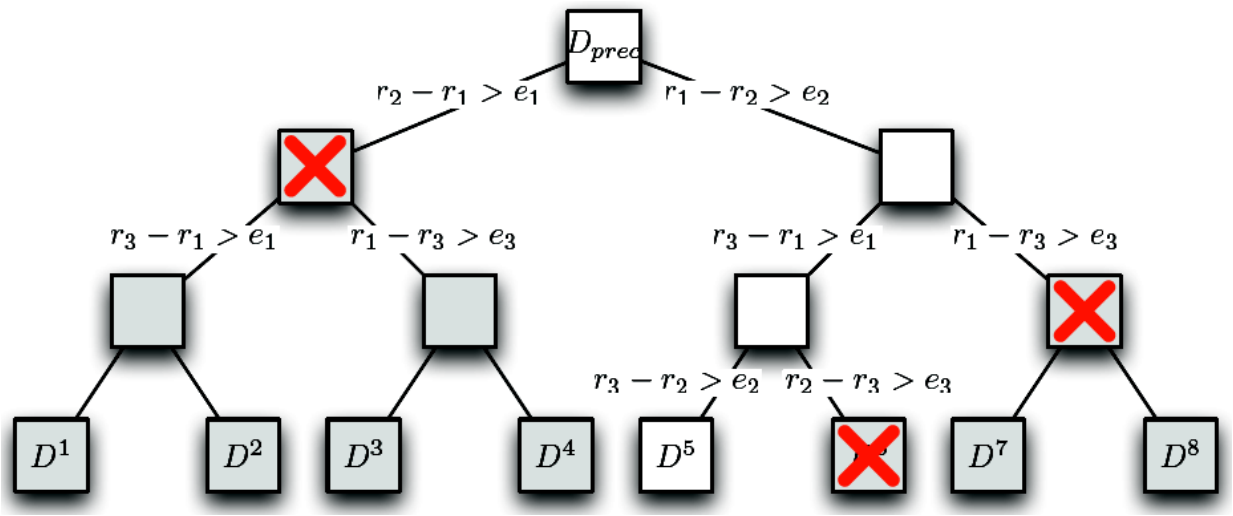
Online scheduling in a biological analysis system (4/5)

- TPN encoding sequencing and timing constraints of the scheduling problem
- An execution is correct if and only if it never visits a state class where:
  - place p_2 has a token while place p_{2e} is empty
  - both places p_2 and p_4 contain a token



Online scheduling in a biological analysis system (5/5)

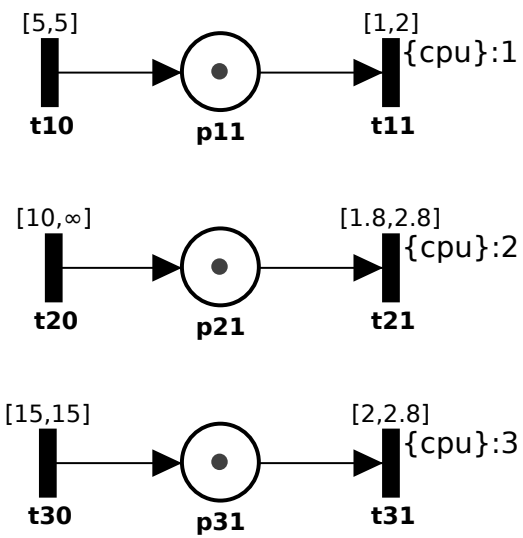
- State class graph enumeration to derive a schedule
- Optimization: heuristic-based enumeration of the state class graph
- Tests on challenging cases (e.g., 24 pipettor actions, 8 read head actions)
- Faster compared to traditional model checking approaches
- Software implementation running on a real system!



Preemptive Time Petri Nets

PTPN syntax

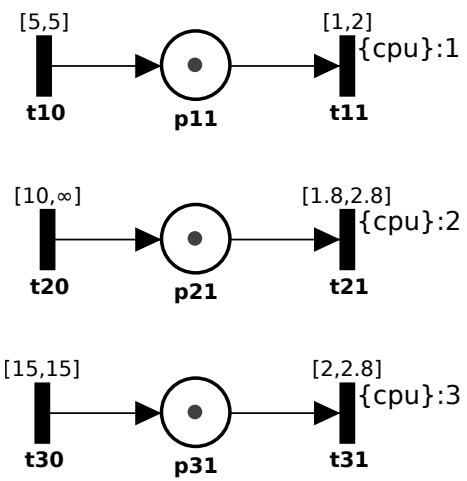
- Preemptive Time Petri Nets (PTPNs) extend TPNs with **resources**
- A PTPN is a tuple $\langle P, T, A^-, A^+, A^\cdot, EFT, LFT, Res, Req, Prio \rangle$
  - P, T, A^-, A^+, $A^\cdot$, EFT, LFT are the elements of a TPNs
  - Res is a set of resources
  - Req associates each transition t with a subset of resources $Req(t) \subseteq Res$
  - $Prio$ associates each transition t with a priority $Prio(t) \in \mathbb{N}$
    (the lower the priority number is, the higher the priority level is)
- Example
  - The model includes a resource cpu
  - `t11`, `t21`, and `t31` are all enabled by the initial marking `p11` `p21` `p31`...
  - ...but only `t11` can be assigned cpu



G. Bucci, A. Fedeli, L. Sassoli, E. Vicario, "Timed State Space Analysis of Real-Time Preemptive Systems," IEEE Transactions on Software Engineering, 2004.

State of a PTPN

- The **state** of a PTPN is a pair $\langle m, \tau \rangle$ where
  - m is a marking (representing the logical location of the system)
  - τ associates each enabled transition t with a **time to fire** $\tau(t) \in \mathbb{R}_{\geq 0}$
- Example
  - t10 can be associated with a time to fire $\tau(\texttt{t10}) = 5$
  - t20 can be associated with a time to fire $\tau(\texttt{t20}) \in [10, \infty]$
  - t30 can be associated with a time to fire $\tau(\texttt{t30}) = 15$
  - t11 can be associated with a time to fire $\tau(\texttt{t11}) \in [1, 2]$
  - t21 can be associated with a time to fire $\tau(\texttt{t21}) \in [1.8, 2.8]$
  - t31 can be associated with a time to fire $\tau(\texttt{t31}) \in [2, 2.8]$

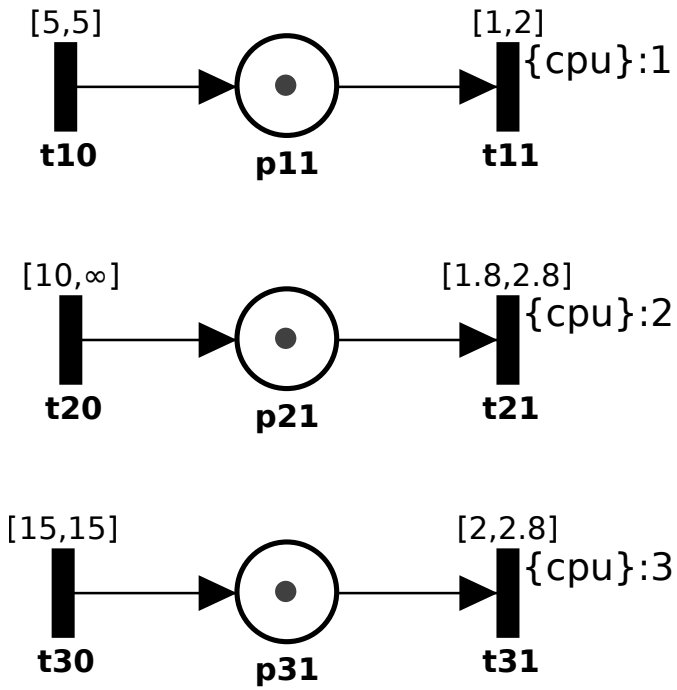


PTPN semantics (1/5)

- As in PNs with inhibitor arcs, a transition t is **enabled** by a marking m iff:
  - $m(p) > 0 \; \forall \, p \in P \mid (p, t) \in A^-$
    (i.e., m assigns at least one token to each input place of t)
  - $m(p) = 0 \; \forall \, p \in P \mid (p, t) \in A^\bullet$
    (i.e., m assigns no token to each inhibitor place of t)
- A transition t is **progressing** in a marking m iff:
  - it is enabled by m
  - $Req(t) \cap Req(t') \neq \varnothing \Rightarrow Prio(t) < Prio(t') \; \forall \, t' \in T_e(m)$ (i.e., every required resource is not required by another enabled transition with higher priority level)
- A transition t is **suspended** in a marking m iff:
  - it is enabled by m
  - it is not progressing in m
- A transition t is **firable** in a state $s = \langle m, \tau \rangle$ iff:
  - it is progressing in m
  - $\tau(t) \leq \tau(t') \; \forall \, t' \in T \mid t'$ is progressing in m (i.e., the time to fire of t is not larger than the time to fire of any other progressing transition)

PTPN semantics (2/5)

- In any possible initial state of the example PTPN:
  - t10, t20, t30, and t11 are progressing
  - t21 and 31 are suspended

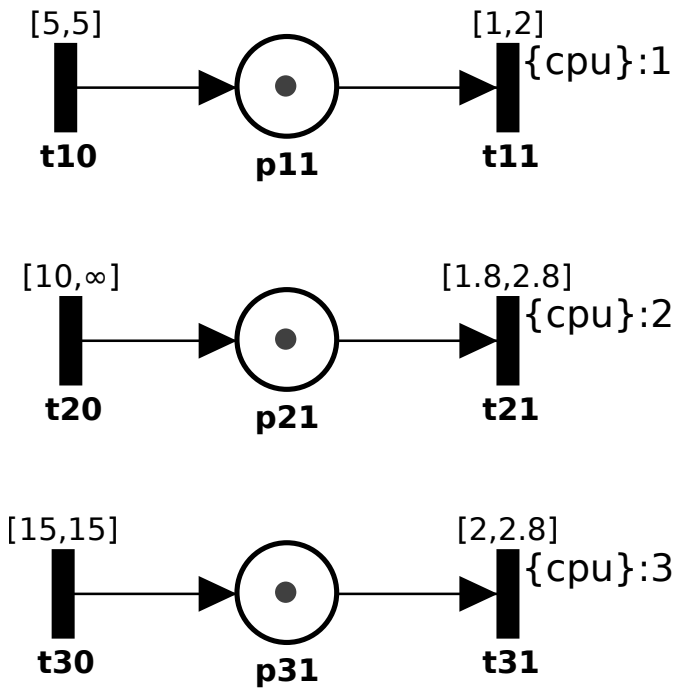


PTPN semantics (3/5)

- The firing of a transition t replaces $s_0 = \langle m_0, \tau_0 \rangle$ by $s_1 = \langle m_1, \tau_1 \rangle$
  - As in PNs, m_1 is derived from m_0 by:
    1. removing a token from each input place of t
       (i.e., $m_{\text{tmp}} = m_0(p) - 1 \; \forall \, p \,|\, (p, t) \in A^-$)
    2. adding a token to each output place of t
       (i.e., $m_1 = m_{\text{tmp}}(p) + 1 \; \forall \, p \,|\, (t, p) \in A^+$)
  - As in TPNs, a transition t' enabled by m_1 is termed:
    - **persistent** if it is not t and it is enabled both by m_0 and m_{tmp}
    - **newly enabled** if it is not enabled by m_0 or by m_{tmp},
      or if it is the fired transition t and it is enabled by m_1
  - As in TPNs, a transition is termed **disabled** if enabled by m_0 but not by m_1
  - τ_1 is derived from τ by:
    1. reducing the time to fire of each **persistent progressing** transition by $\tau_0(t)$
       (i.e., $\tau_1(t') = \tau_0(t') - \tau_0(t) \; \forall \, t' \in T \,|\, t'$ is persistent and progressing in m_1)
    2. leaving the time to fire of each **persistent suspended** transition unchanged
       (i.e., $\tau_1(t') = \tau_0(t') \; \forall \, t' \in T \,|\, t'$ is persistent and suspended in m_1)
    3. eliminating the time to fire of each transition **disabled** by the firing of t
    4. sampling the time to fire of each **newly enabled** transition t' in its firing
       interval (i.e., $\tau_1(t') \in [EFT(t'), LFT(t')] \; \forall \, t' \in T \,|\, t'$ is newly enabled in m_1)

PTPN semantics (4/5)

- Example PTPN
  - t11 is the only firable transition in any initial state with marking p11 p21 p31
  - The firing of t11 in state $\langle \mathtt{p11}\,\mathtt{p21}\,\mathtt{p31}, \tau_0 \rangle$ yields state $\langle \mathtt{p21}\,\mathtt{p31}, \tau_1 \rangle$ where $\tau_1(\mathtt{t10}) = \tau_0(\mathtt{t10}) - \tau_0(\mathtt{t11})$, $\tau_1(\mathtt{t20}) = \tau_0(\mathtt{t20}) - \tau_0(\mathtt{t11})$, $\tau_1(\mathtt{t20}) = \tau_0(\mathtt{t30}) - \tau_0(\mathtt{t11})$, $\tau_1(\mathtt{t21}) = \tau_0(\mathtt{t21})$, $\tau_1(\mathtt{t31}) = \tau_0(\mathtt{t31})$

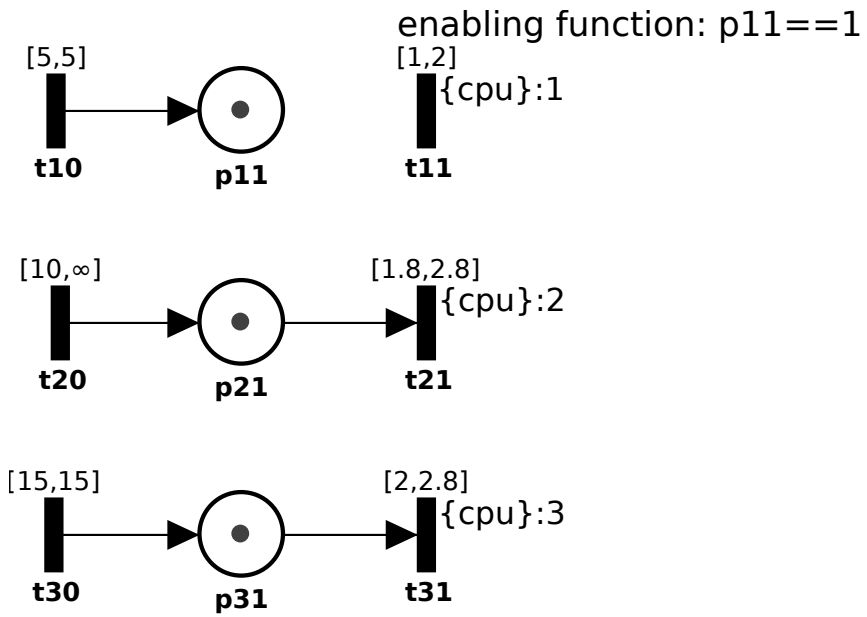


PTPN semantics (5/5)

- An **execution** of a PTPN is a path $\omega = s_0 \xrightarrow{\gamma_1} s_1 \xrightarrow{\gamma_2} s_2 \xrightarrow{\gamma_3} \dots$ such that:
  - $s_0 = \langle m_0, \tau_0 \rangle$ is the initial state
  - γ_i is the i-th fired transition
  - $s_i = \langle m_i, \tau_i \rangle$ is the state reached after the firing of γ_i
  - γ_i is selected among the transitions **firable** in state $s_{i-1} = \langle m_{i-1}, \tau_{i-1} \rangle$
  - m_i is derived from m_{i-1} by
    1. removing a token from each input place of γ_i
       (i.e., $m_{\text{tmp}} = m_{i-1}(p) - 1 \; \forall \, p \,|\, (p, \gamma_i) \in A^-$)
    2. adding a token to each output place of γ_i
       (i.e., $m_i = m_{\text{tmp}}(p) + 1 \; \forall \, p \,|\, (\gamma_i, p) \in A^+$)
  - τ_i is derived from τ_{i-1} by:
    1. reducing the time to fire of each persistent progressing transition by $\tau_{i-1}(\gamma_i)$
       (i.e., $\tau_i(t') = \tau_{i-1}(t') - \tau_{i-1}(\gamma_i) \; \forall \, t' \in T \,|\, t'$ is persistent progressing in s_i)
    2. leaving the time to fire of each persistent suspended transition unchanged
       (i.e., $\tau_i(t') = \tau_{i-1}(t') \; \forall \, t' \in T \,|\, t'$ is persistent suspended in s_i)
    3. eliminating the time to fire of each transition disabled by the firing of γ_i
    4. sampling the time to fire of each newly enabled transition in its firing interval
       (i.e., $\tau_i(t') \in [EFT(t), LFT(t)] \; \forall \, t' \in T \,|\, t$ is newly enabled in m_i)
  - ω is a finite or infinite path

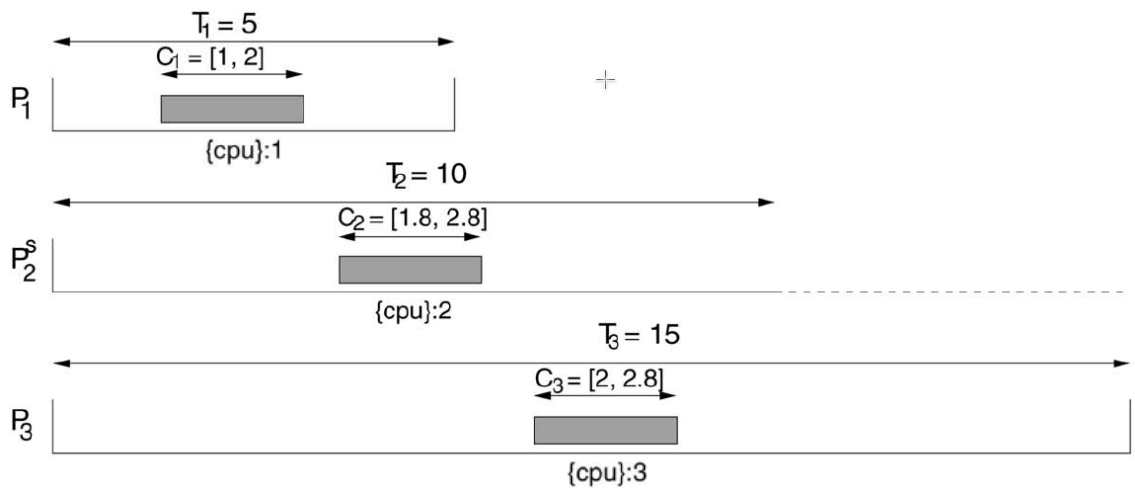
Extension of syntax and semantics of PTPNs

- PTPNs could be extended with enabling functions, update functions, priorities (intended to resolve nondeterminism, not associated with resources)
  - These features would improve the modeling convenience wrt PTPNs
  - These features would not increase the model expressivity wrt PTPNs



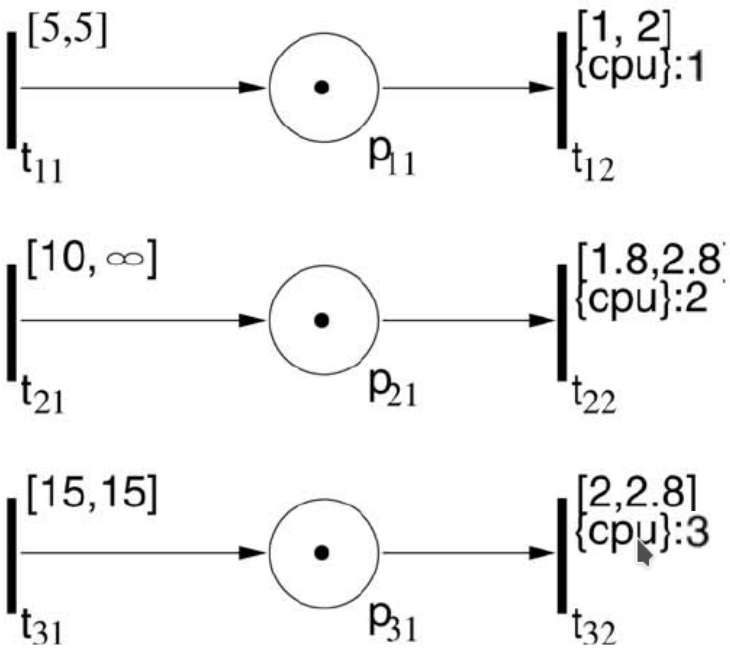
Modeling real-time task-sets with PTPNs: an example (1/4)

- 3 **independent** tasks P_1, P_2, P_3 contending for a **preemptable** resource cpu
  - P_1, P_2, P_3 run under **fixed-priority preemptive scheduling** P_1 is periodic: period $T_1 = 5$, relative deadline $D_1 = 5$, WCET $C_{11} \in [1, 2]$, requires resource cpu with priority 1
  - P_2 is sporadic: min inter-arrival time $T_2 = 15$, relative deadline $D_2 = 15$, WCET $C_{11} \in [1.8, 2.8]$, requires resource cpu with priority 2
  - P_3 is periodic: period $T_3 = 15$, relative deadline $D_3 = 15$, WCET $C_{11} \in [2, 2.8]$, requires resource cpu with priority 3
- Timeline representation of the task-set



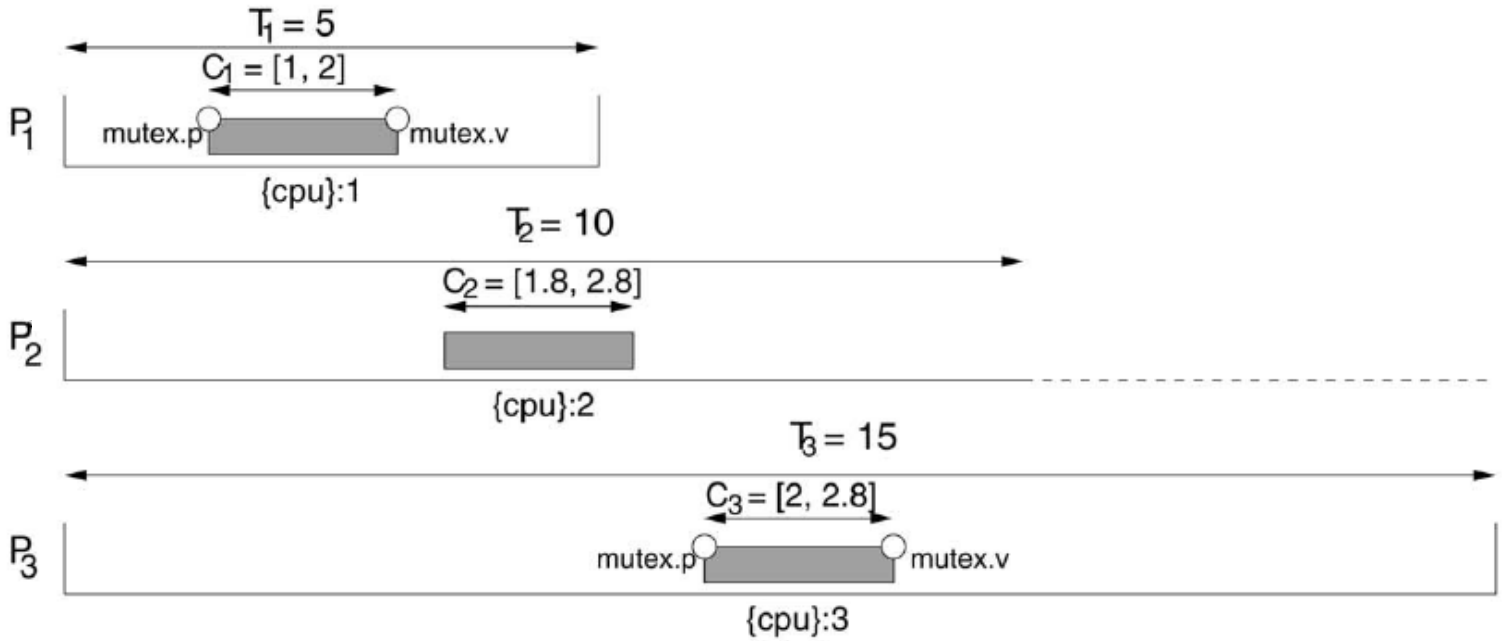
Modeling real-time task-sets with PTPNs: an example (2/4)

- Corresponding PTPN model
  - Are deadlines met? Which is the min/max completion time of each task?
  - If deadlines were not met, what could the software designer do?
  - How could preemptive Earliest Deadline First (EDF) scheduling be modeled?



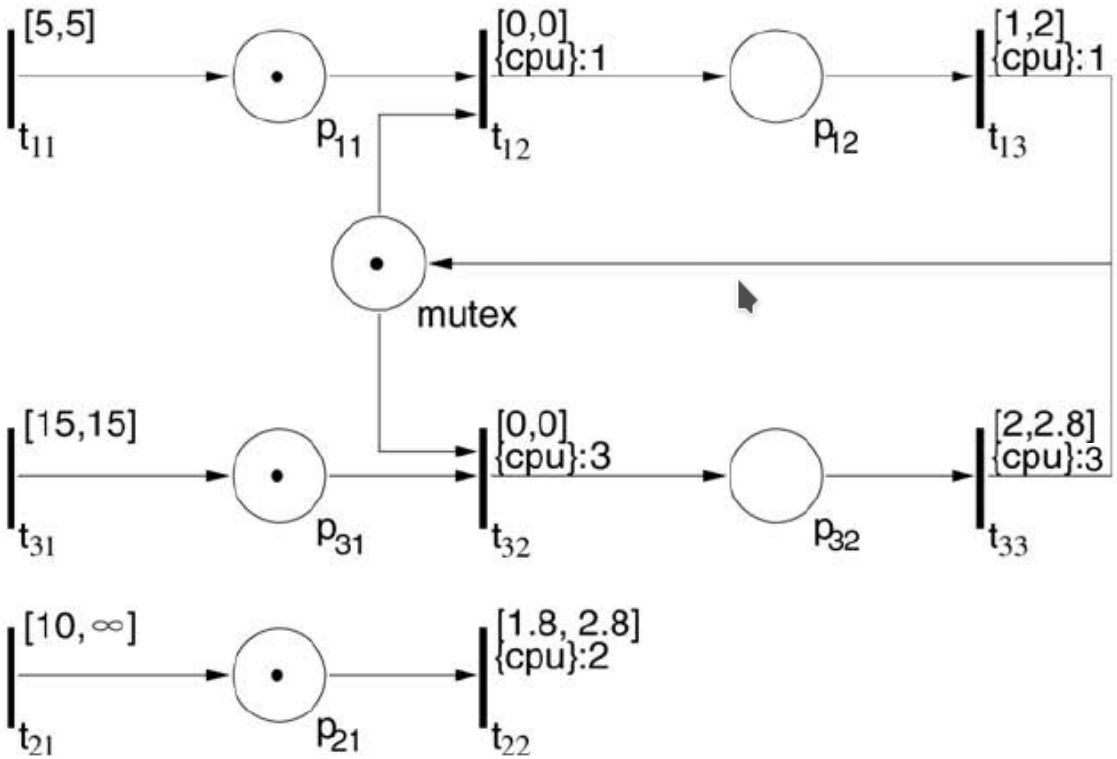
Modeling real-time task-sets with PTPNs: an example (3/4)

- A variant of the task-set where tasks P_1 and P_3 share an **exclusive resource** protected against concurrent accesses by a **binary semaphore** mutex
- Timeline representation of the task-set



Modeling real-time task-sets with PTPNs: an example (4/4)

- Corresponding PTPN model
  - Are deadlines met? Which is the min/max completion time of each task?
  - If deadlines were not met, what could the software designer do?
  - How could preemptive Earliest Deadline First (EDF) scheduling be modeled?



PTPN analysis: preliminaries

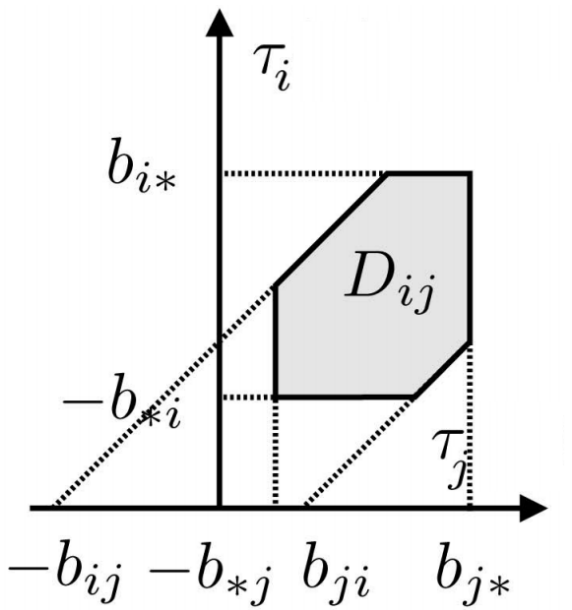
- As in TPNs, a **state class** is a pair $S = \langle m, D \rangle$ where:
  - m is a marking (representing the logical location of the system)
  - D is a continuous set of values for the times to fire of the enabled transitions
- As in TPNs, a state $s = \langle m_s, \vec{\tau_s} \rangle$ is included in a state class $S = \langle m, D \rangle$ if:
  - they have the same marking (i.e., $m_s = m$)
  - the times to fire vector $\vec{\tau_s}$ satisfies the constraints of D (i.e., $\vec{\tau_s} \in D$)
- As in TPNs, a state class S' is **reachable** from class S through transition t (i.e., $S \xrightarrow{t} S'$) if and only if it contains all and only the states that are reachable from some state collected in S through some feasible firing of t, i.e., S' is reachable from S through t if and only if:
  - $\forall\, s' \in S'\ \exists\, s \in S \mid s \xrightarrow{t} s'$
  - $s \in S \wedge s \xrightarrow{t} s' \Rightarrow s' \in S'$

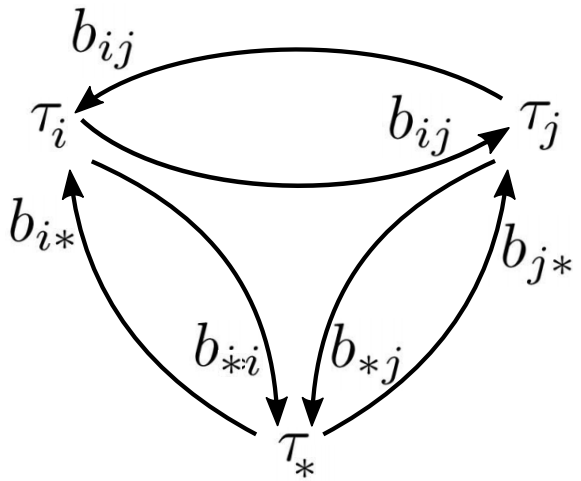
PTPN analysis: an intuitive introduction

TBD

PTPN analysis: initial state class

- As in TPNs, we assume that the firing domain D of the initial state class $S = \langle m, D \rangle$ is represented as a set of linear inequalities in DBM normal form, i.e., $D = \{\tau(t_i) - \tau(t_j) \leq b_{ij} \ \forall\, t_i, t_j \in T_e(m) \cup \{t_\star\}$ with $i \neq j$, where:
  - $b_{ij} \in \mathbb{R} \cup \{\infty\}$
  - $T_e(m)$ is the set of transitions enabled by marking m
  - $\tau(t_i)$ is the time to fire of transition $t_i \ \forall\, t_i \in T_e(m)$
  - $t_\star$ is the fictitious event of entrance into the class
  - $\tau(t_\star)$ is the **ground** time at which the class is entered





PTPN analysis: successor existence

- A state class $S = \langle m, D \rangle$ with D in DBM normal form has a successor through transition t_0 if and only if t_0 is **progressing** in m and D accepts solutions where $\tau(t_0)$ is not larger than the time to fire of any other **progressing** transition, i.e., the restricted firing domain D_{t_0} is non-empty:

$$D_{t_0} = \begin{cases} \tau(t_i) - \tau(t_j) \leq b_{ij} \\ \tau(t_0) - \tau(t_h) \leq 0 \\ \forall\, t_i, t_j \in T_{\mathrm{e}}(m) \cup \{t_\star\} \text{ with } t_i \neq t_j,\ \forall\, t_h \in T_{\mathrm{pr}}(m) \setminus \{t_0\} \end{cases}$$

where $T_{\mathrm{pr}}(m)$ is the set of transitions that are **progressing** in m

Lemma 9

A state class $S = \langle m, D \rangle$ with D in DBM normal form has a **successor** *through transition t_0 if and only if $t_0 \in T_{\mathrm{pr}}(m)$ and $b_{h0} \geq 0\ \forall\, t_h \in T_{\mathrm{pr}}(m)$.*

- Successor detection: linear complexity wrt the number of enabled transitions
- Given $\langle m, D \rangle \xrightarrow{t_0} \langle m', D' \rangle$, how to compute the min embedding DBM of D'?

PTPN analysis: successor computation - conditioning

- Condition D to the fact that t_0 fires first:

$$D_{t_0} = \begin{cases} D \\ \tau(t_0) - \tau(t_h) \leq 0 \\ \forall\, t_h \in T_{\mathrm{pr}}(m) \setminus \{t_0\} \end{cases} = \begin{cases} \tau(t_i) - \tau(t_j) \leq b_{ij} \\ \tau(t_0) - \tau(t_h) \leq 0 \\ \forall\, t_i, t_j \in T_{\mathrm{e}}(m) \cup \{t_\star\} \,|\, t_i \neq t_j \\ \forall\, t_h \in T_{\mathrm{pr}}(m) \setminus \{t_0\} \end{cases}$$

$$= \begin{cases} \tau(t_i) - \tau(t_j) \leq b_{ij} \\ \tau(t_0) - \tau(t_h) \leq \min\{0, b_{0h}\} \\ \forall\, t_i, t_j \in T_{\mathrm{e}}(m) \cup \{t_\star\} \,|\, t_i \neq t_j,\ \forall\, t_h \in T_{\mathrm{pr}}(m) \setminus \{t_0\} \end{cases}$$

- Represent D_{t_0} in normal form and let B_{ij} be its coefficients (quadratic complexity wrt the number of enabled transitions):

$$D_{t_0} = \begin{cases} \tau(t_i) - \tau(t_j) \leq B_{ij} \\ \tau(t_0) - \tau(t_i) \leq B_{0i} \\ \forall\, t_i, t_j \in T_{\mathrm{e}}(m) \cup \{t_\star\} \,|\, t_i \neq t_j,\ t_i \neq t_0 \end{cases}$$

PTPN analysis: successor computation - time adv. (1/4)

- Reduce by the time elapsed in S the time to fire of transitions that are **progressing** in m and **persistent** in S',
  i.e., $\tau'(t_i) = \tau(t_i) - \tau(t_0) \; \forall \, t_i \in T_{\mathrm{pr}}(m) \cap T_{\mathrm{p}}(S') \setminus \{t_0\}$
- Do not change the time to fire of transitions that are **suspended** in m and **persistent** in S',
  i.e., $\tau'(t_x) = \tau(t_x) \; \forall \, t_x \in T_{\mathrm{s}}(m) \cap T_{\mathrm{p}}(S') \setminus \{t_0\}$,
  where $T_{\mathrm{s}}(m)$ is the set of transitions that are **suspended** in m
- Before substituting variables, rewrite the inequalities in order to:
  - make explicit the constraints involving t_0 (fired transition) and $t_\star$ (ground)
  - distinguish progressing transitions from suspended transitions

PTPN analysis: successor computation - time adv. (2/4)

- Rewrite the inequalities in D_{t_0}:

$$
D_{t_0} = \begin{cases}
\tau(t_i) - \tau(t_j) \leq B_{ij} & \text{(progr,progr)} \\
\tau(t_i) - \tau(t_\star) \leq B_{i\star} & \text{(progr,}\star\text{)} \\
\tau(t_\star) - \tau(t_i) \leq B_{\star i} & \text{(}\star\text{,progr)} \\
\tau(t_i) - \tau(t_0) \leq B_{i0} & \text{(progr,0)} \\
\tau(t_0) - \tau(t_i) \leq B_{0i} & \text{(0,progr)} \\
\tau(t_i) - \tau(t_x) \leq B_{ix} & \text{(progr,susp)} \\
\tau(t_y) - \tau(t_j) \leq B_{yj} & \text{(susp,progr)} \\
\tau(t_x) - \tau(t_\star) \leq B_{x\star} & \text{(susp,}\star\text{)} \\
\tau(t_\star) - \tau(t_x) \leq B_{\star x} & \text{(}\star\text{,susp)} \\
\tau(t_y) - \tau(t_0) \leq B_{y0} & \text{(susp,0)} \\
\tau(t_0) - \tau(t_x) \leq B_{0x} & \text{(0,susp)} \\
\tau(t_x) - \tau(t_y) \leq B_{xy} & \text{(susp,susp)} \\
\tau(t_0) - \tau(t_\star) \leq B_{0\star} & \text{(0,}\star\text{)} \\
\tau(t_\star) - \tau(t_0) \leq B_{\star 0} & \text{(}\star\text{,0)} \\
\forall\, t_i, t_j \in T_{\mathrm{pr}}(m) \cap T_{\mathrm{p}}(S') \setminus \{t_0\} \mid t_i \neq t_j,\ t_i, t_j \neq t_0,\ t_i, t_j \neq t_\star \\
\forall\, t_x, t_y \in T_{\mathrm{s}}(m) \cap T_{\mathrm{p}}(S') \mid t_x \neq t_y,\ t_x, t_y \neq t_0,\ t_x, t_y \neq t_\star
\end{cases}
$$

PTPN analysis: successor computation - time adv. (3/4)

- Substitute variables in D_{t_0}:

$$D_{t_0} = \begin{cases} \tau(t_i) - \tau(t_j) \leq B_{ij} \\ \tau(t_i) - \tau(t_\star) \leq B_{i\star} \\ \tau(t_\star) - \tau(t_i) \leq B_{\star i} \\ \tau(t_i) - \tau(t_0) \leq B_{i0} \\ \tau(t_0) - \tau(t_i) \leq B_{0i} \\ \tau(t_i) - \tau(t_x) \leq B_{ix} \\ \tau(t_y) - \tau(t_j) \leq B_{yj} \\ \tau(t_x) - \tau(t_\star) \leq B_{x\star} \\ \tau(t_\star) - \tau(t_x) \leq B_{\star x} \\ \tau(t_y) - \tau(t_0) \leq B_{y0} \\ \tau(t_0) - \tau(t_x) \leq B_{0x} \\ \tau(t_x) - \tau(t_y) \leq B_{xy} \\ \tau(t_0) - \tau(t_\star) \leq B_{0\star} \\ \tau(t_\star) - \tau(t_0) \leq B_{\star 0} \\ \\ \forall\, t_i, t_j \in \ldots,\ \forall\, t_x, t_y \in \ldots \end{cases} \qquad D'_{t_0} = \begin{cases} \tau'(t_i) - \tau'(t_j) \leq B_{ij} \\ \tau(t_0) - \tau(t_\star) \leq B_{i\star} - (\tau'(t_i) - \tau'(t_\star)) \\ \tau(t_\star) - \tau(t_0) \leq B_{\star i} + \tau'(t_i) - \tau'(t_\star) \\ \tau'(t_i) - \tau'(t_\star) \leq B_{i0} \\ \tau'(t_\star) - \tau'(t_i) \leq B_{0i} \\ \tau(t_0) - \tau(t_\star) \leq B_{ix} - (\tau'(t_i) - \tau'(t_x)) \\ \tau_(t_\star) - \tau(t_0) \leq B_{xi} + \tau'(t_i) - \tau'(t_x) \\ \tau'(t_x) - \tau'(t_\star) \leq B_{x\star} \\ \tau'(t_\star) - \tau'(t_x) \leq B_{\star x} \\ \tau(t_\star) - \tau(t_0) \leq B_{y0} + \tau'(t_\star) - \tau'(t_y) \\ \tau(t_0) - \tau(t_\star) \leq B_{0x} - (\tau'(t_\star) - \tau'(t_0)) \\ \tau'(t_x) - \tau'(t_y) \leq B_{xy} \\ \tau(t_0) - \tau(t_\star) \leq B_{0\star} \\ \tau(t_\star) - \tau(t_0) \leq B_{\star 0} \\ \\ \forall\, t_i, t_j \in \ldots,\ \forall\, t_x, t_y \in \ldots \end{cases}$$

PTPN analysis: successor computation - time adv. (4/4)

- Reorder the equations in D'_{t_0}:

$$D'_{t_0} = \begin{cases} \tau'(t_i) - \tau'(t_j) \leq B_{ij} \\ \tau(t_0) - \tau(t_\star) \leq B_{i\star} - (\tau'(t_i) - \tau'(t_\star)) \\ \tau(t_\star) - \tau(t_0) \leq B_{\star i} + \tau'(t_i) - \tau'(t_\star) \\ \tau'(t_i) - \tau'(t_\star) \leq B_{i0} \\ \tau'(t_\star) - \tau'(t_i) \leq B_{0i} \\ \tau(t_0) - \tau(t_\star) \leq B_{ix} - (\tau'(t_i) - \tau'(t_x)) \\ \tau_(t_\star) - \tau(t_0) \leq B_{xi} + \tau'(t_i) - \tau'(t_x) \\ \tau'(t_x) - \tau'(t_\star) \leq B_{x\star} \\ \tau'(t_\star) - \tau'(t_x) \leq B_{\star x} \\ \tau(t_\star) - \tau(t_0) \leq B_{y0} + \tau'(t_\star) - \tau'(t_y) \\ \tau(t_0) - \tau(t_\star) \leq B_{0x} - (\tau'(t_\star) - \tau'(t_0)) \\ \tau'(t_x) - \tau'(t_y) \leq B_{xy} \\ \tau(t_0) - \tau(t_\star) \leq B_{0\star} \\ \tau(t_\star) - \tau(t_0) \leq B_{\star 0} \\ \\ \forall\, t_i, t_j \in \ldots,\ \forall\, t_x, t_y \in \ldots \end{cases} \quad D'_{t_0} = \begin{cases} \tau'(t_i) - \tau'(t_j) \leq B_{ij} \\ \tau'(t_x) - \tau'(t_y) \leq B_{xy} \\ \tau'(t_x) - \tau'(t_\star) \leq B_{x\star} \\ \tau'(t_\star) - \tau'(t_x) \leq B_{\star x} \\ \tau'(t_i) - \tau'(t_\star) \leq B_{i0} \\ \tau'(t_\star) - \tau'(t_i) \leq B_{0i} \\ \\ \tau(t_0) - \tau(t_\star) \leq B_{0\star} \\ \tau(t_0) - \tau(t_\star) \leq B_{i\star} - (\tau'(t_i) - \tau'(t_\star)) \\ \tau(t_0) - \tau(t_\star) \leq B_{ix} - (\tau'(t_i) - \tau'(t_x)) \\ \tau(t_0) - \tau(t_\star) \leq B_{0x} - (\tau'(t_\star) - \tau'(t_0)) \\ \\ \tau(t_\star) - \tau(t_0) \leq B_{\star 0} \\ \tau(t_\star) - \tau(t_0) \leq B_{\star i} + \tau'(t_i) - \tau'(t_\star) \\ \tau(t_\star) - \tau(t_0) \leq B_{xi} + \tau'(t_i) - \tau'(t_x) \\ \tau(t_\star) - \tau(t_0) \leq B_{y0} + \tau'(t_\star) - \tau'(t_y) \\ \\ \forall\, t_i, t_j \in \ldots,\ \forall\, t_x, t_y \in \ldots \end{cases}$$

PTPN analysis: successor computation - projection

- Eliminate the time to fire of the fired transition t_0 through a projection:

$$
D''_{t_0} = \begin{cases}
\tau'(t_i) - \tau'(t_j) \leq C_{ij} = B_{ij} \\
\tau'(t_x) - \tau'(t_y) \leq C_{xy} = B_{xy} \\
\tau'(t_x) - \tau'(t_\star) \leq C_{x\star} = B_{x\star} \\
\tau'(t_\star) - \tau'(t_x) \leq C_{\star x} = B_{\star x} \\
\tau'(t_i) - \tau'(t_\star) \leq C_{i\star} = B_{i0} \\
\tau'(t_\star) - \tau'(t_i) \leq C_{\star i} = B_{0i} \\
\tau'(t_i) - \tau'(t_x) \leq C_{ix} = B_{ix} + B_{\star 0} \\
\tau'(t_x) - \tau'(t_i) \leq C_{xi} = B_{xi} + B_{0\star} \\
\\
(\tau'(t_i) - \tau'(t_x)) + (\tau'(t_y) - \tau'(t_j)) \leq C_{ix} + C_{yj} - c_{ixyj} & \text{with } c_{ixyj} = B_{\star 0} + B_{0\star} \\
(\tau'(t_\star) - \tau'(t_x)) + (\tau'(t_y) - \tau'(t_j)) \leq C_{\star x} + C_{yj} - c_{\star xyj} & \text{with } c_{\star xyj} = B_{0\star} + B_{\star x} - B_{0x} \\
(\tau'(t_i) - \tau'(t_\star)) + (\tau'(t_y) - \tau'(t_j)) \leq C_{i\star} + C_{yj} - c_{i\star yj} & \text{with } c_{i\star yj} = B_{i0} + B_{0\star} - B_{i\star} \\
(\tau'(t_i) - \tau'(t_x)) + (\tau'(t_\star) - \tau'(t_j)) \leq C_{ix} + C_{\star j} - c_{ix\star j} & \text{with } c_{ix\star j} = B_{\star 0} + B_{0j} - B_{\star j} \\
(\tau'(t_i) - \tau'(t_x)) + (\tau'(t_y) - \tau'(t_\star)) \leq C_{ix} + C_{y\star} - c_{ixy\star} & \text{with } c_{ixy\star} = B_{y\star} + +B_{\star 0} - B_{y0} \\
(\tau'(t_\star) - \tau'(t_x)) + (\tau'(t_\star) - \tau'(t_j)) \leq C_{\star x} + C_{\star j} - c_{\star x\star j} & \text{with } c_{\star x\star j} = B_{0j} + B_{\star x} - B_{\star j} - \\
(\tau'(t_i) - \tau'(t_\star)) + (\tau'(t_y) - \tau'(t_\star)) \leq C_{i\star} + C_{y\star} - c_{i\star y\star} & \text{with } c_{i\star y\star} = B_{i0} + B_{y\star} - B_{\star j} - \\
\\
\forall\, t_i, t_j \in T_{\mathrm{pr}}(m) \cap T_{\mathrm{p}}(S') \setminus \{t_0\} \mid t_i \neq t_j,\ t_i, t_j \neq t_0,\ t_i, t_j \neq t_\star \\
\forall\, t_x, t_y \in T_{\mathrm{s}}(m) \cap T_{\mathrm{p}}(S') \mid t_x \neq t_y,\ t_x, t_y \neq t_0,\ t_x, t_y \neq t_\star
\end{cases}
$$

- If coefficients B are in normal form $\Rightarrow$ coefficients C are in normal form

PTPN analysis: successor computation - newly enabling

- Add the time to fire of each newly-enabled transition:

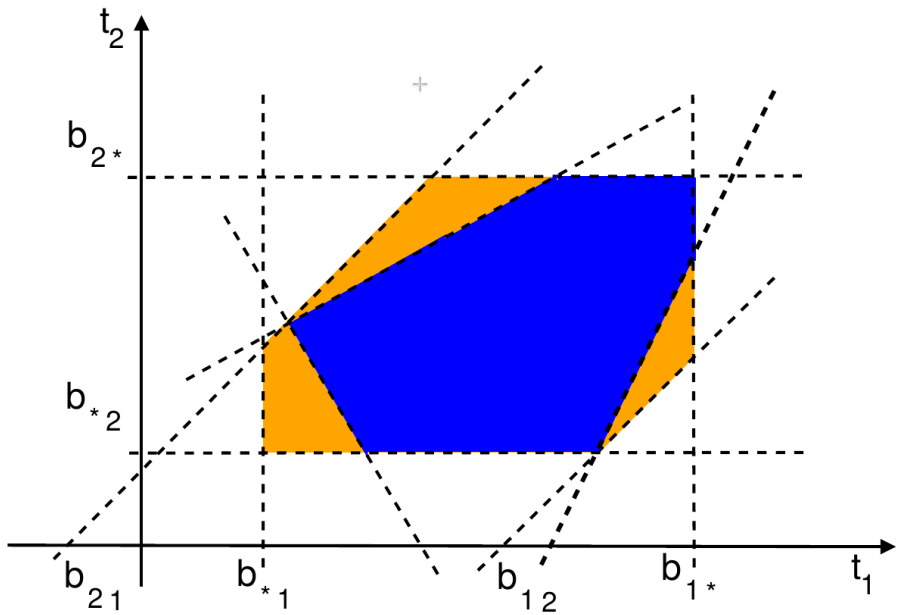
$$D' = \begin{cases} D''_{t_0} \\ \tau'(t_j) - \tau'(t_\star) \leq LFT(t_j) \\ \tau'(t_\star) - \tau'(t_j) \leq -EFT(t_j) \\ \forall\, t_j \in T_{\mathrm{n}}(S') \end{cases}$$

- The firing domain D' is **not in DBM form**!$\Rightarrow$
  $\Rightarrow$ The DBM form is **not closed** wrt the steps of successor computation!
- Repeated derivation of successor classes yields more complex firing domains, with exponential number of inequalities in the number of enabled transitions!

PTPN analysis: successor approximation

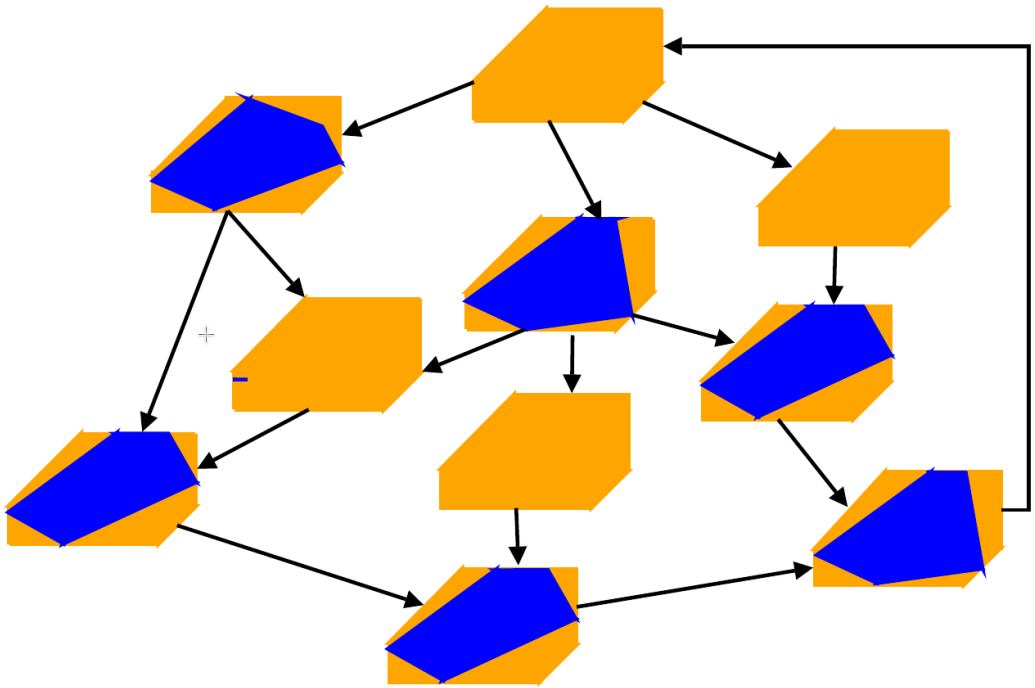
- D' is approximated by its **tightest embedding DBM** $\bar{D}'$:

$$\bar{D}' = \begin{cases} \tau'(t_i) - \tau'(t_j) \leq C_{ij} \\ \tau'(t_x) - \tau'(t_y) \leq C_{xy} \\ \tau'(t_x) - \tau'(t_\star) \leq C_{x\star} \\ \tau'(t_\star) - \tau'(t_x) \leq C_{\star x} \\ \tau'(t_i) - \tau'(t_\star) \leq C_{i\star} \\ \tau'(t_\star) - \tau'(t_i) \leq C_{\star i} \\ \tau'(t_i) - \tau'(t_x) \leq C_{ix} \\ \tau'(t_x) - \tau'(t_i) \leq C_{xi} \\ \\ \tau'(t_n) - \tau'(t_\star) \leq LFT(t_j) \\ \tau'(t_\star) - \tau'(t_n) \leq -EFT(t_j) \\ \\ \forall\, t_i, t_j \in T_{\mathrm{pr}}(m) \cap T_{\mathrm{p}}(S') \setminus \{t_0\} \mid t_i \neq t_j,\ t_i, t_j \neq t_0,\ t_i, t_j \neq t_\star \\ \forall\, t_x, t_y \in T_{\mathrm{s}}(m) \cap T_{\mathrm{p}}(S') \mid t_x \neq t_y,\ t_x, t_y \neq t_0,\ t_x, t_y \neq t_\star \\ \forall\, t_n \in T_{\mathrm{n}}(S') \end{cases}$$



PTPN analysis: state space enumeration

- Enumeration of the approximated reachability relation between state classes yields the **approximated state class graph**



Domain of timings of a symbolic run (1/2)

- Consider a symbolic run $\rho = S_0 \overset{t^0_{f(0)}}{\rightarrow} S_1 \overset{t^1_{f(1)}}{\rightarrow} S_2 \ldots S_{N-1} \overset{t^{N-1}_{f(N-1)}}{\rightarrow} S_N$
  - $S_0 = \langle m_0, D_0 \rangle$ is the initial state class
  - $S_n = \langle m_n, D_n \rangle$ is the n-th state class visited by $\rho \; \forall \, n \in \{1, \ldots, N\}$
  - $f(n)$ is the index of the transition fired in state class $S_n \; \forall \, n \in \{0, 1, \ldots, N-1\}$
  - t_i^k is a transition newly enabled in state class S_k
    (it is denoted as t_i^k in any class S_h such that it is persistent from S_k to S_h)
  - $l(t_i^k)$ is the index of the last class in which t_i^k is persistent
  - $c_n(t_i)$ is 1 or 0 depending on whether t_i is progressng or suspended in S_n, resp.
  - σ_n is the sojourn time in $S_n \; \forall \, n \in \{1, \ldots, N\}$
- The domain of timings of ρ satisfies the following constraints:

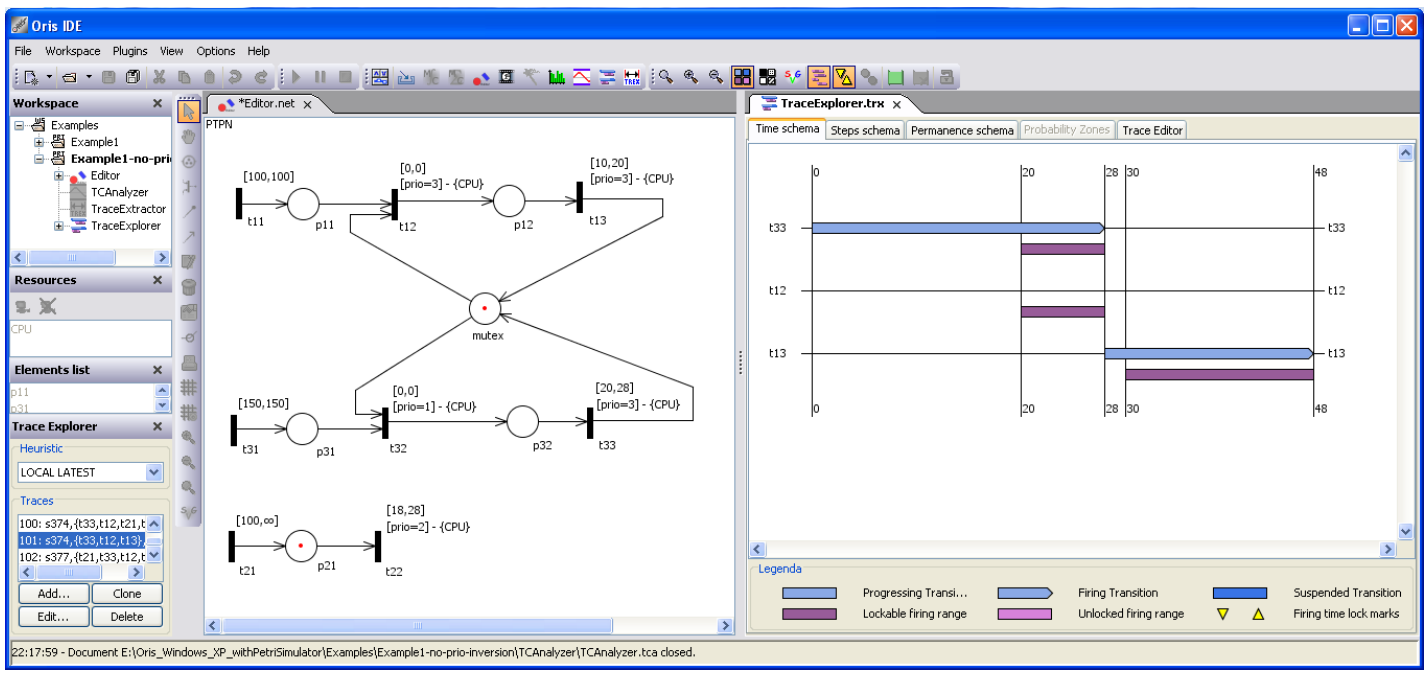
$$
D_\rho = \begin{cases}
-b^k_{\star i} \leq \sum_{n=k}^{l(t_i^k)} c_n(t_i)\sigma_n \leq b^k_{i\star} & \forall \; t_i^k \text{ fired along } \rho \\
\sum_{n=k}^{l(t_w^k)} c_n(t_w)\sigma_n \leq b^k_{w\star} & \forall \; t_w^k \text{ enabled but not fired along } \rho \\
\sum_{n=0}^{l(t_j^0)} c_n(t_j)\sigma_n - \sum_{n=0}^{l(t_z^0)} c_n(t_z)\sigma_n \leq b^0_{ij} & \forall \; t_j^0, t_z^0 \text{ enabled in } S_0 \text{ and fired along } \rho \\
-b^k_{i\star} \leq \sum_{n=0}^{l(t_w^0)} c_n(t_w)\sigma_n - \sum_{n=0}^{l(t_i^0)} c_n(t_i)\sigma_n \leq b^0_{wi} & \forall \; t_i^0 \text{ enabled in } S_0 \text{ and fired along } \rho, \; \forall \; t_w^k \text{ enabled but not fired along } \rho
\end{cases}
$$

Domain of timings of a symbolic run (2/2)

- If D_ρ does not admit solution $\Rightarrow$ ρ is a **false behavior**
- If D_ρ admits solution $\Rightarrow$ any min/max bound on the time elapsed between any two (not necessarily consecutive) transition firings can be derived by solving a linear programming problem (e.g., through the simplex method) so as to **clean up** false behaviors introduced by the approximation
  - Compute the minimum time spent in any feasible execution of ρ:
$$\min_{D_\rho} \left\{ \sum_{n=0}^{N-1} \sigma_n \right\}$$
  - Compute the minimum time spent in any feasible execution of ρ:
$$\max_{D_\rho} \left\{ \sum_{n=0}^{N-1} \sigma_n \right\}$$

ORIS 1: a tool for modeling and analysis of TPNs and PTPNs

- Developed by the **Software Technologies lab** of the University of Florence
  - Home page: `https://stlab.dinfo.unifi.it/oris1.0`
- Main features
  - **Modeling**: graphical editing of TPNs and PTPNs
  - **State space analysis**: state space enumeration of TPNs and PTPNs
  - **Trace analysis**: evaluation of the tight timing profile of any symbolic run selected in the state space of a TPN and in the state space of a PTPN



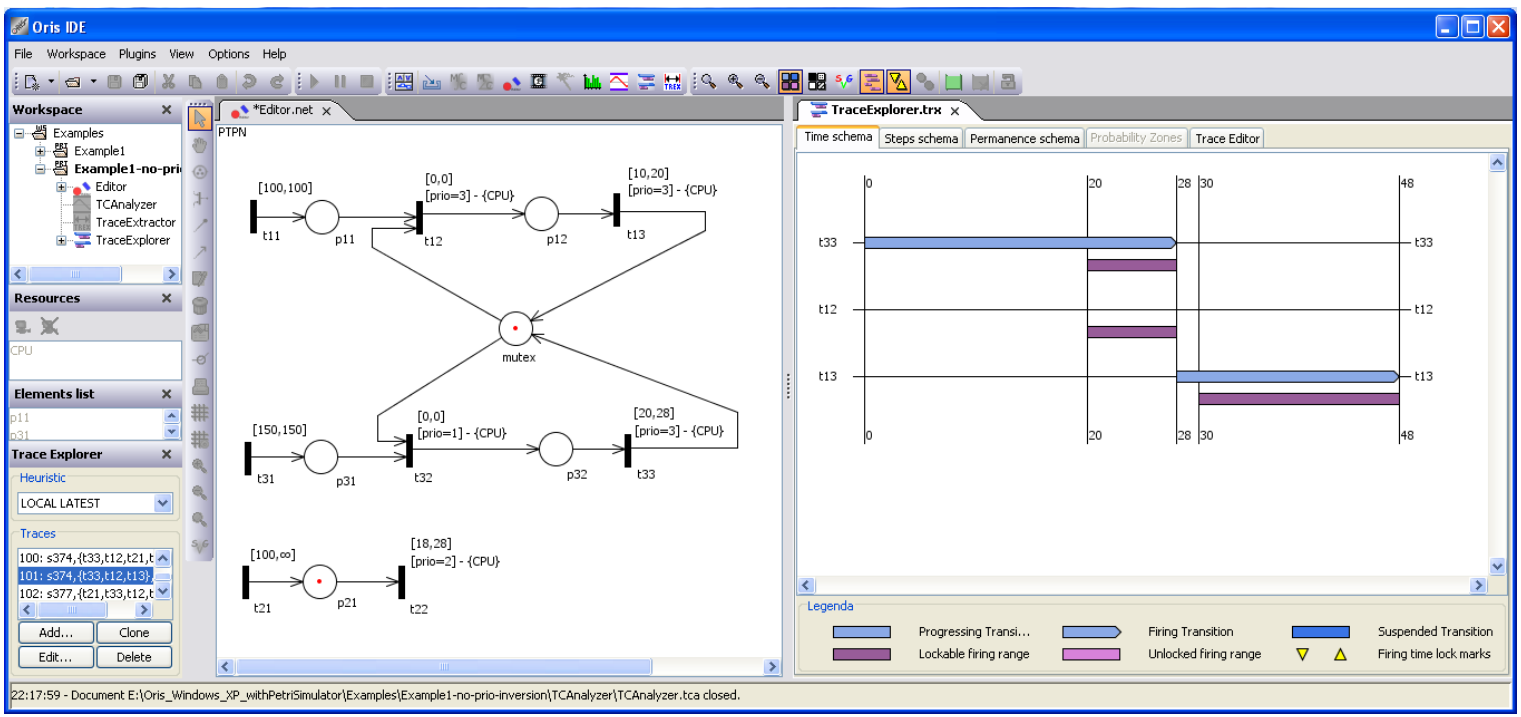
ORIS 1: how to install and use the tool

- How to install ORIS 1
  - Install Virtual Box (a free and open-source hosted hypervisor for virtualizing the x86 computing architecture): `https://www.virtualbox.org`
  - Download the ORIS 1 virtual machine: `https://stlab.dinfo.unifi.it/carnevali/misc/WinXP.ova`
  - Launch Virtual Box and open the ORIS 1 virtual machine
- How to use ORIS 1
  - Manual: `https://stlab.dinfo.unifi.it/carnevali/misc/Oris1.pdf`
  - Note: for PTPNs, the higher the priority number the higher the priority level
- A new release of the tool: ORIS 2
  - Home page: `http://www.oris-tool.org`
  - Support for the analysis of TPNs and Stochastic Time Petri Nets (STPNs)
  - Ongoing porting of PTPN features from ORIS 1 to ORIS 2

Using Preemptive Time Petri Nets in a V-Model SW life cycle

Modeling and analysis of real-time task-sets with PTPN theory

- Representation of task-sets running under **preemptive** scheduling policies
  - e.g., task-sets running under fixed-priority preemptive scheduling
- **Timed reachability**: can the task-set reach a specific logical condition?
  - e.g., verify sequencing correctness such as absence of priority inversion
- **Timeliness analysis**: which is the min/max time between specific events?
  - e.g., compute the best/worst case completion time of each task



Is this only a matter of verification?

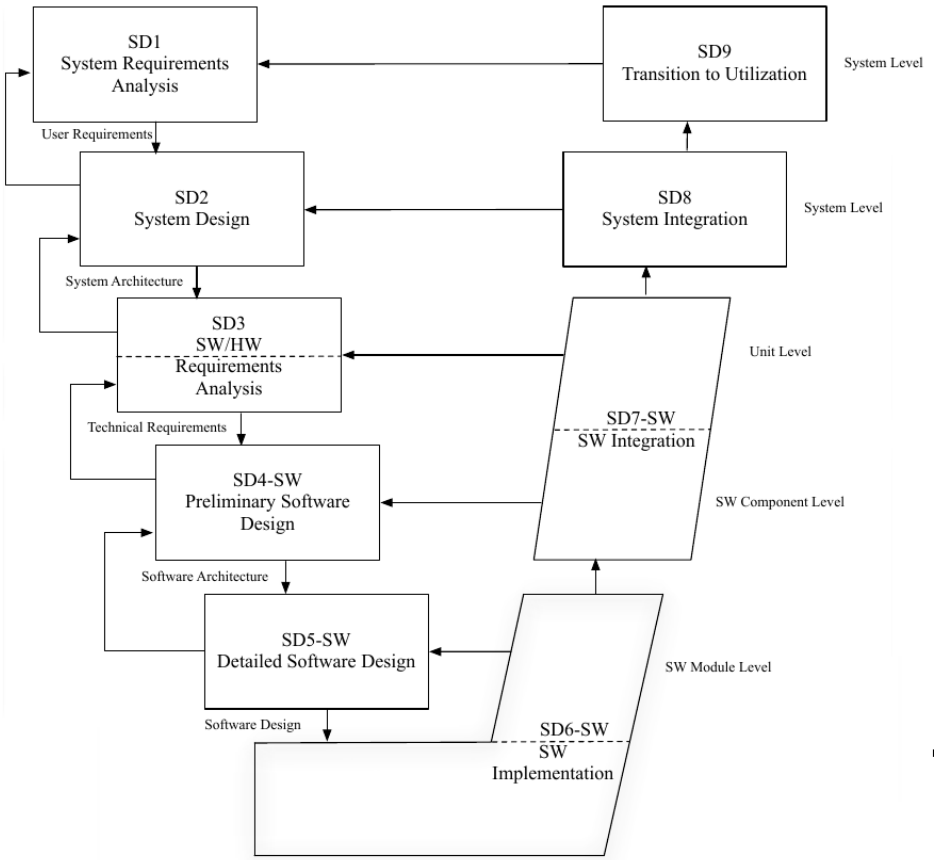
- PTPNs as formal core of a Model Driven Development (MDD) approach...
  - automated derivation of pTPN from timelines
  - automated code generation
  - execution time measurement of entry-point functions
  - test-case selection and execution
  - oracle verdict
- ...organized in a comprehensive software (SW) development life cycle:
  - referred to the practice of industrial development (V-Model)
  - referred to applicable regulatory standards (RTCA-178B)
  - formalized in terms of artifacts of the UML profile for MARTE (Modeling and Analysis of Real-Time and Embedded Systems)

L. Carnevali, L. Ridi, E. Vicario, "Putting preemptive Time Petri Nets to work in a V-Model SW life cycle", IEEE Transactions on Software Engineering, 2011.

I. Bicchierai, G. Bucci, L. Carnevali, E. Vicario, "Combining UML-MARTE and preemptive Time Petri Nets: An Industrial Case Study", IEEE Transactions on Industrial Informatics, 2013.

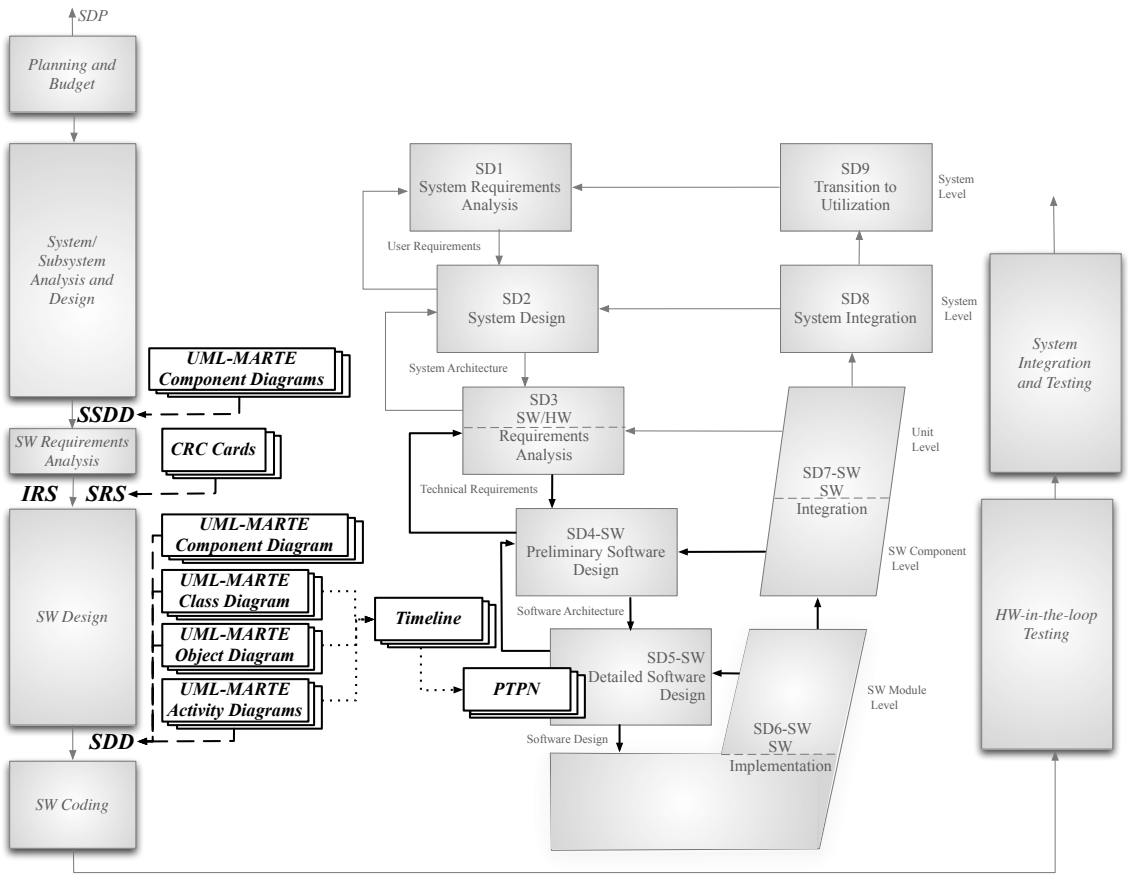
V-Model

- Reference SW development life cycle for safety critical systems
- Integration of *design* and *verification* (left/right)
- Decomposition from *System* to *Modules* (up/down)



V-Model tailored according to MIL-STD-498

- Integration of formal methods within the development life cycle requires SW engineering methods not to disrupt industrial practices
  - V-model development process supported by the pTPN theory
  - MIL-STD-498 documentation process supported by UML-MARTE

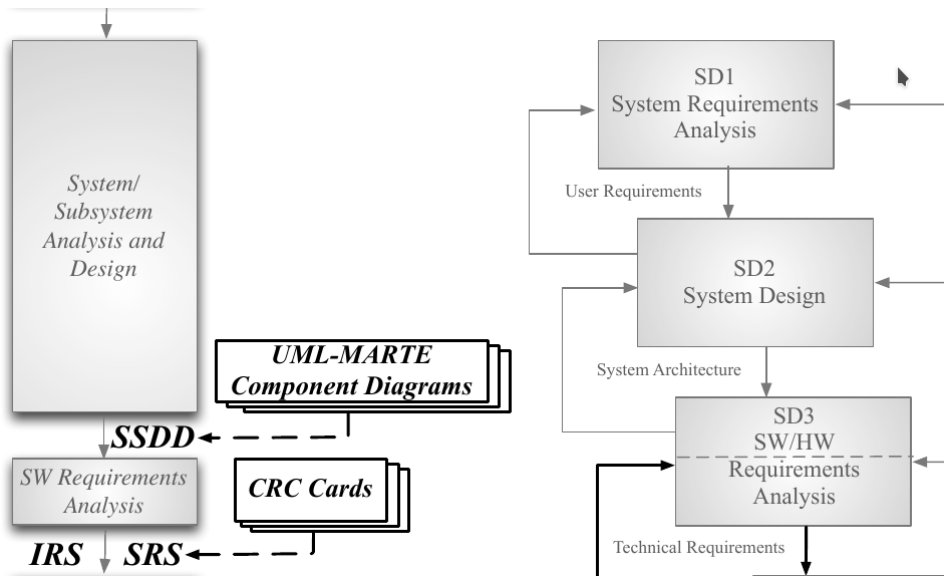


A running case study

- An **electro-optical system** of a **military vehicle**
  - Developed by Selex-Galileo of the FinMeccanica Group (now Leonardo)
  - Guarantees battlefield advantage using visual, infra-red and thermal imaging, long-range target acquisition and illumination, precise aiming
- The system consists of 3 units
  - An **Optical Unit (OU)** made of sensors, cameras, and servo-motors
  - An **Electronic Unit (EU)** responsible for sensor control and image processing
  - A **System Monitoring Unit (SMU)** managing the entire system
- EU plays the role of a bridge in the communication between SMU and OU
  - Forwards the commands periodically sent by SMU to OU
  - Sending the corresponding OU replies to SMU
- EU processes images acquired by OU and sends obtained results to SMU
- Experimentation of the formal methodology in the development of the EU in a one-year-long research collaboration with Selex-Galileo (now Leonardo) supported by the SW-Initiative of the FinMeccanica Group

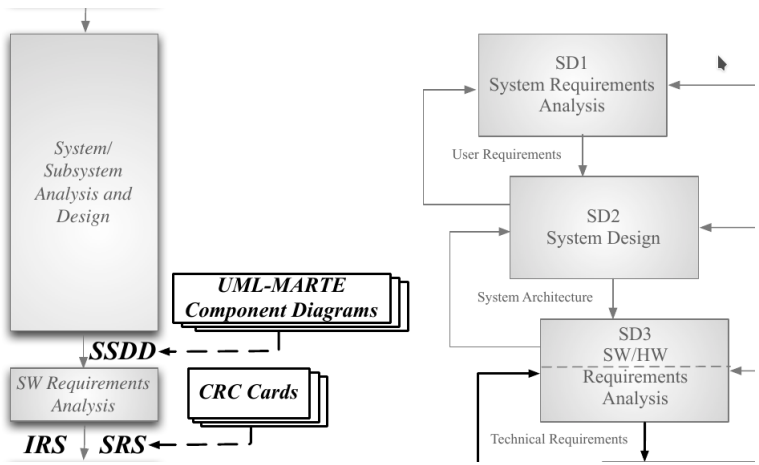
V-Model: SD1, SD2, SD3

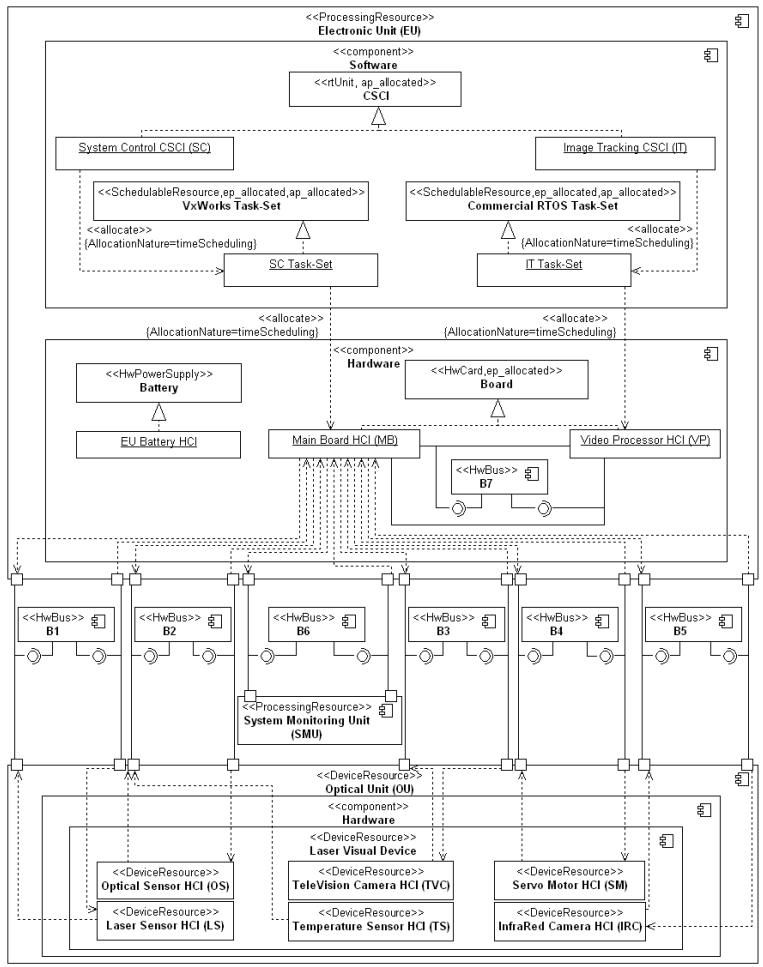
- SD1 (System Requirements Analysis) - system scope
  - Determines high level user requirements of the system
- SD2 (System Design) - System scope
  - Identifies system units and allocates user requirements to them
- SD3 (SW/HW Requirements Analysis) - from system to unit scope
  - Decomposes each unit into HW Configuration Items (HCIs), Computer SW Configuration Items (CSCIs), and Firmware Configuration Items (FCIs)



MIL-STD-498: System/Subsystem Analysis and Design

- Integrates SD1, SD2, and the first part of SD3 (system scope)
- Produces the System/Subsystem Design Description (SSDD)
  - Here in the form of a UML-MARTE component diagram of system units





MIL-STD-498: SW Requirements Analysis

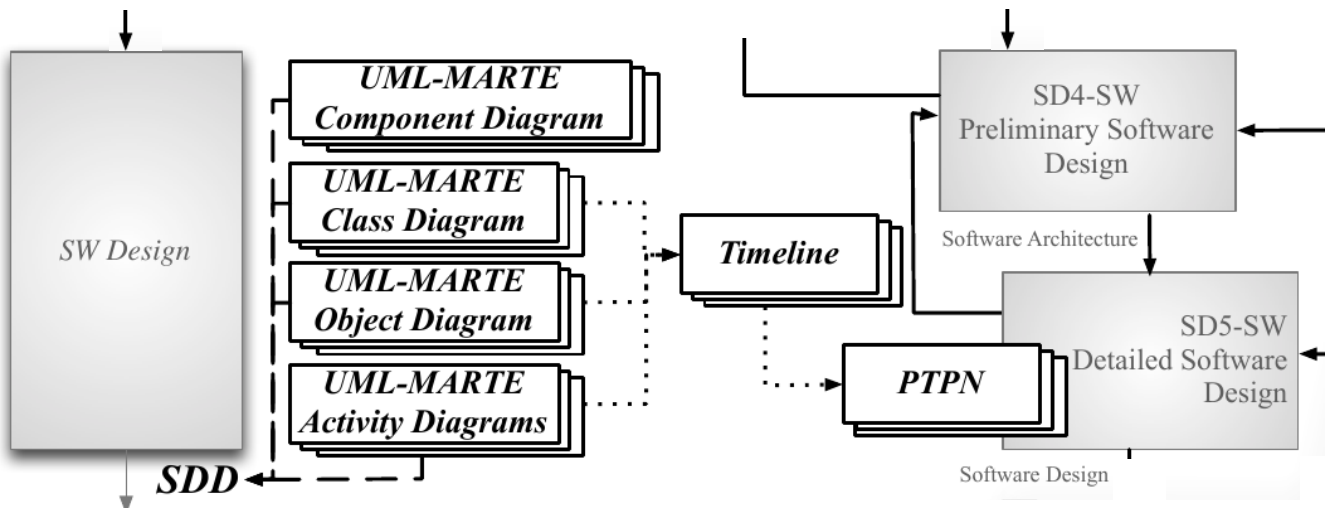
- Corresponds to the second part of SD3 (CSCI scope)
- Produces the Software Requirements Specification (SRS) of each CSCI and the Interface Requirements Specification (IRS) of each unit interface
  - Here SRS in a form similar to Class Responsibility Collaboration (CRC) cards
    - e.g., CRC card specifying capabilities of the System Control CSCI (left)
    - e.g., CRC card specifying sub-capabilities of SMU-OU_COMMANDS (right)

Capability	Description	Collaboration
Init	HW and SW initialization	-
LS-IRC_Power	Switching on/off LS and IRC	LS, IRC
TVC_Config	Management of the TVC configuration	TVC
SMU-OU_Cmd	Management of the messages exchanged by SMU and OU	-
SM_Comm	Communication with SM	SM
SMU_Comm	Communication with SMU	SMU
IT_Comm	Communication with IT	IT
IRC_Comm	Communication with IRC	IRC
TVC_Comm	Communication with TVC	TVC
LS_Comm	Communication with LS	LS

Sub-Capability	Description	Collaboration
SMU_Cmd	Management of the SMU commands and the OU replies	-
TVC_Cmd	Management of the TVC configuration parameters	-
HCIs_Trasm	Activation of the data transmission to IRC, TVC, and LS	-
LS-IRC_State	Management of the switched on/off state of IRC and LS	-
HCIs_Data	Processing of the HCIs data	-
SM_LocationData	Processing of the SM location data elaborated by IT	-
OperationModes	Management of Operation Modes	-

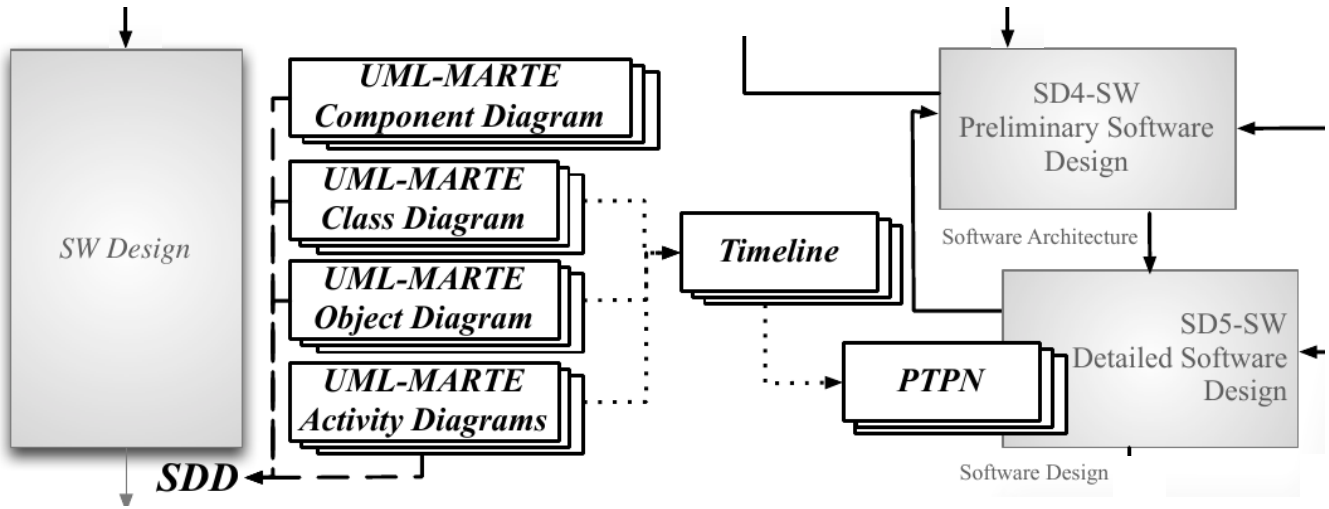
V-Model: SD4-SW, SD5-SW

- SD4-SW (Preliminary SW Design) - SW component scope
  - Defines the **SW architecture** of each CSCI as set of communicating tasks with assigned functional modules and prescribed release times and deadlines
- SD5-SW (Detailed SW Design) - SW module scope
  - Defines the **SW design** of each CSCI allocating resources and time requirements to each software module
  - Includes SD5.2-SW (Analysis of Resources and Time Requirements)



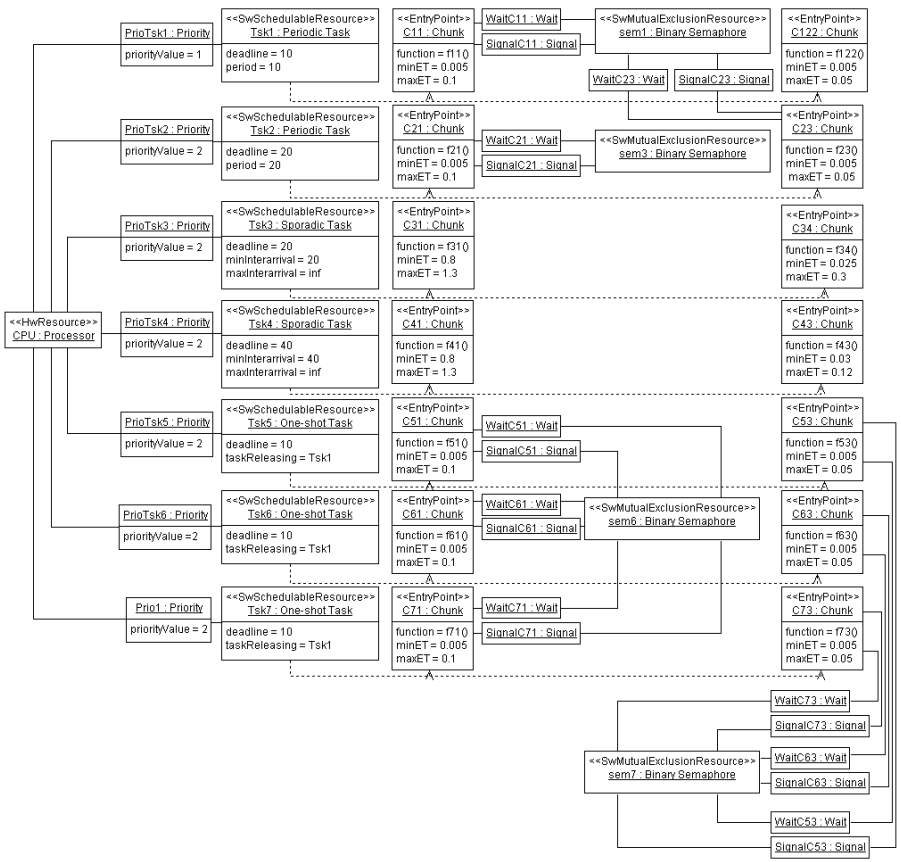
MIL-STD-498: SW Design

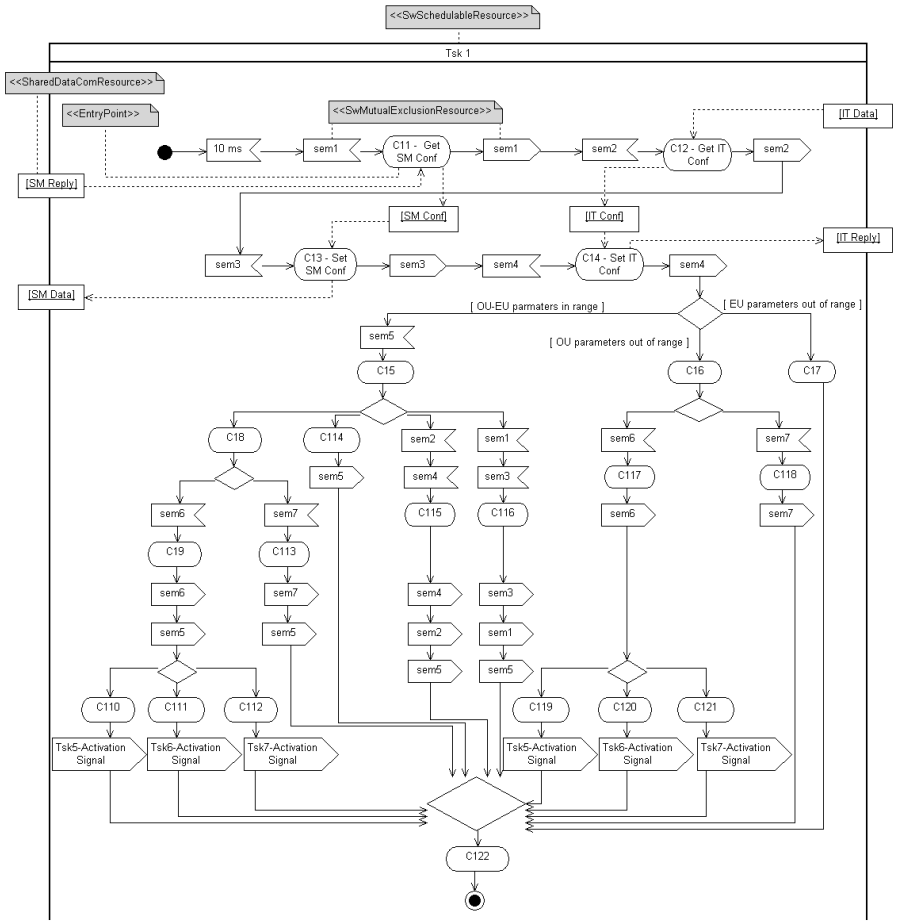
- Integrates SD4-SW and SD5-SW
- Produces the Software Design Descripton (SDD)



SW Design: semi-formal specification through UML-MARTE

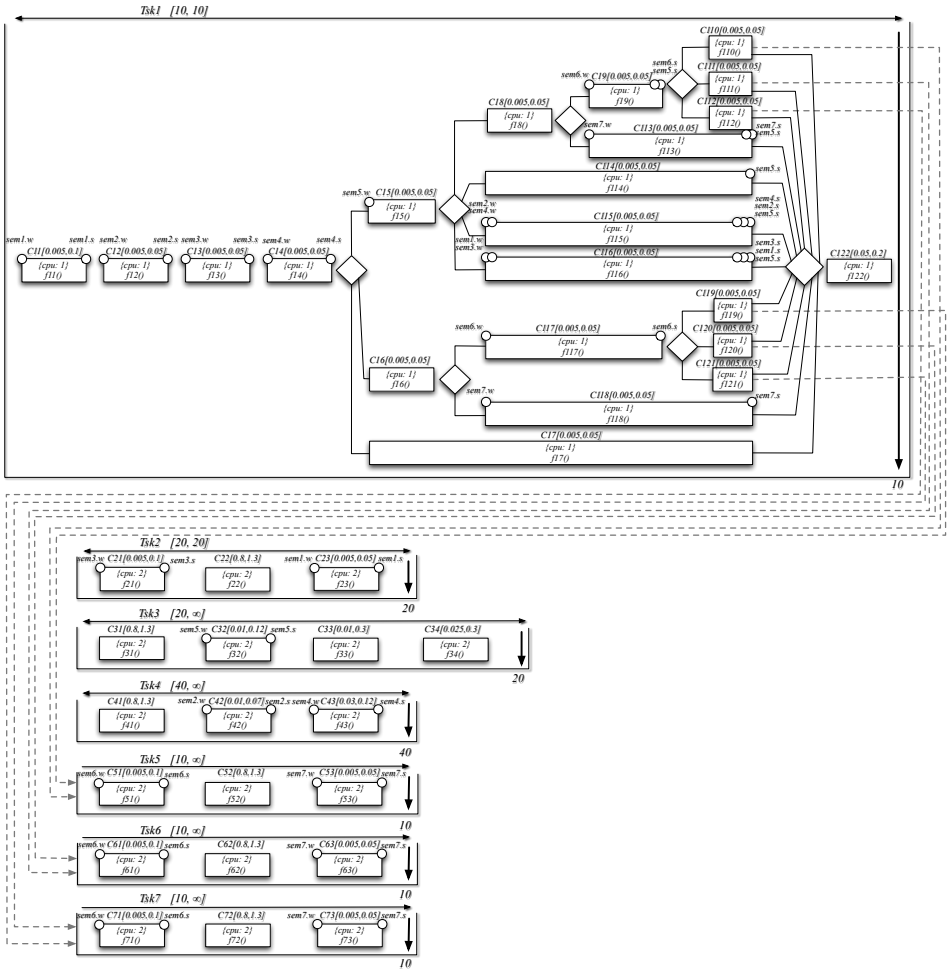
- Specification of non-functional requirements of each CSCI
  - Here in the form of a UML-MARTE object diagram for the whole task-set
- Specification of functional requirements of each CSCI
  - Here in the form of a UML-MARTE activity diagram for each task





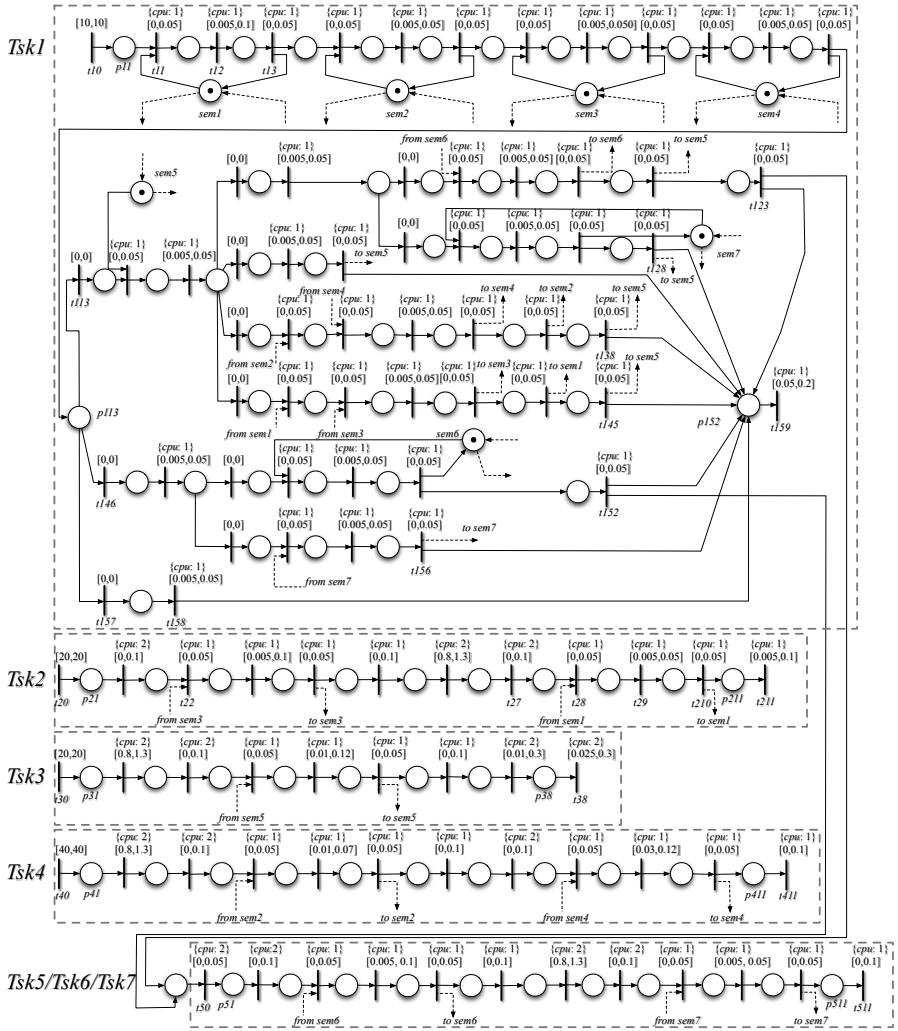
SW Design: semi-formal specification through timelines

- Can be (manually/automatically) derived from UML-MARTE diagrams
- Provide a synthetic and intuitive representation of a task-set
- Do not support automated verification of the model



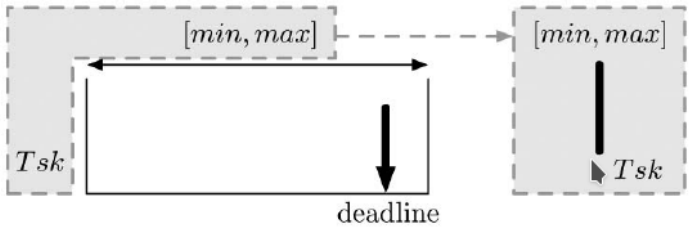
SW Design: formal specification through PTPNs

- Can be (manually/automatically) derived from timelines
- Support formal verification and subsequent development steps

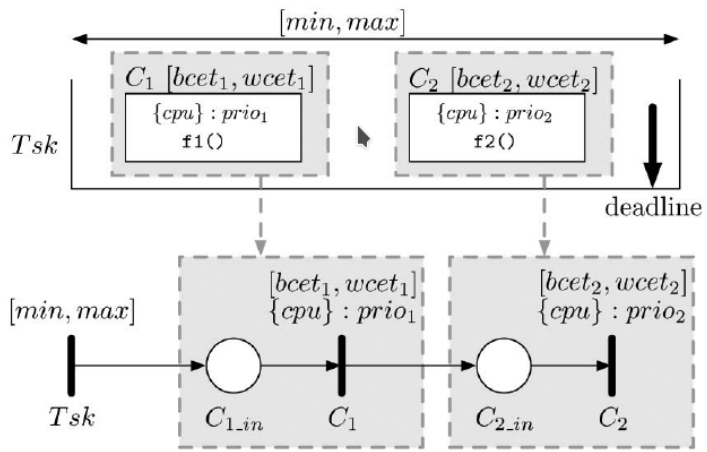


SW Design: derivation of PTPNs from timelines (1/3)

- Representation of task releases

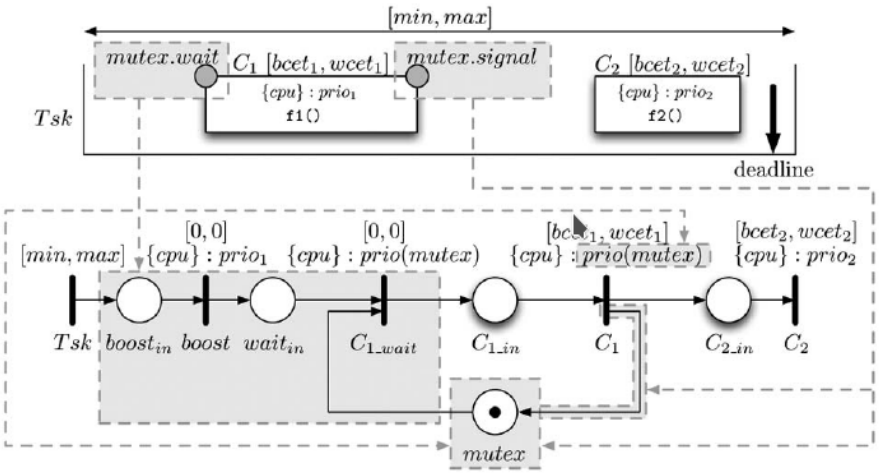


- Representation of tasks with multiple sequential chunks



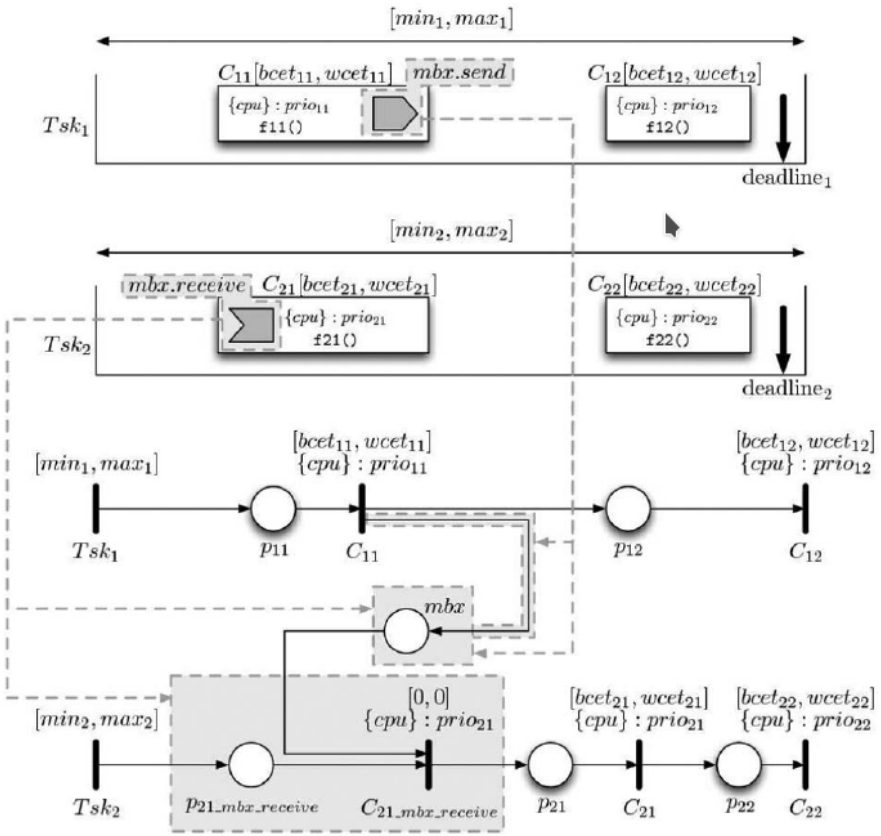
SW Design: derivation of PTPNs from timelines (2/3)

- Representation of tasks synchronized on a semaphore



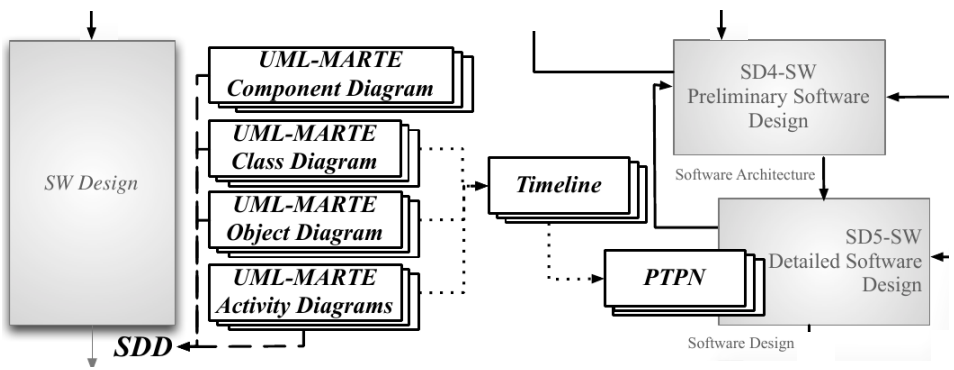
SW Design: derivation of PTPNs from timelines (3/3)

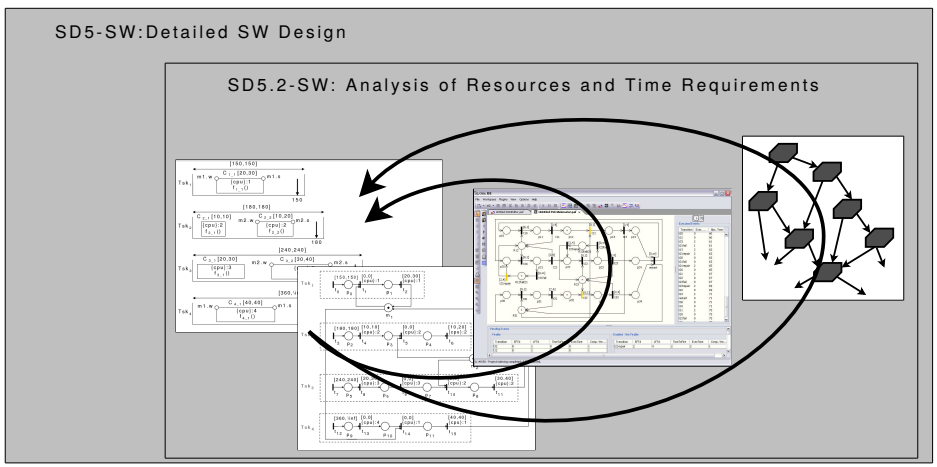
- Representation of tasks synchronized on a mailbox



SD5.2-SW: Analysis of Resources and Time Requirements

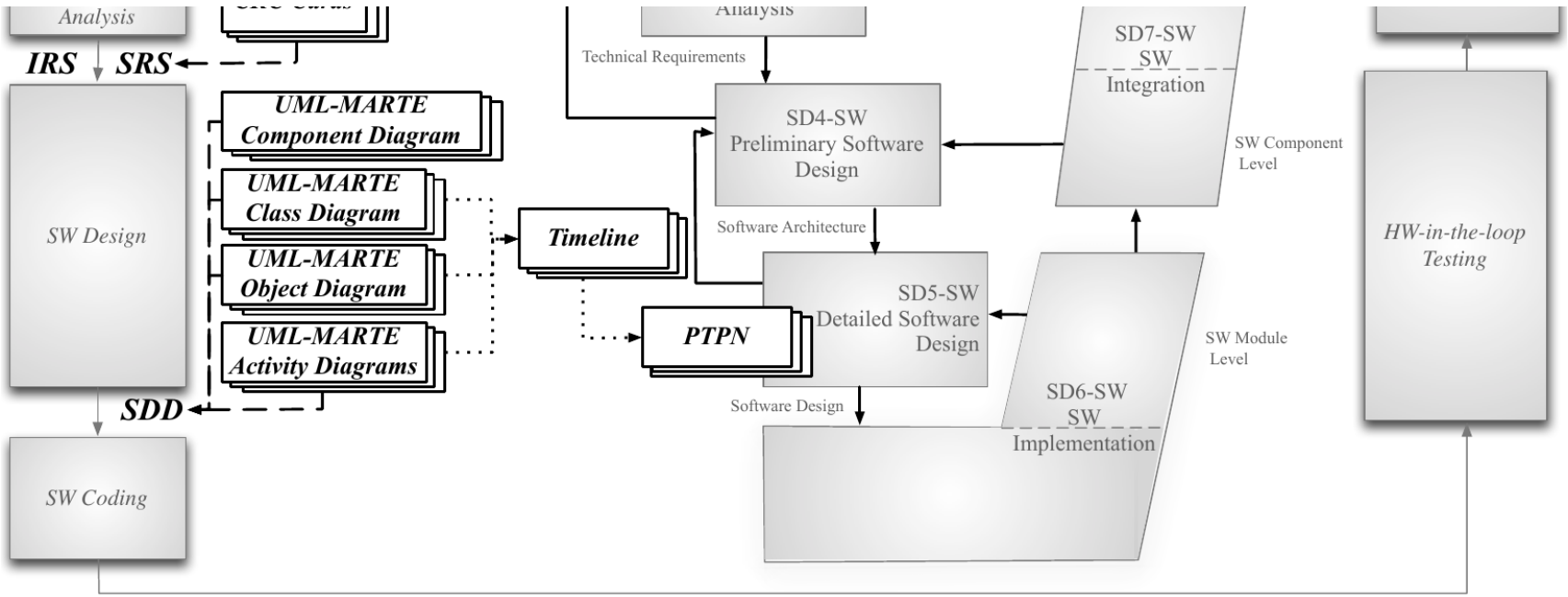
- Simulation of the PTPN model
  - Online interactive token game
  - Offline derivation of statistics
- State-space analysis of the PTPN model
  - Exhaustive state space coverage (unless the state space explodes)
  - Verification of sequencing and timeliness properties





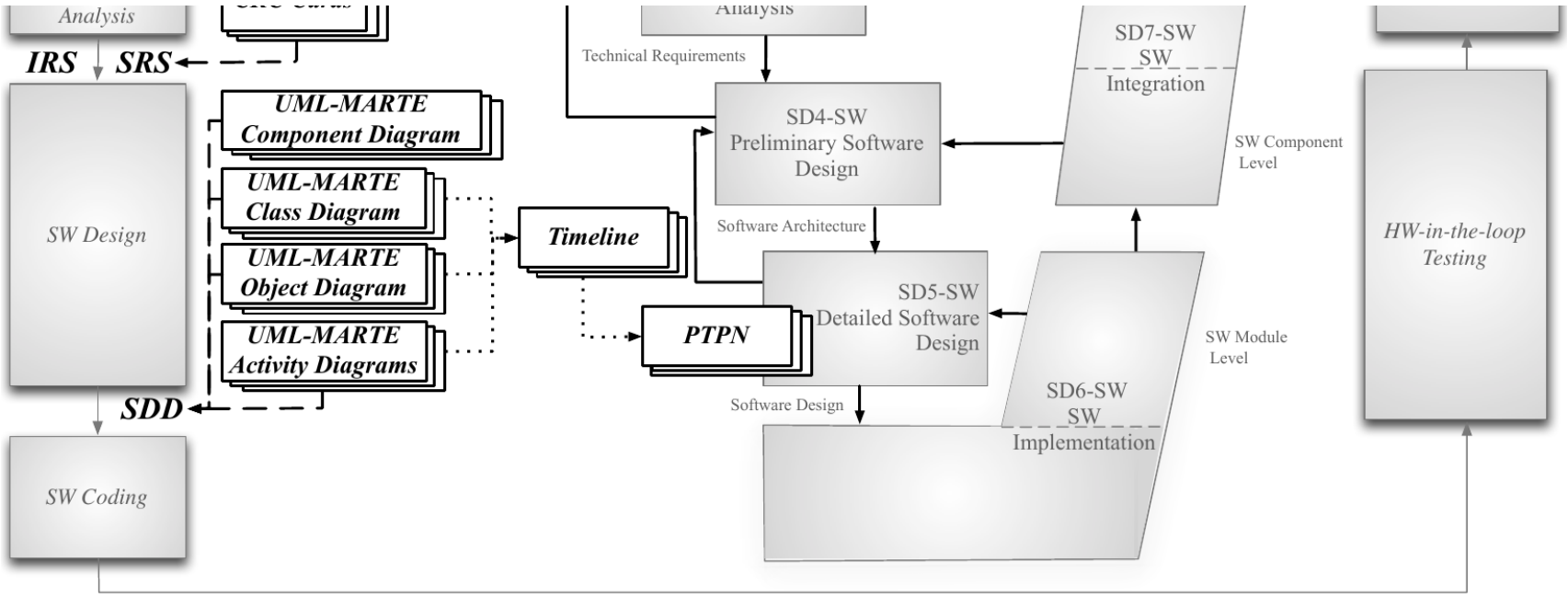
V-Model: SD6-SW

- SD6-SW: SW Implementation - SW module scope
  - Implementation of the task set architecture (executable architecture)
  - Unit testing (execution time profiling of low-level modules)
  - Implementation of entry-point functions (not in the scope of the approach)



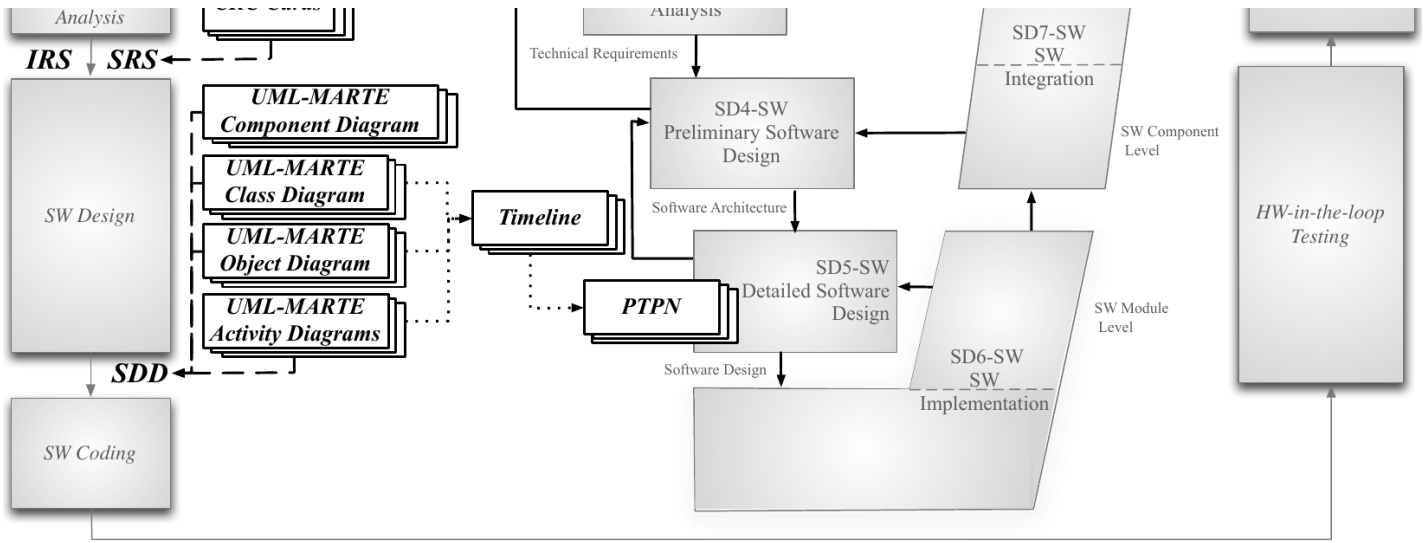
MIL-STD-498: SW Coding

- Corresponds to the first part of SD6-SW
- Produces the task-set architecture of each CSCI (**dynamic architecture**)
  - A skeleton of **control code** that is responsible of implementing **non-functional behavior** of tasks as well as invoking **functional code** of entry-point functions



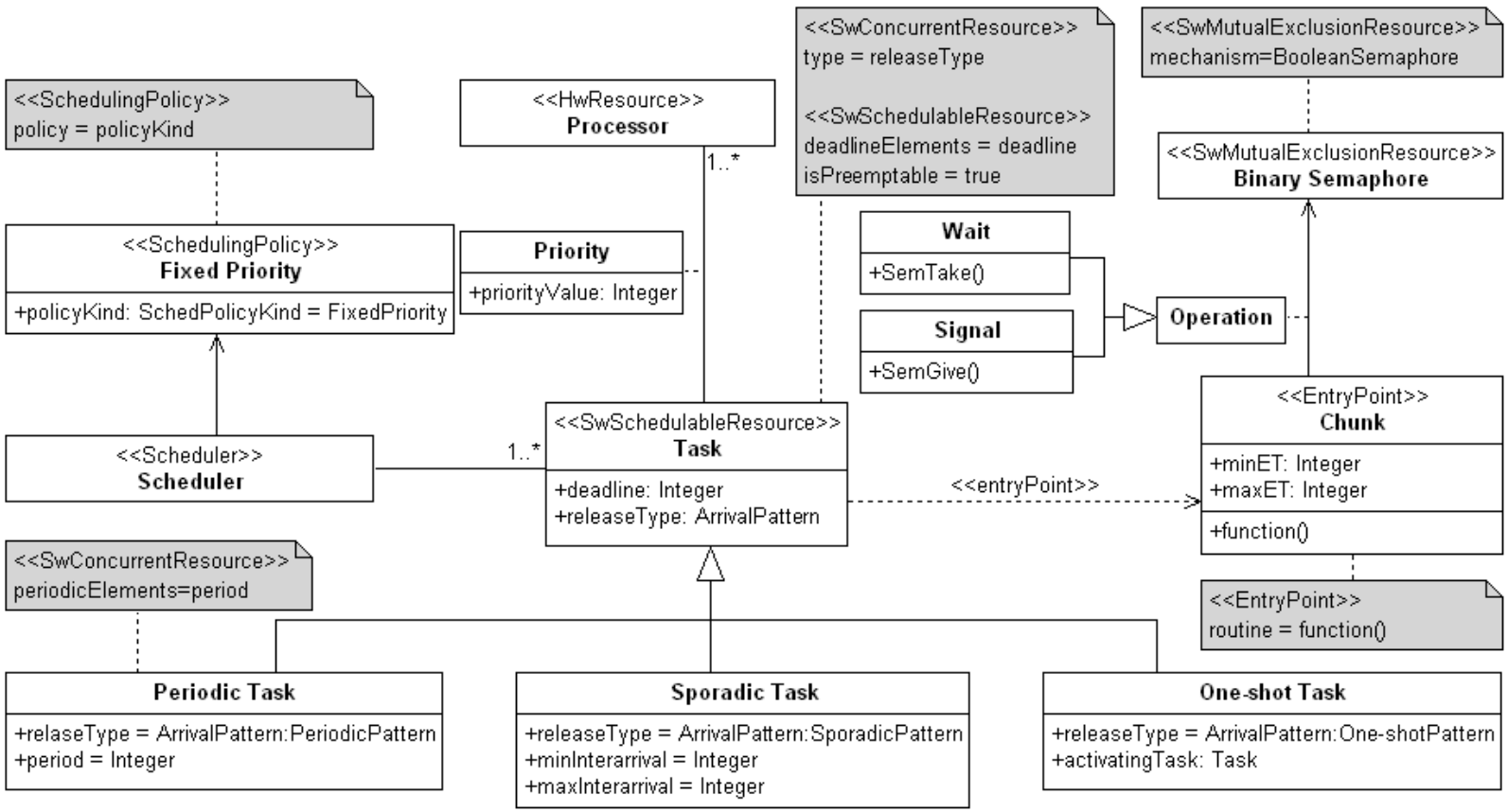
SW Coding: implementation of the task-set architecture (1/5)

- Responsibilities
  - Release task jobs according to their policies
  - Implement semaphore operations and priority handling
  - Sequence the invocation of entry point functions
- Structure
  - Reflects the timeline specification
  - Relies on conventional primitives of a real-time operating system
  - Experiments on c-code of single-processor applications (VxWorks, Linux RTAI)
  - Is this disciplined coding or Model Driven Development (MDD)?



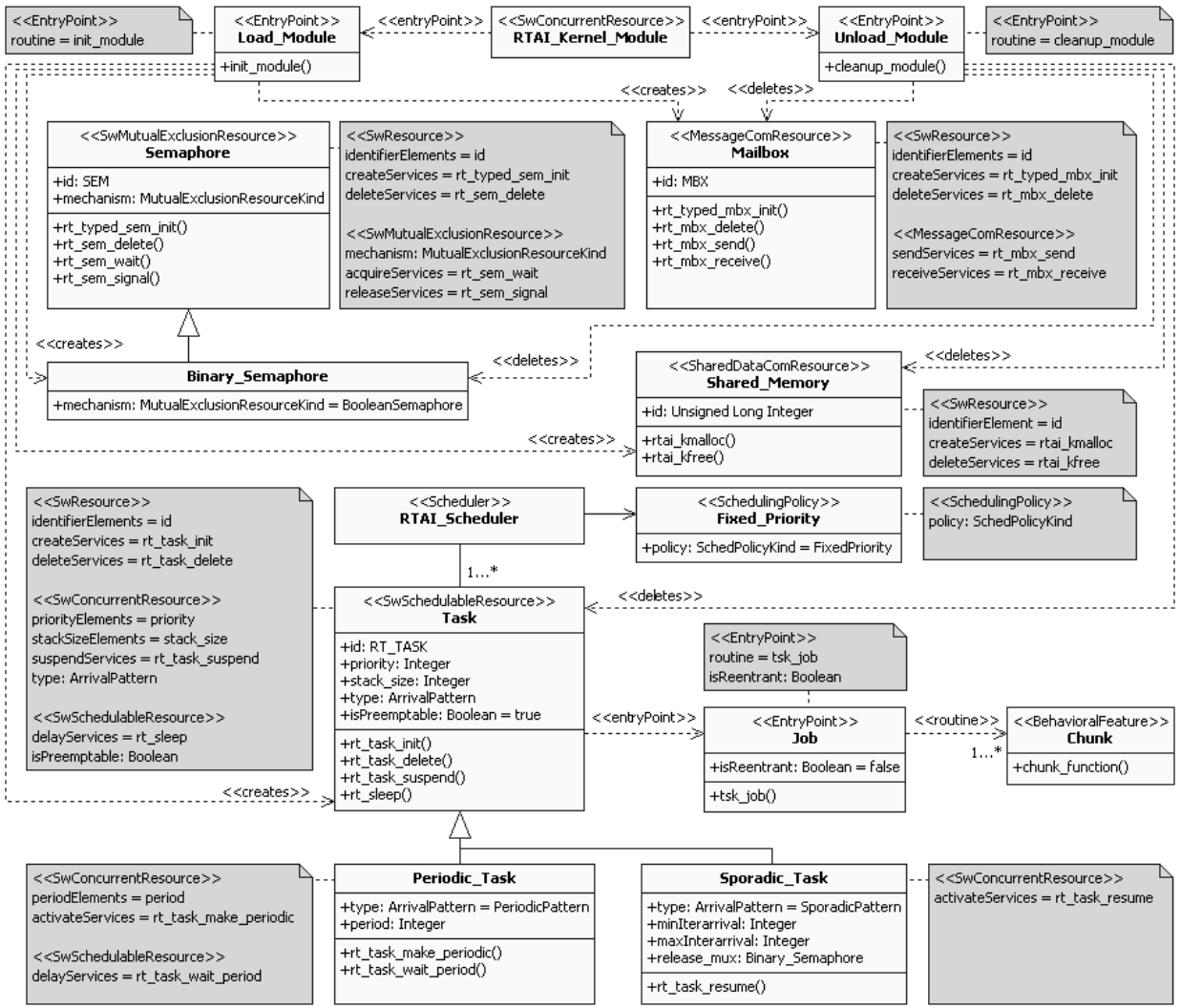
SW Coding: implementation of the task-set architecture (2/5)

- UML-MARTE class diagram of the implementation structure of the dynamic architecture of a CSCI running on **VxWorks 6.5**



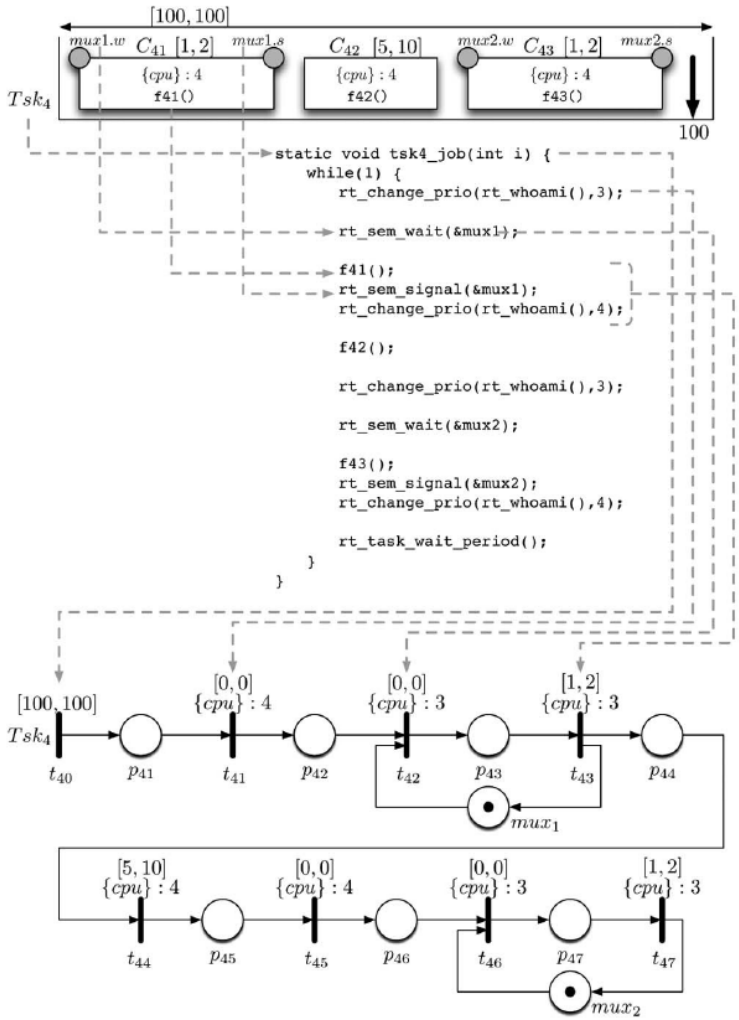
SW Coding: implementation of the dynamic architecture (3/5)

- UML-MARTE class diagram of the implementation structure of the dynamic architecture of a CSCI running on **Linux-RTAI 3.3**



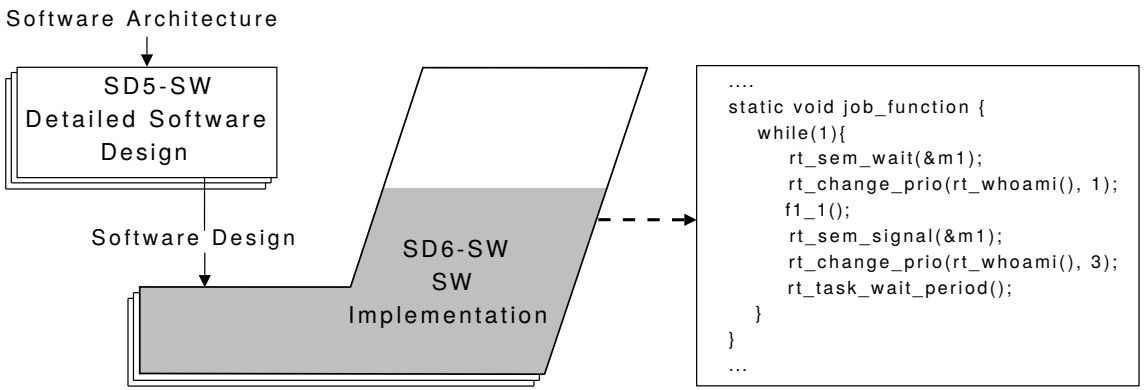
SW Coding: implementation of the dynamic architecture (4/5)

- Fragment of the dynamic architecture of a CSCI running on **Linux-RTAI 3.3**
  - The architecture of the code reflects the PTPN model
  - The developer maintains full control over the code



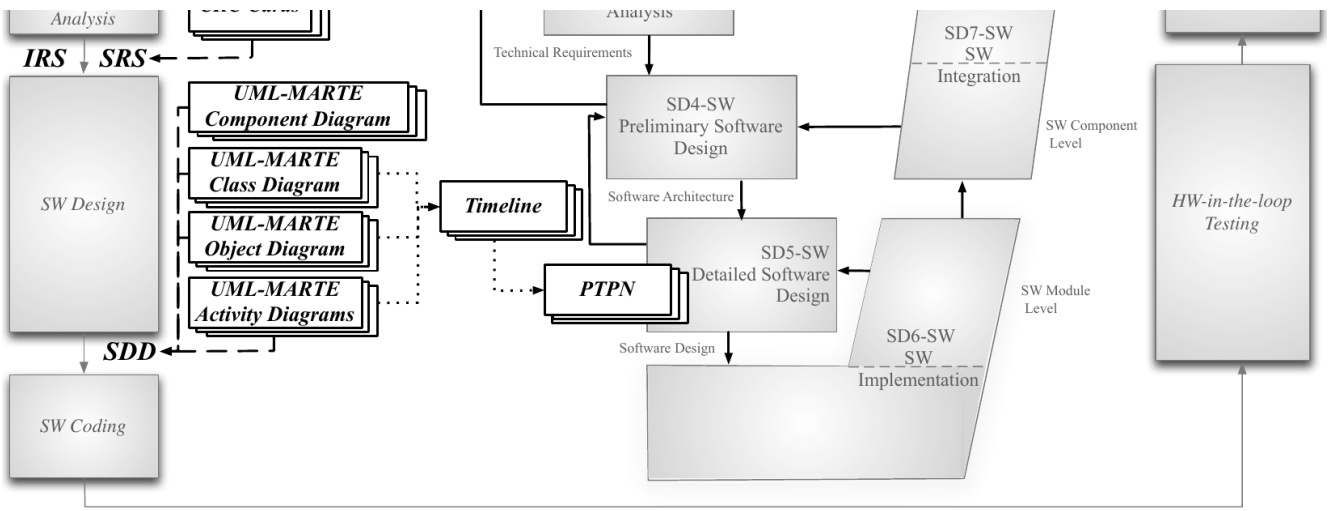
SW Coding: implementation of the dynamic architecture (5/5)

- Emulates the Execution Time (ET) of entry-point functions through a busy-sleep function (e.g., deterministic loop using cpu for a controlled ET)
- Comprises a kind of **executable architecture** (Unified Process)
- Provides a baseline for incremental integration of entry-point functions



MIL-STD-498: first part of HW-in-the-loop testing

- Corresponds to the second part of SD6-SW
- Performs **unit testing**, i.e., testing of each SW module (also referred to as low-level unit) in the dynamic architecture of each CSCI
  - Includes other SW modules emulated through busy-sleep functions
  - Embeds each SW module within the range of behaviors that are feasible for the specification ⇒ permits circumventing the known problem of **determinization of the implementation** with respect to the specification

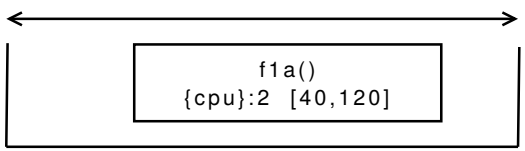


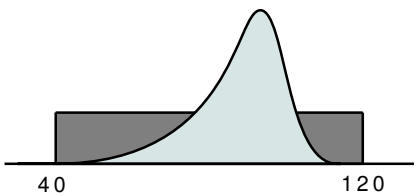
HW-in-the-loop testing: execution time profiling (1/6)

- The executable architecture can be automatically instrumented to produce **time-stamped logs** of the events corresponding to transitions in the PTPN
  - Allows reconstruction of the execution time of each entry-point function:

$$ET(t_i^n) = \sum_{k=n}^{K_{i,n}-1} c_{i,k} \cdot (\tau_{k+1} - \tau_k)$$

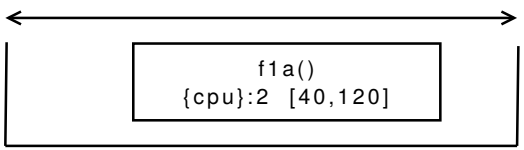
    - t_i^n is the **instance** of transition t_i newly-enabled in the n-th visited state
    - $c_{i,k}$ is 1 or 0 if t_i is progressing or suspended in the k-th visited state, resp.
    - $K_{i,n}$ is the index (in the trace) of the state reached through the firing of t_i^n
    - τ_k is the entrance time in the k-th visited state, i.e. the k-th timestamp
  - Identifies the **implementation behavior** with respect to the specification

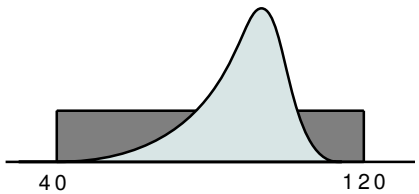




HW-in-the-loop testing: execution time profiling (2/6)

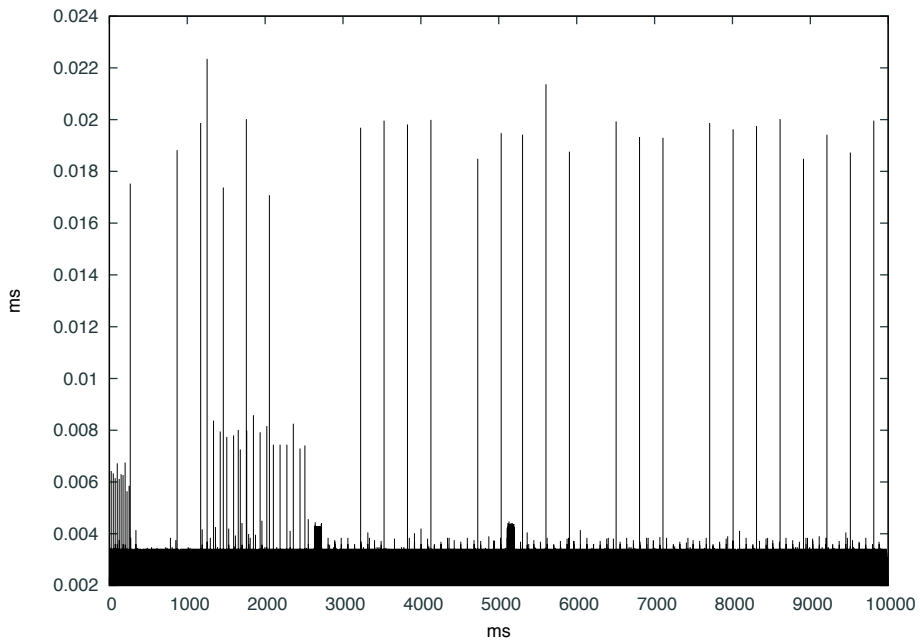
- This is a **measurement-based** approach to execution time analysis
- Runs in interrupted mode ⇒ accounts for context switches, preemption events, HW interrupts, SW interrupts, cache and pipeline effects
- What is the impact of the logging operation on measured execution times?

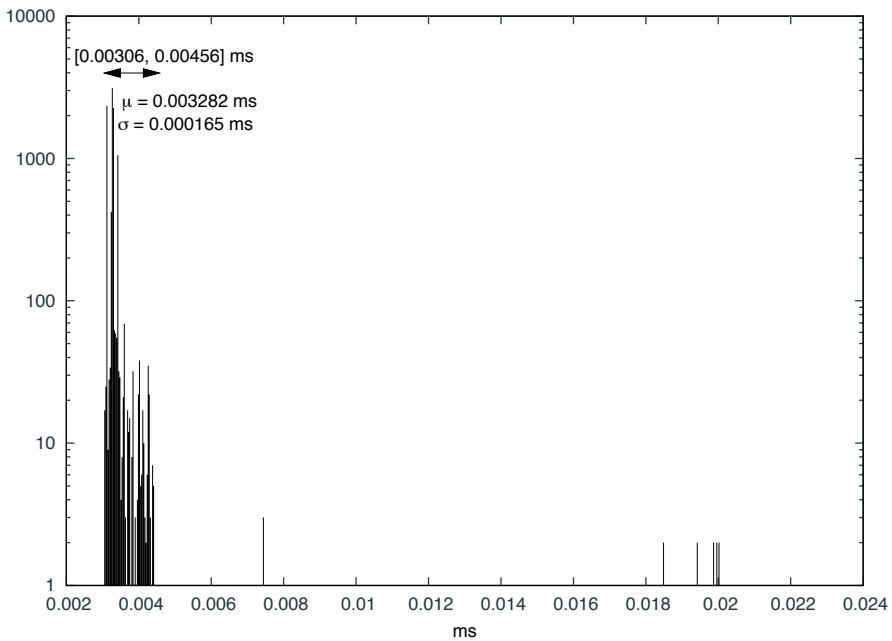




HW-in-the-loop testing: execution time profiling (3/6)

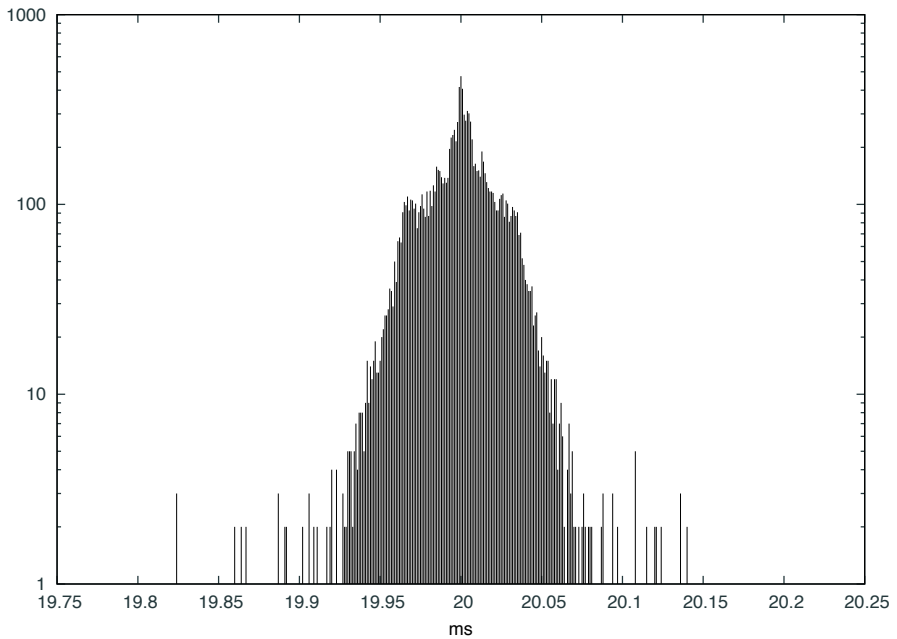
- Perform a large number of repetitions of the logging operation (10,000)
  - Sequence of observed execution times of the logging operation (left)
  - Histogram of observed execution times of the logging operation (right)
- Simple but coarse-grained approach (logging requires tens of microseconds)
  - Min-max intervals associated with temporal parameters were enlarged during iterative refinements of the dynamic architecture of each CSCI





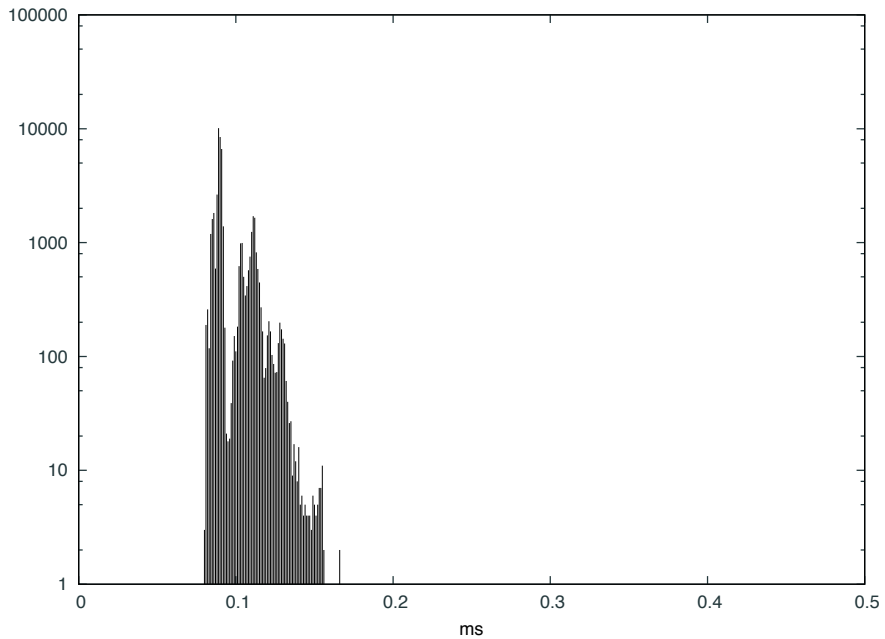
HW-in-the-loop testing: execution time profiling (4/6)

- Histogram of observed inter-release times of the periodic task *Tsk2*
  - The nominal period is equal to 20 ms
  - The observed period falls within $[19.920, 20.069]$ ms in 98.9% of the cases
- The observed variability is not negligible: what can we do?
  - Fix the implementation: not viable given that jitters depend on the RTOS!
  - Refine the model: not convenient due to state-space increase!
  - Raise a warning to subsequent testing stages!



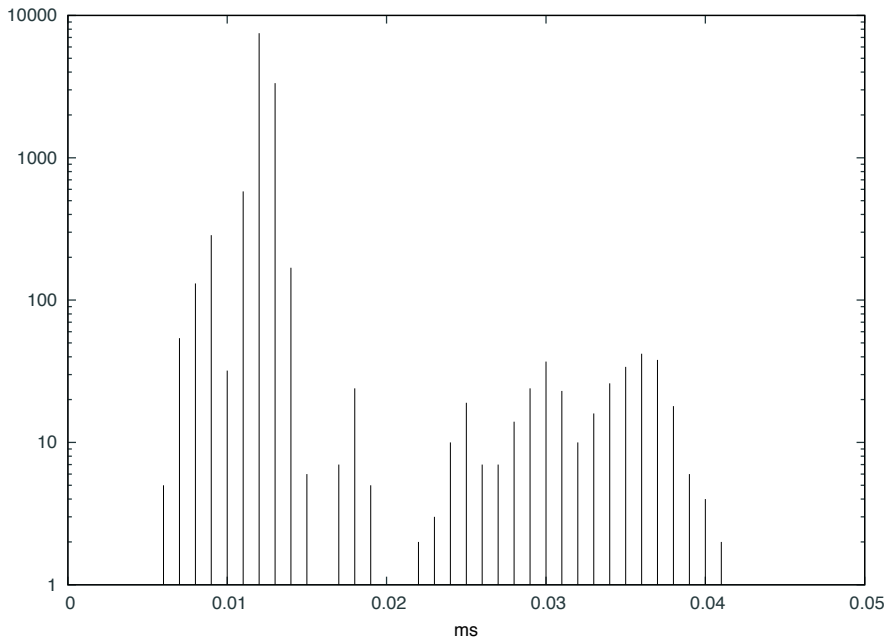
HW-in-the-loop testing: execution time profiling (5/6)

- Histogram of observed execution times of entry-point `f122` of $Tsk1$
  - Nominal execution time within the interval $[0.05, 0.2]$ ms
  - Reflects the absence of data-dependent alternatives in the implementation of the entry-point function (otherwise the histogram would have larger support)



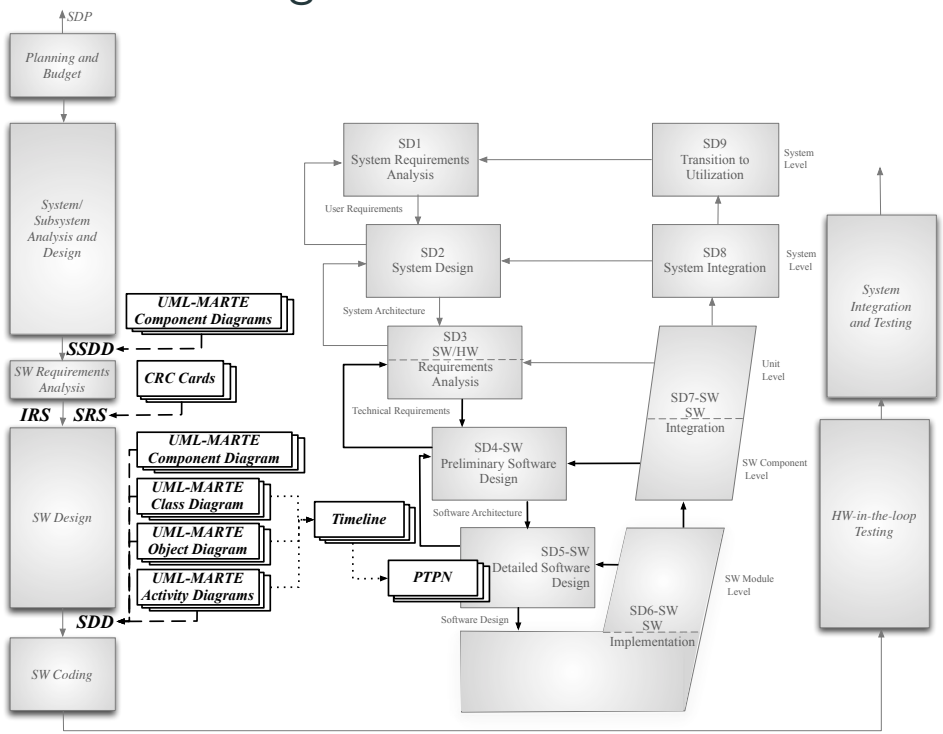
HW-in-the-loop testing: execution time profiling (6/6)

- Histogram of observed execution times of the wait operation performed by the periodic task *Tsk2* on the binary semaphore *sem3*
  - The time spent for semaphore operations on VxWorks 6.5 is not negligible with respect to the order of the execution time of entry-point functions



V-Model: first part of SD7-SW

- First part of SD7-SW: SW Integration - SW component scope
- Corresponds to the second part of HW-in-the-loop Testing in MIL-STD-498
- Activities supported by the formal core of PTPNs:
  - Test case selection and execution
  - Oracle verdict and coverage evaluation

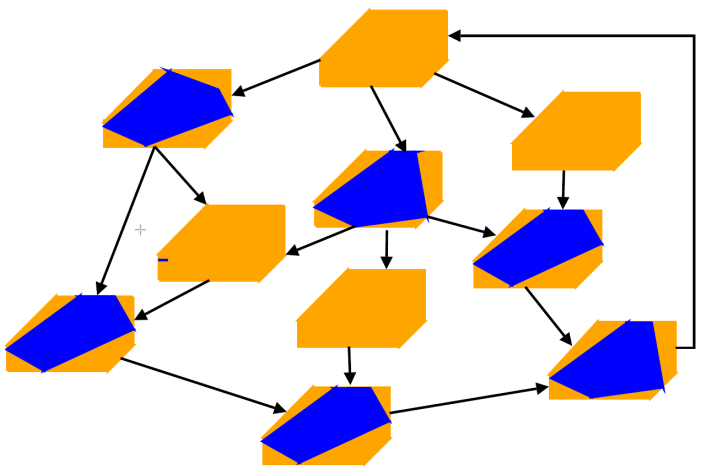


Failure and defect model

- Failures (deviations of a component from its expected delivery, service, result)
  - **Unsequenced execution**: a run breaking sequencing requirements (e.g., chunk sequence, semaphore and priority handling operations, IPC mechanisms, . . .)
  - **Timeframe violation**: a temporal parameter taking values out of its nominal interval (e.g., untimely job release, execution time $\notin$ [BCET, WCET])
  - **Deadline miss**: a job breaking its end-to-end timing requirement
- Defects (flaws in a component that can cause failures of the system)
  - **Task programming defect**: flaw in concurrency control and task interactions (e.g., semaphore operations not properly combined with priority handling, wrong priority assignment, unsequenced invocation of entry-points)
  - **Cycle stealing**: detraction of computational resources due to additional tasks (intentionally or not intentionally omitted in the specification)
- Remarks
  - Deadlines missed by tasks have contractual relevance
  - Failures in chunk sequencing/timing have relevance for design and unit testing
- Functional behavior is not addressed

SD7-SW: which abstraction for test case selection?

- Certification standards prescribe structural coverage criteria measured on the **control-flow graph** of the code
  - e.g., all-statemens, all-decisions, modified condition decision coverage, ...
  - Effectively exert control structures and data-flow dependencies
  - Limited coverage of the variety of behaviors that result from concurrent and interrupted execution of the dynamic architecture of each CSCI
- The **State Class Graph (SCG)** of the PTPN model of a CSCI is an effective abstraction of the variety of behaviors of the CSCI dynamic architecture
  - Qualified code generator $\Rightarrow$ SCG provides a **structural** coverage measure
  - Not qualified code generator $\Rightarrow$ SCG provides a **functional** abstraction



SD7-SW: criteria for test case selection on the SCG (1/3)

- All-markings
  - Requires coverage of each reachable marking
  - Guarantees coverage of all possible **states of concurrency** (i.e., all possible combinations of chunks that are concurrently running/ready/blocked)
  - For each reachable marking m, select any class S_m with marking m and include in the test suite any path from a **Controllable Starting Point (CSP)** to S_m
  - Complexity proportional to the number of reachable markings
- All-marking-edges (includes all-markings)
  - Requires coverage of each edge between any two reachable markings
  - Guarantees coverage of all possible **transitions between concurrency states** (e.g., preemptions following asynchronous releases, chunk completions, semaphore and mailbox operations, priority boost/deboost operations)
  - Complexity proportional to the number of reachable markings multiplied by the maximum number of events in a marking (the latter is $\leq$ # tasks)
  - For each edge between any two markings m_1 and m_2, select any class S_1 with marking m_1 with an event leading to class S_2 with marking m_2, and include in the test suite any path starting from a CSP and covering the edge

SD7-SW: criteria for test case selection on the SCG (2/3)

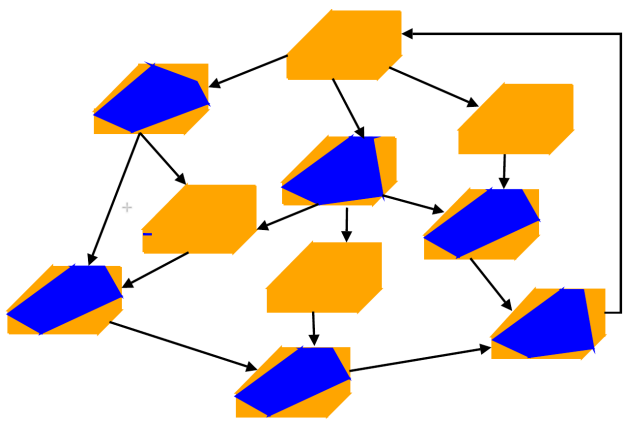
- All-classes (includes all-markings)
  - Requires coverage of all reachable state classes
  - Distinguishes states of concurrency associated with different timings yielded by the execution of different sequences of transition firings
  - Increases complexity with respect top all-markings
- All-class-edges (includes all-marking-edges)
  - Requires coverage of all edges between reachable state classes
  - Distinguishes states of concurrency associated with different timings yielded by the execution of different sequences of transition firings
  - Increases complexity with respect top all-marking-edges

SD7-SW: criteria for test case selection on the SCG (3/3)

- All-symbolic-runs (includes all-class-edges)
  - Requires coverage of all symbolic runs that start with a job release and terminate either with its completion or with a deadline miss
  - Distinguishes concurrency states and transitions visited in $\neq$ job executions
  - Increases complexity with respect top all-class-edges
- All-symbolic-executions
  - Requires coverage of any event sequence that starts with a job release and terminates either with its completion or with a deadline miss
  - Does not distinguish paths with same firing sequence and $\neq$ starting class
  - Reduces the complexity of all-symbolic runs

SD7-SW: test case execution

- How can we determine **timed inputs** that force the **Implementation Under Test (IUT)** to execute each selected test case?
  - Periodic and asynchronous releases times can be controlled
  - Computation times are often impractical to control
  - Problem of **determinization** of the implementation wrt the specification
    - e.g., the specification neglects dependencies in the execution times of chunks, nominal ranges of variation of timers are enlarged to make the model robust, ...
- **Guided testing** (as opposed to **randomized testing**)
  - Exploit SCG to identify state classes that can be selected as starting points
  - Exploit PTPN trace analysis to derive timing restrictions for starting classes, that can lead to the execution of specific sequences of transition firings



SD7-SW: oracle verdict and coverage evaluation

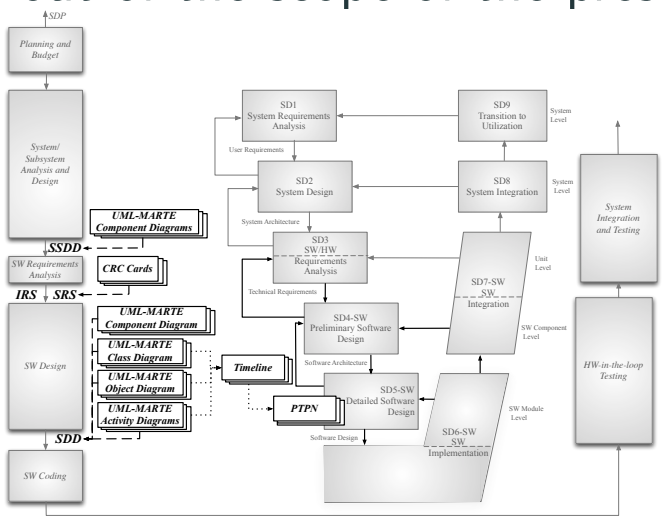
- End-to-end oracle
  - Verifies that end-to-end task deadlines are met
  - Requires time-stamped logging of job releases and completions
- Sequencing oracle
  - Verifies that the qualitative ordering of events conforms with the specification (i.e., that there exists at least one timing makes the logged sequence feasible)
  - Requires logging of all events corresponding to a transition in the PTPN model
- Timing oracle
  - Verifies that the logged sequence is a feasible execution for the specification (i.e., verifies the **timed trace inclusion relation**)
  - Requires time-stamped logging of all events corresponding to a transition
  - Detects all failures detected by the sequencing oracle

SD7-SW: results on the running case study

- Testing in a simulated environment of the code of the System Control (SC) CSCI, code integrated with **functional** entry-points of its chunks
- Detection of an unsequenced execution
  - Caused by a task programming defect: two chunks synchronizing on a semaphore not explicitly represented in the dynamic architecture
  - Fixed by representing the semaphore in the task-set specification and then repeating the formal verification
- Detection of time-frame violations
  - Caused by a cycle stealing due to a VxWorks task named `tNetTask` providing packet-processing network services
  - Fixed by assigning SC tasks larger priority than `tNetTask`

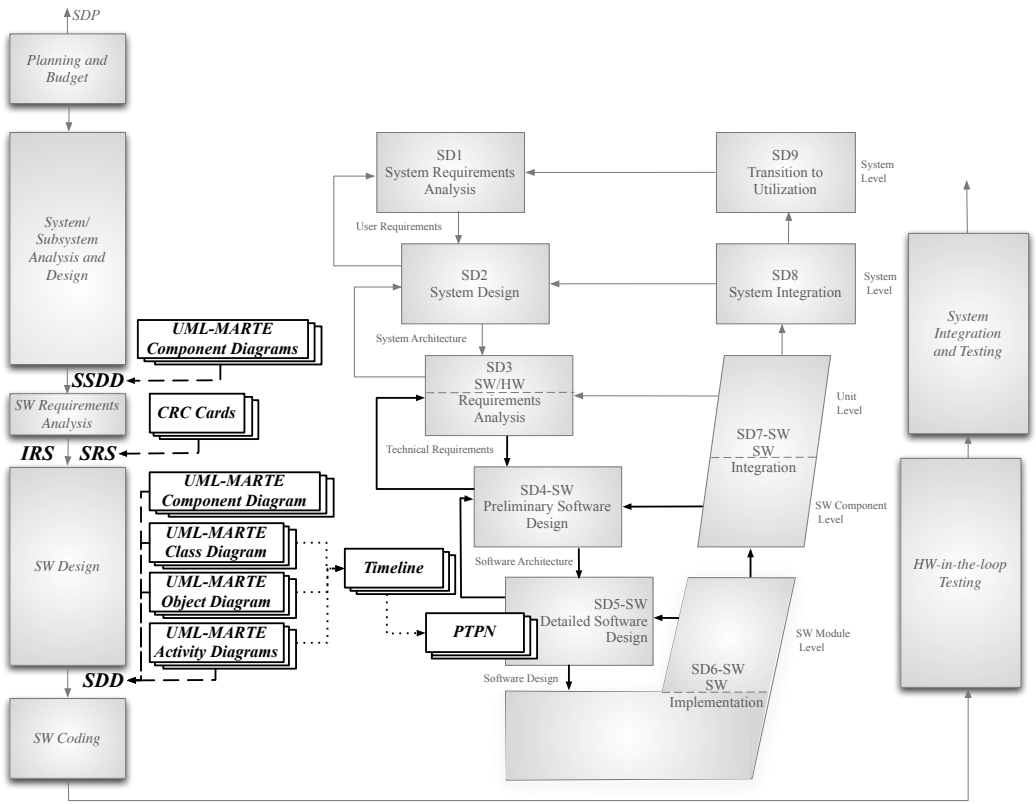
V-Model: second part of SD7-SW, SD8, SD9

- Second part of SD7-SW: SW Integration - unit scope
  - Tests the integration of CSCIs, HCIc, and FCIs within each unit
- SD8: System Integration - system scope
  - Tests the integration of all units within the system
- SD9: Transition to Utilization - system scope
  - Puts the completed system into operation at the intended application site
- The second part of SD7-SW and SD8 correspond to System Integration and Testing in MIL-STD-498
- All these activities are out of the scope of the presented methodology



Is testing really needed in Model Driven Development (MDD)?

- Architectural verification abstracts from a number of details
  - Execution time of operations (execution time may be negligible but not zero), additional tasks, deployment platform, context switches, ...
- Architectural verification could be not exhaustive (state space explosion)
- Testing is in any case needed for certification purposes (for good reasons)



Timed Automata

Introduction to TA (1/2)

- Timed Automata support the representation of **concurrent timed** systems
  - Expressivity comparable with that of Time Petri Nets (TPNs)
- A Timed Automaton (TA) is a finite automaton enriched with:
  - a set of **clocks**
  - **timing constraints** on transitions (also termed guards or enabling conditions)
  - **clock resets** on transitions
- A minimal TA: two locations `H` and `K`, two clocks `x` and `y`, a transition from `H` to `K` (enabled if $x > 0$, labeled with action `a`, resetting clock `y` to 0)

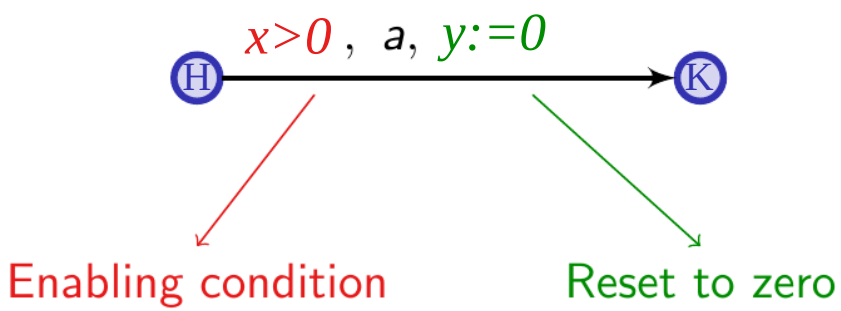
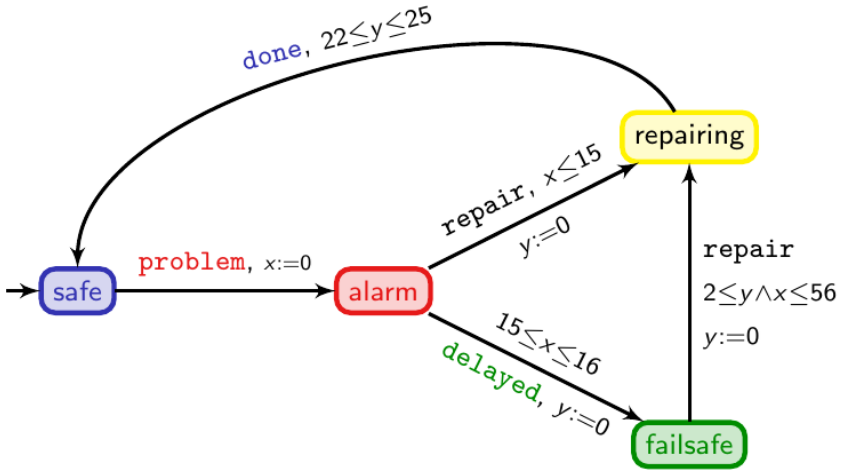


Image adapted from Patricia Bouyer-Decitre, CNRS & ENS Paris-Saclay

R. Alur and D. L. Dill, "A theory of timed automata," Theoretical Computer Science, 1994.

Introduction to TA (2/2)

- The TA of a repair process



- A possible execution of the TA

	safe	23 →	safe	problem →	alarm	15.6 →	alarm	delayed →	failsafe	
x	0		23		0		15.6		15.6	⋯
y	0		23		23		38.6		0	

	failsafe	2.3 →	failsafe	repair →	repairing	22.1 →	repairing	done →	safe
⋯	15.6		17.9		17.9		40		40
	0		2.3		0		22.1		22.1

Image by Patricia Bouyer-Decitre, CNRS & ENS Paris-Saclay

TA syntax

- A TA is a tuple $\langle L, l_0, \Sigma, X, E \rangle$ where:
  - L is a finite set of **locations**
  - $L_0 \subseteq L$ is the set of **initial location**
  - Σ is a finite **action alphabet**
  - X is a set of **clocks**
  - $E \subseteq L \times \Sigma \times G(X) \times 2^X \times L$ is a set of **edges** where $G(X)$ is a set of **guards** with syntax $g := x \sim c \mid g \wedge g$ where $x \in X$, $\sim \in \{<, \leq, =, >, \geq\}$, $c \in \mathbb{N}$ (i.e., a guard is a conjunction of atomic constraints of the form $x \sim c$)
- An example with three locations, two actions, two clocks, and two transitions

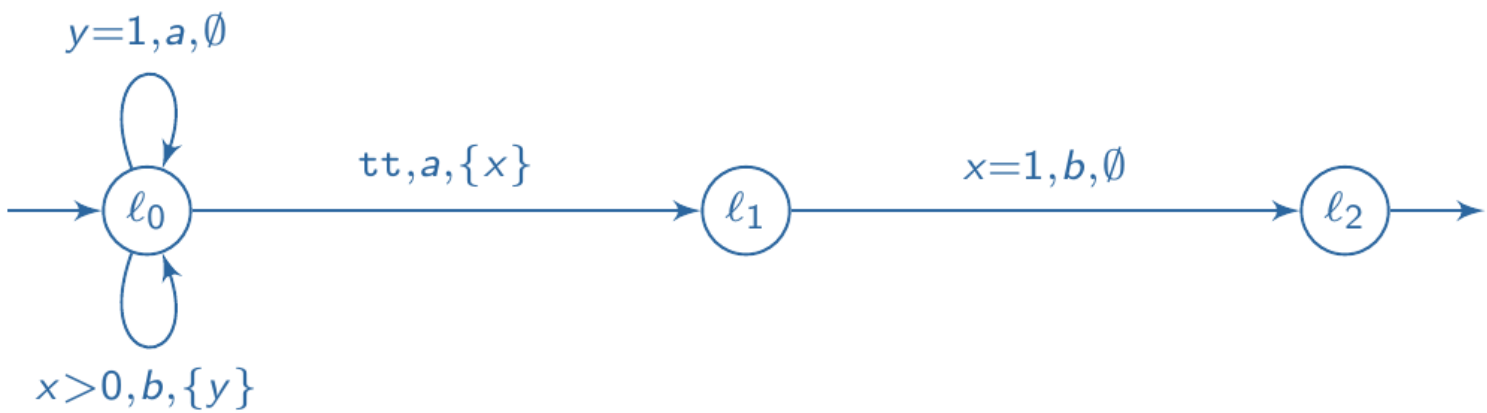


Image by Natalie Bertrand, Inria Rennes Bretagne

TA semantics (1/3)

- A **clock valuation** $v \in \mathbb{R}^X_{\geq 0}$ assigns a value to each clock
  - If a valuation $v \in \mathbb{R}^X_{\geq 0}$ satisfies a guard $g \in G(X)$, then we write $v \models g$
  - $v + \tau$ denotes the valuation $(v + \tau)(x) = v(x) + \tau \; \forall x \in X, \forall \tau \in \mathbb{R}_{\geq 0}$
- The **state** of a TA is a pair (l, v) where:
  - l is a location
  - v is a valuation
- There exist two types of **transition** between states
  - **Delay transition**: $(l, v) \xrightarrow{\tau} (l, v + \tau)$ for any state (l, v) and any delay $\tau \in \mathbb{R}_{\geq 0}$
  - **Discrete transition**: $(l, v) \xrightarrow{a} (l', v')$ if $\exists (l, a, g, R, l') \in E$ such that $v \models g$ and $v'(x) = 0$ if $x \in R$ and $v'(x) = v(x)$ if $x \notin R$
- A **run** of a TA is a sequence of alternating delay and discrete transitions: $(l_0, v_0) \xrightarrow{\tau_1} (l_0, v_0 + \tau_1) \xrightarrow{a_1} (l_1, v_1) \xrightarrow{\tau_2} (l_1, v_1 + \tau_2) \xrightarrow{a_2} \ldots \xrightarrow{a_k} (l_k, v_k)$
  - Equivalently written as $(l_0, v_0) \xrightarrow{\tau_1, a_1} (l_1, v_1) \xrightarrow{\tau_2, a_2} \ldots \xrightarrow{\tau_k, a_k} (l_k, v_k)$

TA semantics (2/3)

- A **time sequence** is a finite non-decreasing sequence of non-negative real values i.e., $s = (t_1)(t_2)\cdots(t_k)$ with $t_i \in \mathbb{R}_{\geq 0}\ \forall\, i \in \{1, \ldots, k\}$
- A **timed word** is a finite sequence of pairs consisting of an action and a time value belonging to a time sequence, i.e., $w = (a_1, t_1)(a_2, t_2)\cdots(a_k, t_k)$ with $a_i \in \Sigma$ and with $(t_1)(t_2)\cdots(t_k)$ being a time sequence
- A timed word $w = (a_1, t_1)(a_2, t_2)\cdots(a_k, t_k)$ is **accepted** by a TA if $\exists\ \rho = (l_0, v_0) \xrightarrow{\tau_1, a_1} (l_1, v_1) \xrightarrow{\tau_2, a_2} \ldots \xrightarrow{\tau_k, a_k} (l_k, v_k)$ with $l_0 \in L_0$ and $t_i = \sum_{1 \leq j \leq i} \tau_j$
- The **accepted timed language** of a TA is the set of its accepted timed words

TA semantics (3/3)

- A run of the example TA is $(l_0,(0,0)) \xrightarrow{0.1,b} (l_0,(0.1,0)) \xrightarrow{0.2,b} (l_0,(0.3,0)) \xrightarrow{1,a} (l_0,(1.3,1)) \xrightarrow{0.2,b} (l_0,(1.5,0)) \xrightarrow{0,a} (l_1,(0,0)) \xrightarrow{1,b} (l_2,(1,1))$
- An accepted timed word is $w = (b,0.1)(b,0.3)(a,1.3)(b,1.5)(b,2.5)$
- Always true guards and empty reset sets are omitted in the representation

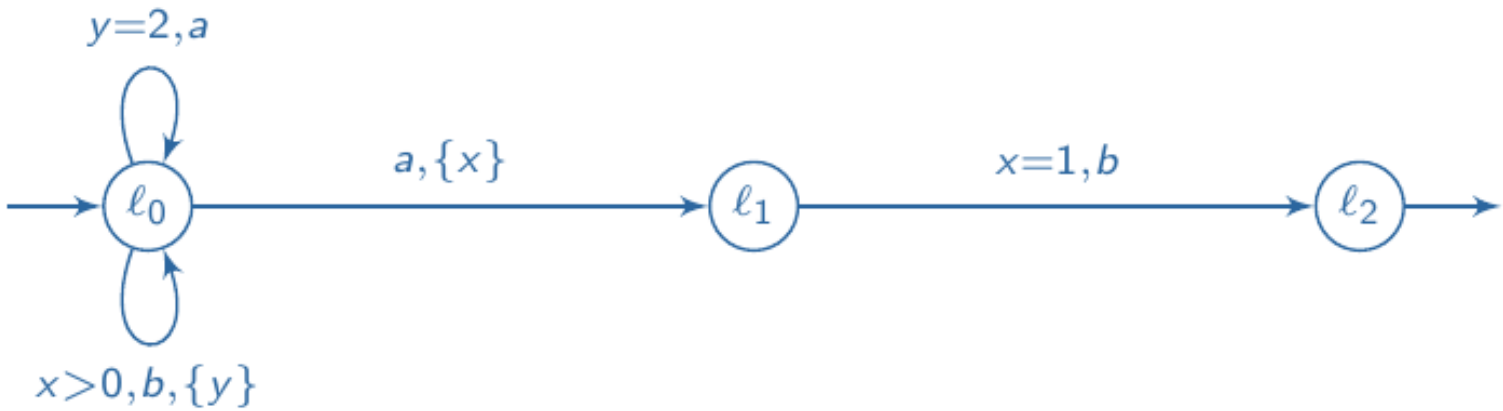
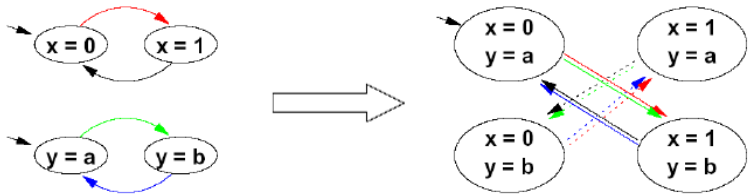


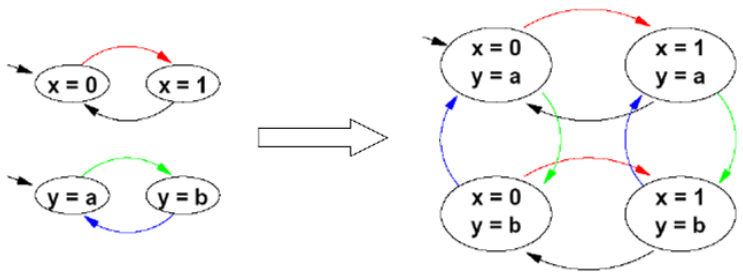
Image by Natalie Bertrand, Inria Rennes Bretagne

Many variants of TA in the literature (1/2)

- Networks of TA (NTA)
  - Product (parallel composition) of two or more TA with different sets of clocks
  - Different semantics depending on the type of composition
    - Synchronous composition



    - Asynchronous composition



    - Composition with explicit synchronization

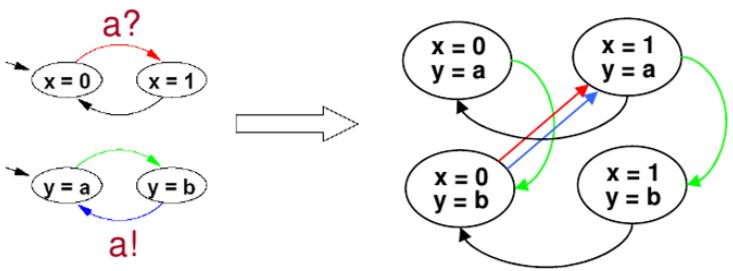


Image by Prof. Marco Di Natale, Scuola Superiore Sant'Anna, Pisa

Many extensions of TA in the literature (2/2)

- TA with diagonal constraints
  - Guards have syntax $g := x \sim c \,|\, x - y \sim c \,|\, g \wedge g$ where $x, y \in X$, $\sim \in \{<, \leq, =, >, \geq\}$, and $c \in \mathbb{N}$ (i.e., a guard is a conjunction of atomic constraints of the form $x \sim c$ and $x - y \sim c$)
- TA with epsilon transitions
  - Epsilon transitions are silent or unobservable transitions
- ...

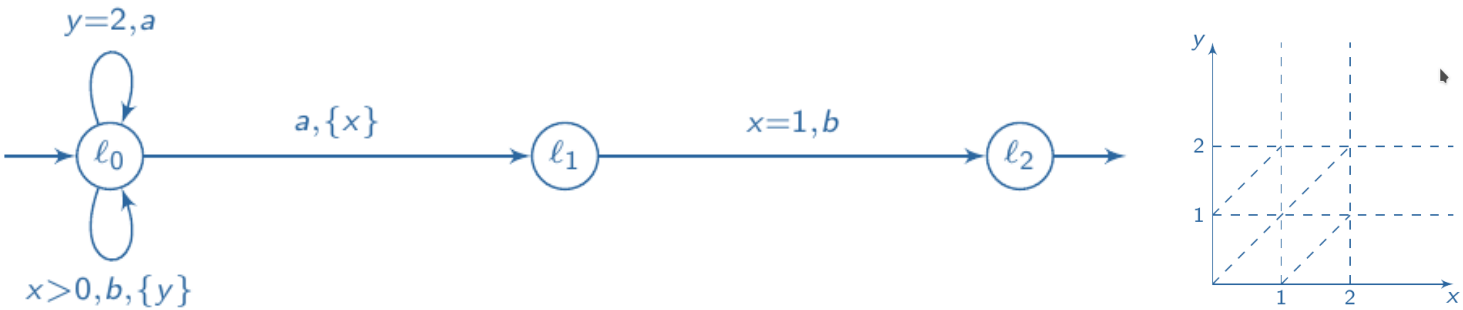
Region partitioning (1/2)

- $\forall\, g \in G$, let $[\![g]\!]$ be the set of valuations $\{v \in \mathbb{R}^X_{\geq 0} \mid v \models g\}$
- $\forall\, Y \subseteq X$, let $[Y \leftarrow 0]v$ be the valuation such that
  $[Y \leftarrow 0]v(x) = 0$ if $x \in Y$ and $[Y \leftarrow 0]v(x) = v(x)$ if $x \in X \setminus Y$

- A finite partition $\mathcal{R}$ of $\mathbb{R}^X_{\geq 0}$ is a **set of regions** (for the set of guards G) if:
  1. $\forall\, g \in G$ and $\forall\, R \in \mathcal{R}$, $[\![g]\!] \subseteq R$ or $[\![g]\!] \cap R = \varnothing$
  2. $\forall\, R, R' \in \mathcal{R}$, if $\exists\, v \in R$ and $t \in \mathbb{R}_{\leq 0}$ with $v + t \in R' \Rightarrow$
     $\Rightarrow \exists\, t' \in \mathbb{R}_{\geq 0}$ with $v' + t' \in R'\ \forall\, v' \in R$
  3. $\forall\, R, R' \in \mathcal{R}$, $\forall\, Y \subseteq X$, if $R_{[Y \leftarrow 0]} \cap R' \neq \varnothing \Rightarrow R_{[Y \leftarrow 0]} \subseteq R$
     (where $R_{[Y \leftarrow 0]}$ is the region obtained from R by resetting clocks in $Y \in X$)
- $\mathcal{R}$ defines an **equivalence relation** over valuations
  - An equivalence class is termed a **region**
- If two valuations are equivalent, then their future behaviors are equivalent
  - condition 1: two equivalent valuations satisfy the same clock constraints
  - condition 2: elapsing of time does not distinguish two equivalent valuations
  - condition 3: resetting clocks does not distinguish two equivalent valuations

Region partitioning (2/2)

- Two valuations v and v' with $v, v' \in \mathbb{R}^X_{\geq 0}$ are equivalent if $\forall\, x, y \in X$:
  - $v(x) > M \Leftrightarrow v'(x) > M$ where M is the **maximal constant** for all clocks
  - $v(x) \leq M \Rightarrow \lfloor v(x) \rfloor = \lfloor v'(x) \rfloor \wedge (\{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0)$
    where $\{v(x)\}$ is the fractional part of $\{v(x)\}$
  - $v(x) \leq M \wedge v(y) \leq M \Rightarrow (\{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\})$
- The partition is compatible with conditions 1, 2, and 3
- Diagonal-free region partitioning for 2 clocks and maximal constant 2

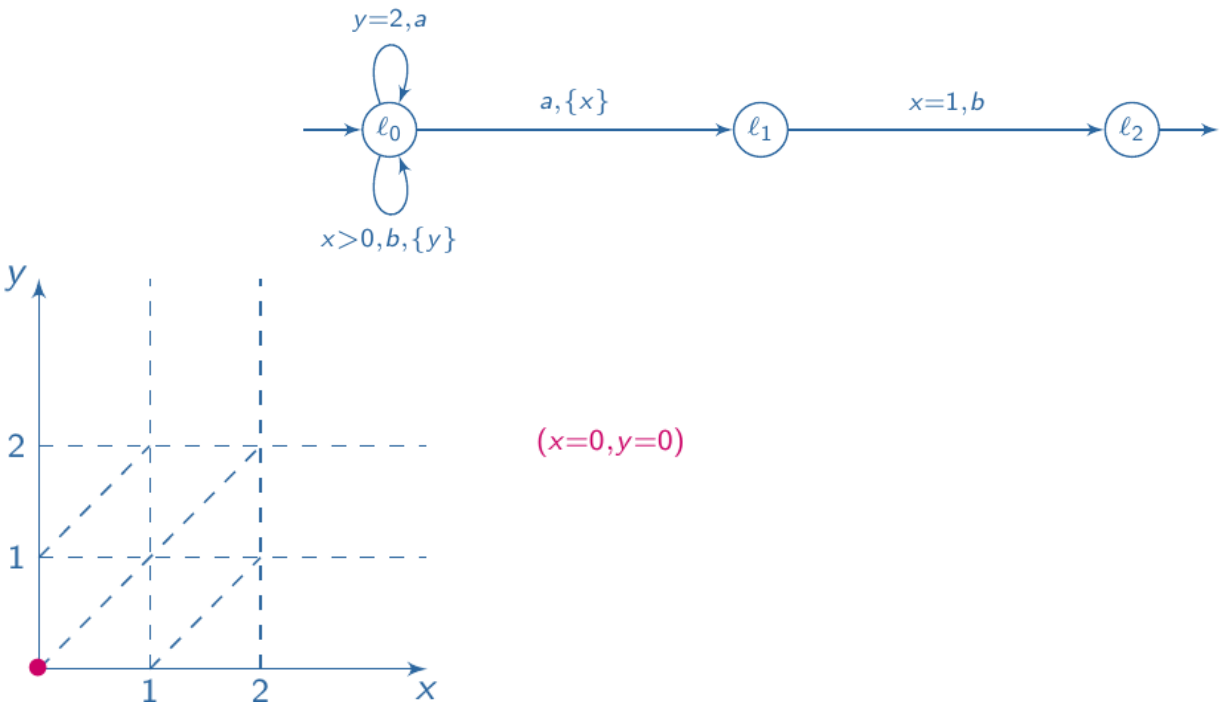


- Shape of **bounded** regions with 2 clocks and no diagonal constraints:

Images by Natalie Bertrand, Inria Rennes Bretagne

Region graph

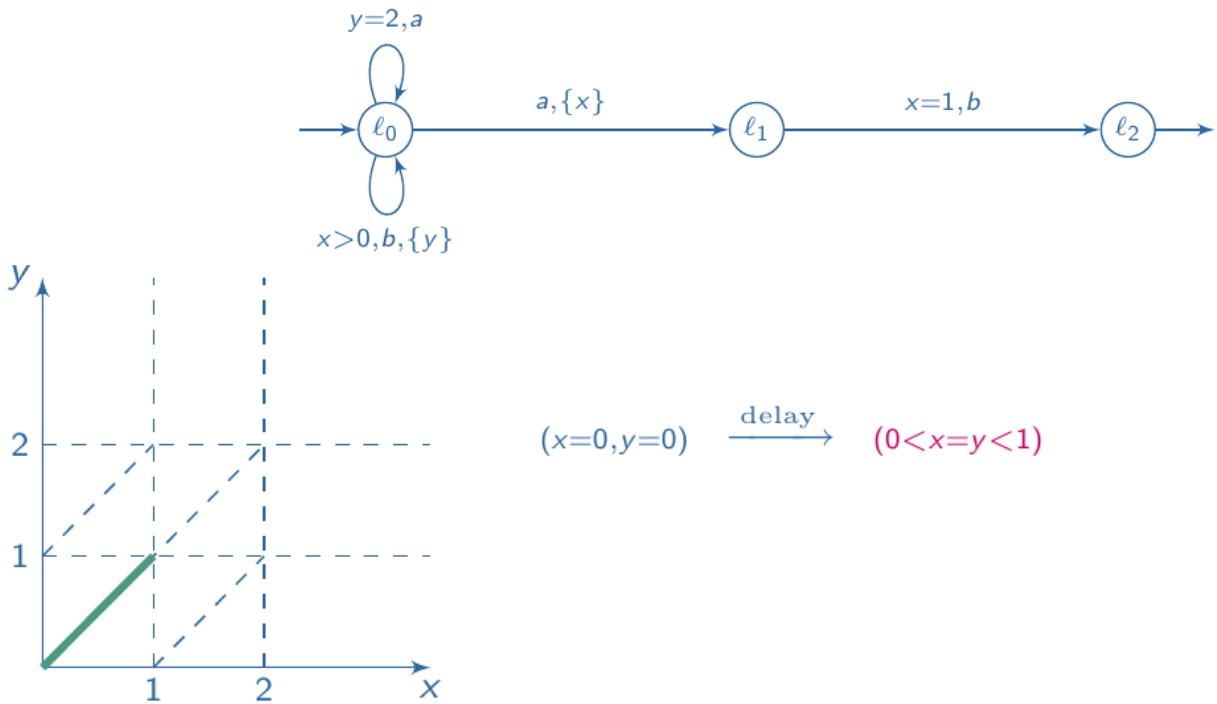
- Is a finite automaton whose states are the regions and whose transitions are:
  - $R \xrightarrow{\tau} R'$ if $\exists\, v \in R,\ v' \in R',\ \tau \in \mathbb{R}_{\geq 0}$ such that $v' = v + \tau$
    (in this case, R' is termed time successor of R)
  - $R \xrightarrow{Y} R'$ if $R_{[Y \leftarrow 0]} \subseteq R'$
- Represents the possible timing evolutions of the system
- Diagonal-free region partitioning for 2 clocks and maximal constant 2



Images by Natalie Bertrand, Inria Rennes Bretagne

Region graph

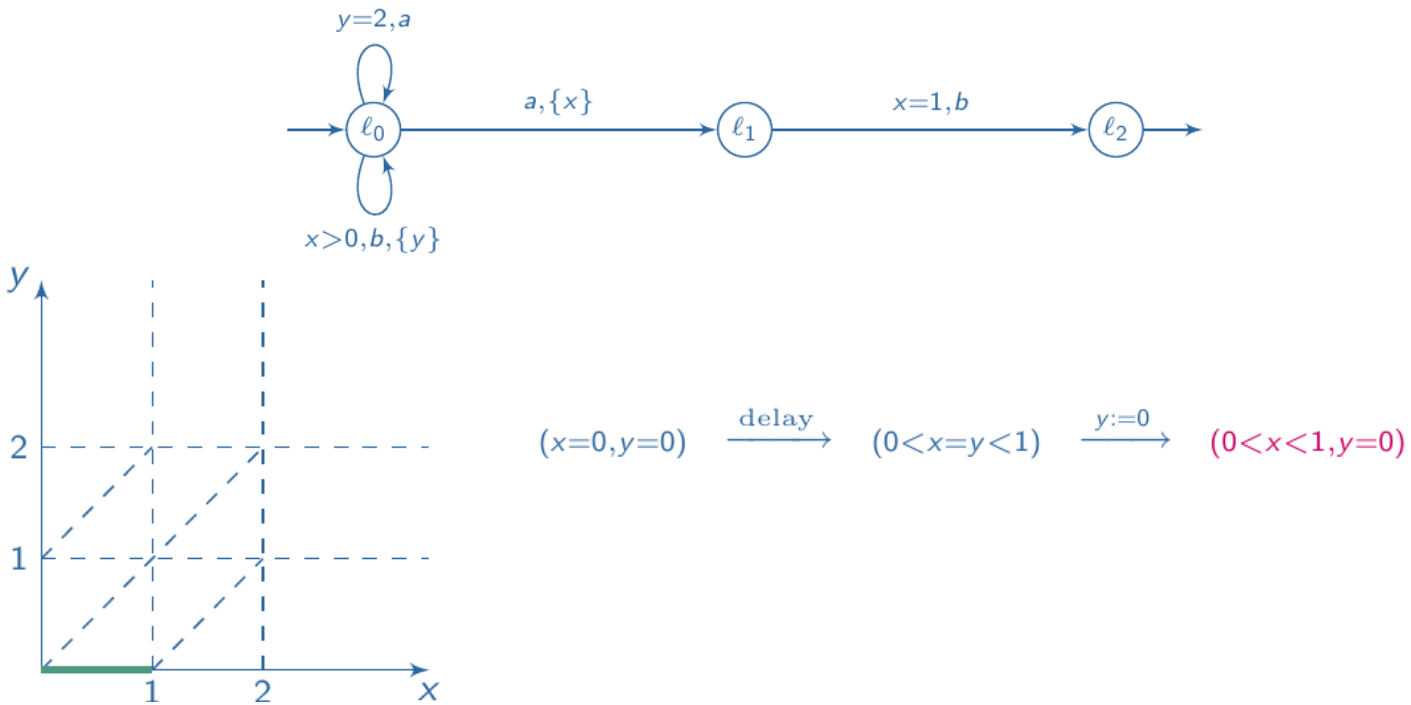
- Is a finite automaton whose states are the regions and whose transitions are:
  - $R \xrightarrow{\tau} R'$ if $\exists\, v \in R,\ v' \in R',\ \tau \in \mathbb{R}_{\geq 0}$ such that $v' = v + \tau$
    (in this case, R' is termed time successor of R)
  - $R \xrightarrow{Y} R'$ if $R_{[Y \leftarrow 0]} \subseteq R'$
- Represents the possible timing evolutions of the system
- Diagonal-free region partitioning for 2 clocks and maximal constant 2



Images by Natalie Bertrand, Inria Rennes Bretagne

Region graph

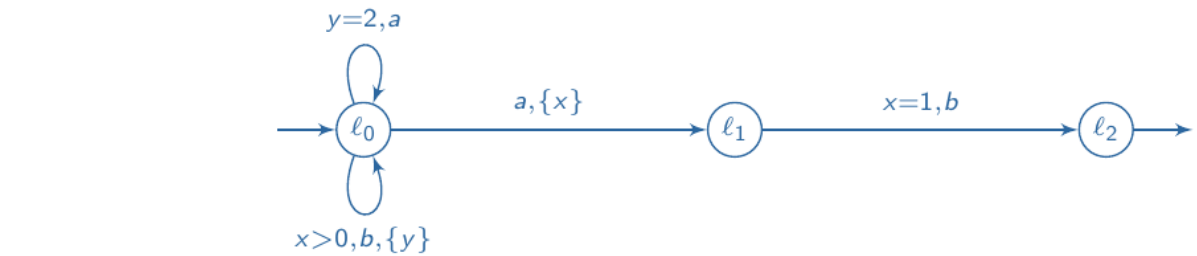
- Is a finite automaton whose states are the regions and whose transitions are:
  - $R \xrightarrow{\tau} R'$ if $\exists\, v \in R,\ v' \in R',\ \tau \in \mathbb{R}_{\geq 0}$ such that $v' = v + \tau$
    (in this case, R' is termed time successor of R)
  - $R \xrightarrow{Y} R'$ if $R_{[Y \leftarrow 0]} \subseteq R'$
- Represents the possible timing evolutions of the system
- Diagonal-free region partitioning for 2 clocks and maximal constant 2

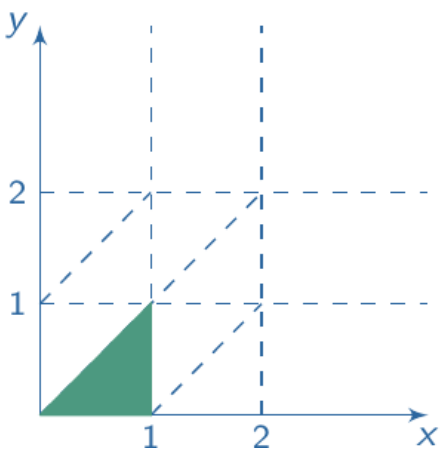


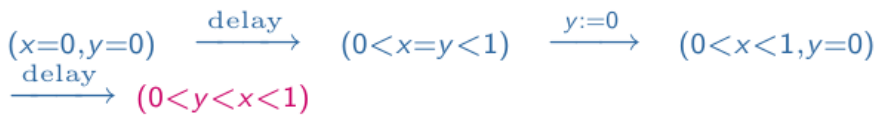
Images by Natalie Bertrand, Inria Rennes Bretagne

Region graph

- Is a finite automaton whose states are the regions and whose transitions are:
  - $R \xrightarrow{\tau} R'$ if $\exists\, v \in R,\ v' \in R',\ \tau \in \mathbb{R}_{\geq 0}$ such that $v' = v + \tau$
    (in this case, R' is termed time successor of R)
  - $R \xrightarrow{Y} R'$ if $R_{[Y \leftarrow 0]} \subseteq R'$
- Represents the possible timing evolutions of the system
- Diagonal-free region partitioning for 2 clocks and maximal constant 2



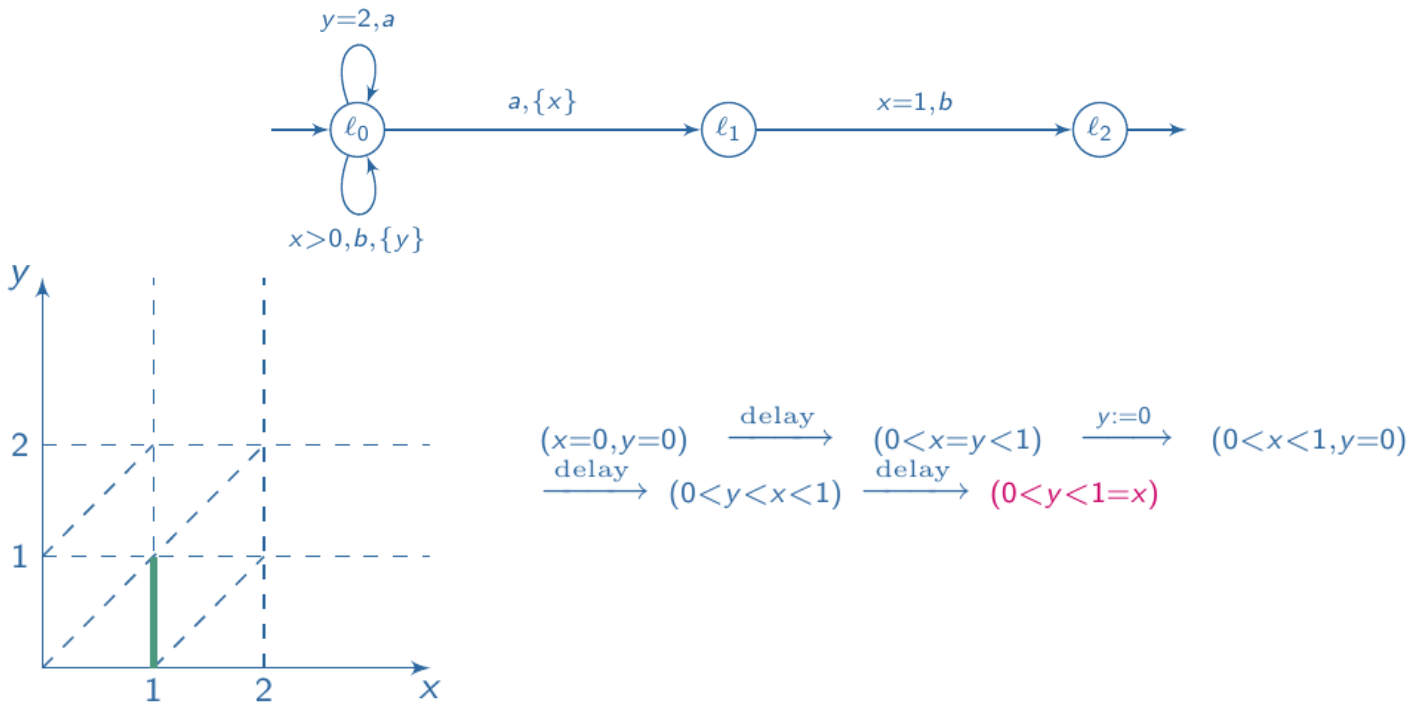




Images by Natalie Bertrand, Inria Rennes Bretagne

Region graph

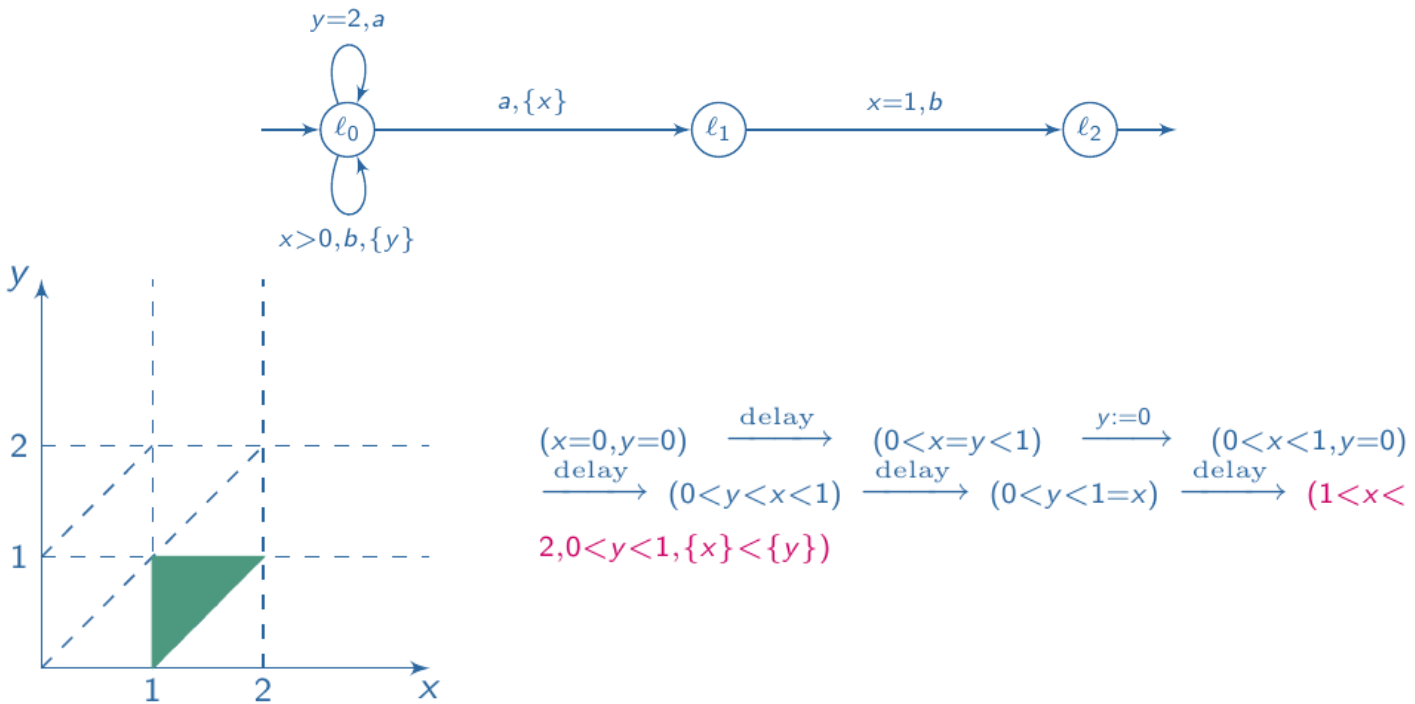
- Is a finite automaton whose states are the regions and whose transitions are:
  - $R \xrightarrow{\tau} R'$ if $\exists\, v \in R,\ v' \in R',\ \tau \in \mathbb{R}_{\geq 0}$ such that $v' = v + \tau$
    (in this case, R' is termed time successor of R)
  - $R \xrightarrow{Y} R'$ if $R_{[Y \leftarrow 0]} \subseteq R'$
- Represents the possible timing evolutions of the system
- Diagonal-free region partitioning for 2 clocks and maximal constant 2



Images by Natalie Bertrand, Inria Rennes Bretagne

Region graph

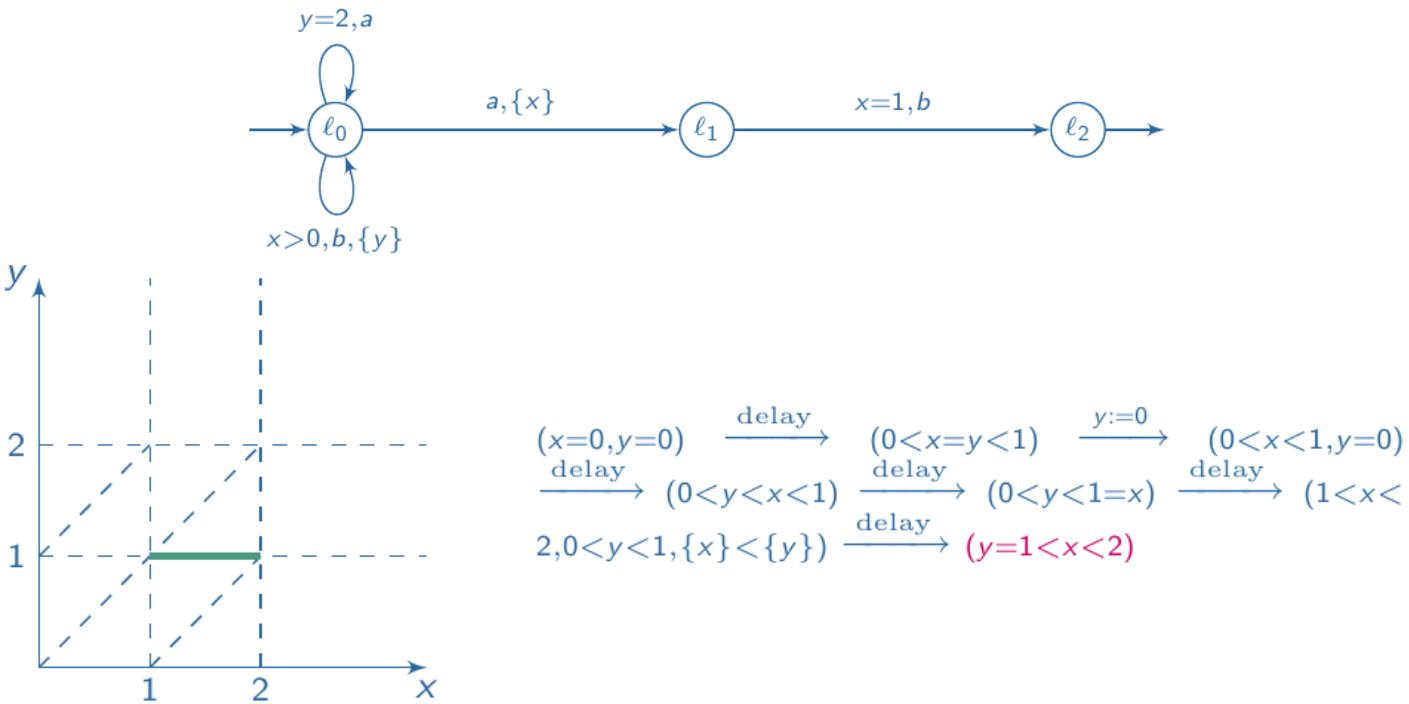
- Is a finite automaton whose states are the regions and whose transitions are:
  - $R \xrightarrow{\tau} R'$ if $\exists\, v \in R,\ v' \in R',\ \tau \in \mathbb{R}_{\geq 0}$ such that $v' = v + \tau$
    (in this case, R' is termed time successor of R)
  - $R \xrightarrow{Y} R'$ if $R_{[Y \leftarrow 0]} \subseteq R'$
- Represents the possible timing evolutions of the system
- Diagonal-free region partitioning for 2 clocks and maximal constant 2



Images by Natalie Bertrand, Inria Rennes Bretagne

Region graph

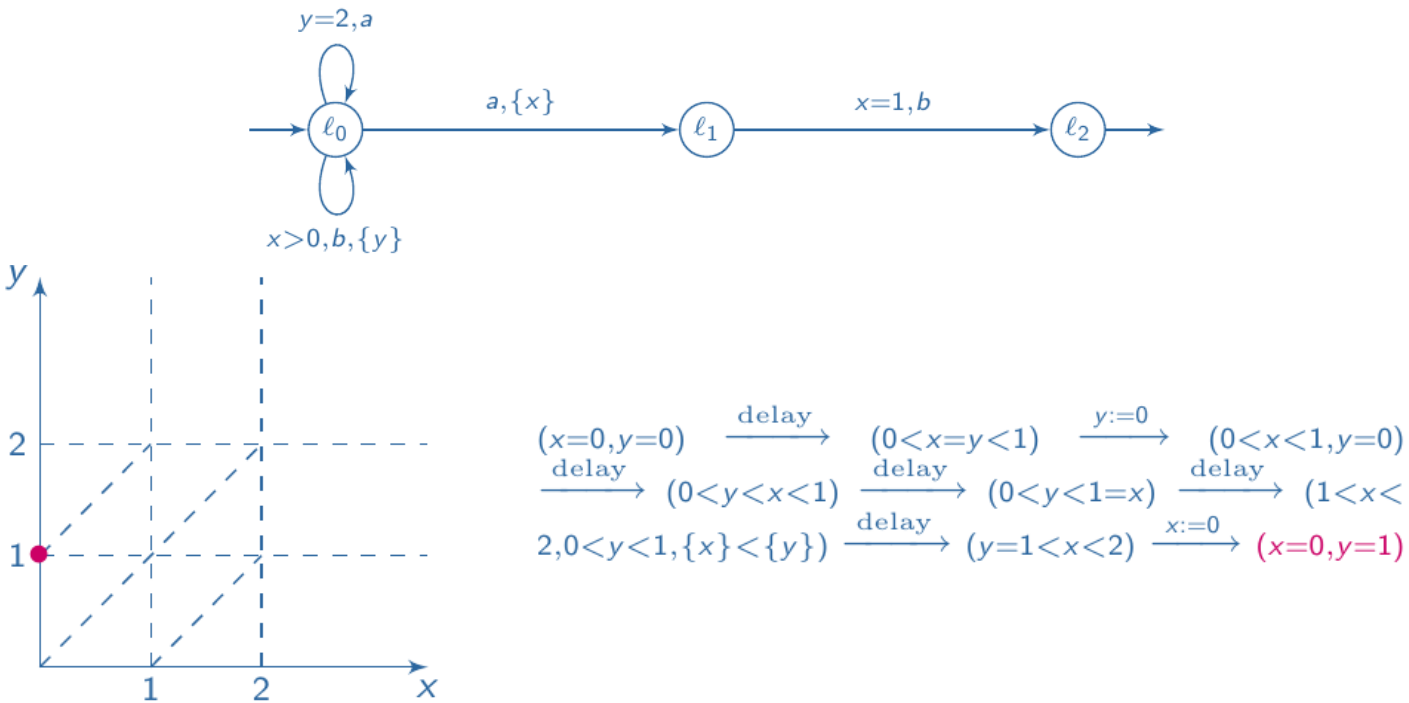
- Is a finite automaton whose states are the regions and whose transitions are:
  - $R \xrightarrow{\tau} R'$ if $\exists\, v \in R,\ v' \in R',\ \tau \in \mathbb{R}_{\geq 0}$ such that $v' = v + \tau$
    (in this case, R' is termed time successor of R)
  - $R \xrightarrow{Y} R'$ if $R_{[Y \leftarrow 0]} \subseteq R'$
- Represents the possible timing evolutions of the system
- Diagonal-free region partitioning for 2 clocks and maximal constant 2



Images by Natalie Bertrand, Inria Rennes Bretagne

Region graph

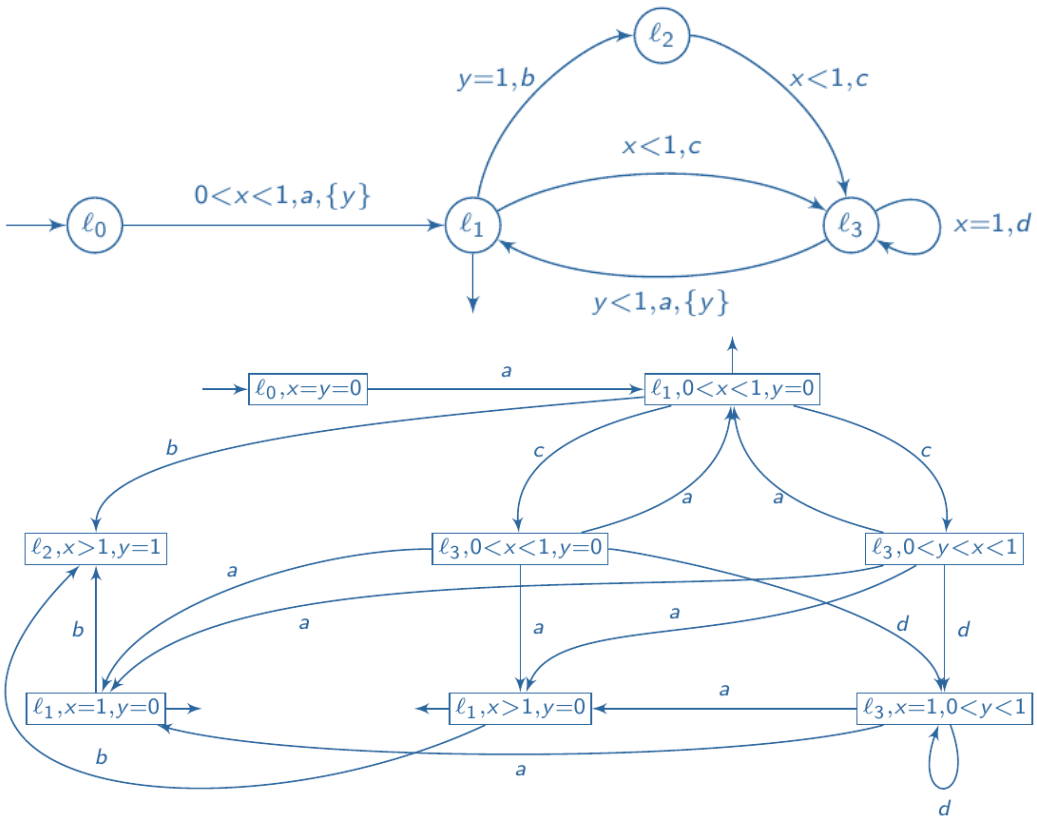
- Is a finite automaton whose states are the regions and whose transitions are:
  - $R \xrightarrow{\tau} R'$ if $\exists\, v \in R,\ v' \in R',\ \tau \in \mathbb{R}_{\geq 0}$ such that $v' = v + \tau$
    (in this case, R' is termed time successor of R)
  - $R \xrightarrow{Y} R'$ if $R_{[Y \leftarrow 0]} \subseteq R'$
- Represents the possible timing evolutions of the system
- Diagonal-free region partitioning for 2 clocks and maximal constant 2



Images by Natalie Bertrand, Inria Rennes Bretagne

Region automaton

- Is a finite automaton whose set of states is $L \times \mathcal{R}$ and whose transitions are:
  - $(l, R) \xrightarrow{a} (l', R')$ if $\exists\, l \xrightarrow{g,a,Y} l'$ is a transition in the TA with $R \subseteq [\![g]\!]$ and $R \xrightarrow{Y} R'$ is a transition in the region graph
  - $(l, R) \xrightarrow{\tau} (l', R')$ if $R \xrightarrow{\tau} R'$ is a transition of the region graph
- An example



Images by Natalie Bertrand, Inria Rennes Bretagne

Zone graph

- The number of regions is **exponential** in the number of clocks and the value of the maximal constants in the guards!
- The **zone graph** provides a more efficient encoding
  - Zones are DBMs representing the union of regions
- An example: a TA and its zone graph

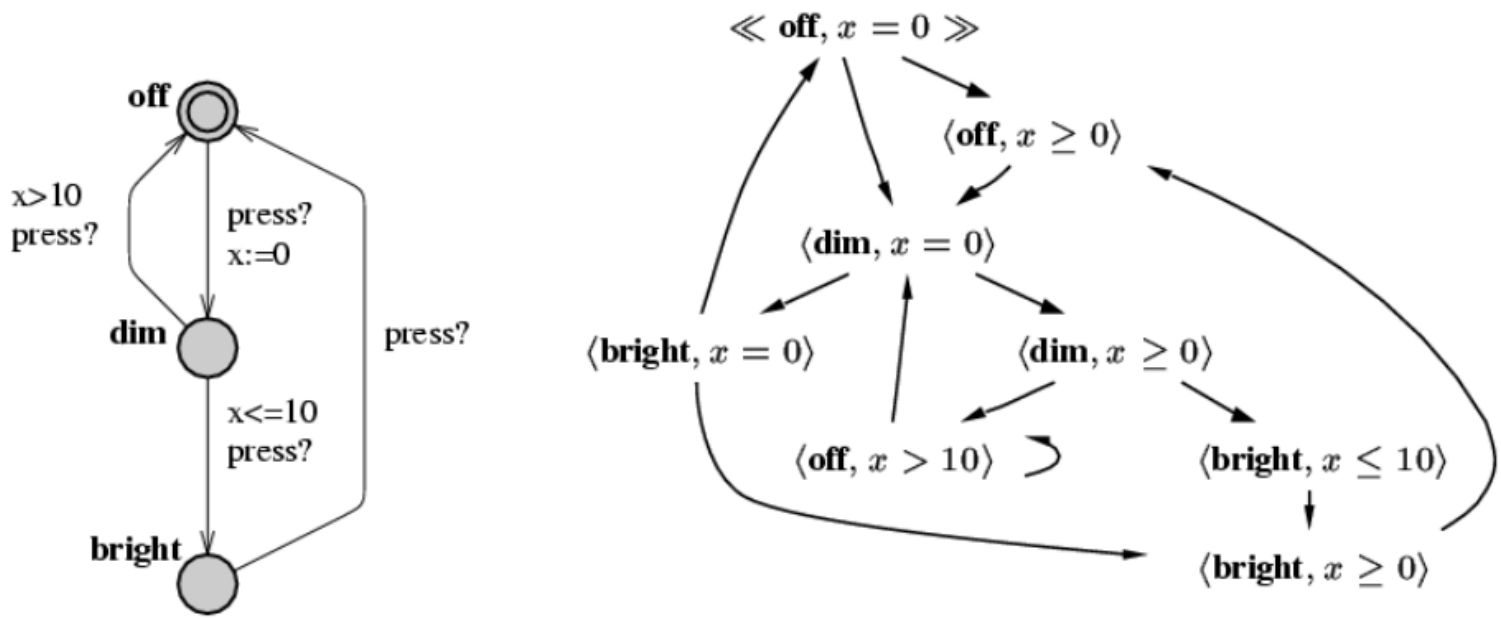
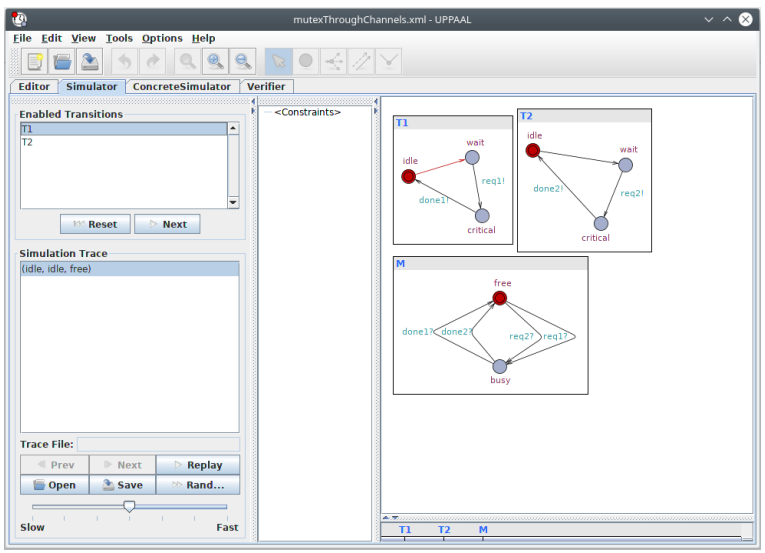


Image by Prof. Marco Di Natale, Scuola Superiore Sant'Anna, Pisa

Uppaal: a tool for modeling and analysis of NTA

- Jointly developed by the Department of Information Technology at Uppsala University and the Department of Computer Science at Aalborg University
  - Home page: `http://www.uppaal.org`
- Main features
  - **Modeling**: graphical editing of NTA (with templates, constants, ...)
  - **Simulation**: manual or random execution of runs of NTA
  - **Analysis**: verification of reachability, safety, and liveness properties



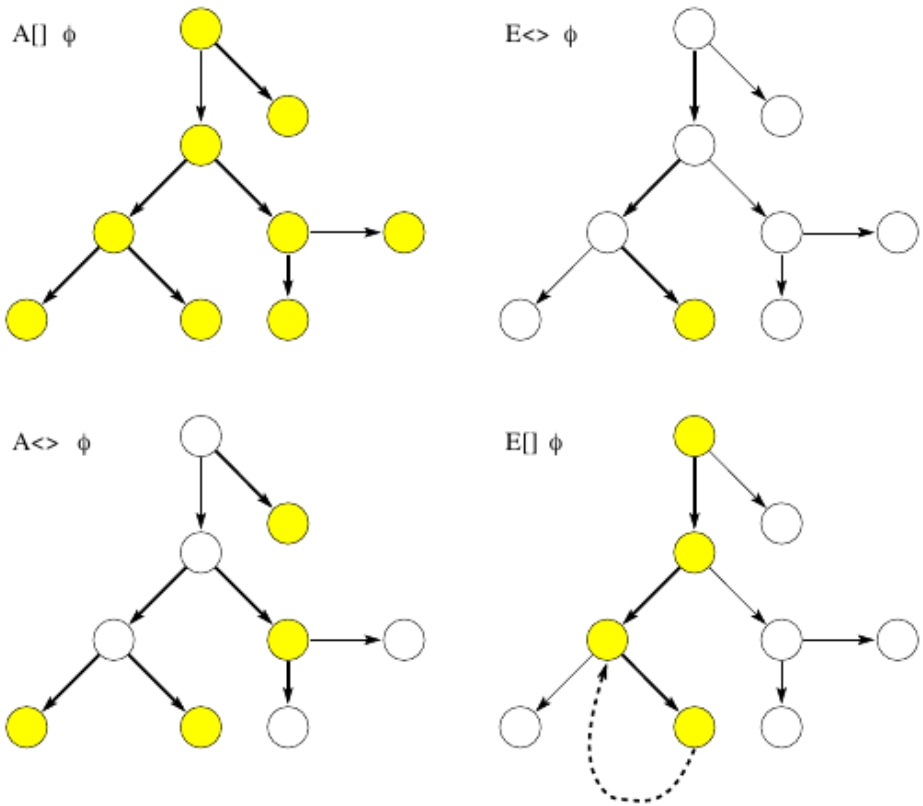
G. Behrmann, A. David, and K. G. Larsen, "A Tutorial on Uppaal 4.0," November, 2006.

Uppaal: verification of temporal logic formulas (1/2)

- The first temporal connective can be:
  - A: on all paths from the current state
  - E: on at least one path from the current state
- The second temporal connective can be:
  - X: the next state
  - G: all future states (denoted as [] in Uppaal)
  - F: some future state (denoted as <> in Uppaal)
  - U: until
- Examples (assuming the system is in some state s)
  - ϕ is true iff it is satisfied by s
  - AXϕ is true iff ϕ is true for every immediate successor state of s
  - AGϕ is true iff ϕ is true for every successor state of s
  - AFϕ is true iff on all paths originating from s there is a state where ϕ holds
  - AϕUθ is true iff all paths from s satisfy ϕ until they reach a state where θ holds
  - EXϕ is true iff ϕ is true for at least one immediate successor state of s
  - ...

Uppaal: verification of temporal logic formulas (2/2)

- Examples (assuming the system is in some state s)
  - `A[]`ϕ is true iff ϕ is true in all states of all paths from s
  - `E<>`ϕ is true iff ϕ is true in some state of at least one path from s
  - `A<>`ϕ is true iff ϕ is true in some state of all paths from s
  - `E[]`ϕ is true iff ϕ is true in all states of at least one path from s



Images by Prof. Marco Di Natale, Scuola Superiore Sant'Anna, Pisa

Credits

Credits

- Part of materials on Petri nets and time Petri nets presented in these slides is taken from the notes of the course "Methods for Verification and Testing" ("Metodi di verifica e testing") given by Prof. Enrico Vicario: `https://stlab.dinfo.unifi.it/vicario/Teaching/Verification_and_Testing_Metho/verification_and_testing_metho.html`
- Part of materials on timed automata presented in these slides is taken from:
  - the slides "An introduction to timed systems" by Patricia Bouyer: `http://www.lsv.fr/~bouyer`
  - the slides "Timed Automata" by Natalie Bertrand: `http://people.rennes.inria.fr/Nathalie.Bertrand`
  - the slides of the course "Embedded Systems - Model-Based Design" given by Prof. Marco Di Natale: `http://retis.sssup.it/~marco/teaching/embeddedsystems`