1

Basic Description of Discrete-event Dynamic Systems

1.1 Introduction

People observe various phenomena of nature and endeavor to comprehend them. The first step in that is a reflection of the phenomena by imagination and description. The reflexive process is a process of abstraction. In this process, the notion of “system” is of basic importance.

A system is defined to be a group of objects separated from the universe and having mutual relations.

Different physical entities can constitute system objects. If time is included among the system objects, their temporal properties or the system dynamics can be considered. The system dynamics is given by the time behavior of the system objects. The behavior is called the process.

A real physical system is represented by an ideal system created by human thinking and understanding. Mathematical representations of real systems are the most abstract and precise descriptions. Since the very beginning of its existence, mankind strives not only to know and to describe natural systems but also to govern and control them.

Control of a system is based on knowledge about the particular system. This knowledge is developed *via* abstraction based on observation of the system. The observation is realized by measurements and if possible, by experimentation with the system. Two main abstractions are to be distinguished, namely:

1. The notion of a continuous system and
2. The notion of a discrete system.

A natural question arises about the substance of these abstractions. A continuous system is specified by a set of continuous variables, a set of continuous functions over the respective domains of these variables, and by derivatives of the variables and functions. Such a system is called a continuous-variable dynamic system (CVDS). One can find a good systematic survey of the CVDS control theory in the book by Jörgl (1993). A discrete system is specified by a set of discrete variables and relations defined on them. A hybrid system is a combination of both.

Sometimes the relations of system variables are not treated with respect to time. Then they describe the static behavior of the studied systems. However, time is mostly involved in the analysis and synthesis of systems and dynamic system behavior is considered.

Figure 1.1 illustrates the classification of continuous and discrete systems considering dynamic behavior. In order to simplify the illustration, a system with one variable is depicted. Figure 1.1.a shows the case when the continuous variable $x(t)$ is a continuous function over a continuous time interval. The function domain and co-domain are real numbers. A system is continuous if it is defined by continuous variables and continuous functions such as the function depicted in Figure 1.1.a. A discrete system is given by discrete variables and discrete functions or relations as illustrated by Figure 1.1.b. The system in Figure 1.1.b consists of one object in the form of one variable that takes on values from the set of real numbers in discrete time points.

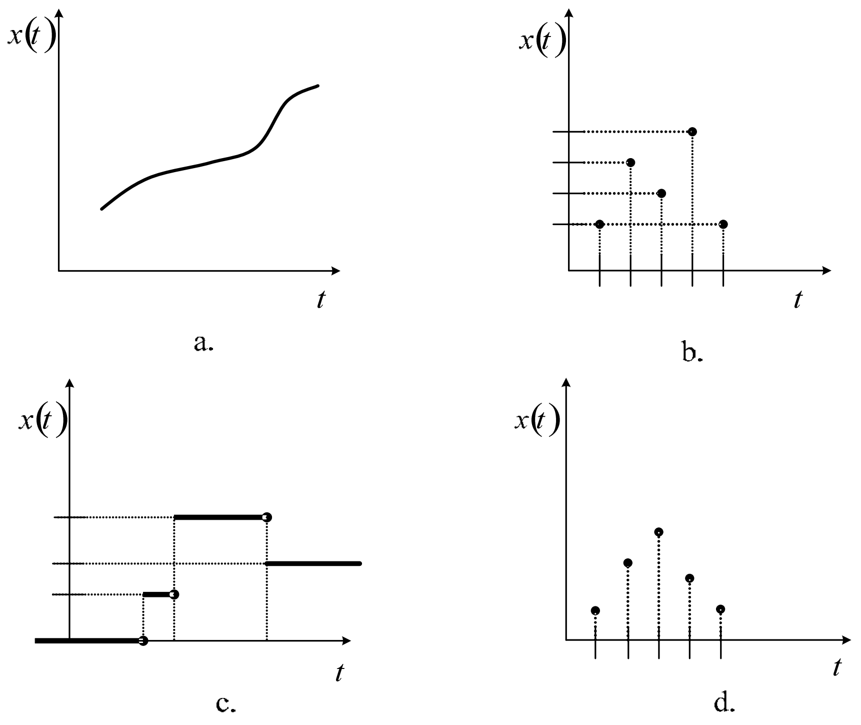

Figure 1.1. Properties of one-variable system

Figures 1.1.c and 1.1.d show the mixed/hybrid cases when the system is semi-continuous. Usually, a system has more than one object or variable. Then a set of variables can be aggregated into one or more vector variables. Note that the case depicted in Figure 1.1.b can be understood either as a discrete representation of a continuous system or as a representation of the system that is discrete by its nature. The role of the semantics or interpretation is obvious. Therefore, the discrete or digital representation of continuous systems has to be distinguished from the representation of systems that are discrete in their nature and substance. The discrete representation of a continuous system is obtained by sampling continuous variables at discrete time points.

Continuity and discreteness of a system is one aspect of the view on system properties. Another aspect is that CVDS are time-driven systems. The reason for the dynamic development of system states is time. On the other hand, discrete systems can be time-driven or event-driven.

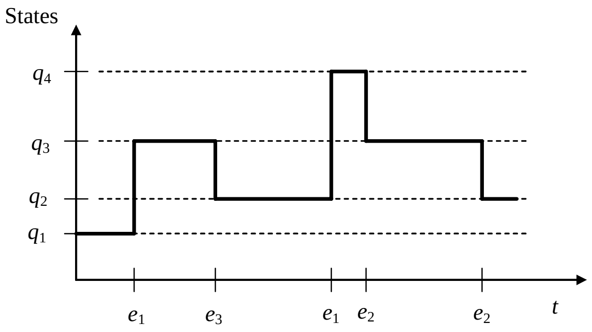

Figure 1.2. An event-driven system

Let us compare Figures 1.1 and 1.2. The discrete variable q describes the state of the system. There are four states q_1, q_2, q_3, and q_4, $q_{1\text{-}4}$ for short, and three events e_1, e_2, and e_3, $e_{1\text{-}3}$ for short. Figure 1.2 shows that the state change is event-driven. The events occur at discrete time points and the state changes depend on the events only. Such systems are called discrete event dynamic systems or DEDS for short (Ho 1991; Ho and Cassandras 1983). They are also called discrete event systems (Ramadge and Wonham 1987; Zhou and DiCesare 1993; Jafari 1995; Bogdan *et al.* 2006). As mentioned earlier, Figure 1.1.b has a double meaning. It can represent either time-driven CVDS or event-driven DEDS when the events occur at discrete time points and cause the change of system states as depicted in Figure 1.1.b.

The applied system analysis and synthesis methods depend on the system nature. In this textbook we will study systems that are fully discrete in their nature and event-driven, *i.e.*, discrete event dynamic systems. Their name expresses their specific character. DEDS are characterized by a set of states which the system can take, and by the set of events that cause the state changes at discrete time points. The events may take place asynchronously as opposed to the synchronous nature in a discrete time system. The change of states and occurrence of events are the essence of the DEDS dynamic behavior.

A primary task of the DEDS theory is creating a DEDS model. Without such a model it would be impossible to analyze and control DEDS just as it is true in classic CDVS control theory. Obviously we are interested in a model that is sufficiently general and includes the DEDS dynamics. There are two ways to consider the dynamics:

1. To specify values of the system variables and system relations in defined discrete time points; and
2. To specify time order of the states or events.

The latter case means that time is not explicitly expressed and only the precedence relations for DEDS states and events are given. The order of events can be determined by means of their indexing. In other words, it is given which event happens before some other events. Such an approach is more abstract and avoids problems related to the time relativity.

The control of DEDS can be designed if there is a DEDS model available. Control engineering design methods perform the following tasks:

- Formulation and specification of the given system control tasks;
- Determination of control algorithms;
- Design of technical means necessary for the control implementation;
- Creation and verification of control programs; and
- Implementation, testing and maintenance of the control system function.

Control engineering is an applied interdisciplinary technical science. To a considerable extent, the solution methods are independent of the technological substance of controlled systems. For a control it is important to achieve such an influence of various agents on the system that parameters and behavior of the system are as required (Kozák 2002; Jörgl 1993). The system behavior and various influences on it are given by physical, chemical, biological or other quantity values. What is important from the viewpoint of control is the information the quantities carry, but not their physical substance.

Automatic control is based on the information manifestations of the system. In other words, a system is described by means of information about the spatial location of objects, time, system parameters, properties, characteristics, *etc*. Time is substantial for the dynamics of events. As mentioned earlier, the time evolution of system variables is called the process and in the context of DEDS, it is called the discrete process.

1.2 Discrete Variables and Relations

The notion of DEDS has been specified in the previous section. It is based on the discrete character of the individual variables and relations. It is useful to study the property of discreteness in some detail.

Definition 1.1. Let D be a finite set of n elements, *i.e.*,

$$D = \{d_1, d_2, ..., d_n\} \tag{1.1}$$

Let v be a variable taking on values only from set D, *i.e.*,

$$v = d_i \in D \tag{1.2}$$

then v is a discrete variable.

Definition 1.2. Let two non-empty finite sets D and E be given:

$$D = \{d_1, d_2, ..., d_n\} \tag{1.3}$$
$$E = \{e_1, e_2, ..., e_m\} \tag{1.4}$$

A binary relation R from D into E is defined by

$$R \subseteq D \times E \tag{1.5}$$

where symbol × denotes the Cartesian product.

If a relation is defined on the sets for which $D = E = A$ then $R \subseteq A \times A$ and we say that R is a binary relation on A. The relation R can be empty. If, *e.g.*, $(d_2, e_3) \in R$ we write $d_2 \, R \, e_3$. Functions or mappings are subsets of relations. They are special relation cases as formally given next.

Definition 1.3. Let a binary relation R from $D = \{d_1, d_2, ...d_n\}$ into $E = \{e_1, e_2, ..., e_m\}$ be given. Let for any two elements of $D \times E$

$$\begin{gathered}(d_i, e_j) \in D \times E, \quad (d_k, e_l) \in D \times E, \\ i \in \{1,2,...,n\}, j \in \{1,2,...,m\}, k \in \{1,2,...,n\}, l \in \{1,2,...,m\}\end{gathered} \tag{1.6}$$

If the following implication holds true

$$(d_i = d_k \ \ and \ \ e_j \neq e_l) \Rightarrow ((d_i, e_j) \notin R \ and \ (d_k, e_l) \notin R) \tag{1.7}$$

then the relation R is a discrete function or a discrete mapping notated *f* defined on the domain

$$DOM = \{d_{i_1}, d_{i_2}, ..., d_{i_s}\} \tag{1.8}$$

where DOM is the set of all first elements of the pairs (d_i, e_j) belonging to the relation R. A co-domain of the function is set *CDOM* that consists of all the second elements of the pairs (d_i, e_j) belonging to the relation R

$$CDOM = \{e_{i_1}, e_{i_2}, ..., e_{i_u}\} \tag{1.9}$$

We write

$$e_j = f(d_i),\ d_i \in DOM,\ e_j \in CDOM \tag{1.10}$$

The right-hand side of Equation (1.7) is an AND conjunction of two propositions. If the premise is true, they both are true. It means that both ordered pairs (d_i, e_j) and (d_k, e_l) cannot belong to the relation *R*. However, one of them can be in *R*. Another formulation of this can be as follows. A function is a binary relation from set $D = \{d_1, d_2, ..., d_n\}$ into set $E = \{e_1, e_2, ..., e_m\}$ if there are no two ordered pairs $(d_p, e_r), (d_p, e_v)$ in *R* such that $e_r \neq e_v$.

1.3 Discrete Processes

Let a finite set Σ be given as

$$\Sigma = \{e_1, e_2, ..., e_n\} \tag{1.11}$$

The set Σ is called the event set. We assume that an event $e_{i_k} \in \Sigma$ occurs at the time point τ_{i_k}. Let a sequence of events be given as

$$\tilde{\sigma} = e_{i_1}, e_{i_2}, ..., e_{i_k}, ..., e_{i_N} \tag{1.12}$$

where $e_{i_1} \in \Sigma$ occurs in the discrete time point τ_{i_1}, $e_{i_2} \in \Sigma$ in time point τ_{i_2}, *etc.*, e_{i_k} in time point τ_{i_k}, *etc.*, and e_{i_N} in time point τ_{i_N}, $\tau_{i_1} \langle \tau_{i_2} \langle .....\langle \tau_{i_k} \langle .....\langle \tau_{i_N}$. The sequence $\tilde{\sigma}$ is called a discrete process. In this particular case when elements of a sequence are events we speak about the event string.

Figure 1.3 shows layout of a manufacturing system including a milling machine M, a grinding machine G and three belt conveyors C1–C3. The parts to be processed in the manufacturing system come into the system irregularly with various gaps as a sequence one by one part. Maximum three parts can be fed up on the conveyor C1. Input of a part is detected by a photo-sensor P11. The part is stopped by a stopper at the end of C1. Presence of the part at the end of the conveyor is signalized by a photo-sensor P12. If the milling machine M is free and a part is available at the end of C1, the part is transferred by the transportation means T1 into the milling machine. After milling the part is transferred by T2 onto the conveyor C2. The photo-sensor P21 detects input of the part on the conveyor C2. In the conveyor section between the sensors P21 and P22 there can be maximum two parts. The same mechanism holds for loading of the grinding machine G. Maximum four parts can be loaded on the conveyor section P31–P32.

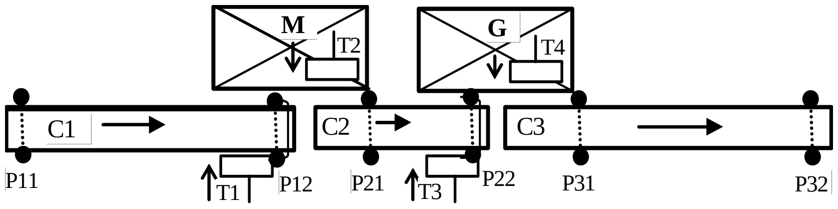

Figure 1.3. Manufacturing system layout

In a manufacturing system, typical events are the input of a part into a conveyor section, arrival of a part in some position on the conveyor, start of an operation, *e.g.*, start of milling, end of an operation, *e.g.*, end of milling.

As an example, consider the following event set:

$$\Sigma = \{s_{C1}, e_{C1}, m_{C1}, m_{C2}, m_{C3}, s_M, e_M, s_{C2}, e_{C2}, g_{C2}, g_{C3}, s_G, e_G, s_{C3}, e_{C3}\} \quad (1.13)$$

where s_{C1} stands for input of a part on conveyor C1, e_{C1} means arrival of a part at the end of the conveyor C1, m_{C1} means the transfer of a part from the conveyor C1 into the milling machine, s_M is the start and e_M the end of milling, m_{C2} is the transfer of a part from M on C2. Similarly, g_{C2} and g_{C3} denote transfers from C2 in G and from G on C3, respectively. The other events are denoted similarly.

Suppose that the manufacturing system is empty in its initial state. Both machines and conveyors are free. A possible sequence of events starting from the initial state is

$$\tilde{\sigma}_1 = s_{C1} e_{C1} m_{C1} s_M\, s_{C1} e_M\, m_{C2} s_{C2} e_{C1} m_{C1} s_M\, e_{C2}\, g_{C2} s_G \quad (1.14)$$

Event s_{C1} occurs at the time point τ_1, event e_{C1} at τ_2 … whereas $\tau_1 \langle \tau_2 \langle ......$.

Let us consider another event sequence example:

$$\tilde{\sigma}_2 = s_{C1} e_{C1} s_{C1} s_{C1} m_{C1} s_M\, e_{C1} e_M\, m_{C2} s_{C2} m_{C1} s_M\, e_{C2} g_{C2}\, e_M\, s_{C1} \quad (1.15)$$

Consider the following sequence starting from the initial state when the system is without parts (empty system):

$$\tilde{\sigma}_3 = s_{C1} e_{C1} e_{C1} \quad (1.16)$$

It represents an example of a technologically unfeasible event sequence in the given system. Consider a sequence from the beginning:

$$\tilde{\sigma}_4 = s_{C1} e_{C1} m_{C1} s_M\, s_{C1} e_{C1} m_{C1} \quad (1.17)$$

This is an example of a feasible event string, but not an admissible one due to the requirement that only one part can be present in the milling machine.

As mentioned before, an event is associated with a change of state. For example, the empty state when all conveyors are empty and both machines are free is denoted q_0. Arrival of a part on conveyor C1 is an event. State q_0 turns into state q_1 characterized by the presence of a part on C1 moving toward the stopper, while other conveyors and machines are still free.

The manufacturing system in Figure 1.3 is a serially arranged production line. A serial-parallel production cell example is shown in Figure 1.4. Suppose that four kinds of semi-products are produced from one kind of parts coming in *via* conveyor C1 and transported through the cell via conveyors C2–C4. Table 1.1 describes the options how to produce them.

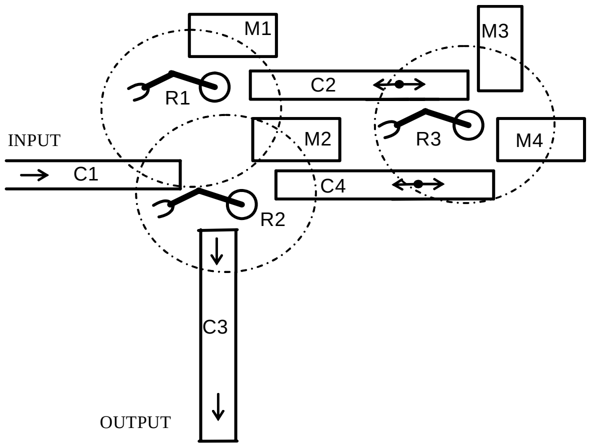

Figure 1.4. Manufacturing system arranged in a serial-parallel structure

Table 1.1. Job options in the manufacturing system

Operation	A	B	C	D
1	M1	M3	M2	M1
2	M2 or M3	M2 or M4	M4	M3
3	M4	M3 or M4	M3 or M4	M3 or M4
4		M4		M2

The system is flexible in that there are several ways to finish the production tasks having the job alternatives given in Table 1.1. The optimal route of the processed parts is to be found. A related problem to this is the job scheduling. Both problems can be solved with respect to the given optimality criterion, *e.g.,* to minimize the overall production work-span (work-time), also called makespan and completion time. The operation times have to be available for that task. A joblist breakdown with respect to operation time specifications is given in Table 1.2. Time durations for the semi-products A, B, C, and D are denoted by a, b, c, and d, respectively. The scheduling problem will be treated in detail later.

Individual events of the system depicted in Figure 1.4 can be specified as before:

$$\Sigma = \left\{ \begin{array}{l} a_{C1}, b_{C1}, a_{C2}, b_{C2}, a_{C3}, b_{C3}, a_{C4}, b_{C4}, \\ s_{M1}, e_{M1}, \ldots, s_{M4}, e_{M4}, \\ s_{R1C1M1}, e_{R1C1M1}, s_{R1C1C2}, e_{R1C1C2}, s_{R1C1M2}, e_{R1C1M2}, s_{R1C1C4}, e_{R1C1C4}, s_{R1M1C2}, e_{R1M1C2}, \\ s_{R1M1M2}, e_{R1M1M2}, s_{R1M2C2}, e_{R1M2C2}, \\ s_{R2C1M2}, e_{R2C1M2}, s_{R2C1C4}, e_{R2C1C4}, s_{R2M2C4}, e_{R2M2C4}, s_{R2C4M2}, e_{R2C4M2}, s_{R2C4C3}, \\ e_{R2C4C3}, s_{R2M2C3}, e_{R2M2C3}, \\ s_{R3C2M3}, e_{R3C2M3}, s_{R3M3C2}, e_{R3M3C2}, s_{R3C2M4}, e_{R3C2M4}, s_{R3M4C2}, e_{R3M4C2}, \\ s_{R3C2C4}, e_{R3C2C4}, s_{R3C4C2}, e_{R3C4C2}, s_{R3M3M4}, e_{R3M3M4}, s_{R3M3C4}, e_{R3M3C4}, s_{R3C4M3}, \\ e_{R3C4M3}, s_{R3M4C4}, e_{R3M4C4}, s_{R3C4M4}, e_{R3C4M4} \end{array} \right\} \tag{1.18}$$

Table 1.2. Job duration times specifications

	Products													
	A				B			C			D			
Operation	Machines				Machines			Machines			Machines			
	M1	M2	M3	M4	M2	M3	M4	M2	M3	M4	M1	M2	M3	M4
1	a_{11}					b_{13}		c_{12}			d_{11}			
2		a_{22}	a_{23}		b_{22}		b_{24}			c_{24}			d_{23}	
3				a_{34}		b_{33}	b_{34}		c_{33}	c_{34}			d_{33}	d_{34}
4							b_{44}					d_{42}		

Occurrence of a part at one side of the conveyor is denoted as event a, on the other side as b. s_{R1C1M1} is the start of the part transfer via Robot R1 from conveyor C1 to the machine M1, e_{R1C1M1} is the end of the transfer. The system control depends on the conveyor capacities.

It is assumed that the transfer times consumed by the robots and conveyors are negligible because they are much smaller than operation times. If such an assumption is not acceptable times of transfer operations can be considered separately. In the latter case Table 1.2 would be extended by further time specifications. Sometimes the transportation times can be included in the operation times of the processing machines.

1.4 Basic Properties of DEDS and their Specification

Characteristic properties of DEDS can be best illustrated on examples. DEDS include flexible manufacturing systems, digital computers, local or global computer networks, operation centers, and transportation systems on surface or in air. Various properties of DEDS events are to be studied:

- Event synchronization
- Concurrency
- Parallelism
- Conflict
- Mutual exclusion
- Deadlock
- System liveness
- Reversibility
- State reachability
- Event scheduling

Mankind strives not only to observe the natural phenomena but also to govern, to control and to benefit from them. Many various tools for the specification and analysis of DEDS are used nowadays. In addition, there is a need for tools that are suitable for the design of DEDS control. They should be able to specify the required properties of DEDS and to ensure the real-time reactivity of the controlled DEDS.

Basically the tools can be divided in three groups:

- Graphical tools
- Algebraic tools
- Formal language-based tools

The graphical tools are frequently used due to their transparency and ability to provide rich visual information. The main graphical tools include:

- State-transition diagrams or finite-automata
- Reactive (real-time) flow diagrams
- Statecharts
- Petri nets
- Grafcet
- Ladder logic diagrams

The algebraic tools are the following:

- Boolean algebra
- Algebraic expressions based on the respective state space
- Temporal logic
- Max-plus algebra

The tools based on formal languages are as follows:

- Formal language models

- Standard programming languages combined with real-time operating systems
- Real-time programming languages

The different tools listed above are not equivalent with respect to the application field. For example, max-plus algebra is effective especially for the analysis and control of job scheduling, while Grafcet is useful for the specification of the sequence control (Frištacký *et al*. 1981, 1990; Zhou and Venkatesh 1998; Zhou 1995).

The first two basic groups can serve as intermediate means between the requirements imposed on the system and the control that ensures them. A final specification and implementation of control requires the third group, namely a programming language. The specified control will then be implemented on appropriate hardware components, *e.g.,* personal computer, process computer, programmable logic controller, *etc*. From a graphical or algebraic specification a control program can be generated automatically. Also the transformations among the different specification tools are useful.

On the other hand, sometimes and for someone there is no need to use any intermediate means and it is possible to write a control program directly. However, for most people the opposite is true. A program formulated in any procedural language is a string of instructions to be performed separately and to force the system to behave as required. Intermediate means help to avoid the programming incorrectness.

Each specification tool listed above is based on the concept of the system state and state transitions described earlier as events. This fact can be illustrated by the following generally valid system behavior scheme:

$$\textbf{...} \quad \rightarrow \textbf{STATE} \rightarrow \textbf{TRANSITION} \rightarrow \textbf{STATE} \rightarrow \textbf{TRANSITION} \rightarrow \quad \textbf{...}$$

Because of the generality of this scheme, the "state and transition" concept is dealt with in more detail in the following section.

1.5 Basic Transition System

Various DEDS can be described uniquely by means of the so-called basic transition model proposed by Manna and Pnueli (1991), which serves us as a general description framework. It is defined by the quadruple

$$SYST = (\Pi, Q, \Sigma, \Theta) \tag{1.19}$$

where

$\Pi = \{u_1, u_2, ..., u_n\}$ is the finite set of state variables;

Q is the set of states where each state is given by the particular values of the variables from set Π. This value assignment is called the interpretation of variables belonging to the set Π;

Σ is the set of transitions whereby a transition $e \in \Sigma$ is a partial function $e : Q \xrightarrow{C_e} 2^Q$. Note that 2^Q is the power set defined as a set of all subsets created from set Q including the empty set $\varnothing$, C_e is a condition imposed on a transition e so that e can occur only if C_e is fulfilled, and C_e can be empty (meaning no condition); and

Θ is the set of initial conditions of the system. It includes states in which the execution of potential events can start.

The modeling power of the Manna and Pnueli model is that any correct specification by means of any tool described earlier can be transformed into a basic transition system. In other words, suitable transformations can be established between different system specifications. We can see that transitions in this model correspond to events introduced before. The time is not explicitly expressed in a basic transition. Rather, a possible sequence of events or an event precedence relation is used.

The function $Q \xrightarrow{C_e} 2^Q$ defining an event e is quite abstract. A standard particular case by excluding system control (if only controlled system is represented) is when one state is mapped into one another state due to the condition C_e or because there is only one-to-one mapping. The case when a state is mapped into state subsets presents indeterminism and its significance is purely theoretical. An example of that is the indeterministic finite automaton (see Section 5.4). In practical system control the indeterminism should be removed. If control is included the function $Q \xrightarrow{C_e} 2^Q$ can map a state to more states. See Chapter 4 for more details.

A general form of the specification of an event e is a transition relation given as an assertion for each transition e:

$$\rho_e\left(\Pi, \Pi' \right) \tag{1.20}$$

which relates the interpretation of state variables given as a state s with the interpretation of state variables given as a succeeding state s'. Under assertions we understand Boolean expressions extended by quantificators $\exists, \forall$, *etc.* In other words, in each state the relation $\rho_e\left(\Pi, \Pi'\right)$ determines the next state or states after transition e takes place. The following form describing the transition relation e can be used:

$$\rho_e\left(\Pi, \Pi'\right) = C_e\left(\Pi\right) \wedge \left(u'_1 = ex_1\right) \wedge \left(u'_2 = ex_2\right) \wedge \ldots \wedge \left(u'_n = ex_n\right) \tag{1.21}$$

where $C_e\left(\Pi\right)$ is an assertion stating a condition for state s under which transition e is enabled and the system comes over into state s'; ex_{1-n} are logic expressions. Assertion $C_e\left(\Pi\right)$ is constructed over state variables such that if the variable values lead to a true Boolean value from $C_e\left(\Pi\right)$, e is enabled and state variables u'_{1-n}

are given the values according to expressions ex_{1-n}. These expressions are built up of state variables u_{1-n}. Notation ex_i is shortened in Equation (1.21). It has the following meaning:

$$u'_i = lex_{i1} \textit{ if and only if } lex_{i2} \textit{ is true} \tag{1.22}$$

State variables in DEDS are discrete ones. If they are logical variables or expressions built up of logical variables, then Equation (1.21) can be put together directly based on them. Other than logic variables can be represented by means of a set of the auxiliary logic variables further used in Equation (1.21).

Figure 1.5 shows an example of a simple discrete event dynamic system. The system is an input portion of a flexible manufacturing system. The parts to be processed are transported into the system by belt conveyor C1. They arrive as an irregular stream. There are different gaps between individual parts. A video system VS scans each part when the latter enters the VS range (detected by sensor P_0). It evaluates the parameters of shape and quality of an incoming part and sorts it in two groups. These two groups are routed *via* turntable TT_1. Intervals between individual parts are so that a new part comes in the range of sensor P_0 when the preceding part is already on conveyor C2 or C3. The parts of the first group are placed on conveyor C2 while those of the second group on conveyor C3.

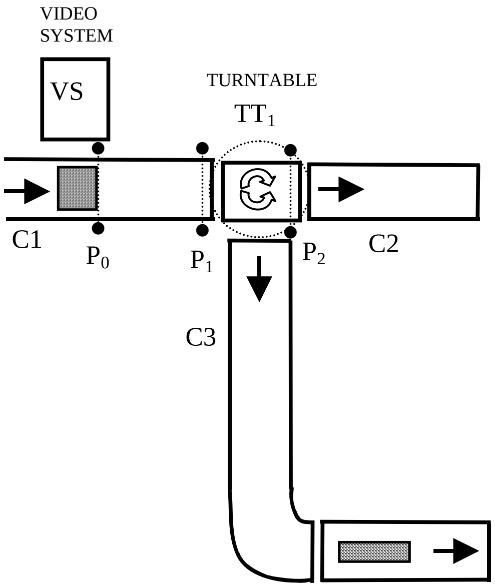

Figure 1.5. A manufacturing system

Now, let us model the system described in the example in a form given by Equation (1.19). We have

$$SYST = (\Pi, Q, \Sigma, \Theta) \tag{1.23}$$

$$\Pi = \begin{Bmatrix} P_0, P_1, P_2; \gamma_{01}, \gamma_{02}; T1; ETT1H, ETT1V; \\ TT1H, TT1V \end{Bmatrix} \tag{1.24}$$

where P_0-P_2 are logic variables corresponding to sensors P$_0$-P$_2$, respectively. If a part is under sensor P$_0$ then $P_0 = 1$, *etc*. Note that the variables are written in italic in order to distinguish them from the corresponding sources of the variables. γ_{01} and γ_{02} serve for the group distinction: γ_{01}=1 and γ_{02}=0 for the first group and γ_{01}=0 and γ_{02}=1 for the second one. $T1$ is a state variable signaling the presence of a part in turntable TT1 when $T1 = 1$; otherwise $T1 = 0$. *ETT1H* and *ETT1V* indicate the turntable horizontal and vertical positions, respectively. *TT*1*H*, and *TT*1*V* are commands to set the turntable horizontally or vertically. Before the start of the system operation, conveyors C1–C3 are switched on and remain in this state during the operation. Let there be the logic *C*1–*C*3 corresponding to the conveyor state; $C1 = 1$ if conveyor C1 is switched on and similarly for C2 and C3.

The set of states is as follows:

$$\begin{array}{ll}
Q = \{q_0 = (0,0,0;\ 0,0;\ 0;\ 1,0;\ 1,0), & q_{12} = (0,0,1;\ 0,1;\ 1;\ 0,1;\ 0,1), \\
q_1 = (1,0,0;\ 0,0;\ 0;\ 1,0;\ 1,0), & q_{13} = (0,0,0;\ 0,0;\ 0;\ 0,1;\ 0,1), \\
q_2 = (1,0,0;\ 1,0;\ 0;\ 1,0;\ 1,0), & q_{14} = (1,0,0;\ 0,0;\ 0;\ 0,1;\ 0,1), \\
q_3 = (0,0,0;\ 1,0;\ 0;\ 1,0;\ 1,0), & q_{15} = (1,0,0;\ 0,1;\ 0;\ 0,1;\ 0,1), \\
q_4 = (0,1,0;\ 1,0;\ 0;\ 1,0;\ 1,0), & q_{16} = (0,0,0;\ 0,1;\ 0;\ 0,1;\ 0,1), \\
q_5 = (0,0,0;\ 1,0;\ 1;\ 1,0;\ 1,0), & q_{17} = (0,1,0;\ 0,1;\ 0;\ 0,1;\ 0,1), \\
q_6 = (0,0,1;\ 1,0;\ 1;\ 1,0;\ 1,0), & q_{18} = (0,0,1;\ 0,1;\ 1;\ 0,1;\ 0,1), \\
q_7 = (1,0,0;\ 0,1;\ 0;\ 1,0;\ 1,0), & q_{19} = (1,0,0;\ 1,0;\ 0;\ 0,1;\ 0,1), \\
q_8 = (0,0,0;\ 0,1;\ 0;\ 1,0;\ 1,0), & q_{20} = (0,0,0;\ 1,0;\ 0;\ 0,1;\ 0,1), \\
q_9 = (0,1,0;\ 0,1;\ 0;\ 1,0;\ 1,0), & q_{21} = (0,1,0;\ 1,0;\ 0;\ 0,1;\ 0,1), \\
q_{10} = (0,0,0;\ 0,1;\ 1;\ 1,0;\ 1,0), & q_{22} = (0,0,0;\ 1,0;\ 1;\ 0,1;\ 0,1), \\
q_{11} = (0,0,1;\ 0,1;\ 1;\ 1,0;\ 0,1), & q_{23} = (0,0,1;\ 0,1;\ 1;\ 1,0;\ 1,0)\}
\end{array} \tag{1.25}$$

The set of transitions Σ is given by the set of the following partial functions:

$$e_1 : e_1(q_0) = q_1,\ \ e_2 : e_2(q_1) = q_2,\ \ e_3 : e_3(q_2) = q_3,\ \ e_4 : e_4(q_3) = q_4, \ldots etc. \tag{1.26}$$

All functions can be given *via* Tables 1.3 and 1.4. Function values are in the table cells. Let the initial conditions Θ for the occurrence of the events from set Σ are these: the system is in state q$_0$ and $C1 = C2 = C3 = 1$. Establishing the system in q$_0$ is indicated by the logic variable *INIT*.

Table 1.3. The first part of the transition set functions

	Function argument											
	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}	q_{11}
e_1	q_1	-	-	-	-	-	-	-	-	-	-	-
e_2	-	q_2	-	-	-	-	-	-	-	-	-	-
e_3	-	-	q_3	-	-	-	-	-	-	-	-	-
e_4	-	-	-	q_4	-	-	-	-	-	-	-	-
e_5	-	-	-	-	q_5	-	-	-	-	-	-	-
e_6	-	-	-	-	-	q_6	-	-	-	-	-	-
e_7	-	-	-	-	-	-	q_0	-	-	-	-	-
e_8	-	q_7	-	-	-	-	-	-	-	-	-	-
e_9	-	-	-	-	-	-	-	q_8	-	-	-	-
e_{10}	-	-	-	-	-	-	-	-	q_9	-	-	-
e_{11}	-	-	-	-	-	-	-	-	-	q_{10}	-	-
e_{12}	-	-	-	-	-	-	-	-	-	-	q_{11}	-
e_{13}	-	-	-	-	-	-	-	-	-	-	-	q_{12}
e_{14}	-	-	-	-	-	-	-	-	-	-	-	-
e_{15}	-	-	-	-	-	-	-	-	-	-	-	-
e_{16}	-	-	-	-	-	-	-	-	-	-	-	-
e_{17}	-	-	-	-	-	-	-	-	-	-	-	-
e_{18}	-	-	-	-	-	-	-	-	-	-	-	-
e_{19}	-	-	-	-	-	-	-	-	-	-	-	-
e_{20}	-	-	-	-	-	-	-	-	-	-	-	-
e_{21}	-	-	-	-	-	-	-	-	-	-	-	-
e_{22}	-	-	-	-	-	-	-	-	-	-	-	-
e_{23}	-	-	-	-	-	-	-	-	-	-	-	-
e_{24}	-	-	-	-	-	-	-	-	-	-	-	-
e_{25}	-	-	-	-	-	-	-	-	-	-	-	-
e_{26}	-	-	-	-	-	-	-	-	-	-	-	-

Table 1.4. The second part of the transition set functions

	Function argument											
	q_{12}	q_{13}	q_{14}	q_{15}	q_{16}	q_{17}	q_{18}	q_{19}	q_{20}	q_{21}	q_{22}	q_{23}
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-	-
	q_{13}	-	-	-	-	-	-	-	-	-	-	-
	-	q_{14}	-	-	-	-	-	-	-	-	-	-
	-	-	q_{15}	-	-	-	-	-	-	-	-	-
	-	-	-	q_{16}	-	-	-	-	-	-	-	-
	-	-	-	-	q_{17}	-	-	-	-	-	-	-
	-	-	-	-	-	q_{18}	-	-	-	-	-	-
	-	-	-	-	-	-	q_{13}	-	-	-	-	-
	-	-	q_{19}	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	q_{20}	-	-	-	-
	-	-	-	-	-	-	-	-	q_{21}	-	-	-
	-	-	-	-	-	-	-	-	-	q_{22}	-	-
	-	-	-	-	-	-	-	-	-	-	q_{23}	-
	-	-	-	-	-	-	-	-	-	-	-	q_6

All possible event sequences in the analyzed manufacturing system are built up from the sequences

$$\begin{array}{lll} \tilde{\sigma}_1 = e_1 & \tilde{\sigma}_2 = e_2\, e_3\, e_4\, e_5\, e_6\, e_7 & \tilde{\sigma}_3 = e_8\, e_9\, e_{10}\, e_{11}\, e_{12}\, e_{13}\, e_{14} \\ \tilde{\sigma}_4 = e_{15} & \tilde{\sigma}_5 = e_{16}\, e_{17}\, e_{18}\, e_{19}\, e_{20}\, e_{14} & \tilde{\sigma}_6 = e_{21}\, e_{22}\, e_{23}\, e_{24}\, e_{25}\, e_{26}\, e_7 \end{array} \tag{1.27}$$

The buildup or concatenations of the event sequences at Equation (1.27) follows the scheme as shown in Figure 1.6. Activities of the system can start when the condition Θ is fulfilled and it means that the first event sequence can be only $\tilde{\sigma}_1$. Equation (1.21) for the investigated system are

$$\begin{aligned} \rho_{e_1}(\Pi,\Pi') = {} & (C1 \wedge C2 \wedge C3 \wedge INIT) \wedge \left(P_0' = lex_1 \textit{ if and only if } lex_2 = 1 \right) \wedge \\ & \left(P_1' = \overline{lex_1} \textit{ if and only if } lex_2 = 1 \right) \wedge \left(P_2' = \overline{lex_1} \textit{ if and only if } lex_2 = 1 \right) \wedge \\ & \left(\gamma_{01}' = \overline{lex_1} \textit{ if and only if } lex_2 = 1 \right) \wedge \left(\gamma_{02}' = \overline{lex_1} \textit{ if and only if } lex_2 = 1 \right) \wedge \\ & \left(T_1' = \overline{lex_1} \textit{ if and only if } lex_2 = 1 \right) \wedge \left(ETT1H' = lex_1 \textit{ if and only if } lex_2 = 1 \right) \wedge \\ & \left(ETT1V' = \overline{lex_1} \textit{ if and only if } lex_2 = 1 \right) \wedge \left(TT1H' = lex_1 \textit{ if and only if } lex_2 = 1 \right) \\ & \wedge \left(TT1V' = \overline{lex_1} \textit{ if and only if } lex_2 = 1 \right) \end{aligned} \tag{1.28}$$

where

$$\begin{aligned} & lex_1 = lex_2, \\ & lex_2 = \overline{P_0} \wedge \overline{P_1} \wedge \overline{P_2} \wedge \overline{\gamma_{01}} \wedge \overline{\gamma_{02}} \wedge \overline{T1} \wedge ETT1H \wedge \overline{ETT1V} \wedge TT1H \wedge \overline{TT1V} \end{aligned} \tag{1.29}$$

True logic value is 1 and false 0 in Equation (1.29). For event e_2 we have

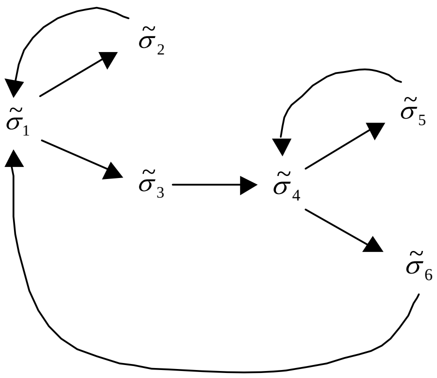

Figure 1.6. Event sequence patterns

$$\begin{aligned}\rho_{e_2}(\Pi,\Pi') = {} & (C1 \wedge C2 \wedge C3 \wedge INIT) \wedge \left(P_0' = lex_1 \text{ if and only if } lex_2 = 1\right) \wedge \\ & \left(P_1' = \overline{lex_1} \text{ if and only if } lex_2 = 1\right) \wedge \left(P_2' = \overline{lex_1} \text{ if and only if } lex_2 = 1\right) \wedge \\ & \left(\gamma_{01}' = lex_1 \text{ if and only if } lex_2 = 1\right) \wedge \left(\gamma_{02}' = \overline{lex_1} \text{ if and only if } lex_2 = 1\right) \wedge \\ & \left(T_1' = \overline{lex_1} \text{ if and only if } lex_2 = 1\right) \wedge \left(ETT1H' = lex_1 \text{ if and only if } lex_2 = 1\right) \wedge \\ & \left(ETT1V' = \overline{lex_1} \text{ if and only if } lex_2 = 1\right) \wedge \left(TT1H' = lex_1 \text{ if and only if } lex_2 = 1\right) \\ & \wedge \left(TT1V' = \overline{lex_1} \text{ if and only if } lex_2 = 1\right)\end{aligned} \tag{1.30}$$

where

$$\begin{aligned} & lex_1 = lex_2, \\ & lex_2 = P_0 \wedge \overline{P_1} \wedge \overline{P_2} \wedge \overline{\gamma_{01}} \wedge \overline{\gamma_{02}} \wedge \overline{T1} \wedge ETT1H \wedge \overline{ETT1V} \wedge TT1H \wedge \overline{TT1V} \end{aligned} \tag{1.31}$$

Other events would be expressed in a similar way.

Now consider an extension to the above system so that the parts of the first group are processed in a batch of three by R_{A1} or R_{A2}. Both robots perform the similar operations. The parts of the second group to a cell are routed *via* turntable TT_2. They are processed in two by R_B robotic cell. The number of parts is checked by means of photo-sensors P_{A1}–P_{A3} and P_{B1}–P_{B2}, respectively. Gate G1 (G2) goes up when three (two) parts are prepared for the next processing. Transport conveyor capacities are three transported parts for C2–C5, C8–C9 and two for C6–C7, respectively. Only one part can be allowed between sensors P_0 and P_1.

When a triple is prepared under sensor P_{A5}, robot R_{A1} performs the required processing operations and then transfers the triple onto conveyor O1. R_{A2} and R_B operate similarly. The aim of the control is to coordinate and control operations and movement of parts within the system. The co-ordination control level is superior to the process control one. The process control examples are the vision system's detection of parts, robots' movement control, and conveyor speed control. The vision system start is an event commanded from the coordination control level. Other facts concerning the FMS function are evident from the layout in Figure 1.7.

Now, let us outline the model of the system depicted in Figure 1.7 in a form given by Equation (1.19). We have

$$\Pi = \left\{\begin{array}{l} P_0, P_1, P_2, P_{A1}, P_{A2}, P_{A3}, P_{A4}, P_{A5}, P_{A6}, P_{B1}, P_{B2}, P_{B3}, ETT1H, \\ ETT1V, ETT2H, ETT2V, ERA1, ERA2, ERB; \\ SC1, EC1, SC2, EC2, SC4, EC4, SC5, EC5, SC7, EC7, SC8, \\ EC8, SC9, EC9, G1, G2, TT1H, TT1V, TT2H, TT2V, RA1, RA2, RB; \\ \gamma_{01}, \gamma_{02}, \gamma_{11}, \gamma_{12}, \gamma_{21}, \gamma_{22}, LC_{21}, LC_{22}, LC_{31}, LC_{32}, LC_{41}, LC_{42}, LC_{31}, \\ LC_{51}, LC_{52}, LC_{61}, LC_{62}, LC_{71}, LC_{72}, LC_{81}, LC_{82}, LC_{91}, LC_{92} \end{array}\right\} \tag{1.32}$$

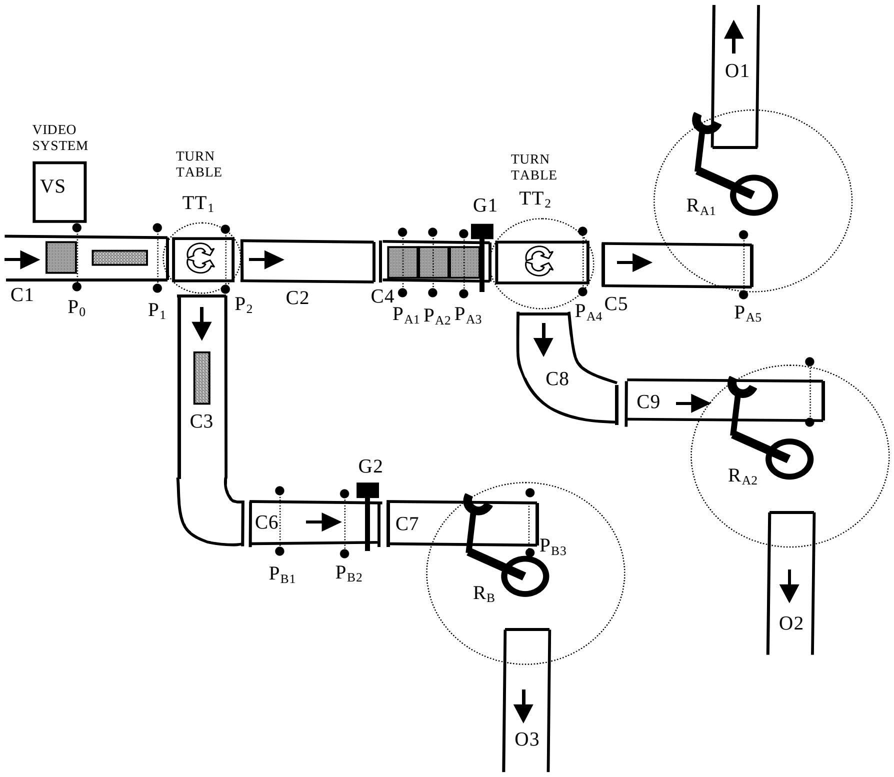

Figure 1.7. Example of a basic transition system: FMS with three robots

where the first part (the first two rows of Equation (1.28)) is inputs fed from the controlled system to the control system, the second part (the third and fourth rows) is outputs from the latter fed into the former and the third part is internal state variables. The variables Π are determined at the higher coordination control level. The model is built for the coordination control purpose. As in the preceding example, variable *ETT*1*H* indicates the straight position of turntable TT1, *ETT*1*V* its transversal position, *etc*. We assume that initially conveyors C3 and C6 and conveyors in both turntables are switched on and moving during the FMS operation. Conveyor C1 is started by command variable *SC1* and stopped by *EC1*. Other conveyor variables listed in Equation (1.32) have analogous meanings. The gates are operated by variables *G*1 and *G*2. Variables *TT*1*H*, *TT*1*V*, *etc*., are used to control the turntables, while *RA1*, *RA2*, and *RB* start robot operations. *ERA*1, *ERA*2, and *ERB* signals the end of the part processing by robots RA1, RA2, and RB, respectively. Information about routing a part is transferred from γ_{01} and γ_{02} to γ_{11} and γ_{22} when the part moves from sensors P_0 to P_1 (γ_{01} and γ_{02} should be free for the next part), and analogously for γ_{11} and γ_{22} and sensor P_2. LC_{21} and LC_{22} stand for storing the number of parts loaded on C2 so that no part gives $LC_{21} = 0, LC_{22} = 0$; one part gives $LC_{21} = 0, LC_{22} = 1$; two parts $LC_{21} = 1, LC_{22} = 1$; and three parts

$LC_{21}=1, LC_{22}=1$. Analogously, this holds true for LC_{ij} , but in terms of conveyors C6 and C7, which have the capacity equal 2.

All variables in Equation (1.32) are the Boolean ones taking values of 0 and 1. The states in set Q are given by pertinent variable values. For example, if $P_{A2}=1$ and $P_{A3}=1$ and all other variables are zero, the system is in a state when two parts are located before gate $G1$ and otherwise it is empty. Then the arrival of a new part in $C1$, signaled by $P_0=1$, is an event given by mapping the previously described state into one with $P_0=1, P_{A2}=1, P_{A3}=1$ and all remaining variables being zero. For example, an event $e=WP3$ - the arrival of a next part – in P_0=0, P_1=0, …, P_{A1}=0, P_{A2}=1, P_{A3}=1, P_{A4}=0, ..., is

$$\begin{aligned}\rho_{WP3}(\Pi,\Pi') = 1 \wedge \left(P_0' = \overline{P_0} \wedge \overline{P_1} \wedge \ldots \wedge \overline{P_{A1}} \wedge P_{A2} \wedge P_{A3} \wedge \overline{P_{A4}} \wedge \ldots \wedge \overline{LC_{92}}\right) \wedge \\ \left(P_1' = \overline{\overline{P_0} \wedge \overline{P_1} \wedge \ldots \wedge \overline{P_{A1}} \wedge P_{A2} \wedge P_{A3} \wedge \overline{P_{A4}} \wedge \ldots \wedge \overline{LC_{92}}}\right) \wedge \ldots \wedge \\ \left(P_{A1}' = \overline{\overline{P_0} \wedge \overline{P_1} \wedge \ldots \wedge \overline{P_{A1}} \wedge P_{A2} \wedge P_{A3} \wedge \overline{P_{A4}} \wedge \ldots \wedge \overline{LC_{92}}}\right) \wedge \\ \left(P_{A2}' = \overline{P_0} \wedge \overline{P_1} \wedge \ldots \wedge \overline{P_{A1}} \wedge P_{A2} \wedge P_{A3} \wedge \overline{P_{A4}} \wedge \ldots \wedge \overline{LC_{92}}\right) \wedge \\ \left(P_{A3}' = \overline{P_0} \wedge \overline{P_1} \wedge \ldots \wedge \overline{P_{A1}} \wedge P_{A2} \wedge P_{A3} \wedge \overline{P_{A4}} \wedge \ldots \wedge \overline{LC_{92}}\right) \wedge \\ \left(P_{A4}' = \overline{\overline{P_0} \wedge \overline{P_1} \wedge \ldots \wedge \overline{P_{A1}} \wedge P_{A2} \wedge P_{A3} \wedge \overline{P_{A4}} \wedge \ldots \wedge \overline{LC_{92}}}\right) \wedge \ldots \wedge \\ \left(LC_{92}' = \overline{\overline{P_0} \wedge \overline{P_1} \wedge \ldots \wedge \overline{P_{A1}} \wedge P_{A2} \wedge P_{A3} \wedge \overline{P_{A4}} \wedge \ldots \wedge \overline{LC_{92}}}\right)\end{aligned} \tag{1.33}$$

where Θ represents the condition that the system has to be initialized and empty before the first event can be accepted.

The example illustrates that in a little more complex case the modeling using a basic transition system is complicated, not transparent and very difficult for analysis and control design. In the following chapters of this book, we try to develop systematically theory and a way for practical use of other tools aiming to model, analyze, evaluate, simulate and control DEDS.

1.6 Problems and Exercises

1.1. Cite some CDVS and DEDS examples from your daily life.

1.2. Derive the basic transition system models for Figures 1.3 and 1.4.

1.3. In the system in Figure 1.5, change the assumption that only one part is processed in it so that a part can come in when another part is between sensors P_0 and P_1. Consider capacities of the conveyors.

1.4. Write the expression for the event next to that given by Equation (1.33) when a part moves from sensors P_0 to P_1.

1.5. A robotic cell is depicted in Figure 1.8. A-Parts are loaded into it *via* input conveyor I1. The input has capacity of 1 part. The same holds for input I2 and output O. Robot R2 picks up an A-workpiece from I2 and transfers it onto table T. R1 picks up a B-work-piece from I1 and puts it into the free milling machine M1 or M2. If both are free, M1 is preferred. After the machining, R1 transfers it onto palette P. If there is an A-workpiece on T, R2 transfers it from the palette to T and an assembly starts. After it, R2 transfers the product onto O. a) Analyze the system as DEDS; and b) Create an event set and event strings corresponding to the required behavior of the system, a realizable but not admissible event string and a non-realizable event string.

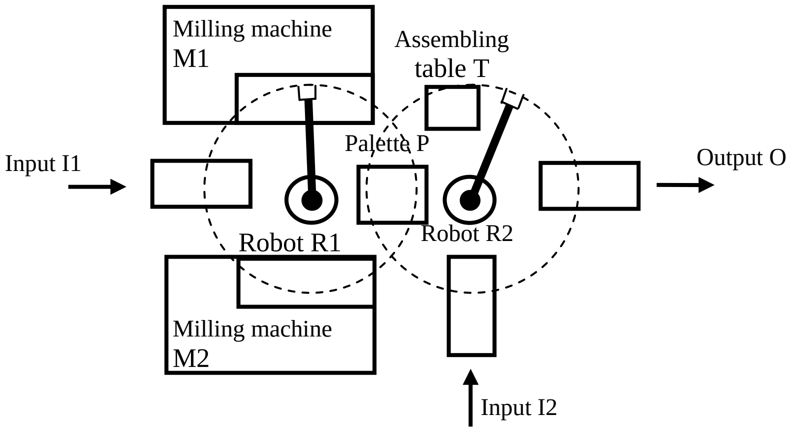

Figure 1.8. A robotic cell with two milling machines

1.6. For the system depicted in Figure 1.4 write a realizable event string corresponding to the technological process A in Table 1.1, a realizable but not admissible event string and a non-realizable event string.