Joint Advanced Student School 2008, St. Petersburg

Supervisory Control of Manufacturing Systems with Time Specifications

Final Report by

Alexander Schaub

born on 21.02.1985
address:
Ferdinand-Millerstraße 9
82256 Fürstenfeldbruck
Tel.: 08141 290591

Institute of
AUTOMATIC CONTROL ENGINEERING
Technische Universität München

Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss
Univ.-Prof. Dr.-Ing. Sandra Hirche
Univ.-Prof. Dr.-Ing. Olaf Stursberg

Supervisor: Dipl.-Ing. Tina Paschedag
Submitted: 07.04.2008

Contents

Chapter 1

Introduction

The intension of [3] is to introduce a formal constructive method to decide first if a stable supervisory control exists for a given manufacturing system. Moreover, the second step is to compute the stable supervisory control if it exists. It is even possible to state every 'safe' sequence that will not harm any of the given specifications and constraints.

Nevertheless, a few requirements must be conformed by the manufacturing systems in order to make the introduced procedure of finding the stable supervisory control possible.

First of all the manufacturing system is flexible and consists of reconfigurable processors, whereas processor denotes rather an executing unit than a cpu, and buffers with limited capacities. Flexible in this case means that the system can execute different tasks on different types of parts, whereas the processor must be reconfigured to perform another kind of task. This reconfiguration process takes a certain time and a possible example could be a setup process of a computer-numerical-control (CNC) machine. Before executing another task the tools must be changed e.g. for a rougher milling.

Another requirement is the fixed supply and demand rate, which characterizes the dynamics of the system. The supply rate expresses how many new parts of a kind enter the system per time and the demand rate denotes the output of the single types of parts. Since the rates are static, time constraints are entailed on the processor. Input buffers must be emptied and output buffers must be filled in due time.

Stability of such a manufacturing system means that the capacity of the individual buffers of the system must not be exceeded during operation.

The task of the supervisory controller is to achieve the stability of the system and to ensure that the specifications are maintained. It decides when to start processing which part and when to reconfigure a processor.

An important requirement is the hard-real-time characteristic. This means that such a manufacturing system must guarantee the completion of a given task by a specific deadline. It is not sufficient to execute a task properly, but the time bounds must be kept, too. The supply rate of one part per t_1 minutes and demand rate of one part per t_2 minutes imply that the intake or delivery of a part must occur in narrow time slots. An earlier or later point in time is strictly forbidden. The hard-real-time approach to manufacturing is not usual, but provides some advantages according to [3]:

1. Manufacturing resources can be more efficiently allocated due to the known exact time and the known duration of utilization of a device.
2. The fact that the completion by specific deadlines of a manufacturing tasks is guaranteed permits a more efficient planning of enterprise-wide activities.
3. The whole production flow becomes more manageable as problems caused by tardiness of task execution can be avoided.

1.1 Discrete-Event Systems

The underlying approach is based on the theory of discrete-event systems (DESs)(see [18],[15]). A DES can be described as a tuple:

$$G = (\Sigma, Q, \delta, q_0, Q_m) \tag{1.1}$$

Here Σ denotes a finite alphabet of event labels. Those events indicates e.g. the occurrence of a physical phenomenon that caused a change in state. Furthermore, Q is the set of all states q of the DES.
δ stands for the activity transition function and maps a state in combination with an event to a state, which can be the same state, $\delta : \Sigma \times Q \to Q$.
A transition is a triple $[q, \sigma, q']$ with $q, q' \in Q$ and $\sigma \in \Sigma$. This results in $a' = \delta(\sigma, a)$.
Moreover, $q_0 \in Q$ denotes the initial state and $Q_m \subseteq Q$ is the subset of marker states. Marker states are for examples states that should be entered during a process to achieve its goal.

1.2 Further Readings

The state of research of this topic is discussed in [3] and many further readings are given. First of all nearly the complete theory of TDES was developed by W.M. Wonham and his colleagues e.g. see [2], [15], [14], [19], and [18].
According to [3] the control of manufacturing systems has been an intensive research field for nearly the last fifty years with its origin found in [7], [9]. Those investigations concentrate on the development of methods for managing the activities of various resources to receive a manufacturing system, which operates optimal, whereas there is still a certain tolerance, like missing a deadline by a certain amount of time, permitted. Consequently, there is no hard-real-time. The optimality criteria cover the whole range from minimum processing time to maximum utilization of

resources or to least penalty paid for completing a task before the deadline or for missing the deadline, see e.g.[13], [8], [12], [4], [1], [17] and [10].
In [17] for example the problem of a single-machine earliness-tardiness scheduling for a given common due date with tolerances is investigated. It is optimized by minimizing the mean absolute deviation of job completion times with the due date tolerance varying from job to job. Furthermore, [13] presents a class of manufacturing policies for manufacturing systems with fixed process sequences of parts, with specific rates for producing parts, whereas there is a certain tolerance for deviations, and with considerable reconfiguration times.
Different formal techniques have been developed to model, analyze, and control the behavior of more complex manufacturing systems. Those techniques use for example max-plus algebra [5] or Petri nets [6].
Moreover, a hard-real-time environment is still an open problem for managing manufacturing operations, although the first research works were made in the seventies [11].
Of course many more methods, approaches or theories about controlling manufacturing systems can be found in control literature.

The method, which is discussed in this report, for computing a stable supervisory is based on timed discrete-event systems (TDES). The step from DES to TDES is made at the beginning of the following chapter. After that a few synchronization operations on (T)DES are explained and an overview of the procedure of computing supervisory control is given. In the third chapter a stable supervisory controller for a computer-numerical-control (CNC) machine is computed to illustrate the method. Finally, some conclusions are drawn from the discussed papers.

Chapter 2

Supervisory Control of TDES

2.1 Timed Discrete-Event Systems

The extension from a discrete-event system to a timed discrete-event system (TDES) is made for example in [2].
Firstly, a DES is considered with $G_{act} = (\Sigma_{act}, A, \delta_{act}, a_0, A_m)$. For a better differentiation to the TDES the single components of the DES are symbolized with an additional $'act'$ for activity. And the states and state set are denoted by a respectively A. Activities have a duration in time and events are instantaneous.

A discrete time event *tick* is now added to the finite alphabet of event labels: $\Sigma := \Sigma_{act} \cup \{tick\}$. Every transition label $\sigma \in \Sigma_{act}$ is equipped with a lower time bound $l_\sigma \in \mathbb{N}$ and an upper time bound $u_\sigma \in \mathbb{N} \cup \{\infty\}$.

There are two possible kinds of transition labels $\Sigma_{act} = \Sigma_{spe} \cup \Sigma_{rem}$:

1. prospective events σ_{spe} with $0 \leq l_\sigma \leq u_\sigma < \infty$
2. remote events σ_{rem} with $0 \leq l_\sigma < u_\sigma = \infty$

The time bounds can be interpreted as a delay for l_σ , in communication or in control enforcement, and as a hard deadline for u_σ, burdened by specification or physical constraints [2].

Furthermore, the timed events are a triple $(\sigma, l_\sigma, u_\sigma)$ and $\Sigma_{tim} := \{(\sigma, l_\sigma, u_\sigma) | \sigma \in \Sigma_{act}\}$.
To describe the state set

$$Q := A \times \sqcap\{T_\sigma | \sigma \in \Sigma_{act}\} \tag{2.1}$$

a time interval T_σ is defined as:

$$T_\sigma = \begin{cases} [0, u_\sigma] & \text{if } \sigma \in \Sigma_{spe} \\ [0, l_\sigma] & \text{if } \sigma \in \Sigma_{rem} \end{cases} \tag{2.2}$$

Consequently, a state is described as:

$$q := \{a, \{t_\sigma | \sigma \in \Sigma_{spe}\}\} \text{ with } a \in A, t_\sigma \in T_\sigma \tag{2.3}$$

For a prospective event σ every activity a, in which the occurrence of an event σ is defined $\delta_{act}(\sigma, a)!$, needs a separate state at every discrete time point until the upper time bound is reached. For remote events it is only important to know when the l_σ is reached and every time step after the lower time bound does not need to be considered separately as $u_\sigma = \infty$. The initial state is

$$q_0 := \{a_0, \{t_{\sigma 0} | \sigma \in \Sigma_{spe}\}\} \text{ with } a_0 \in A, t_{\sigma 0} := \begin{cases} u_\sigma & \text{if } \sigma \in \Sigma_{spe} \\ l_\sigma & \text{if } \sigma \in \Sigma_{rem} \end{cases} \tag{2.4}$$

Whereas $t_{\sigma 0}$ is the default value of a timer. The marker state subset contains all marker states, which consist of marker activities in combination with valid timers.

$$Q_m \subseteq := A_m \times \sqcap\{T_\sigma | \sigma \in \Sigma_{act}\} \tag{2.5}$$

Although the state transition function looks similar to the DES case a few more rules must be defined concerning the transitions. The time event *tick* only occurs at real time moments $t = n$ with $n \in \mathbb{N}$. There are three cases that a transition $\delta(\sigma, q)$ can be taken:

1. $\sigma = tick$ and no deadline of a prospective event in q is zero. Otherwise the prospective event would occur.

2. σ is prospective, a is the current activity state $q = (a, \_)$, there exists an activity transition $\delta_{act}(\sigma, a)!$ and the timer is within the bounds $0 \leq t_\sigma \leq u_\sigma - l_\sigma$.

3. σ is remote, a is the current activity state $q = (a, \_)$, there exists an activity transition $\delta_{act}(\sigma, a)!$ and the timer $t_\sigma = 0$.

An event is *enabled* if $\delta(\sigma, a)!$ and *eligible* if in addition $\delta(\sigma, q)!$, which cannot happen prior to l_σ *ticks*. In a TDES only *eligible* events can occur. If $\sigma \in \Sigma_{act}$ is *enabled* the timer t_σ is decreased with every *tick*, whereas the *tick* has no influence to any activity component, until

- it reaches zero.

- or σ is disabled because of the occurrence of some eligible transition.

- or the time is reseted to its default value with the occurrence of σ.

Finally the tuple of a TDES is obtained by combination of all preceding definitions:

$$G = (\Sigma, Q, \delta, q_0, Q_m) \tag{2.6}$$

2.2 Extending a DES to a TDES

The impact of introducing discrete time is visualized by the following example. Firstly, there is just a simple DES $G_{act} = (\Sigma_{act}, A, \delta_{act}, a_0, A_m)$ (see Figure 2.1) with:

- $\Sigma_{act} = \{\alpha, \beta\}$, $a_0 = 0$
- $\delta_{act}(\alpha, 0) = \delta_{act}(\beta, 0) = 0$
- $A = A_m = \{0\}$

An entering arrow symbolizes the initial state and a leaving arrow denotes a marker state. This DES consists just of a single state $'0'$ with a self-loop that is taken at the occurrence of either the event α or β.

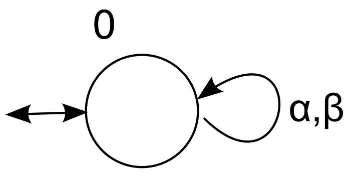


Figure 2.1: DES

Now both events are considered as timed events. The lower time bound $l_{\sigma\alpha}$ of alpha is 1 and the upper time bound $u_{\sigma\alpha}$ is 1, too. Consequently, α must happen after the first *tick* and before the second *tick* after being activated. The time slot of β is a bit larger, as it has to occur between the second and the third *tick* after being enabled. The resulting tuples of the timed events are $(\alpha, 1, 1)$ and $(\beta, 2, 3)$.
The state set of generator G does not consist of only one state but eight states. $Q = \{'0'\} \times \{0, 1\} \times \{0, 1, 2, 3\}$ means that the activity $'0'$ can be combined with two discrete times of t_α and four discrete times of t_β. Hence, the state set of the TDES generator has the size $|Q| = 8$.
An *enabled* but not *eligible* event is called *pending*. The event *tick* is always preempted by the events $\alpha, \beta \in \Sigma_{act}$ if one of those has an imminent deadline. The resulting TDES is shown in Figure 2.2 and the timer values for each state can be looked up in Table 2.1. Considering the event α, it is *pending* at states 0, 2, 5, 7 and *eligible* at states 1, 3, 4, 6. The event β is *pending* at 0, 1, 2, 4 and eligible at 3, 5, 6, 7.

State	0	1	2	3	4	5	6	7
$[t_\alpha, t_\beta]$	[1,3]	[0,2]	[1,2]	[0,1]	[0,3]	[1,1]	[0,0]	[1,0]

Table 2.1: TDES timers

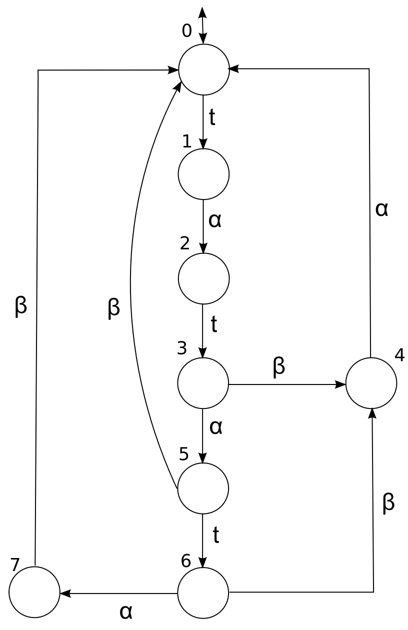


Figure 2.2: TDES

2.3 Supervisory Control of Discrete-Event Systems

A supervisory controller disables certain events in the transition structure of a (T)DES G in order to meet certain specifications. The supervisor is usually represented by an automaton V, which monitors G ([3],[15],[2]).
Considering the disabling of events the alphabet of event labels is divided into *controllable* and *uncontrollable* events $\Sigma_{act} = \Sigma_c \cup \Sigma_u$. Moreover, there are also $forcible$ events Σ_{for}. A $forcible$ event can be forced to occur, as the name says, but the difference to *controllable* events is that they are not necessarily preventable. *Controllable* events can be also $forcible$ as well as *uncontrollable* events can be forcible. In [2] the author gives the example that 'landing' a plane is a $forcible$ event, but you can not prevent it from landing for ever and so it is not a *controllable* event.

The supervisory map s specifies the control input κ for every output sequence of transitions w of G: $\kappa = s(w)$. All possible transition sequences of a (T)DES G are included in its language $L(G)$.
The closed-loop behavior of the system as shown in Figure 2.3 is denoted by $L(V|G) := K$, whereas:

- $\epsilon \in K$, ϵ expresses that no changes are made (null event).
- $w\kappa \in K$ iff $w \in K, \kappa \in V(w), w\kappa \in L$

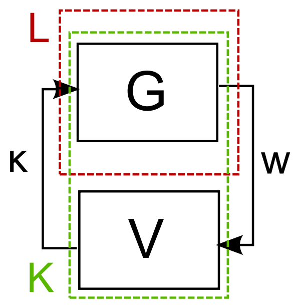


Figure 2.3: Supervisor TDES

The goal is to find the supremal controllable language $K^{\uparrow}$, which is the largest controllable language $K^{\uparrow} \subseteq K$. It is defined as *controllable* if: $\overline{K}\Sigma_u \cap L \subseteq \overline{K}$. This means, that if an arbitrary *uncontrollable* event is combined with the supremal controllable language the intersection with the language of the generator G must still be a subset of the supremal controllable language.
The following example, shown in 2.4 and taken from [16], illustrates this:

- $\Sigma_c = \{\alpha, \beta\}$, $\Sigma_u = \{\lambda\}$, $q_0 := z_1$, $Q_m = \{z_6\}$.
- $L = (\alpha(\alpha\alpha + \beta)(\lambda + \alpha) + \beta(\alpha\lambda + \alpha\alpha + \lambda))\beta^*$. Language of the generator
- $L_m = (\alpha(\alpha\alpha + \beta)\alpha + \beta(\alpha\alpha + \lambda))\beta^*$. Language of the marker states.
- $K^{\uparrow} = (\alpha\alpha + \beta)\lambda\beta^*$.

The only unsafe state is z_5, consequently the desired K should contain all possible paths between all safe states. If K is tested for stability by adding an uncontrollable event, it can be seen that the resulting language is no longer subset of K. An occurrence of λ in state z_4 would lead to the unsafe state and this transition can not be prevented since $\lambda \in \Sigma_u$. Hence, the state z_4 must be excluded from K. The resulting DES, containing z_1, z_2, z_3, z_6, is described by the supremal controllable language $K^{\uparrow}$. β^* in a language expresses that β can occur arbitrarily often, including zero times.
A supervisory controller of TDES must also consider time bounds (l_σ, u_σ) as specifications.
Furthermore, a minimal restrictive supervisor will be computed in the process from [3]. This means that the supervisor disables certain events only if necessary and therefore creates the largest possible subset of legal sequences.
The computation of the supremal controllable language can be executed automatically by a program called TTCT, which was developed by W.M. Wonham, one of the authors of [3], [15], [2] and [18].

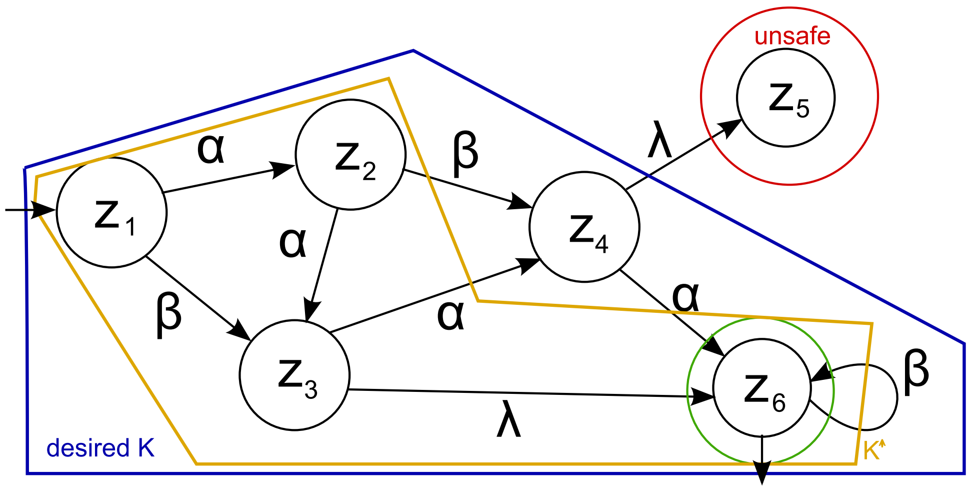


Figure 2.4: Supremal Controllable Language

2.4 Parallelization of Generators

The formal constructive method to compute a supervisory controller, introduced in [3], requires certain synchronization operations on (T)DESs. The first one is the parallelization of generators $G_3 = G_1 || G_2$. In TTCT it is called *sync* or synchronization.

Both alphabets are combined by:

$$\sigma \in \Sigma_{3,act} : \sigma \in (\Sigma_{1,act} - \Sigma_{2,act}) \cup (\Sigma_{2,act} - \Sigma_{1,act}) \tag{2.7}$$

This guarantees that an event contributes to $\Sigma_{3,act}$ only once if it is contained in both $\Sigma_{1,act}, \Sigma_{2,act}$.
Timed events must be synchronizable:

1. $\sigma \in \Sigma_{1,act} \cap \Sigma_{2,act}$
2. $(l_\sigma, u_\sigma) = (\max(l_{1,\sigma}, l_{2,\sigma}), \min(u_{1,\sigma}, u_{2,\sigma}))$

If the maximal lower time bound is higher than the minimal upper time bound the parallelization of the generators is not possible. The *tick* event must occur always at the same continuous time in both automata, which is more a problem of real systems. A detailed definition of *sync* can be found for example in [18].

To illustrate this operator two TDES will be combined by the *sync* operator in the following example, which was described in [2], to model the TDES of both together.
In this scenario a pedestrian walks on the road and a car is approaching very fast. To avoid a collision the pedestrian should jump on the curb. The TDES of the pedestrian is PED $= (\{j\}, \{r, c\}, \{[r, j, c]\}, r, \{c\})$ (Figure 2.5):

- j: is the event "jump on the curb".
- r, c: are the states "on the road" and "on the curb".
- $[r, j, c]$: denotes the transition from the road to the curb by jumping.
- r: is also the initial state. The pedestrian walks on the road first.
- c: the curb is the marker state, which should be reached.
- $\Sigma_{tim} = (j, 1, \infty)$: is the whole notation for the timed event. The pedestrian needs at least one *tick* of reaction time before he jumps and has no upper time bound for jumping.

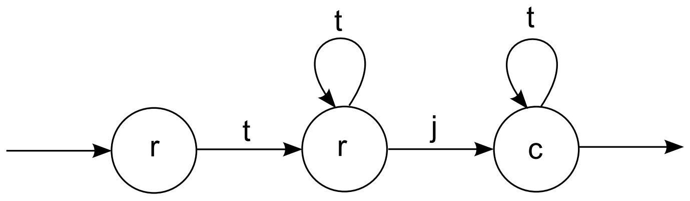


Figure 2.5: TDES of the Pedestrian

The behavior of the car can be expressed by $CAR = (\{p\}, \{a, g\}, \{[a, p, g]\}, a, \{g\}$ (Figure 2.6):

- p: is the event "car passes".
- a, g: are the states "car is approaching" and "car has gone by".
- $[a, p, g]$: denotes the transition from the approach to being gone by after the occurrence of passing.
- a: is also the initial state. The car is approaching first.
- g: is the marker state.
- $\Sigma_{tim} = (p, 2, 2)$: is the whole notation for the timed event. Nothing can prevent the car from passing after two *tick*.

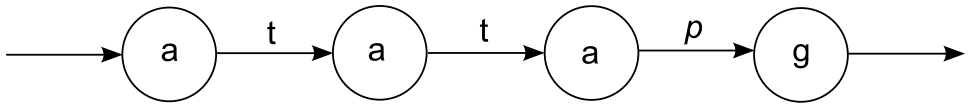


Figure 2.6: TDES of the Car

Both TDES are now combined to model the whole scenario $CP = CAR||PED$, with $\Sigma_{for} = \{j\}$. This TDES is shown in Figure 2.7, whereas the red state is a strongly undesired state, which stands for the case that the car passes before the pedestrian jumps and consequently hits him. Moreover it can be seen, that the car still must pass after two *tick* and the pedestrian can jump after one *tick*.

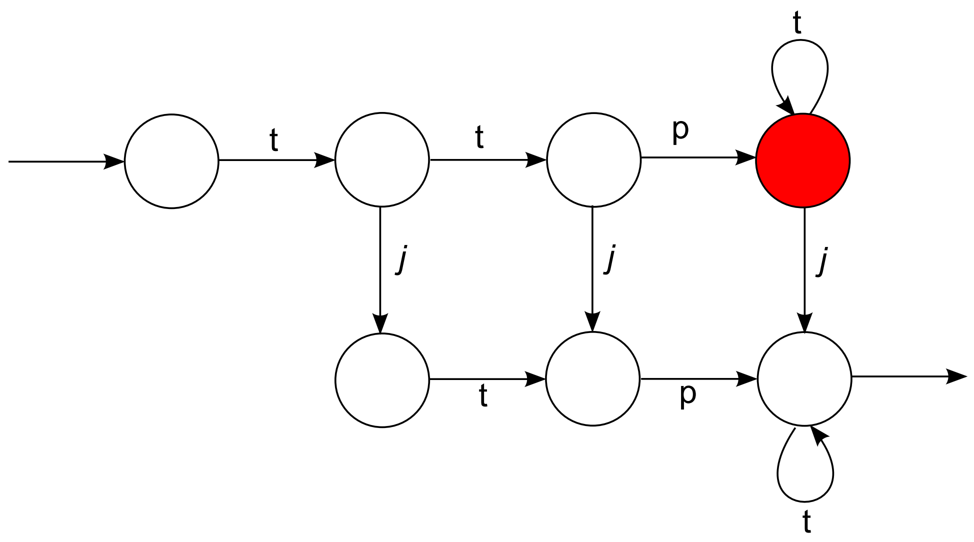


Figure 2.7: Combined TDES

2.5 Restriction of Synchronization on Common Symbols

The second operator, denoted by *meet* in TTCT, is a restriction of synchronization on common symbols and denoted by $G_3 = G_1 \sqcap G_2$. If two TDESs are combined with this operation the resulting TDES will consist only of states and transitions that exist in each parent TDES. The *meet* is described as a special case of *sync* with $\Sigma_1 = \Sigma_2$ in [2]. Every event that is not contained in both G_1 and G_2 will be blocked by *meet*.

The endangered pedestrian example from the preceding section is now extended to illustrate the functionality of *meet*. A safety specification is introduced, which expresses that the jump must occur before the car passes. The number of ticks before, between and after those events is arbitrary.
The corresponding TDES is SAVE $= (\{j, p\}, \{s0, s1, s2\}, \{[s0, j, s1], [s1, p, s2]\}, s0, \{s2\})$ (Figure 2.8):

- j, p: are the events "jump" and "car passes" as in the example of Section 2.4.
- $s0, s1, s2$: are just the states before jumping, before passing and and after passing.
- $[s0, j, s1], [s1, p, s2]$: denote the activity transitions between the single states.
- $s0$: is also the initial state. The jump has not happened yet.
- $s2$: is the marker state, which is reached if first j and then p have occurred.

- $\Sigma_{tim} = \{(j, 0, \infty), (p, 0, \infty)\}$ is the whole notation for the timed events. Both events are *remote* and can occur immediately after being activated.

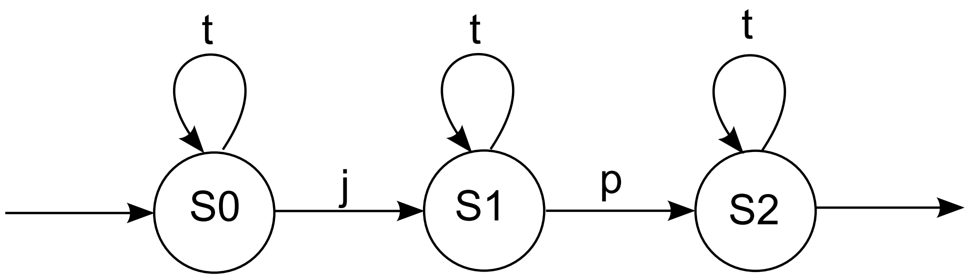


Figure 2.8: TDES of Safety Specification

The *meet* operation is applied to the combined TDES CP (see Figure 2.7), which stands for the car-pedestrian scenario, and the safety specification SAVE: $CPSAVE = CP \sqcap SAVE$.

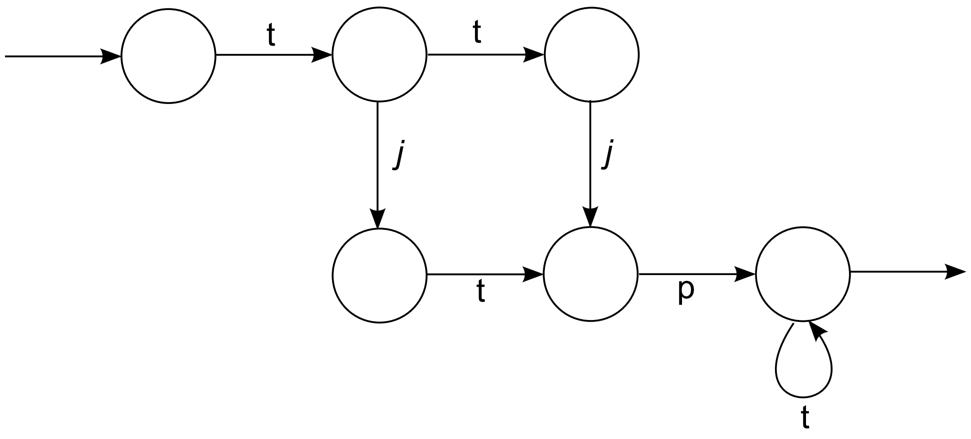


Figure 2.9: $CPSAVE$

Obviously, the red state and its transitions from Figure 2.7 are not contained in Figure 2.9, as they are not contained in the specifications. The jump will now occur always before the car passes.

2.6 Computation of the Supremal Controllable Sublanguage

The computation of the supremal controllable sublanguage $K^{\uparrow}$ is discussed generally in Section 2.3. It is here mentioned again in the context of the formal constructive method to find a supervisory control for a manufacturing system [3].
The task is to find the supremal controllable sublanguage for a certain TDES generator G and its specification S, which are modeled as TDES as well. It exists a

command *supcon* in TTCT for this task: $V = \Phi(G, S)$.
Every sequence, which is contained in V observes the specification and all undesired transition paths are erased. There is the possibility that V is empty. Consequently, the constraints or specifications were chosen too hard and a supervisory control is simply not feasible. The solution would be an ease of the specifications if possible.
A very general description of the *supcon* operation is given in [3]. Namely, that for a TDES G and its set of specifications S, a minimally restrictive supervisor V can be computed by an iterative algorithm based on a fix point characterization of a certain operator [14].
Moreover, it is possible to show that V is the largest fix point of this operator. The maximum number of iterations needed to compute V is finite, and a worst-case bound can be calculated [14].

2.7 Procedure for Computing a Supervisor

Figure 2.10 shows the procedure, which was developed in [3], for computing a supervisor of a manufacturing system as described in Chapter 1.

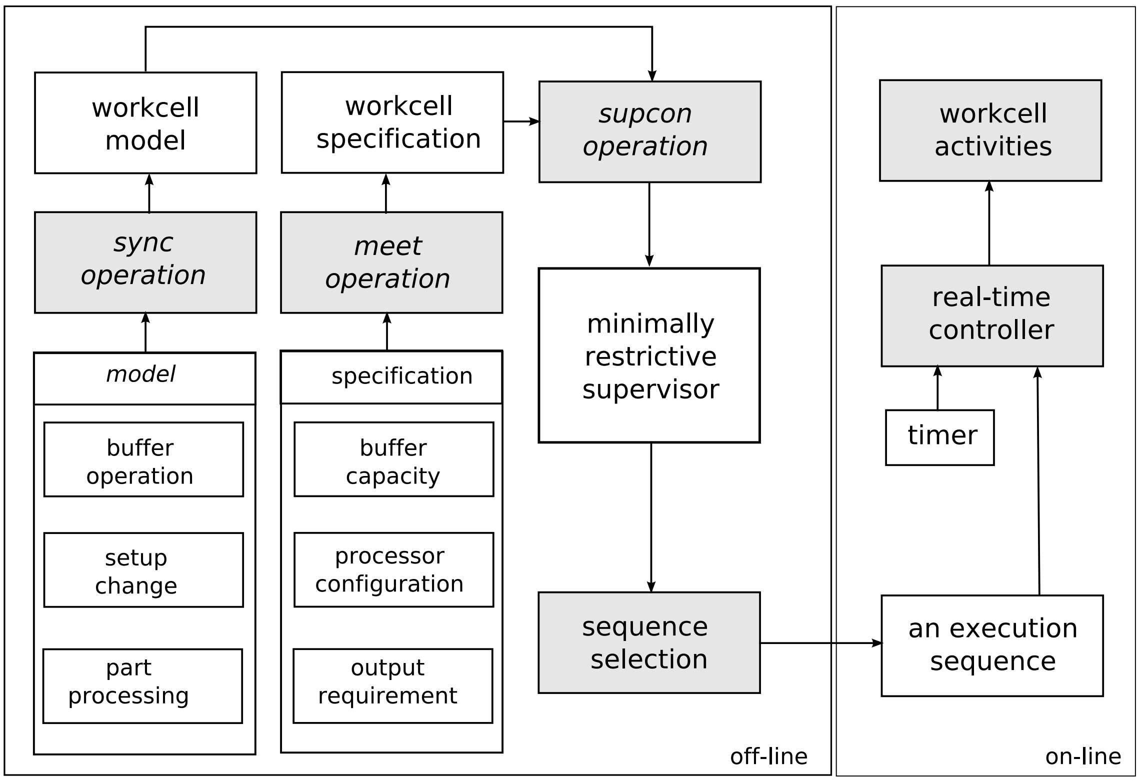


Figure 2.10: Computing Procedure for V

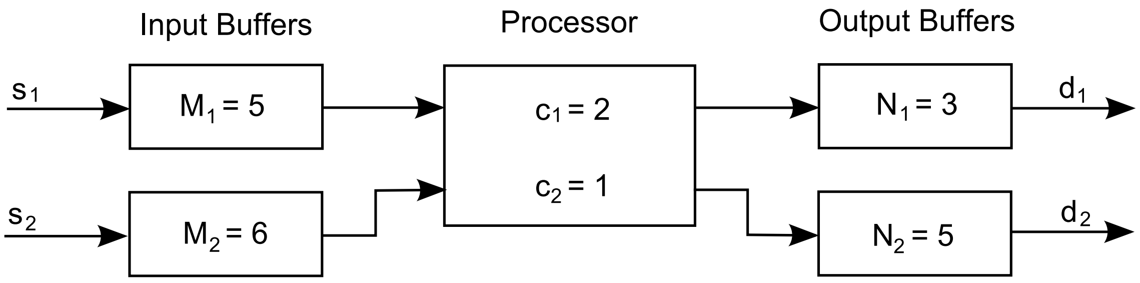
First of all the single parts of the manufacturing system are modeled by TDESs.

The operations of the input and output buffers must be defined. As the processing unit is flexible and it is able to execute different tasks, not only the part processing must be modeled, but also the setup change from one configuration to another. All those different TDESs are combined with *sync* in order to create the workcell model TDES G_w, which contains all single parts of the system simultaneously.
Furthermore, the features and constraints of the system must be specified. First the capacities of the single input and output buffers, as all buffers are limited, are set. To process parts of a given type, the processor must be in the corresponding configuration. To ensure this and to avoid being stuck in an infinite reconfiguration loop [3], the processor configuration must be specified, too. Another important specification is the desired output rate. It is obvious that all specifications must be fulfilled concurrently and hence the *meet* operation combines all single specification TDES to the workcell specification S_w.
After the computation of the workcell model G_w and its specification S_w, the *supcon* is executed for the controller synthesis and produces the TDES of V, which represents the minimally restrictive supervisor of G_w with respect to S_w.
After selecting a sequence from V, it can be executed on-line on a real-time controller to control the workcell activities.

Chapter 3

Supervisory Control for a CNC Machine

In order to illustrate the procedure from Chapter 2.7 an example from [3] is discussed in this chapter. The supervisory control is computed for the CNC machine in Figure 3.1 that processes two different types of parts and has the following settings:



Setup time

from \ to	a_0	a_1	a_2
a_0	0	0	1
a_1	0	0	—
a_2	1	—	0

Figure 3.1: CNC Machine

- Processor P with processing time $c_1 = 2$ for type-1 parts and $c_2 = 1$ for type-2 parts.
- Input buffers F_1 with capacity $M_1 = 5$ and F_2 with capacity $M_2 = 6$.
- Output buffers H_1 with capacity $N_1 = 3$ and H_2 with capacity $N_2 = 5$.

- Input rates: $s_1 = 0.5$ parts/min, $s_2 = 1/3$ parts/min.
- Output rates: $d_1 = 1/3$ parts/min, $d_2 = 0.25$ parts/min.
- The time between two *tick*s is exact one minute.

The setup time table in Figure 3.1 shows the time that is required to reconfigure from one processor configuration to another, whereas a_0 is the neutral and initial configuration. The processor cannot change from a_i to a_j directly. It must reach firstly the neutral state. Hence the reconfiguration actually means a_i to a_0 to a_j.

3.1 TDES of the Model

As described in Chapter 2.7 the TDESs to model the single parts are created. It should be said at this point, that if two or more TDESs have the same structure, e.g. the model of the input buffer 1 G_{F_1} and the one of input buffer 2 G_{F_2}, only one of those will be discussed.
Figure 3.2 shows the input buffer G_{F_1}, whereas α_1 stands for "requesting a raw part of type-1" and β_1 means that it has "physically entered the buffer". Between those two events are two *ticks* and hence the input rate s_1 is 0.5 parts/min. The straight line perpendicular to the transition arrow of α_1 denotes that this event is *controllable*. Again an entering arrow symbolizes the initial state and a leaving arrow denotes a marker state.

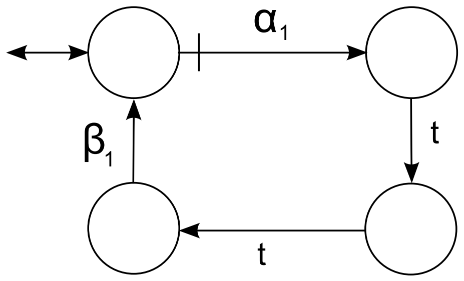


Figure 3.2: Input Buffer G_{F_1}

The TDES model of the output buffer G_{H_1} can be seen in Figure 3.3 and α_3 expresses that the part has "left the buffer" and that it was "fetched" is denoted by event β_3. At the very beginning there is no part available in the output buffer. This fact is modeled by the self-loop for the *tick* event at the initial transition. After the first part has left the buffer this initial state will not be reached again to safe the outputrate $d_1 = 1/3$ parts/min. There must always be three *ticks* between α_3 and β_3.
Furthermore, the TDES of the setup change is shown by Figure 3.4. The event λ_{i0} stands for "start reconfiguration from a_i to a_0" and μ_{i0} that the "reconfiguration is finished". It can be seen that the configuration from a_0 to a_1 and back takes no time

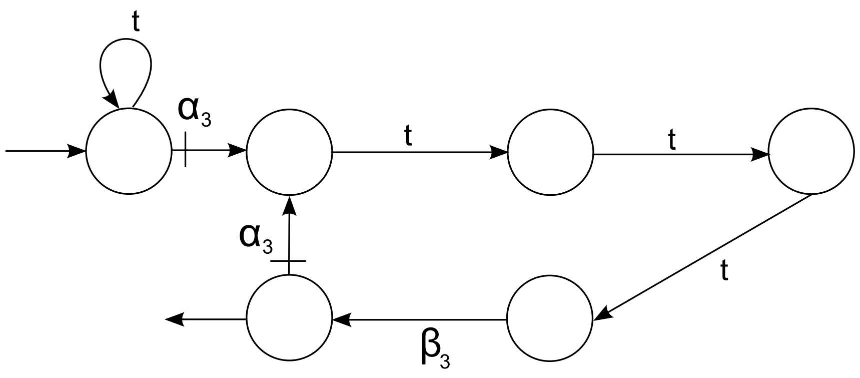


Figure 3.3: Output Buffer G_{H_1}

and from a_0 to a_2 or the other way around one *tick* or one minute, respectively. λ_{ij} is a *controllable* event.

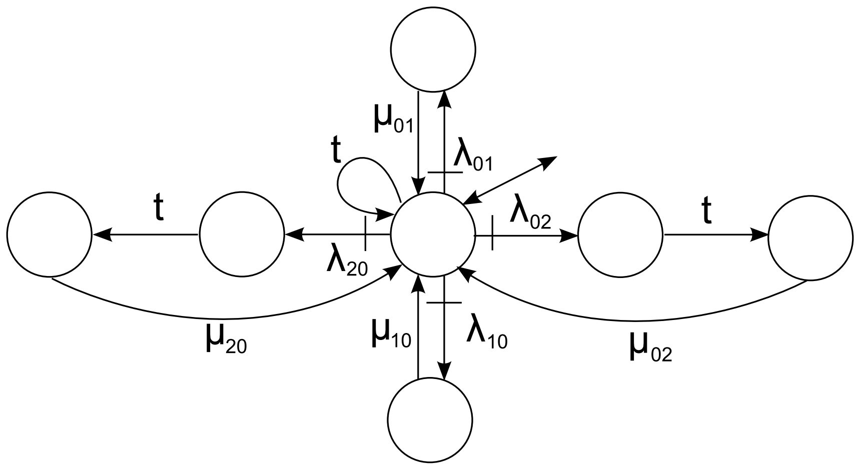


Figure 3.4: Setup Change

The part processing itself is modeled by G_p in Figure 3.5, where as the *controllable* event γ_i denotes the start of "producing part i" and σ_i denotes that the "producing has finished". There are exact c_i *ticks* between those two events , e.g. two for parts of type-1.

3.2 TDES of the Specification

The following TDESs are created to set the specifications of the system. First of all the buffers' capacities are specified like output buffer S_{H_1} in Figure 3.6. The events are of course the same as for the modeling TDESs. Again α_3 means that the part has "left the buffer" and σ_1 denotes that the "producing has finished". As $N_1 = 3$ $\sigma_1 =$ can occur three times before a part must leave the buffer. There

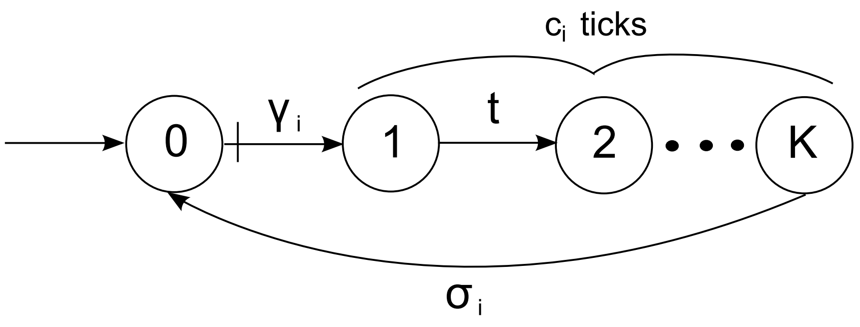


Figure 3.5: Part Processing

are self-loops defined for all other possible events, even if they do not affect the output specification directly. The reason is that if they are not prohibited, they must be defined in order to obtain a correct result of the *meet* operation, since only transitions are considered that are part of all TDESs. At the last state β_3, γ_1 are not defined. As no part of type-1 has recently left the buffer, no type-1 part can be fetched (β_3) and as the buffer is full, it is not allowed to start processing another type-1 part (γ_1).

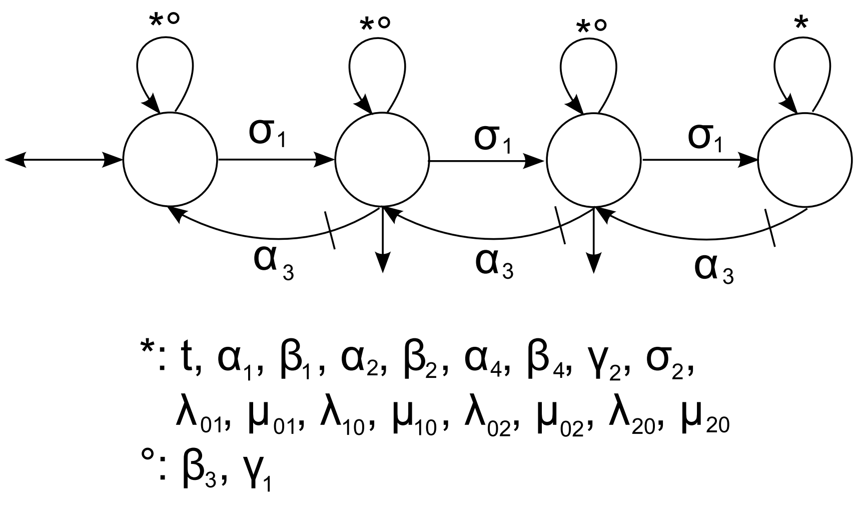

*°
*°
*°
*
σ_1
σ_1
σ_1
α_3
α_3
α_3
*: t, α_1, β_1, α_2, β_2, α_4, β_4, γ_2, σ_2,
λ_{01}, μ_{01}, λ_{10}, μ_{10}, λ_{02}, μ_{02}, λ_{20}, μ_{20}
°: β_3, γ_1

Figure 3.6: Output Buffer Specification S_{H_1}

The specification of the processor configuration S_P can be seen in Figure 3.7. In the initial state nearly all events are defined except γ or σ, as nothing can be produced in this initial configuration. μ_i denotes that "reconfiguration i is finished" and the "start producing part type-i"denotes the transition to the next state. In this state the "completion of the i part processing" σ_i is defined and the processing of new parts of type-i can be started until the processor is "reconfigured to the neutral configuration" λ_{i0}. One remarkable thing of this specification is that it prevents the processor from being stuck in an infinite reconfiguration loop, which is possible as some reconfiguration processes do not take time [3], e.g. a_1 to a_0. Again in every state all transition events that are not prohibited are defined even if just in a self-loop, as this is necessary for the *meet* operation.

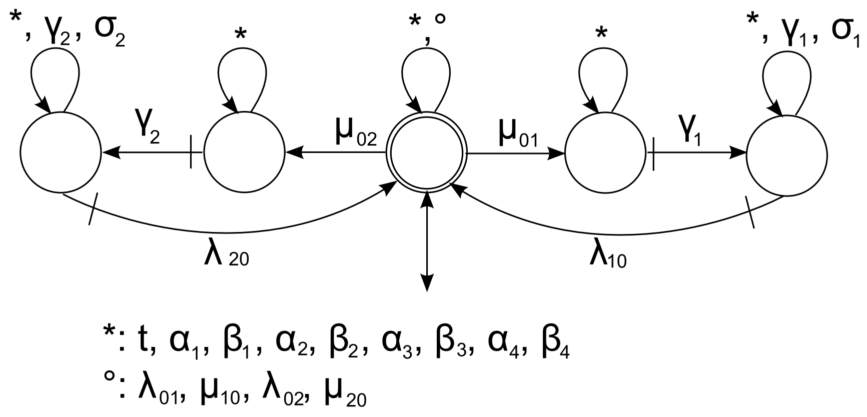
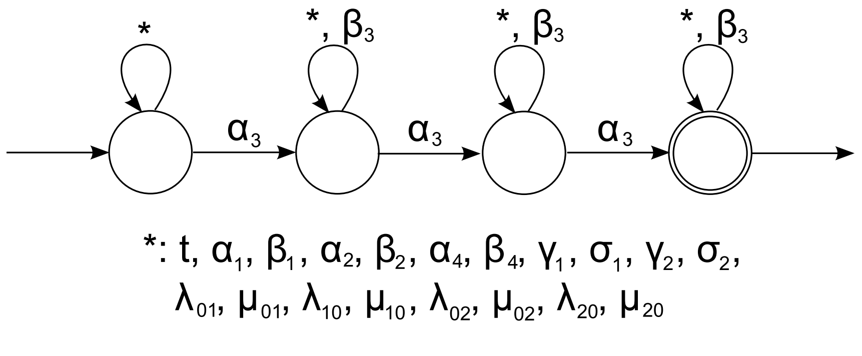


Figure 3.7: Processor Configuration S_P

The output specification for type-1 parts S_{O_1} is shown in Figure 3.7, whereas α_3 stands for "part of type-1 has left the outputbuffer" and β_3 means that it was "fetched". The states of this TDES look very similar, but β_3 is not defined in the first state, as if nothing has left the buffer nothing could be fetched. The requirement for on-time delivery can be expressed as:

$$\gamma_i(t) = \begin{cases} 0 & t \leq t_0 \\ (t-t_0)d_i & t > t_0 \end{cases} \quad \text{and } \gamma_i(\tau) = D_i \tag{3.1}$$

Whereas t_0 is the time, when the output buffer is activated for the very first time and $\tau \geq t_0 > 0$ is the production period. The marked state is reached after D_i consecutive α_{H_i} events. In this case $D_1 = 3$ parts of type-1 have been delivered at the time, when the marker state is reached. As this TDES will be combined with the other specification TDESs by *meet*, in every state all not prohibited events must be defined.



*
*, β3
*, β3
*, β3
α3
α3
α3
*: t, α1, β1, α2, β2, α4, β4, γ1, σ1, γ2, σ2,
λ01, μ01, λ10, μ10, λ02, μ02, λ20, μ20

Figure 3.8: Output Specification S_{O_1}

3.3 Resulting Supervisor

The first step to receive the resulting supervisor is the parallelization of all the single generators, which model the input and output buffers, the part processing and the setup change.

$$G_w = G_{F_1} \| G_{F_2} \| G_{H_1} \| G_{H_2} \| G_p \| G_r \tag{3.2}$$

The workcell model G_w consists of 35280 states and 129792 transitions.
The next step could be the computation of the workcell specifications S_w by applying the operation of restriction of synchronization on common symbols (*meet*) to all single specifications like input and output buffer capacities, output requirements and processor reconfiguration.

$$S_w = S_{F_1} \sqcap S_{F_2} \sqcap S_{O_1} \sqcap S_{O_2} \sqcap S_{H_1} \sqcap S_{H_2} \sqcap S_P \tag{3.3}$$

But in [3] the approach is a bit different, as the computation of the supremal controllable sublanguage (*supcon*) is applied iteratively first to G_w and the processor reconfiguration specification S_p, then the resulting TDES is combined with the *meet* of the input buffer specifications by *supcon* and so on:

$$V = \Phi(\Phi(\Phi(\Phi(G_w, S_p), S_{F_1} \sqcap S_{F_2}), S_{H_1} \sqcap S_{H_2}), S_{O_1} \sqcap S_{O_2}) \tag{3.4}$$

According to [3] this is done due to memory limitations. The resulting supervisor V consists of 2538 states and 5945 transitions.
One possible sequence starts with: $\alpha_1\beta_2\alpha_2\sigma_1\gamma_1 tt\beta_1\alpha_1\sigma_1\lambda_{10}\mu_{10}\lambda_{02}t$...

3.4 Simulated Activities

In this section the activities of the simulated system from [3], which is controlled by the supervisor V, are shown. The following Figures 3.9-3.12 show the level of type-1 part input buffer, level of type-2 part input buffer, level of type-1 part output buffer and level of type-2 part output buffer. It can be easily seen that the capacities are never exceeded and thus the specifications are held.

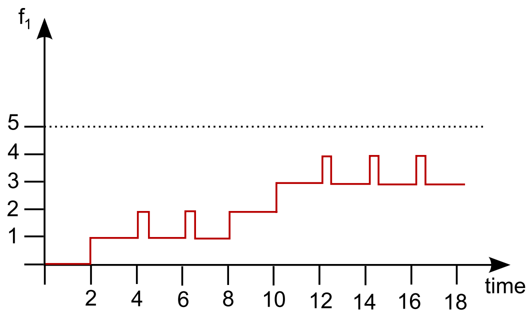


Figure 3.9: Level of type-1 Part Input Buffer

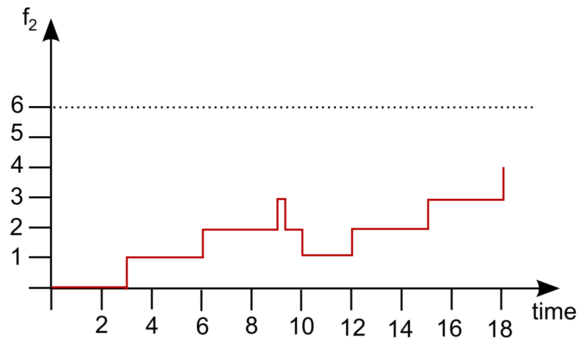


Figure 3.10: Level of type-2 Part Input Buffer

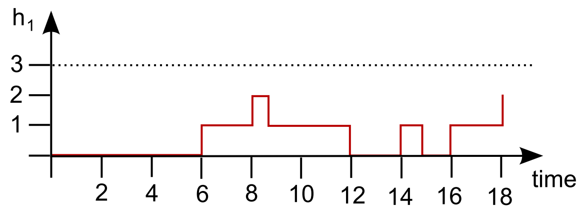


Figure 3.11: Level of type-1 Part Output Buffer

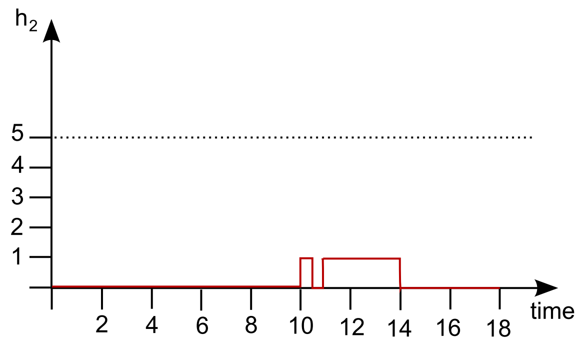


Figure 3.12: Level of type-2 Part Output Buffer

The processor activity under supervisory control is illustrated in Figure 3.13. First of all the processor configures from the initial neutral setup a_0 to a_1, which is instantaneous as shown in Figure 3.1, and then starts processing parts of type-1. At time step 8 it reconfigures back into the neutral configuration to be able to change to a_2, what takes one *tick*. The CNC machine is now processing parts of type-2 until it reconfigures from a_2 to a_0 and then into a_1 to process type-1 parts again.

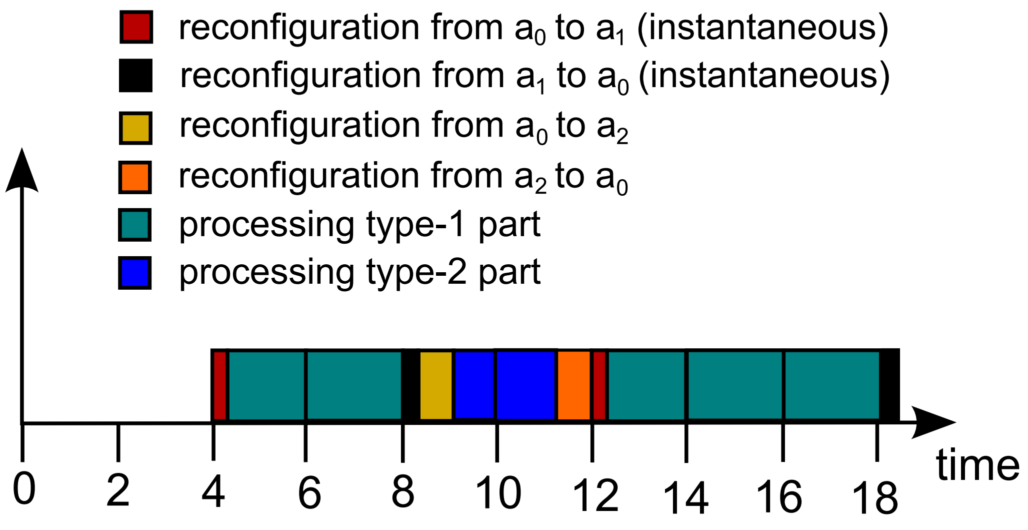


Figure 3.13: Processor Scheduling

Chapter 4

Conclusions

The formal constructive method, which is introduced in [3] and based on the theory of TDES ([2], [15], and [18]), is able to decide firstly if a stable supervisory control exists for a given manufacturing system and if it exists, the method has the ability to find the supervisor V, which contains all safe sequences.
Furthermore the supervisor is a minimally restrictive controller. It only erases transitions if it is absolutely necessary. Consequently, there is an optimization possibility of the supervisor by adding another specification like e.g. that the workcell should produce a certain mix of parts with a minimum number of reconfigurations [3].
An advantage is that the computation of the supremal controllable sublanguage is made in polynomial time [2].
Unfortunately, there is an exponential increase of the number of states of a composite TDES by using *sync*. A solution, which is suggested in [3], for this problem is called modular synthesis. This means a set of concurrently operating modular supervisors. Every part of the system has its own supervisor and the goal is to achieve an overall result as if there was just a single global supervisor.

List of Figures

List of Tables

Bibliography

[1] K. Asano and H. Ohta. Single machine scheduling using dominance relation to minimize earliness subject to ready and due times. *International Journal of Production Economics*, (44), 1996.

[2] B.A. Brandin and W.M. Wonham. Supervisory control of timed discrete-event systems. 1994.

[3] P.C.Y. Chen and W.M. Wonham. Stable supervisory control of flexible manufacturing systems with fixed supply and demand rates. 2001.

[4] T. C. E. Cheng and A. Janiak. Resource optimal control in some single-machine scheduling problems. *IEEE Transactions on Automatic Control*, (39), 1994.

[5] G. Cohenm, D. Dubois, J.P. Quadrat, and M. Viot. A linear-system-theoretical view of discrete-event processes and its use for performance evaluation in manufacturing. *IEEE Transactions on Automatic Control*, (30), 1985.

[6] A. A. Desrochers and R. Y. Al-Jaar. *Application of Petri Nets in Manufacturing Systems*. 1997.

[7] J.R Jackson. Scheduling a production line to minimizemaximum tardiness. *Management Science Research Project Report of the University of California, Los Angeles.*, (43), 1955.

[8] N. I. Karacapilidis and C. P. Pappis. Optimal due date determination and sequencing of n jobs on a single machine using the slk method. *Computers in Industry.*, (21), 1993.

[9] E. L. Lawler. Recent results in the theory of machine scheduling. *In Mathematical Programming: The State of the Art.*, 1983.

[10] K.R. Lipske. A greedy-based decision support system for scheduling a manufacturing operation. *Production and Inventory Management Journal*, (37), 1996.

[11] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, (20), 1973.

[12] P. B. Luh and D. J. Hoitomt. Scheduling of manufacturing systems using the lagrangian relaxation technique. *IEEE Transactions on Automatic Control*, (38), 1993.

[13] J. R. Perkins and P. R. Kumar. Stable, distributed, real-time scheduling of fleexible manufacturing/assembly/disassembly systems. *IEEE Transactions on Automatic Control.*, (34), 1989.

[14] P.J.G. Ramagde and W.M. Wonham. On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, (25), 1987.

[15] P.J.G. Ramagde and W.M. Wonham. The control of discrete-event systems. 1989.

[16] O. Stursberg. *Ereignisdiskrete und Hybride Systeme*. 2007.

[17] M.X. Weng and J.A. Ventura. Single-machine earliness-tardiness scheduling about a common due date with tolerances. *International Journal of Production Economics*, (42), 1996.

[18] W.M. Wonham. *Supervisory control of Discrete-Event Systems*. 2006.

[19] W.M. Wonham and P. J. Ramagde. Modular supervisory control of discrete event systems. *Mathematics of Control, Signal and Systems.*, (1), 1988.