CAS 745
Supervisory Control of Discrete-Event Systems

Slides 6: Timed DES

Dr. Ryan Leduc

Department of Computing and Software
McMaster University

Material based on W. M. Wonham, Supervisory Control of Discrete-Event Systems, Department of Electrical and Computer Engineering, University of Toronto, July 2004. Lecture notes of Professor Wonham also used.

Timed DES Intro - §9.1

- We extend the framework of DES to include timing information by introducing a new event tick that corresponds to a "tick" of a "global" clock that the system is assumed to be synchronized with.
- For instance, if three *tick* events occur, then that means three time units have ended.
- Allows us to specify an upper and lower time bound for an event to occur, relative to when the event first becomes enabled.
- ie. if event α can occur after β occurs, we can then express things like:
- After β occurs, α can not occur for 2 *ticks,* but must occur before 5 *ticks* have elapsed.

Timed DES Intro - II

- We are also able to introduce the new concept of forcible events. These are events that we can ensure occur before the next *tick* event.
- As we can see, our concepts of controllability, nonblocking, modular supervision, and maximally permissive supervisors extend to this new area.
- To investigate timed DES issues, we will use an enhanced version of TCT called TTCT.

Timed Discrete-Event Systems - §9.2

- A timed discrete-event systems (TDES) requires some preparation to define.
- We start by defining an activity transition structure as:

$$\mathbf{G_{act}} = (A, \Sigma_{\text{act}}, \delta_{\text{act}}, a_o, A_m)$$

  which is defined as normal except instead of a "state set," we have an *activity set*.
- We assume that the sets A and Σ_{act} are finite.
- Let $\mathbb{N} := \{0, 1, 2, \ldots\}$ denote the set of natural numbers.
- We associate with each transition label $\sigma \in \Sigma_{\text{act}}$, a lower time bound $l_\sigma \in \mathbb{N}$ and an upper time bound $u_\sigma \in \mathbb{N} \cup \{\infty\}$.
- l_σ typically represents a delay, such as in communication or control enforcement; u_σ a hard deadline imposed by physical requirement or a legal specification.

Timer Details

- We assume our event set is partitioned as follows:

$$\Sigma_{\text{act}} = \Sigma_{\text{spe}} \dot{\cup} \Sigma_{\text{rem}}$$

- If event σ is prospective (Σ_{spe}), its upper time bound is finite $(0 \leq u_\sigma < \infty)$ and $0 \leq l_\sigma \leq u_\sigma$.

- If event σ is remote (Σ_{rem}), we set $u_\sigma = \infty$ and require $0 \leq l_\sigma < \infty$.

- The triple $(\sigma, l_\sigma, u_\sigma)$ is called a timed event.

- For $j, k \in \mathbb{N}$ we will use the notation $[j, k]$ to represent the set of integers i with $j \leq i \leq k$. ie. $[1, 3] = \{1, 2, 3\}$

- **Defn:** We define the timer interval for σ, T_σ as follows:

$$T_\sigma = \begin{cases} [0, u_\sigma] & \text{if } \sigma \in \Sigma_{\text{spe}} \\ [0, l_\sigma] & \text{if } \sigma \in \Sigma_{\text{rem}} \end{cases}$$

TDES Definition

- We are now ready to give the definition of a timed discrete-event system (TDES):

$$\mathbf{G} = (Q, \Sigma, \delta, q_o, Q_m)$$

- We define the state set of $\mathbf{G}$ as follows:

$$Q := A \times \prod \{T_\sigma | \, \sigma \in \Sigma_{\text{act}}\}$$

- A state $q \in Q$ is of the form:

$$q = (a, \{t_\sigma | \, \sigma \in \Sigma_{\text{act}}\}), \text{ for some } a \in A$$

- **Defn:** We refer to the component t_σ of q as the timer of σ in q.

- **NOTE:** The counter for a variable is initialized at its max value, and then counts downwards.

TDES Definition - II

- **Defn:** If $\sigma \in \Sigma_{\text{spe}}$, we refer to t_σ as the *current deadline* for σ. If $t_\sigma = 0$, we say that σ is *imminent*.
- The *current delay* is $\max(t_\sigma + l_\sigma - u_\sigma, 0)$.
- If $\sigma \in \Sigma_{\text{rem}}$, we refer to t_σ as the *current delay* for σ. The current deadline may be considered infinite.
- The value u_σ (l_σ) for a prospective (remote) event σ is referred to as the *default value* ($t_{\sigma o}$) of t_σ.
- The *initial state* is:

$$q_o := (a_o, \{t_{\sigma o} | \, \sigma \in \Sigma_{\text{act}}\})$$

- The *marker state subset* will be taken to be of the form:

$$Q_m \subseteq A_m \times \prod \{T_\sigma | \, \sigma \in \Sigma_{\text{act}}\}$$

TDES Definition - III

- **Defn:** We introduce a new event *tick* to represent the "tick of a global clock." Our *event set* is thus:

$$\Sigma := \Sigma_{\text{act}} \dot{\cup} \{tick\}$$

- Our *state transition function* is:

$$\delta : Q \times \Sigma \to Q \quad (pfn)$$

- We will give an overview of how δ is defined, leaving the formal definition for the textbook.

- As usual, we assume that $\mathbf{G}$ generates strings in Σ^*. As we consider that $\mathbf{G}$ includes our global digital clock, it also generates *tick* events.

Defining δ for TDES

- Conceptually, $\mathbf{G}$ starts at state q_o and at time $t = 0$, where time $t = 1$ represents that our first tick has occurred (and one time unit has passed since $t = 0$), etc.

- For $\sigma \in \Sigma$, $q \in Q$, $\delta(q, \sigma)$ is defined if:

  1. $\sigma =$ *tick* and no prospective event is imminent;

  2. σ is prospective, $q = (a, \_)$, $\delta_{\mathsf{act}}(a, \sigma)!$, and $0 \leq t_\sigma \leq u_\sigma - l_\sigma$ (t_σ is from q); or

  3. σ is remote, $q = (a, \_)$, $\delta_{\mathsf{act}}(a, \sigma)!$, and $t_\sigma = 0$ (t_σ is from q)

- **Defn:** An event $\sigma \in \Sigma_{\mathsf{act}}$ is *enabled* at $q = (a, \_)$ if $\delta_{\mathsf{act}}(a, \sigma)!$ and is *eligible* if, in addition, $\delta(q, \sigma)!$.

- An event that is enabled but not eligible is said to be *pending.*

Defining Next State for TDES

- For $\sigma \in \Sigma$ and $q = (a, \_) \in Q$, if $\delta(q, \sigma)!$, we need to now determine what the next state $q' \in Q$ would be.
- **Case $\sigma =$ tick:** An occurrence of a *tick* does not change the activity component a of q, but the timer components may be altered.
- For each $\tau \in \Sigma_{\text{act}}$, if τ is enabled at q, then $t_\tau = \max(t_\tau - 1, 0)$, otherwise it is unchanged.
- **Case $\sigma \in \Sigma_{\text{act}}$:** The new activity is taken to be $a' = \delta_{\text{act}}(a, \sigma)$
- The occurrence of σ always resets its timer to its default value.
- For each $\tau \in \Sigma_{\text{act}}$, if $\neg\delta_{\text{act}}(a', \tau)!$, then t_τ is reset to its default value, otherwise it is unchanged.

Defining Next State for TDES - II

- $\sigma \in \Sigma_{\text{act}}$ can not occur until after l_σ *ticks* of the clock after it has last become enabled.
- If σ is prospective, it must occur before $u_\sigma + 1$ *ticks* have occurred since it last became enabled, unless disabled by a preemptive occurrence of some other eligible event.
- A remote event need never occur.
- **Defn:** For $\mathbf{G}$, we refer to $\mathbf{G}$'s normal transition graph (explicitly showing *tick* events) as the *timed transition graph* (TTG) of $\mathbf{G}$.
- We refer to $\mathbf{G_{act}}$'s normal transition graph as the *activity transition graph* (ATG) of $\mathbf{G}$.

Activity-Loop-Free Condition

- We need to add a final technical condition on the definition of a TDES, to ensure the physically unrealistic possibility of a *tick* event being preempted indefinitely.

- **Defn:** A TDES has an *activity loop* if:

$$(\exists q \in Q)(\exists s \in \Sigma_{\mathsf{act}}^{+})\, \delta(q, s) = q$$

- We require that a TDES be activity-loop-free (ALF), namely:

$$(\forall q \in Q)(\forall s \in \Sigma_{\mathsf{act}}^{+})\, \delta(q, s) \neq q$$

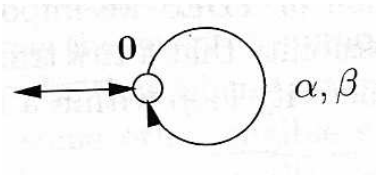
- Note, a TDES never "Stops the clock." We always have either a $\sigma \in \Sigma_{\mathsf{act}}$ or a *tick* event that is possible to occur.

TDES Example

- Let $\mathbf{G_{act}}$ be:

$$\mathbf{G_{act}} = (A, \Sigma_{\text{act}}, \delta_{\text{act}}, a_o, A_m)$$

with $\Sigma_{\text{act}} = \{\alpha, \beta\}$, time bounds $(l_\sigma, u_\sigma) = (1, 1)$ for α and $(2, 3)$ for β , and ATG shown below.



- The state set for $\mathbf{G}$ is:

$$\begin{aligned} Q &= \{0\} \times T_\alpha \times T_\beta \\ &= \{0\} \times [0, 1] \times [0, 3] \\ &= \{0\} \times \{0, 1\} \times \{0, 1, 2, 3\} \end{aligned}$$

- We take: $Q_m = \{(0, 1, 3)\}$.

TDES Example - II

- The TTG for $\mathbf{G}$ is shown in figure, and has event set $\Sigma = \{\alpha, \beta, \mathit{tick}\}$.

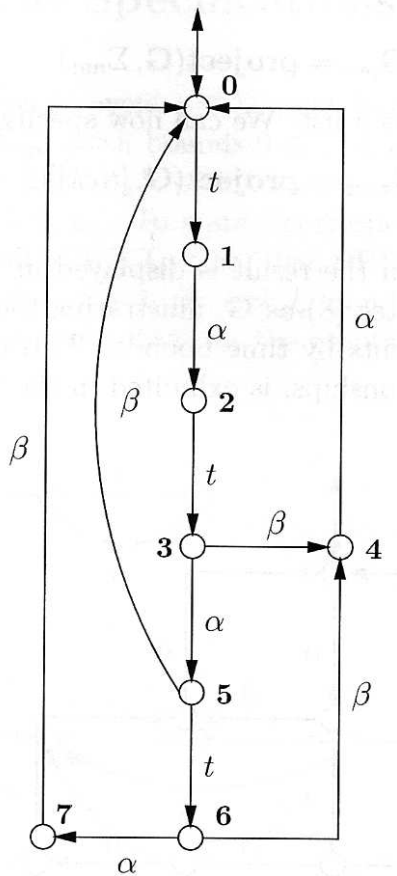


Fig. 9.3.1
Timed transition graph, Ex. 1

State (node of TTG):	0	1	2	3	4	5	6	7
Components $[t_\alpha, t_\beta]$:	[1,3]	[0,2]	[1,2]	[0,1]	[0,3]	[1,1]	[0,0]	[1,0]

Composition of TDES - §9.6

- As in untimed DES, we want to model a plant modularly. We then combine them using the composition operator, **comp.**

- Let $\mathbf{G}_1$ and $\mathbf{G}_2$ be TDES over alphabets Σ_1 and Σ_2, where $\Sigma_i = \Sigma_{i,\text{act}} \cup \{tick\}$.

- The *composition*

$$\mathbf{G} = \mathbf{comp}(\mathbf{G}_1, \mathbf{G}_2) = (Q, \Sigma, \delta, q_o, Q_m)$$

  is defined as follows:

  - The alphabet is: $\Sigma = \Sigma_1 \cup \Sigma_2$

  - The activity structure of $\mathbf{G}$ is:

$$\mathbf{G_{act}} = \mathbf{sync}(\mathbf{G}_{1,\mathbf{act}}, \mathbf{G}_{2,\mathbf{act}})$$

Composition of TDES - II

- We now need to define the time bounds for the non-tick events in $\mathbf{G}$.
- For non-tick events unique to a TDES $(\sigma \in (\Sigma_{1,\text{act}} - \Sigma_{2,\text{act}}) \cup (\Sigma_{2,\text{act}}) - \Sigma_{1,\text{act}})$, the time bounds (l_σ, u_σ) are unchanged from their original definition.
- For non-tick events in common $(\sigma \in \Sigma_{1,\text{act}} \cap \Sigma_{2,\text{act}})$ we define the new time bounds as follows:

$$(l_\sigma, u_\sigma) = (\max(l_{1,\sigma}, l_{2,\sigma}), \min(u_{1,\sigma}, u_{2,\sigma}))$$

  as long as $l_\sigma \leq u_\sigma$.
- If this condition is violated for any $\sigma \in \Sigma$, then the composition $\mathbf{G}$ is undefined.
- As long as event sets are explicitly defined, then **comp** is associative.

Composition of TDES - III

- **WARNING:** The result of **comp**$(\mathbf{G}_1, \mathbf{G}_2)$ is not always the same as **sync**$(\mathbf{G}_1, \mathbf{G}_2)$.
- Forcing synchronization on the *tick* event places a constraint on how the components interact that is often unrealistic in many applications.
- It could even cause *temporal deadlock* (no events possible, including *tick*) as in example in §9.7.

Controllability of TDES - §9.8

- We start by defining a set of events that can be disabled; means the events can be prevented from ever occurring, thus must be remote events.

- **Defn:** We introduce a new type of event called prohibitible events , where $\Sigma_{\text{hib}} \subseteq \Sigma_{\text{rem}}$. These are events that can be disabled by a supervisor.

- **Defn:** We introduce another new type of event called forcible events, where $\Sigma_{\text{for}} \subseteq \Sigma_{\text{act}}$. These are events that can preempt a *tick* of the clock. ie. they can be forced to occur before the next *tick* of the clock, if needed.

- If $\mathbf{G}$ is in a state where the *tick* event is possible, and a forcible event is possible, then our supervisor can "disable" the *tick* event, knowing that at a forcible event will occur if needed to prevent the clock from "stopping."

Controllability of TDES - II

- **Defn:** we define *uncontrollable* events as follows:

$$\Sigma_{\text{unc}} := \Sigma_{\text{act}} - \Sigma_{\text{hib}}$$

- We define controllable events as:

$$\Sigma_{\text{con}} := \Sigma - \Sigma_{\text{unc}} = \Sigma_{\text{hib}} \cup \{\mathit{tick}\}$$

- Before we can give the definition of controllable languages, we first need to define the *eligibility operator.*

- **Defn:** For $L \subseteq \Sigma^*$, we define the eligibility operator $\text{Elig}_L : \Sigma^* \to \text{Pwr}(\Sigma)$ as follows:

$$\text{Elig}_L(s) := \{\sigma \in \Sigma \,|\, s\sigma \in L\}, \quad \text{for } s \in \Sigma^*$$

- From now on, we will use the term *eligible* in the sense above (ie. extends term to also include *tick*).

Controllable Languages and Nonblocking

- To put things into prospective, we give an alternate version of the **untimed** controllability definition.

- **Defn:** A language $K \subseteq \Sigma^*$ is *controllable* with respect to $\mathbf{G}$ if:

$$(\forall s \in \overline{K} \cap L(\mathbf{G}))\, \mathsf{Elig}_{L(\mathbf{G})}(s) \cap \Sigma_u \subseteq \mathsf{Elig}_{\overline{K}}(s)$$

- For TDES, the closed and marked behavior are defined as normal as is nonblocking.

- **Defn:** For timed DES, we define an arbitrary language $K \subseteq L(\mathbf{G})$ to be *controllable* with respect to $\mathbf{G}$ if, for all $s \in \overline{K}$,

$$\mathsf{Elig}_{\overline{K}}(s) \supseteq \begin{cases} \mathsf{Elig}_{L(\mathbf{G})}(s) \cap (\Sigma_{\mathsf{unc}} \cup \{\mathit{tick}\}) & \text{if } \mathsf{Elig}_{\overline{K}}(s) \cap \Sigma_{\mathsf{for}} = \emptyset \\ \mathsf{Elig}_{L(\mathbf{G})}(s) \cap \Sigma_{\mathsf{unc}} & \text{if } \mathsf{Elig}_{\overline{K}}(s) \cap \Sigma_{\mathsf{for}} \neq \emptyset \end{cases}$$

TTCT and Creating Plant Model

- The TTCT program has two sets of functions, one for ATGs (activity transition graphs) and one for TTGs (Timed transition Graph).
- Use the ATG functions to enter in ATGs and then to compose the plant components together to create the monolithic plant model.
- The result is a new ATG, as long as the lower and upper bound constraint haven't been violated.
- Select the "Timed Graph" function to translate the resulting ATG into a TTG.

TTCT and Creating Supervisors

- For supervisors, you can create them as ATGs, or TTGs.
- If you create multiple supervisors as ATGs, you can combine them using compose, or convert them individually to TTG, and use the TTG **meet** function to combine them.
- Once every thing is in TTG form, use the TTG functions.
- A supervisor can be created directly as a TDES by using the TTG **create** function.
- Use the TTG **meet** command to combine the plant and supervisor TDES, to construct the closed loop behavior.
- The TTG menu contains many functions that are analogous to the untimed DES functions of the same name. See the manual that comes with the software for more information.

Controllability, Nonblocking, optimality, and TDES

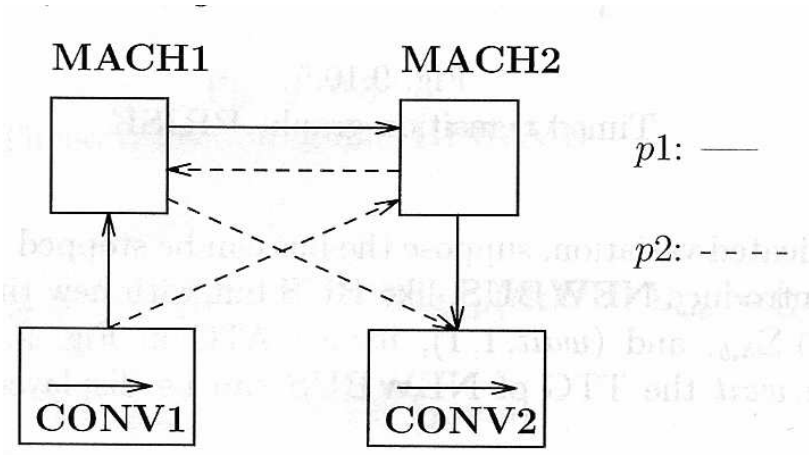
- As in untimed DES, there exists a supremal controllable and nonblocking sublanguage for a given specification language, $E \subseteq \Sigma^*$.
- If the language E is represented by the TDES **EDES**, use TTCT's TTG **supcon** function.
- To verify nonblocking, use the TTG **nonconflict** command as normal.
- To verify a supervisor is controllable for a given plant TDES, use the TTG **condat** command.
- It will display which events are to be disabled at a given state, as well as when an event must be forced.

Controllability, Nonblocking, and TDES - II

- As usual, an odd number represents controllable events, thus any even numbered events listed means the supervisor is uncontrollable.
- If *tick* (event 0) is disabled, it lists the events available to be forced (they are followed by an "f"). If the list is empty, the supervisor is uncontrollable.
- If designing *modular supervisors,* combine them into a single supervisor, and then verify that it is controllable for the plant, and that the closed loop behavior is nonblocking.
- See the manual that comes with the software. It contains a small design example that illustrates the key functions of the software.

TDES Design Example - §9.11

- Read Example 4 (§9.10) on your own.
- We now examine the manufacturing cell shown in block diagram below.
- It consists of machines **MACH1**, **MACH2**, and input conveyor **CONV1** and an output conveyor **CONV2**.



- Each machine can process two types of parts, $p1$ and $p2$. Each can break down and be repaired.

TDES Design Example - II

- The ATG for each machine is shown below as well as the timed events.

- Event α_{ij} means "**MACHi** starts work on a pj-part."
- Event β_{ij} means "**MACHi** finishes working on a pj-part."
- Events λ_i, μ_i represent breakdown and repair of **MACHi**.

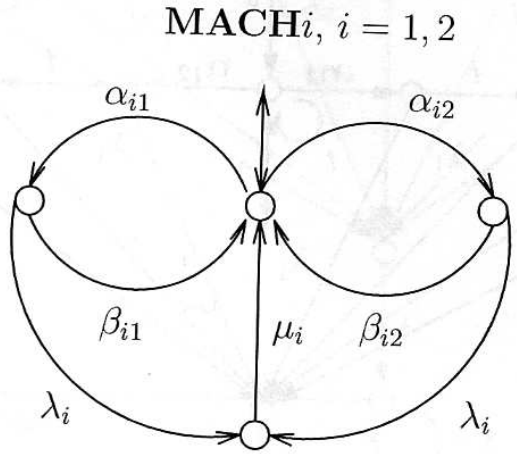


Fig. 9.11.2
Numerically controlled machines

MACH1 : $(\alpha_{11}, 1, \infty)$ $(\beta_{11}, 3, 3)$ $(\alpha_{12}, 1, \infty)$ $(\beta_{12}, 2, 2)$ $(\lambda_1, 0, 3)$ $(\mu_1, 1, \infty)$

MACH2 : $(\alpha_{21}, 1, \infty)$ $(\beta_{21}, 1, 1)$ $(\alpha_{22}, 1, \infty)$ $(\beta_{22}, 4, 4)$ $(\lambda_2, 0, 4)$ $(\mu_2, 1, \infty)$

TDES Design Example - III

- We choose the following event partitions:

$$\Sigma_{\text{for}} = \{\alpha_{ij} | \, i, j = 1, 2\}, \;\; \Sigma_{\text{unc}} = \{\lambda_i, \beta_{ij} | \, i, j = 1, 2\}$$

$$\Sigma_{\text{hib}} = \Sigma_{\text{for}} \cup \{\mu_1, \mu_2\}$$

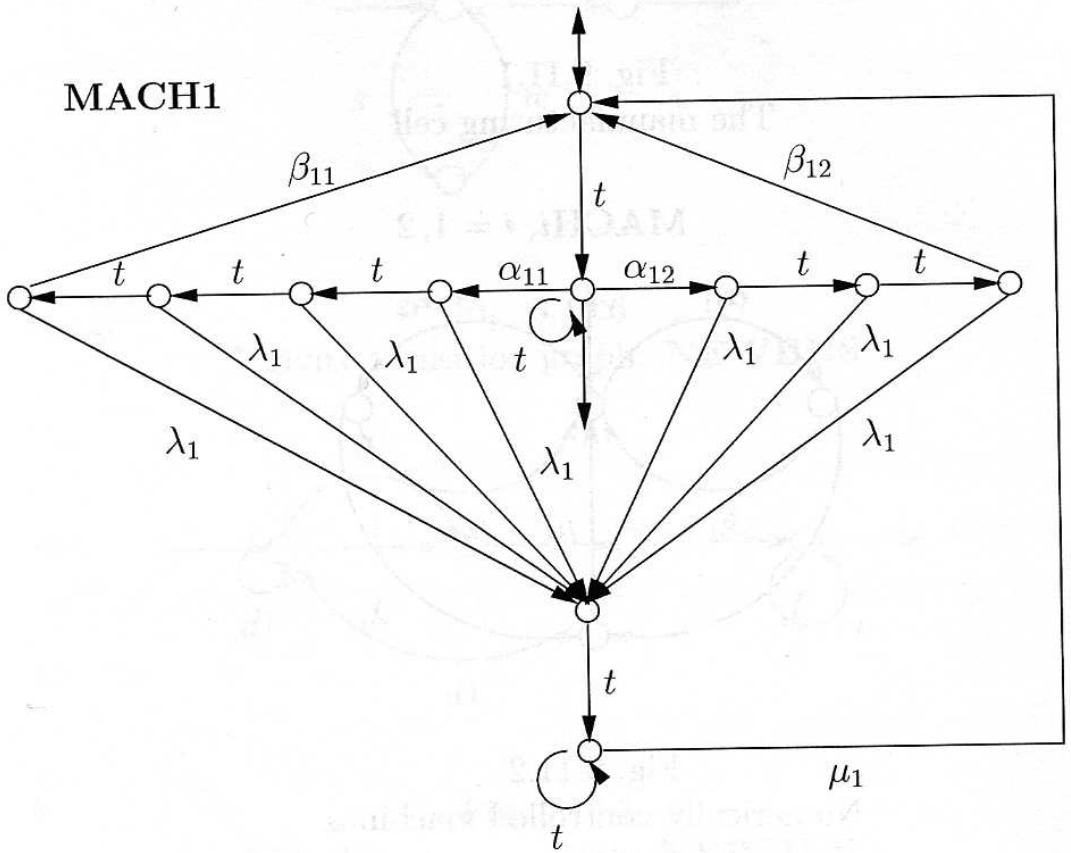


Fig. 9.11.3
The TTG of **MACH1**

TDES Design Example - IV

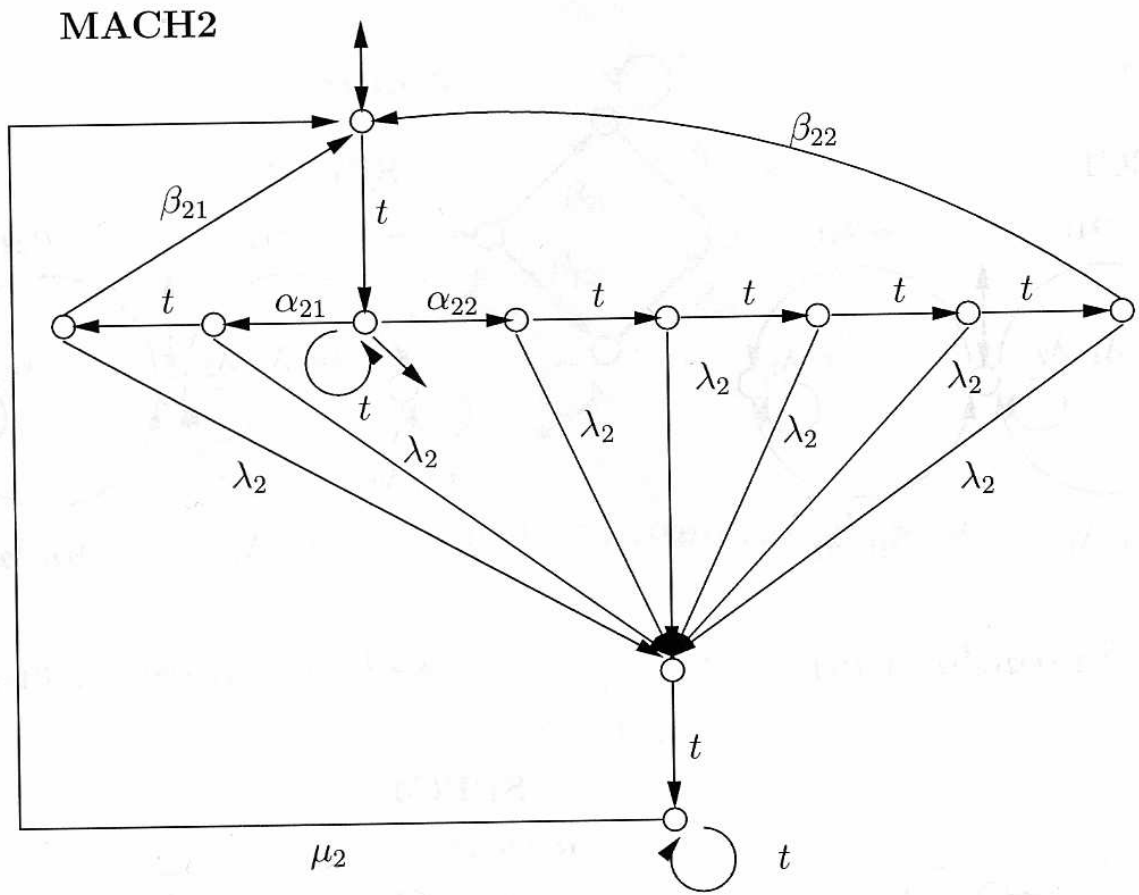


Fig. 9.11.4
The TTG of **MACH2**

TDES Design eg.: Specifications

- We first impose *logic based* specifications:

  (i) - a given part can be processed by only one machine at a time.
  - a $p1$-part must be processed first by **MACH1** then by **MACH2**.
  - $p2$-part must be processed first by **MACH2** then by **MACH1**.
  - one $p1$-part and one $p2$-part must be processed in each production cycle.
  - if both machines down, repair **MACH2** first.

- Then a *temporal* specification:

  (ii) in the absence of breakdown/repair events, a production cycle must be completed in at most 10 time units.

- Finally, a quantitative optimality specification:

  (iii) subject to *(ii)*, production cycle time is to be minimized.

TDES Design eg.: Specs 1-4

- **SPEC1-SPEC4** represent first three points of *(i)*.

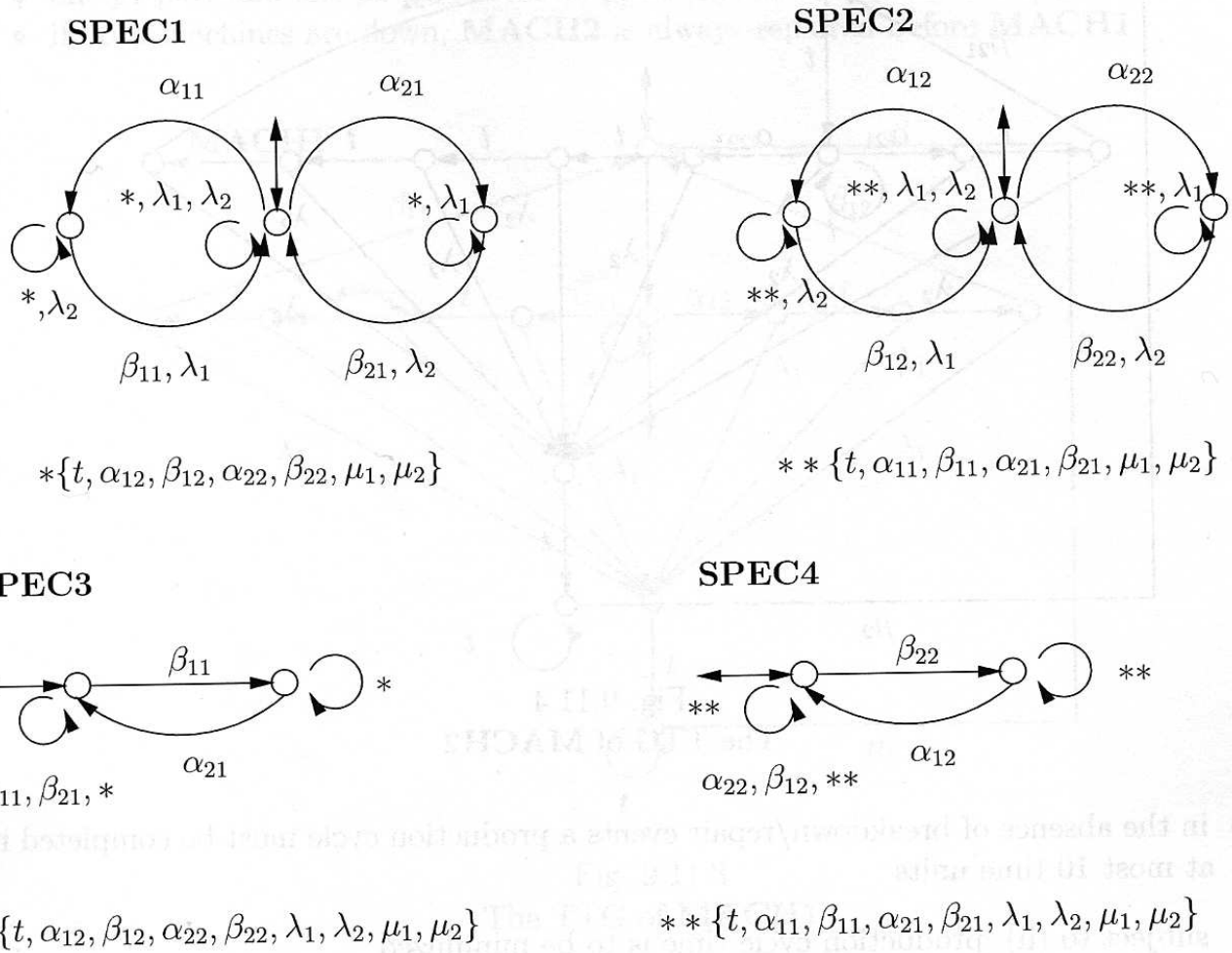
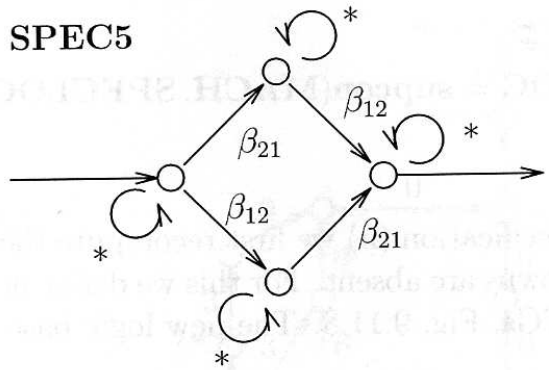


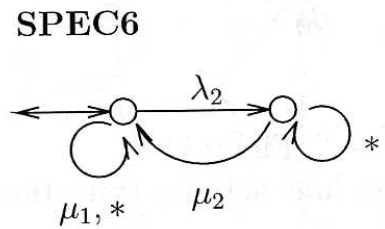
Fig. 9.11.5
SPEC1 – SPEC4

TDES Design eg.: Specs 5-6

- Fourth point of *(i)* enforced by **SPEC5** and fifth point by **SPEC6**.



$$*\{t, \alpha_{11}, \beta_{11}, \alpha_{12}, \alpha_{21}, \alpha_{22}, \beta_{22}, \lambda_1, \lambda_2, \mu_1, \mu_2\}$$



$$*\{t, \alpha_{11}, \beta_{11}, \alpha_{12}, \beta_{12}, \alpha_{21}, \beta_{21}, \alpha_{22}, \beta_{22}, \lambda_1\}$$

TDES Eg: Logic-Based

- Can show that **SPEC3-SPEC4** actually enforce **SPEC1-SPEC2** as well.

- We thus define our logic-based spec as:

$$\textbf{SPECLOG} = \textbf{meet}(\textbf{SPEC3}, \textbf{SPEC4}, \textbf{SPEC5}, \textbf{SPEC6})$$

- We define our plant to be:

$$\textbf{MACH} = \textbf{comp}(\textbf{MACH1}, \textbf{MACH2})$$

- **NOTE:** The **comp** operator is equivalent to the TTCT's ATG composition of the ATG, followed by the ATG "Timed Graph" function to the resulting ATG.
- We then construct the centralized supervisor for **MACH** as:

$$\textbf{SUPLOG} = \textbf{supcon}(\textbf{MACH}, \textbf{SPECLOG})$$

TDES Eg: Temporal Spec

- For our temporal spec, we need to remove breakdowns/repairs. To do this, we create the simplified machine models **NMACHi** shown below.

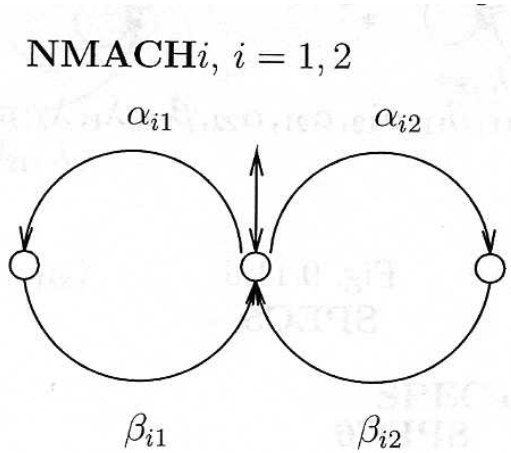


Fig. 9.11.8
The new machine activity transition graphs

- Giving us our new plant:

$$\textbf{NMACH} = \textbf{comp}(\textbf{NMACH1}, \textbf{NMACH2})$$

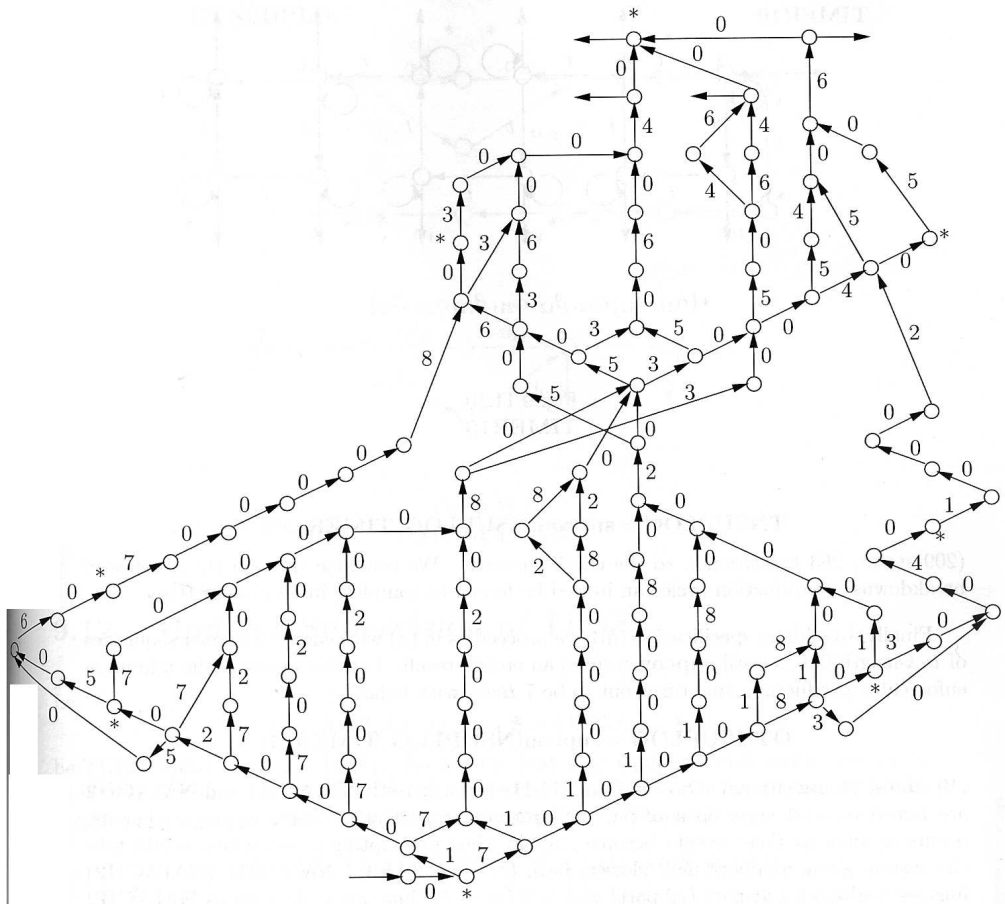
- Our new logic-based specification becomes:

$$\textbf{NSPECLOG} = \textbf{meet}(\textbf{SPEC3}, \textbf{SPEC4}, \textbf{SPEC5})$$

TDES Eg: Temporal Spec - II

- And our new supervisor:

$$\mathbf{NSUPLOG} = \mathbf{supcon}(\mathbf{NMACH}, \mathbf{NSPECLOG})$$

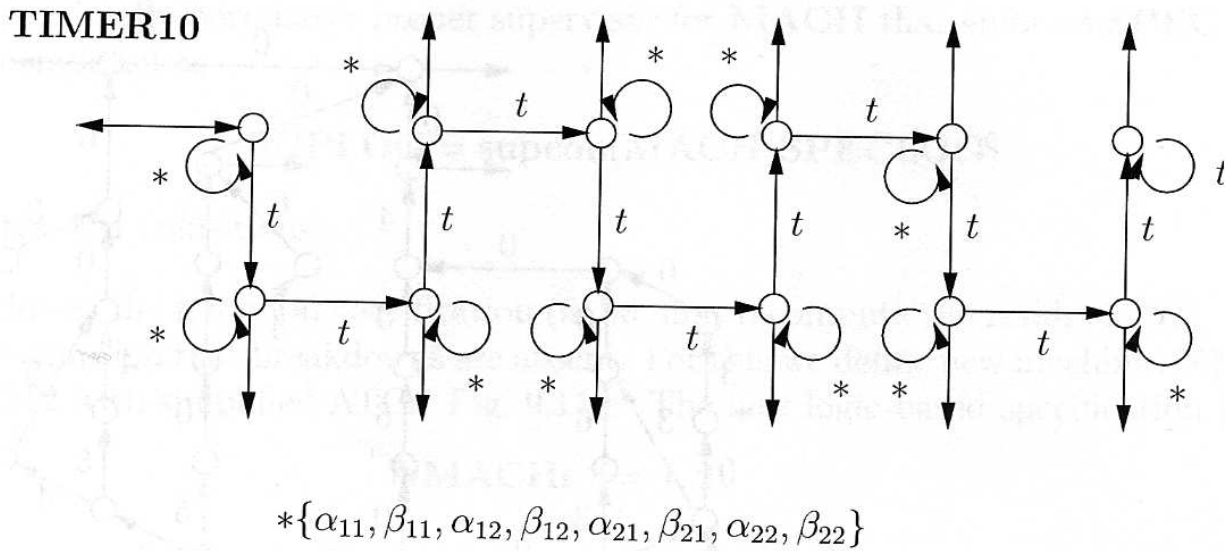


$1 : \alpha_{11}, 2 : \beta_{11}, 3 : \alpha_{12}, 4 : \beta_{12}, 5 : \alpha_{21}, 6 : \beta_{21}, 7 : \alpha_{22}, 8 : \beta_{22}, 0 : tick, * :$ selfloop in $tick$.

Fig. 9.11.9
NSUPLOG

TDES Eg: Temporal Spec - III

- To enforce temporal spec *(ii)*, we use the TDES **TIMER_10** which says no events can occur after the eleventh *tick*.
- As all states marked, system can stop at any time sooner.



- This is then implemented by:

$$\textbf{TNSUPLOG} = \textbf{supcon}(\textbf{NSUPLOG}, \textbf{TIMER\_10})$$

which is (209 states, 263 transitions) thus we conclude that *(ii)* can be satisfied.

TDES Eg: Quantitative Optimality

- To handle the quantitative optimality spec *(iii)*, we try different timers as in **TIMER_10** with progressively shorter sequences.
- ie. must complete by at most nine *ticks*, then eight, etc. until the result of **supcon** is empty.
- Turns out the minimum enforceable time is seven ticks or less, with closed loop behavior:

$$\mathbf{OTNSUPLOG} = \mathbf{supcon}(\mathbf{NSUPLOG}, \mathbf{TIMER\_7})$$

TDES Eg: Quantitative Optimality - II

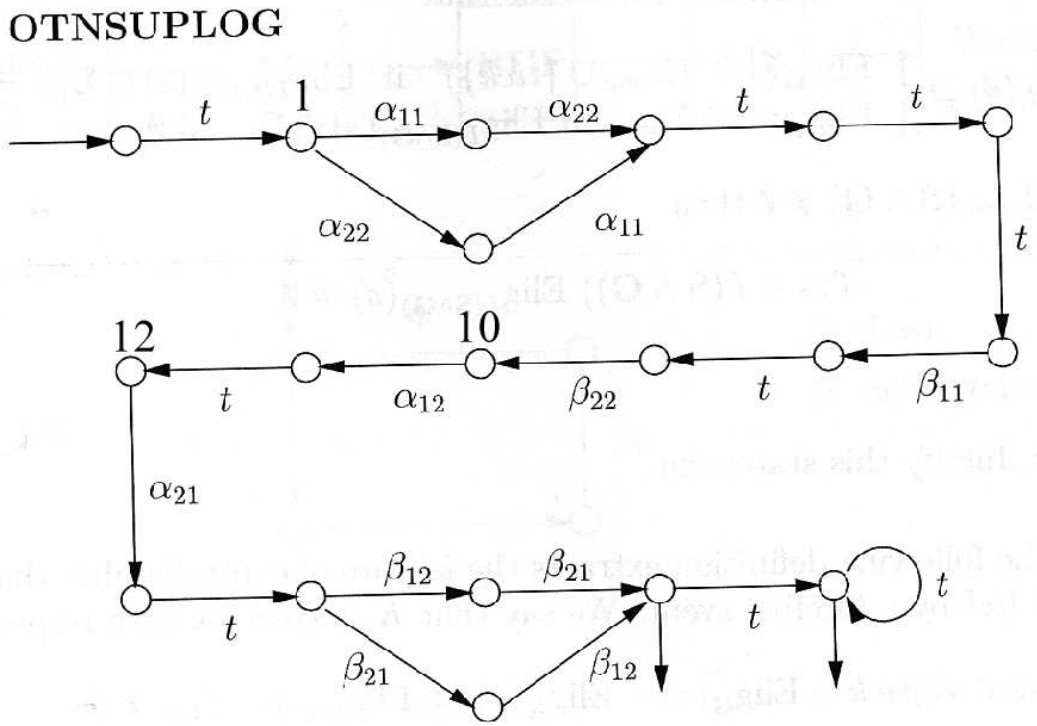


Fig. 9.11.11
OTNSUPLOG

Alternate TDES Modelling

- So far, we have modelled our plant as ATG and time bounds, and then used the **comp** operator.
- Alternatively, we can model all the plant components as TDES from the beginning and combine them using the synchronous product operator.
- To do this, we have to also require the plant satisfy:
- **Defn:** TDES $\mathbf{G}$ has *proper time behavior* if

$$(\forall q \in Q_r)(\exists \sigma \in \Sigma_u \cup \{\tau\})\ \delta(q, \sigma)!$$

where Q_r is the set of reachable states for $\mathbf{G}$.

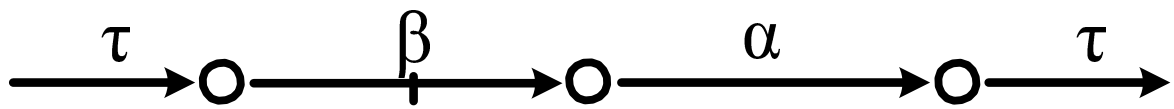
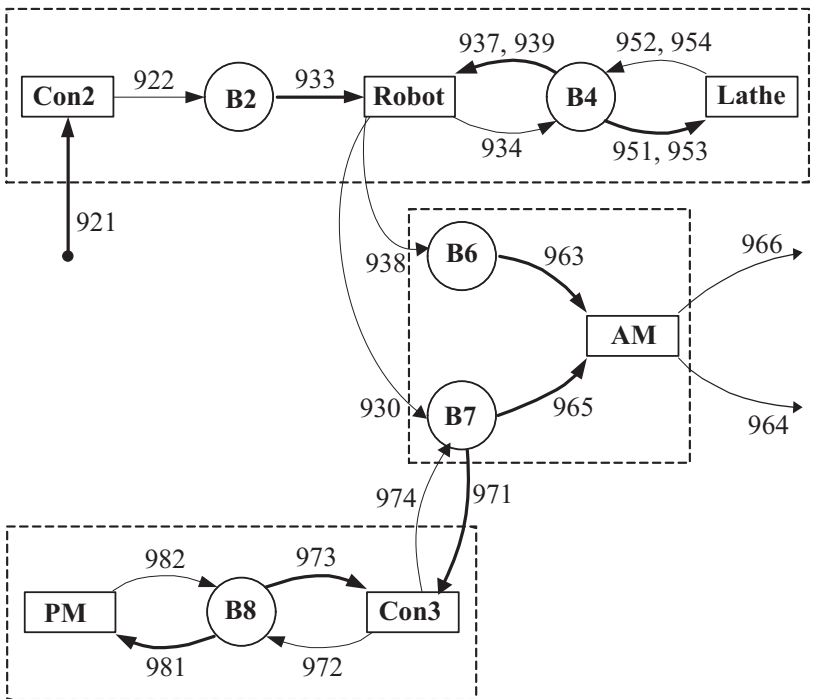


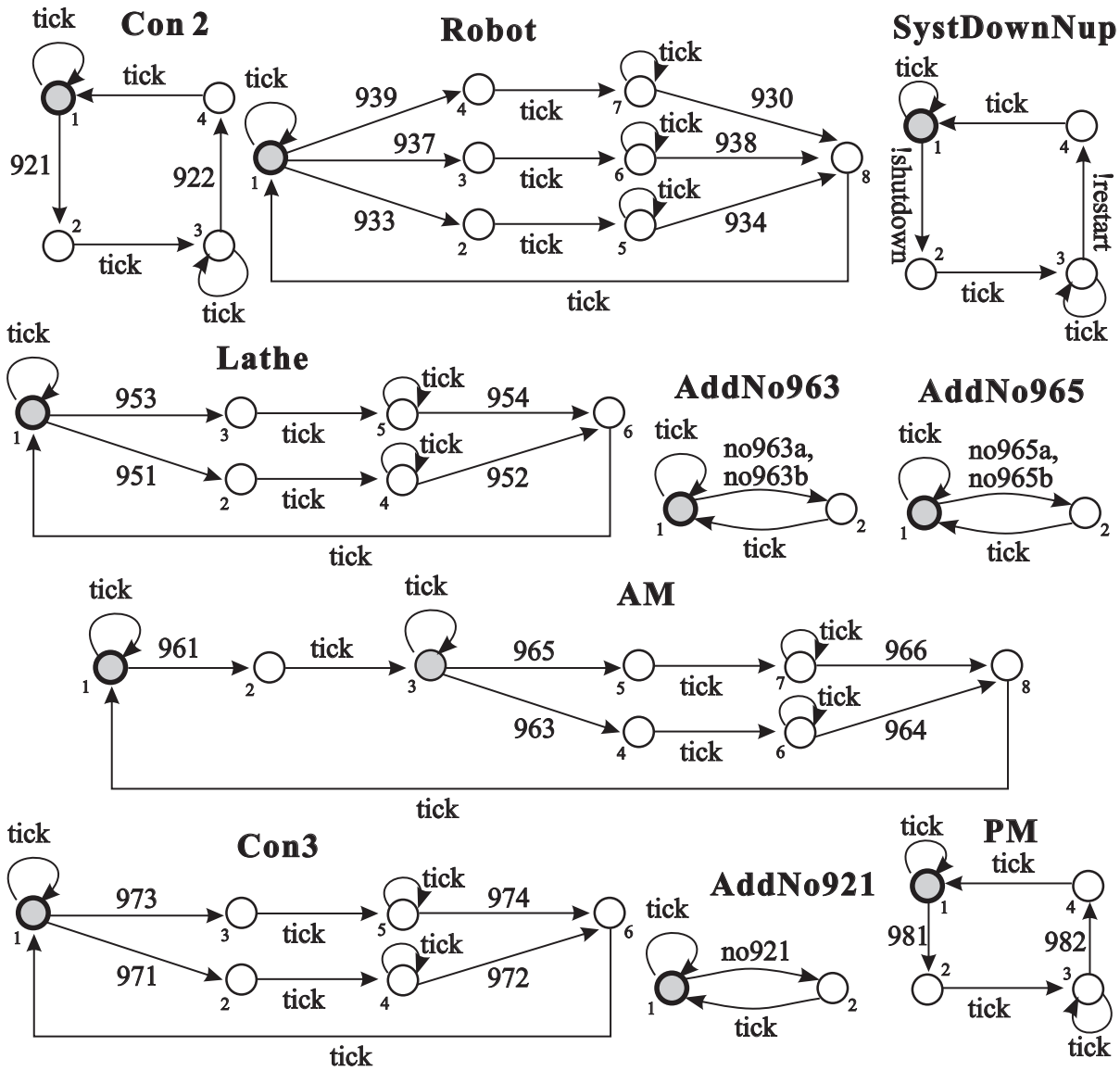
Figure : Non Proper Time Behavior Example

Flexible Manufacturing System Example



Label	Meaning	Label	Meaning
921	Part enters system	*922*	Part enters B2
933	Robot takes from B2	*934*	Robot to B4
937	B4 to Robot for B6	*939*	B4 to Robot for B7
938	Robot to B6	*930*	Robot to B7
951	B4 to Lathe (A)	*953*	B4 to Lathe (B)
952	Lathe to B4 (A)	*954*	Lathe to B4 (B)
971	B7 to Con3	*974*	Con3 to B7
972	Con3 to B8	*973*	B8 to Con3
981	B8 to PM	*982*	PM to B8
961	Initialize AM	*963*	B6 to AM
965	B7 to AM	*966*	Finished from B7
964	Finished from B6		

Plant Models



Supervisors