Chapter 11
Fault Diagnosis of Discrete-Event Systems

Abstract This chapter presents diagnostic methods for discrete-event systems that are described by deterministic, nondeterministic or stochastic automata. Based on the solution to the state observation problem for discrete systems, the fault diagnostic problem is solved for all model classes by observing the unknown state of the model of the faulty systems and, hence, by deciding which model is currently consistent with the system behaviour.

11.1 Overview of Part III

This and the next chapters are devoted to discrete-event dynamical systems whose behaviour is described by sequences of discrete inputs and outputs. In contrast to the preceding chapters where the continuous changes of the signals occurring in the system have been investigated, the class of discrete-event systems is characterised by sequences of abrupt signal changes, because the signals have a finite value set.

Discrete-event systems occur naturally in the engineering practice. If the actuators like switches or valves can only jump between discrete positions, the input signal is binary and the signal values 0 and 1 are associated with the closed and the open position. Sensors may indicate that a physical quantity like the liquid level in a tank or a voltage exceeds a prescribed bound. Alarm sensors are typical examples for such sensors. Also the internal state of the system is often a discrete variable. For example, a robot gripper is empty or has grasped some part, a production step prescribed by a recipe has been carried out or not.

Whether or not a given dynamical system is considered as a discrete-event system depends upon the purpose of the investigations. It is typical for process supervision problems that a rather broad view on the system behaviour can be adopted which is based on a qualitative assessment of the signal values. If the supervisor should make a robot carry out a given sequence of movements or apply a certain recipe, then its decisions depend on discrete signal values like those mentioned in the examples and, hence, a discrete-event view-point is adequate. However, if signals have to remain in a narrow tolerance band, the continuous changes of these signals have to be considered and, thus, the continuous-systems point of view used in the preceding chapters

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_11

is appropriate. The alternative considerations of the tank system as a continuous-system for level control or as a discrete system for carrying out a batch process demonstrates the dependence of the representation level of a dynamical system upon the task to be performed. Note that the terms "continuous" and "discrete" refer to the signal spaces and, hence, to the different description levels. If a distinction has to be made concerning the temporal behaviour, the notions of continuous-time systems and discrete-time systems will be used.

As the dynamical behaviour of discrete systems is described by the changes of the discrete signal values that are called *events*, such systems are said to be discrete-event systems. However, the behaviour of such systems can be equivalently represented by the sequence of discrete values that the input, the state and the output assume in the considered time interval. This notation will be used here, where the kth value of the input is denoted by $v(k)$ and the input sequence in the time horizon $k = 0 \ldots k_e$ by

$$V(0 \ldots k_e) = (v(0), v(1), \ldots, v(k_e)). \tag{11.1}$$

Similarly, the kth value of the output is denoted by $w(k)$ and the output sequence by

$$W(0 \ldots k_e) = (w(0), w(1), \ldots, w(k_e)). \tag{11.2}$$

Based on this information, the diagnostic system should identify which fault $f \in \mathcal{F}$ has occurred (Fig. 11.1). The result is, in general, a set $\mathcal{F}(k_e) \subseteq \mathcal{F}$ of faults rather than a unique element of the fault set $\mathcal{F}$. Note that for discrete systems also the faults are usually considered to be discrete phenomena with the symbol f denoting a single fault that may or may not occur in the system.

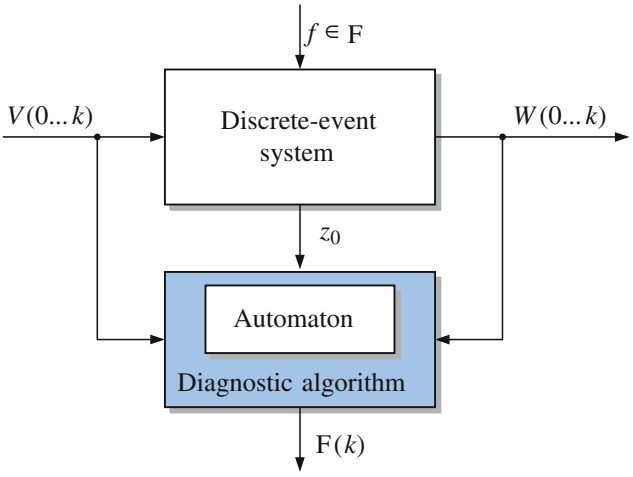


Fig. 11.1 Diagnostic problem

Consistency-based diagnosis of discrete-event systems. The main idea for solving the diagnostic problem is to test the consistency of the measured I/O pair with a set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$ of models that describe the system for all fault situations $f \in \mathcal{F}$ considered. This model will have the form of a deterministic, nondeterministic or stochastic automaton in this chapter. As in case of continuous-variable systems, the consistency of a pair $(V(0 \ldots k_e), W(0 \ldots k_e))$ of input and output sequences (11.1), (11.2) with finite time horizon k_e (in short: an I/O pair) is defined with respect to the behaviour $\mathcal{B}_f$ that the system subject to fault f possesses (cf. Sect. 1.3). To show the conceptual similarity of discrete-event and continuous-variable system diagnosis, assume that the model $\mathcal{A}_f$ of a discrete-event system is represented by a map ϕ_f that associates with each initial state z_0 and with every input sequence $V(0 \ldots k_e)$ the output sequence $W(0 \ldots k_e)$ of the system:

$$W(0 \ldots k_e) = \phi_f(z_0, V(0 \ldots k_e)). \tag{11.3}$$

The *behaviour* is the set of all I/O pairs (V, W) of arbitrary length $k_e \geq 0$ that are consistent with the model $\mathcal{A}_f$:

$$\mathcal{B}_f = \{(V(0 \ldots k_e), W(0 \ldots k_e)) \mid W(0 \ldots k_e) = \phi_f(z_0, V(0 \ldots k_e)); k_e \geq 0\}.$$

Diagnosis is based on the investigation to which behaviour $\mathcal{B}_f, (f \in \mathcal{F})$ the measured I/O pair belongs. An I/O pair is said to be *consistent with the model* $\mathcal{A}_f$ if it belongs to the behaviour $\mathcal{B}_f$:

$$(V(0 \ldots k_e), W(0 \ldots k_e)) \in \mathcal{B}_f. \tag{11.4}$$

In this relation, the left-hand side represents the measurement data and the right-hand side the behaviour of the system subject to fault f as it is represented by the model ϕ_f. If this relation holds, the fault f is called a *fault candidate*. The set of all fault candidates, which is denoted by $\mathcal{F}^*$, is the best possible diagnostic result:

$$\mathcal{F}^*(V(0 \ldots k_e), W(0 \ldots k_e)) = \{f \in \mathcal{F} \mid (V(0 \ldots k_e), W(0 \ldots k_e)) \in \mathcal{B}_f\}. \tag{11.5}$$

It depends upon the time horizon k_e and usually shrinks with increasing horizon because the more information about the system behaviour is obtained by measurements, the more faults can be excluded from the set of fault candidates.

The main problem of fault diagnosis of discrete-event systems is, hence, the elaboration of methods to test the consistency of I/O pairs with a model. This chapter develops consistency tests for systems that are described by deterministic, nondeterministic and stochastic automata.

Chapter overview. The theory of discrete-event systems has been developed rather separately with respect to the theory of continuous-variable systems. Therefore, the state of the art is different from what is known about continuous systems. The main theoretical results concern the modelling, analysis and supervisory control of discrete systems, but only a few results are available for diagnosis and fault-tolerant control.

This is the reason why Part III of this book concerns mainly the fault diagnostic problem and presents merely preliminary results on fault control reconfiguration.

In Sect. 11.2 models of discrete-event systems are introduced. Section 11.3 gives a survey of the solutions to the diagnostic problems, which will be dealt with in more detail in Sects. 11.4–11.7 for deterministic, nondeterministic and stochastic automata. The common aspect is given by the fact that all diagnostic methods are based on state observation methods for the corresponding automata, which will be explained first and later extended to fault detection and to fault identification.

11.2 Models of Discrete-Event Systems

11.2.1 Deterministic and Nondeterministic Systems

This section explains the basic dynamical properties of discrete-event systems and shows how such systems can be described. It extends the brief introduction to discrete-event modelling given in Sect. 3.6.

Discrete-valued signals. The discrete input, state and output of the system are denoted by the symbols v, z and w (Fig. 11.2) and the elements of their discrete value sets are enumerated as follows:

$$\begin{aligned} v &\in \mathcal{V} = \{1, 2, \ldots, M\} \\ z &\in \mathcal{Z} = \{1, 2, \ldots, N\} \\ w &\in \mathcal{W} = \{1, 2, \ldots, R\}. \end{aligned}$$

It is assumed that the numbers M, N or R are finite.

In order to use scalar signals, in many applications the physical variables have to be encoded. For example, if a tank system with 4 on/off valves is considered, the input can be represented by a 4-vector $\boldsymbol{v} = (v_1\ v_2\ v_3\ v_4)^{\mathrm{T}}$ whose ith component describes the position of the ith valve. As each component can assume either the value 0 or 1 in correspondence with the closed or the open position, $\boldsymbol{v}$ has one of 16 different values. These 16 values are represented in the following by a scalar input symbol v with the value set $\mathcal{V} = \{1, 2, \ldots, 16\}$. To do so, a mapping from the set of the 16 values of $\boldsymbol{v}$ onto the set $\mathcal{V}$ of the scalar input v has to be defined.

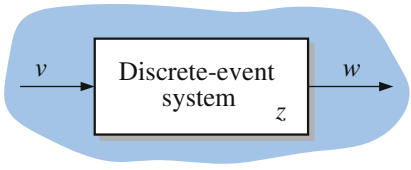


Fig. 11.2 Discrete-event system

Every change of the symbolic value of v, z or w is called an event. For example, if the state z jumps from the value j to the value i, a state event denoted by e_{ij} occurs (Fig. 11.3). However, the models introduced in this section use the sequences (11.1) and (11.2) of symbolic values rather then the sequences of events to characterise the behaviour of the discrete-event system under consideration.

Logical behaviour. The events occur at the time instants t_k which are enumerated as $k = 0, 1, 2, \ldots$ (Fig. 11.3). In the following, only the number k of the event is considered but not the actual time t_k. Therefore, the models used are called untimed or logical. They describe in which order the events occur but they say nothing about the temporal distance of these events. Besides the sequences (11.1) and (11.2) of inputs and outputs, the sequence of states

$$Z(0 \ldots k_e) = (z(0), z(1), z(2), \ldots, z(k_e))$$

is used to represent the logical behaviour of the system.

The motivation for using untimed models is twofold. First, the basic ideas of diagnosis and fault-tolerant control of discrete-event systems can be explained using untimed models, which are much simpler than timed models. Extensions of the methods to timed models are mentioned in the bibliographical notes. Second, in many practical circumstances, the untimed model yields the wanted results. For example, for many discrete-event systems faults can be detected due to the change of the order in which the events occur and no temporal information is necessary.

A further simplification is made in this chapter like in literature with respect to the synchronisation of the input, the state and the output. It is assumed that these events occur synchronously. That is, the state can only change if the input changes, which, at the same time, results in a change of the output. Repetitions of the symbols indicate that no real event occurs but a signal remains constant (Fig. 11.4). To shift the input, state and output events to the same time point is an abstraction that is reasonable for many applications.

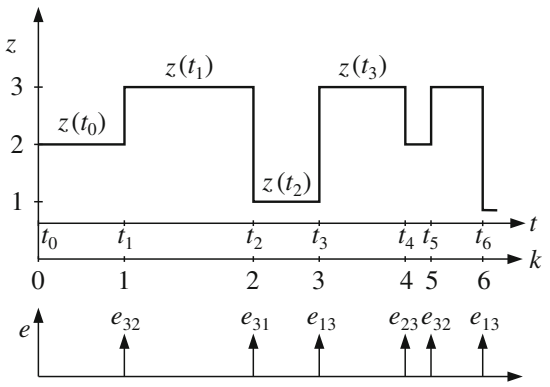


Fig. 11.3 Sequences of symbolic states and event sequences

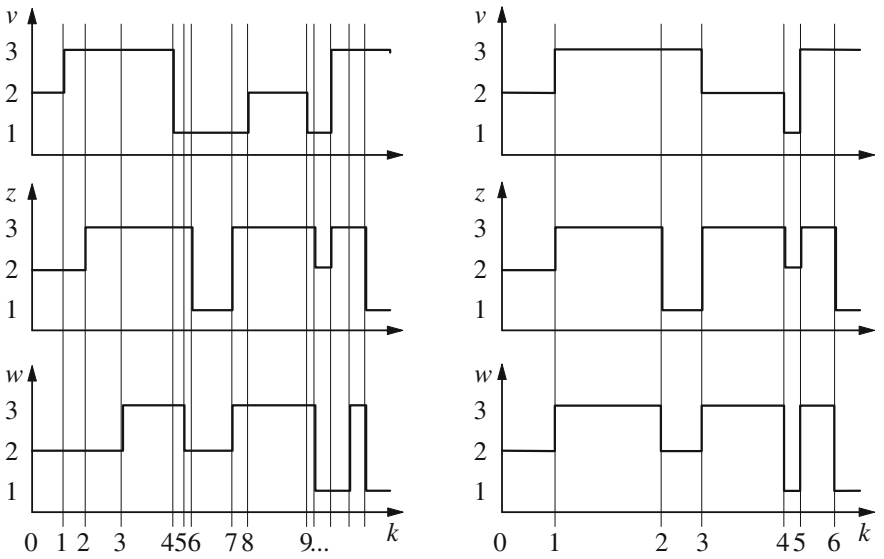


Fig. 11.4 Asynchronous (*left*) and synchronous (*right*) input, state and output sequences

Deterministic and nondeterministic systems. The assumed synchronous occurrence of the input, state and output events lead to the following "working principle" of discrete-event systems: At time $k = 0$, the system is in state $z(0) = z_0$ and obtains the input $v(0)$ (Fig. 11.2). The system generates the output $w(0)$ and its state jumps to the next value $z(1)$. Under the next input $v(1)$ the system changes its state from $z(1)$ to $z(2)$ and so on. In this way, for given initial state z_0 and input sequence $V(0 \ldots k_e)$ the discrete system follows a state sequence $Z(0 \ldots k_e)$ and generates an output sequence $W(0 \ldots k_e)$.

For deterministic systems, the generated state and output sequences Z and W are uniquely defined by z_0 and V. A standard form for describing deterministic systems is the automaton, which will be introduced in this section. It is a formalisation of the function ϕ, which has been used in Eq. (11.3) to represent a model of a deterministic discrete-event system.

For nondeterministic systems the state and output sequences are not unique but the system may generate any sequences of the sets $\mathcal{Z}(z_0, V)$ and $\mathcal{W}(z_0, V)$. This nondeterminism has to be understood in the following way. The technological system under consideration has a unique performance, because it cannot assume different states at the same time. Hence, for a fixed initial state and a fixed input sequence, an unambiguous state sequence Z and an unambiguous output sequence W occur. Nondeterministic behaviour occurs if the information about the system is not sufficient to predict these sequences unambiguously. If the system is brought into a particular initial state z_0 for several times and gets the same input sequence V, then the generated state and output sequences may differ. Hence, the model used to diagnose or control

this system has to describe some sets $\mathcal{Z}(z_0, V)$ and $\mathcal{W}(z_0, V)$ of possible sequences Z and W from which the real system "selects" one sequence. Such models have the form of nondeterministic automata, stochastic automata or Petri nets.

11.2.2 Deterministic Automata

This and the next section introduce models which can be used to describe the relation between the initial state $z(0)$ and the input sequence $V(0 \ldots k_e)$ on the one hand and the state sequence $Z(0 \ldots k_e)$ and output sequence $W(0 \ldots k_e)$ that the discrete system generates on the other hand.

The deterministic input–output automaton (I/O automaton) is defined by a 6-tuple

$$\mathcal{A} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, G, H, z_0)$$

with

- $\mathcal{Z}$ - set of states,
- $\mathcal{V}$ - set of input values (also called the input alphabet),
- $\mathcal{W}$ - set of output values (also called the output alphabet),
- $G : \mathcal{Z} \times \mathcal{V} \to \mathcal{Z}$ - state transition function,
- $H : \mathcal{Z} \times \mathcal{V} \to \mathcal{W}$ - output function,
- z_0 - initial state.

The dynamics of the automaton are described by the functions G and H in the following recursive way:

$$z(k+1) = G(z(k), v(k)), \quad z(0) = z_0 \tag{11.6}$$

$$w(k) = H(z(k), v(k)). \tag{11.7}$$

For the initial state z_0 and the input sequence $V(0 \ldots k_e)$, these functions yield the state sequence $Z(0 \ldots k_e + 1)$ and output sequence $W(0 \ldots k_e)$. If in the later investigations the functions G and H should be analysed, the time counter k does not matter and the equations above are written shorter as

$$z' = G(z, v) \tag{11.8}$$

$$w = H(z, v), \tag{11.9}$$

where z' denotes the "next state" following the state z.

The diagnostic problem may be considered for situations where the initial state z_0 is known or unknown. The initial state is known, for example, if the system to be diagnosed performs a cyclic function and the diagnostic system can be invoked whenever the system moves through its intial state. Also in the start-up phase of a system, the initial state is usually known and one says that the automaton is *initialised*.

Knowing the initial state z_0 considerably simplifies the diagnosis. If z_0 is unknown, a state observation problem is included in the diagnostic problem, which makes the overall problem much more involved. Then it is assumed that a set $\mathcal{Z}_0$ of possible initial states is known (with $\mathcal{Z}_0 = \mathcal{Z}$ as the trivial assumption). Both situations will be considered in this chapter.

Automaton graph. A nice graphical interpretation of an automaton is the *automaton graph*, whose vertices depict the states $z \in \mathcal{Z}$ and whose edges show how the state of the automaton can change (Fig. 11.5). Every directed edge represents a state transition described by Eq. (11.8) together with the output that is generated according to Eq. (11.9). In part (b) of the figure, the state transitions belonging to the input $v = 2$ are drawn by dashed arrows to illustrate the influence of the input upon the state transitions. The initial state is marked by an arrow not emerging from any other vertex. For a given initial state z_0 and input sequence V, the dynamical behaviour of the automaton is represented by the path through the automaton graph that starts in the vertex z_0 and whose edges are associated with the input prescribed by V.

The automaton is deterministic because the state and the input unambiguously determine the edge to follow in the automaton graph and, hence, the successor state and output. For example, if the automaton in Fig. 11.5 is in state 3, depending on the input $v \in \{1, 2\}$ the automaton goes either towards state 4 or towards 5. In this example, in both cases it generates the output 2.

Automaton map and behaviour. The *automaton map*

$$\phi : \mathcal{Z} \times \mathcal{V}^* \to \mathcal{W}^*$$

associates with each initial state $z_0 \in \mathcal{Z}$ and input sequence $V(0 \ldots k_e) \in \mathcal{V}^*$ the output sequence $W(0 \ldots k_e) \in \mathcal{W}^*$ of the automaton, which is obtained by applying Eqs. (11.46), (11.47) for $z(0) = z_0$:

$$W(0 \ldots k_e) = \phi(z_0, V(0 \ldots k_e)). \tag{11.10}$$

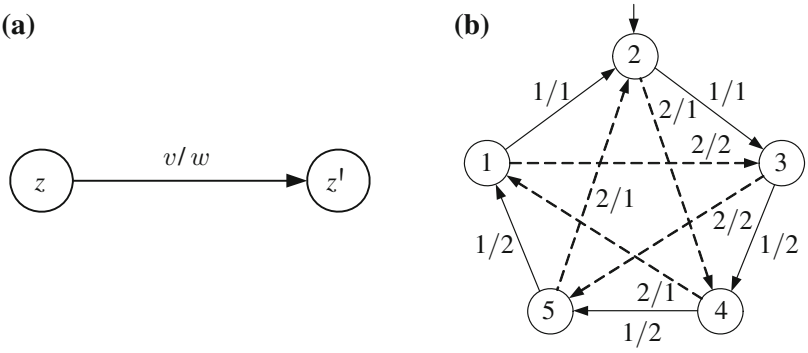


Fig. 11.5 Automaton graph of a deterministic automaton

$\mathcal{V}^*$ and $\mathcal{W}^*$ denote the sets of arbitrary sequences $V(0\ldots k_e)$ or $W(0\ldots k_e)$ of any length k_e that can be built using the input or output symbols of $\mathcal{V}$ or $\mathcal{W}$, respectively. The automaton map shows that an I/O pair (V, W) is consistent with an automaton if and only if the relation (11.10) holds.

The *behaviour* of an initialised automaton $(\mathcal{A}, z_0)$ is the set of all I/O pairs (V, W) that can be generated by the automaton map ϕ:

$$\mathcal{B} = \{(V(0\ldots k_e), W(0\ldots k_e)) \mid W(0\ldots k_e) = \phi(z_0, V(0\ldots k_e)); k_e \geq 0\}.$$

In applications, usually not the automaton map ϕ, but the functions G and H are used to carry out the consistency test included in the diagnostic problem. Therefore, it is important to represent the behaviour $\mathcal{B}$ in terms of G and H. Obviously, the relation (11.10) holds if there exists a state sequence

$$Z(0\ldots k_e) = (z(0), z(1), \ldots, z(k_e))$$

that satisfies for the given input sequence $V(0\ldots k_e)$ the relation (11.6) and for which the outputs generated by Eq. (11.7) coincides with the output sequence $W(0\ldots k_e)$:

$$\mathcal{B} = \{(V(0\ldots k_e), W(0\ldots k_e)) \mid \exists Z(0\ldots k_e): \quad z(k+1) = G(z(k), v(k)), \\ w(k) = H(z(k), v(k))\}. \qquad (11.11)$$

11.2.3 Nondeterministic Automata

In the nondeterministic automaton

$$\mathcal{N} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L_n, \mathcal{Z}_0)$$

the functions G and H occurring in the deterministic automaton are replaced by the function

$$L_n : \mathcal{Z} \times \mathcal{W} \times \mathcal{Z} \times \mathcal{V} \rightarrow \{0, 1\}$$

that describes for every state z and input v which successor state z' can be assumed while generating the output w. This function represents the behaviour of the automaton in terms of all 4-tuples (z', w, z, v) that are consistent with the automaton and form the set

$$\mathcal{R}(L_n) = \{(z', w, z, v) : L_n(z', w, z, v) = 1\} \subseteq \mathcal{Z}\times\mathcal{W}\times\mathcal{Z}\times\mathcal{V}. \qquad (11.12)$$

As the set (11.12) is, mathematically, a *relation*, the function L_n is called *behavioural relation* of the nondeterministic automaton.

For nondeterministic automata, the initial state is usually assumed to belong to a set $\mathcal{Z}_0$ of possible initial states. However, in some particular situations, the initial state z_0 may be unambiguously known.

The description of the dynamical behaviour of a nondeterministic automaton differs from that of a deterministic automaton given by Eqs. (11.6) and (11.7). Instead of a unique successor state z' and output w, the behavioural relation L_n yields, for the current state $z(k)$ and input $v(k)$, the following sets of possible successor states and output values:

$$\mathcal{Z}'(z(k), v(k)) = \{z' : \exists w \in \mathcal{W} \text{ such that } L_\mathrm{n}(z', w, z(k), v(k)) = 1\} \quad (11.13)$$

$$\mathcal{W}(z(k), v(k)) = \{w : \exists z' \in \mathcal{Z} \text{ such that } L_\mathrm{n}(z', w, z(k), v(k)) = 1\}. \quad (11.14)$$

Figure 11.6 depicts a part of the automaton graph of a nondeterministic automaton. It shows that for the input $v = 1$, the state may change from 1 towards 2 or towards 10, whereas for $v = 2$ only the state transition $1 \to 2$ can occur. The state change from 1 to 10 may either cause the output $w = 1$ or the output $w = 2$. Examples for the sets defined in Eqs. (11.13) and (11.14) are the following:

$$\begin{aligned} \mathcal{Z}'(1, 1) &= \{2, 10\} \\ \mathcal{W}(1, 2) &= \{2\}. \end{aligned}$$

If the behavioural relation L_n associates with each pair z, v a unique successor state z' and output w it can be represented by a state transition function G and an output function H and the nondeterministic automaton becomes deterministic.

The behaviour $\mathcal{B}$ of the nondeterministic automaton can be represented in terms of the behavioural relation l_n as follows. An I/O pair belongs to $\mathcal{B}$ if there exists a state sequence

$$Z(0 \ldots k_\mathrm{e} + 1) = (z(0), z(1), \ldots, z(k_\mathrm{e} + 1))$$

that satisfies for the input sequence $V(0 \ldots k_\mathrm{e})$ and the output sequence $W(0 \ldots k_\mathrm{e})$ the function L_n:

$$L_\mathrm{n}(z(k + 1), w(k), z(k), v(k)) = 1, \quad k = 0, 1, \ldots, k_\mathrm{e}.$$

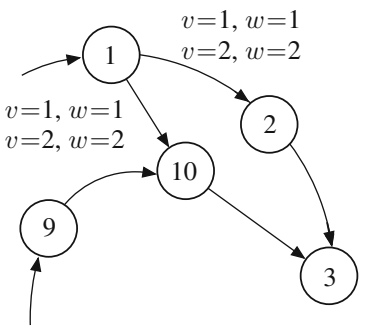


Fig. 11.6 Part of the automaton graph of a nondeterministic automaton

Hence, $\mathcal{B}$ can be represented as

$$\mathcal{B} = \{(V(0\ldots k_e), W(0\ldots k_e)) \mid \exists Z(0\ldots k_e+1) : L_n(z(k+1), w(k), z(k), v(k)) = 1,\ k = 0, \ldots, k_e\}. \tag{11.15}$$

Markov property. Both the deterministic and the nondeterministic automaton possess an important property, which is referred to as the *Markov property* of these models. This property means that the successor state $z(k+1)$ depends *only* upon the current state $z(k)$ and the current input $v(k)$ but it does not depend on the whole state sequence $Z(0\ldots k)$ or the whole input sequence $V(0\ldots k)$ that the automaton has generated or obtained until time k. The Markov property makes it possible to find the recursive relations (11.6) and (11.13) both of which are a relation between the next state $z(k+1)$ and the current state $z(k)$ and input $v(k)$ only. In case of the nondeterministic automaton, this relation does not determine the future state unambiguously, but it fixes the set of future states. Note that the set $\mathcal{Z}'(z(k), v(k))$ given in Eq. (11.13) depends merely upon $z(k)$ and $v(k)$.

In applications, the question whether or not the system under investigation possesses the Markov property depends upon the definition of the state z. Roughly speaking, if the state z includes all the information about the signals up to time k which is necessary to determine the future behaviour of the system, then the future state can be represented only in terms of the current state $z(k)$ and there is no need to refer to earlier states or input values occurring in the sequences $Z(0\ldots k-1)$ or $V(0\ldots k-1)$. To define the state appropriately is an important modelling step.

11.2.4 Stochastic Automata

The stochastic automata introduced in this section extend the description of nondeterministic discrete-event systems in such a way that the frequency of the occurrence of the different events can be assessed. They provide very useful additional information, because nondeterministic systems often produce a large set of different state and output sequences, but in practice these sets do by no means occur with the same frequency. In particular, fault diagnosis has to deal with nominal state sequences that occur (hopefully) with a very large frequency but the models have to include faulty sequences that a system follows seldom. The model should provide information about the frequency of occurrence.

The following explains how nondeterministic automata can be associated with the probabilities with which the different sequences occur. First, the notion of a stochastic process has to be introduced.

Stochastic processes. A stochastic process is a nondeterministic system for which the state and output sequences are generated with a certain probability. Its nondeterminism is not only considered in terms of the sets $\mathcal{Z}'(z_0, V)$ and $\mathcal{W}(z_0, V)$ defined

in Eqs. (11.13) and (11.14) but also in terms of the frequency with which the different elements of these sets are generated. Some of them may occur very often whereas others occur rarely.

Capital letters V, Z and W are used to denote the random variables of the input, state and output of the stochastic process. They are stochastic variables, that is, variables whose values are determined by chance. In every experiment, these variables assume values from the sets $\mathcal{V}$, $\mathcal{Z}$ or $\mathcal{W}$, respectively (Fig. 11.7). As the ranges of these variables and the time k are discrete, the process is called more precisely a *discrete stochastic process*.

The stochastic automaton should describe the probability with which the system assumes at time k the state $z \in \mathcal{Z}$ and generates the output $w \in \mathcal{W}$

$$\text{Prob}(Z(k) = z), \quad \text{Prob}(W(k) = w)$$

or with which the system follows a state trajectory $Z(0 \ldots k_e)$. To do so, it has to represent the generation law underlying the stochastic process, which is given by the transition probability

$$\text{Prob}(Z(k+1) = z(k+1), W(k) = w(k) \mid Z(k) = z(k), V(k) = v(k)).$$

Representation of stochastic processes by stochastic automata. Stochastic processes with finite sets of input values, output values and states are represented by finite-state stochastic automata

$$\mathcal{S} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L, p_0) \tag{11.16}$$

with $\mathcal{Z}$, $\mathcal{V}$ and $\mathcal{W}$ defined as before and

- $L : \mathcal{Z} \times \mathcal{W} \times \mathcal{Z} \times \mathcal{V} \longrightarrow [0, 1]$ - state transition probability (behavioural relation)
- p_0 - initial state probability distribution.

p_0 is the set of the N probability values $\text{Prob}(Z(0) = z_0)$ for the N possible initial states $z_0 \in \mathcal{Z}$. Hence, the automaton may start its behaviour in any state of the set

$$\mathcal{Z}_0 = \{z_0 \in \mathcal{Z} : p_0(z_0) > 0\}.$$

For each state $z \in \mathcal{Z}_0$ it is known with which probability $\text{Prob}(Z(0) = z) = p_0$ this state occurs as initial state of the system under investigation.

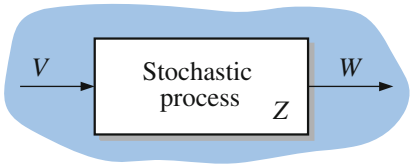


Fig. 11.7 Stochastic process

The function L represents the transition probability

$$L(z', w, z, v) = \text{Prob}(Z(1)=z',\ W(0)=w \mid Z(0)=z,\ V(0)=v). \tag{11.17}$$

In extension of the terminology used for the nondeterministic automaton, the function L is called *behavioural relation* of the stochastic automaton. In order to indicate that the behavioural relation is a conditional probability distribution, the symbol $L(z', w \mid z, v)$ is used instead of $L(z', w, z, v)$.

The probability distribution has the properties

$$\begin{aligned} 0 \le L(z', w \mid z, v) &\le 1, \quad \forall z', z \in \mathcal{Z},\ v \in \mathcal{V},\ w \in \mathcal{W} \\ \sum_{z' \in \mathcal{Z}} \sum_{w \in \mathcal{W}} L(z', w \mid z, v) &= 1, \quad \forall z \in \mathcal{Z},\ v \in \mathcal{V} \end{aligned} \tag{11.18}$$

and it leads to the two boundary distributions

$$G(z' \mid z, v) = \sum_{w \in \mathcal{W}} L(z', w \mid z, v) \tag{11.19}$$

$$H(w \mid z, v) = \sum_{z' \in \mathcal{Z}} L(z', w \mid z, v). \tag{11.20}$$

$G(z'|z, v)$ is called the *state transition relation* and $H(w|z, v)$ the *output relation* of the stochastic automaton. Due to Eq. (11.17) these relations represent the conditional probability distributions

$$G(z' \mid z, v) = \text{Prob}(Z(1)=z' \mid Z(0)=z, V(0)=v) \tag{11.21}$$

$$H(w \mid z, v) = \text{Prob}(W(0)=w \mid Z(0)=z, V(0)=v) \tag{11.22}$$

and possess the properties

$$\sum_{z' \in \mathcal{Z}} G(z' \mid z, v) = 1 \tag{11.23}$$

$$\sum_{w \in \mathcal{W}} H(w \mid z, v) = 1 \tag{11.24}$$

for all $z \in \mathcal{Z}$ and all $v \in \mathcal{V}$. Note that the functions G and H defined in Eqs. (11.21) and (11.22) include less information than the behavioural relation L because L also reflects the stochastic dependence between z' and w. Therefore, the following investigations refer to L rather than G and H. G is useful for problems in which only the state sequence is considered and the output sequence ignored.

Stochastic automata for which the behavioural relation L can be reconstructed from G and H are called stochastic *Mealy automata*. For them the relation

$$L(z', w \mid z, v) = G(z' \mid z, v) \cdot H(w \mid z, v)$$

holds for all $z, z' \in \mathcal{Z}$, $v \in \mathcal{V}$ and $w \in \mathcal{W}$. For these automata, the conditional probability distributions G and H defined in Eqs. (11.21) and (11.22) replace the state transition function G and the output function H of the deterministic automaton used in Eqs. (11.8) and (11.9).

The behaviour $\mathcal{B}$ of the stochastic automaton is the set of all I/O pairs for which a state sequence

$$Z(0 \ldots k_e + 1) = (z(0), z(1), \ldots, z(k_e + 1))$$

exists with positive probability such that the behavioural relation L is satisfied for all k in the time horizon considered:

$$L(z(k+1), w(k) \,|\, z(k), v(k)) > 0, \quad k = 0, 1, \ldots, k_e.$$

Hence, $\mathcal{B}$ can be represented as follows:

$$\begin{aligned} \mathcal{B} = \{&(V(0 \ldots k_e), W(0 \ldots k_e)) \,| \\ &\exists Z(0 \ldots k_e + 1) : L(z(k+1), w(k), z(k), v(k)) > 0,\ k = 0, \ldots, k_e\}. \end{aligned} \tag{11.25}$$

Autonomous stochastic automaton. If the automaton has no input, it is called an autonomous automaton. If, furthermore, the output coincides with the state, this automaton is given by the triple

$$\mathcal{S}_a = (\mathcal{Z}, G, p_0),$$

where G denotes the state transition relation given by Eq. (11.21) after neglecting the input v:

$$G(z' \mid z) = \text{Prob}(Z(1) = z' \mid Z(0) = z).$$

Graph of stochastic automata. The automaton graph is a directed graph, whose vertices denote the states and whose edges denote the possible state transitions. Figure 11.8 gives an example. The edges are associated with the state transition probability given by the value of the behavioural relation L for the pair of states connected by the edges and for the input v obtained and the output w generated for this state transition. The edge from state 4 towards state 5 shows the abbreviated labels, where 1/2/1 means that the state transition occurs for the input $v = 1$ while generating the output $w = 2$ with probability 1.

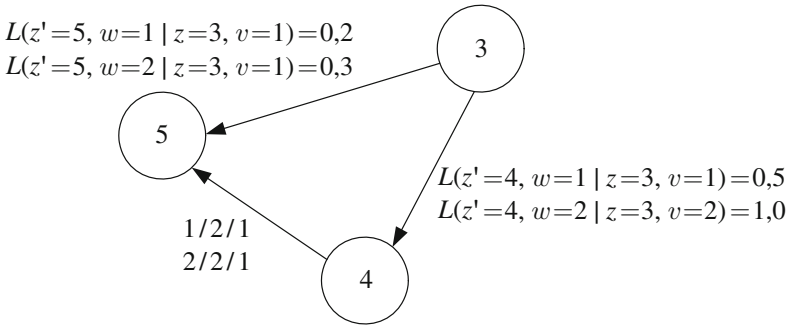


Fig. 11.8 Autonomous stochastic automaton

Beginning in any state z_0 with $\text{Prob}(Z(0) = z_0) > 0$, the automaton moves along the directed edges according to the corresponding probabilities. If more than one edge starts in a state, then all these edges can be followed which results in alternative state and output sequences. The frequencies with which the automaton follows these edges are described by the transition probabilities.

Example 11.1 Properties of stochastic automata
In the example shown in Fig. 11.9, the automaton may step from state 1 towards state 2 or state 10 if it obtains input $v=1$, but it is known to step towards state 2 if the input $v=2$ is applied. Moreover, the automaton may produce either output $w=1$ or $w=2$. The behavioural relation says that the probability to step from state 1 towards state 10 when getting the input $v=1$ is 0.2 if this step is associated with the output $w=1$ and 0.3 if $w=2$ occurs. The sum of 0.5 of both values is the probability that the automaton steps from 1 to 10 under the input $v=1$ while producing *some* output. Hence, $G(10 \mid 1, 1) = 0.5$ holds. Alternatively, the automaton may step from 1 towards 2. It definitely does this step if it obtains input $v=2$ and it is known to produce output $w=2$ during this step. If the automaton gets input $v=1$, then the probability to move to state 2 while generating output $w=1$ is 0.5.

The property (11.18) of the behavioural relation is satisfied, because for $z=1$ and $v=1$ the example yields

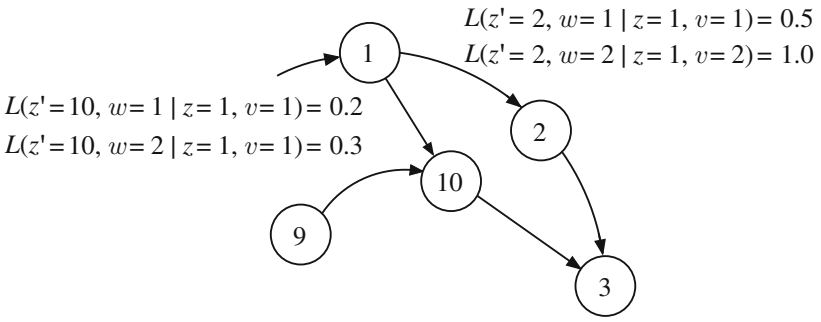


Fig. 11.9 Part of the automaton graph of a stochastic automaton with input and output

$$\sum_{z'}\sum_{w} L(z', w \mid z = 1, v = 1) =$$
$$= L(z'=10, w=1 \mid z=1, v=1) + L(z'=10, w=2 \mid z=1, v=1) +$$
$$+ L(z'=2, w=1 \mid z=1, v=1)$$
$$= 0.2 + 0.3 + 0.5 = 1$$

and for $z = 1$ and $v = 2$

$$\sum_{z' \in \mathcal{Z}} \sum_{w \in \mathcal{W}} L(z', w \mid z = 1, v = 2) \; = \; L(z'=2, w=2 \mid z = 1, v = 2) \; = \; 1.$$

The state transition relation G defined in Eq. (11.19) ignores the output and considers merely the transition between the states in dependence upon the input. For the example, the following relations hold

$$\begin{aligned} G(z'=10 \mid z=1, v=1) &= 0.2 + 0.3 = 0.5 \\ G(z'=2 \mid z=1, v=1) &= 0.5 \\ G(z'=2 \mid z=1, v=2) &= 1. \end{aligned}$$

They say that for the input $v=1$ the automaton goes from state 1 to state 2 or to state 10 with probability 0.5, but if it obtains input $v=2$ the automaton is known to go towards state 2.

The output relation H defined in Eq. (11.20) says which output is produced independently of the next state that is assumed by the automaton. For the example, the output relation has the values

$$\begin{aligned} H(w=1 \mid z=1, v=1) &= 0.7 \\ H(w=2 \mid z=1, v=1) &= 0.3 \\ H(w=2 \mid z=1, v=2) &= 1, \end{aligned}$$

which means that the automaton is known to produce the output $w = 2$ if it obtains the input $v = 2$ in state 1, but for the input $v = 1$ it may generate the output $w = 1$ with probability 0.7 and $w = 2$ with probability 0.3. Note that there is in general no way to reconstruct L from given G and H as mentioned above. □

Prediction. Stochastic automata can be used to predict the behaviour of a discrete-event system when starting from some state

$$z(0) = \mathcal{Z}_0 = \{z \in \mathcal{Z} \; : \; p_0(z) > 0\}$$

and getting the input sequence

$$V(0 \ldots k_e) = (v_0, v_1, \ldots, v_{k_e}).$$

For the initial state, the probability distribution is given by $p_0(z)$:

$$\text{Prob}(Z(0) = z) = p_0(z).$$

If the first input symbol v_0 has been obtained, the stochastic automaton carries out a state transition $z_0 \xrightarrow{v_0} z_1$ with $z_0 \in \mathcal{Z}_0$ according to the state transition probability $G(z_1 \mid z_0)$:

$$\text{Prob}(Z(1) = z_1 \mid v(0)) = \sum_{z_0 \in \mathcal{Z}_0} G(z_1 \mid z_0) \cdot \text{Prob}(Z(0) = z_0).$$

More generally, after the input symbols up to time k_e have been received, the automaton is in the state $z_{k_\text{e}+1}$ with the probability

$$\begin{aligned} &\text{Prob}(Z(k_\text{e}+1) = z_{k_\text{e}+1} \mid V(0 \ldots k_\text{e})) \\ &\quad = \sum_{z_{k_\text{e}} \in \mathcal{Z}} G(z_{k_\text{e}+1} \mid z_{k_\text{e}}) \cdot \text{Prob}(Z(k_\text{e}) = z_{k_\text{e}} \mid V(0 \ldots k_\text{e}-1)). \end{aligned} \quad (11.26)$$

Markov property. Stochastic automata describe discrete-event systems only if several assumptions are satisfied, which are summarised now.

A discrete stochastic process is defined by the probability with which a certain state change appears and an output symbol occurs at time k for a given sequence of input symbols. In general, for the state $z(k+1)$ and the output $w(k)$ this probability depends on the sequence of states and the sequence of input symbols up to time k and is thus described by the conditional probability

$$\text{Prob}\left(\begin{array}{c|l} Z(k+1)=z(k+1), & Z(0)=z(0), \ldots, Z(k)=z(k), \\ W(k)=w(k) & V(0)=v(0), \ldots, V(k)=v(k) \end{array} \right).$$

In the following, only those stochastic processes are considered that possess the *Markov property*. For such processes, the relation

$$\begin{aligned} &\text{Prob}(Z(k+1)=z(k+1), W(k)=w(k) \mid Z(k)=z(k), V(k)=v(k)) \\ &= \text{Prob}\left(\begin{array}{c|l} Z(k+1)=z(k+1), & Z(0)=z(0), \ldots, Z(k)=z(k), \\ W(k)=w(k) & V(0)=v(0), \ldots, V(k)=v(k) \end{array} \right) \end{aligned} \quad (11.27)$$

holds for all k, $z(k+1), z(k), \ldots, z(0)$, $w(k), w(k-1), \ldots, w(0)$ and $v(k), v(k-1), \ldots, v(0)$. It is common to say that $z(k+1)$ and $w(k)$ are *conditionally independent of the past variables* for given $z(k)$ and $v(k)$. The consequence of this assumption is the fact that the model of the system has only to include information of all single-state transitions, which in the stochastic automaton is represented by the behavioural relation L.

If the Markov property were not valid for a system, the model has to represent relations over a longer time interval; for example, the information about the next state if the system came into the current state 1 from the precessor state 5 or from predecessor state 4.

Furthermore, it is assumed that the process is homogeneous which means that the transition probability does not explicitly depend on the time variable k. Whenever the 4-tuple of successor state z', output w, current state z and input v is considered, the transition probability is the same. Hence, the relation

$$\begin{aligned}&\text{Prob}(Z(k+1)=z',\, W(k)=w \mid Z(k)=z,\, V(k)=v)\\ &\quad=\text{Prob}(Z(1)=z',\, W(0)=w \mid Z(0)=z,\, V(0)=v)\end{aligned} \tag{11.28}$$

holds for all k. A discrete stochastic process whose generation law is described by the state transition probability (11.28) is called *homogenous Markov process* with input and output.

For describing the stochastic process by a stochastic automaton, it is assumed furthermore that the appearance of a certain input symbol at time k is independent of the states and of the input values that have occurred up to that time, and independent of the time k. Therefore, the relation

$$\begin{aligned}&\text{Prob}\left(V(k)=v \;\middle|\; \begin{array}{l} Z(0)=z(0),\ldots,Z(k)=z(k),\\ V(0)=v(0),\ldots,V(k-1)=v(k-1)\end{array}\right)\\ &\quad=\text{Prob}(V=v)\end{aligned} \tag{11.29}$$

holds, where $\text{Prob}(V=v)$ describes the probability with which the input symbol v occurs. This assumption is not satisfied, for example, if the input $v(k)$ is prescribed by a supervisor that acts in dependence upon the measured output $w(k)$. Then this "feedback" needs to be included into the stochastic description of the process in order to satisfy the assumption (11.29) which will be used in the following.

11.2.5 Model of the Faulty System

This section explains how the system representation by automata can be extended to include information about the occurrence of a fault and the effect that the fault has on the future behaviour of the system. In literature, two main ideas have been followed in the past, both of which will be compared now.

Both ideas start from the interpretation of a fault as an *unobservable event* f. The attribute "unobservable" means that the fault event does not directly generate a measurement event that indicates this fault. The diagnostic method to be elaborated should find the fault from the changes that this fault causes in the future state or output sequences.

The first idea, which will be used later in this book, emphasises the fact that in many technological applications a fault changes the dynamics of the system under consideration. That means that the behaviour before the occurrence and after the occurrence of a fault f distinguishes from one another and, hence, has to be described by

different models. For example, if in a batch process a valve is blocked, the behaviour of the process changes qualitatively because the effect of the fault may influence the behaviour of the process at any future time.

This situation is depicted in Fig. 11.10. Part (a) of the figure shows two models, which describe the system for the fault cases f_0 and f_1, where as usual f_0 indicates the faultfree situation. For shortage of notation, the I/O pairs that are represented by the labels of the edges of the automaton graph are abbreviated here by the symbols a, b, c and d. As long as the system is faultless, it generates I/O pairs that are repetitions of (a, b, c, d). Hence, the left model, which is denoted by $\mathcal{A}_0$, is valid. After the fault has occurred, the system behaviour is decribed by the right model, which is denoted by $\mathcal{A}_f$ and shows that the future I/O pairs are represented by repetitions of (b, a).

For this kind of faults, the diagnostic problem concerns the question which of the two models represents the current behaviour of the system. Stated again in the sense of consistency-based diagnosis, the diagnostic problem leads to the consistency check for the measured I/O pair with respect to a set of models that is denoted by $\{\mathcal{A}_f, f \in \mathcal{F}\}$. The appearance of a fault is represented by the unobservable event, which changes the valid model from $\mathcal{A}_0$ to $\mathcal{A}_f$.

The alternative approach is depicted in Fig. 11.11. The fault is again considered as an unobservable event, but now such events occur as additional state transitions in the model of the faultless system. Consequently, after the occurrence of the fault the system behaves as before the fault. This situation is typical, for example, for computer or control systems, where a fault event represents an erroneous modification of a data set. The application of algorithms denoted by the labels a, b, c and d in the automaton graph does not change, but their sequence is modified by the fault. Before and after the fault f, the same algorithms are applied, possibly in a temporarily changed order.

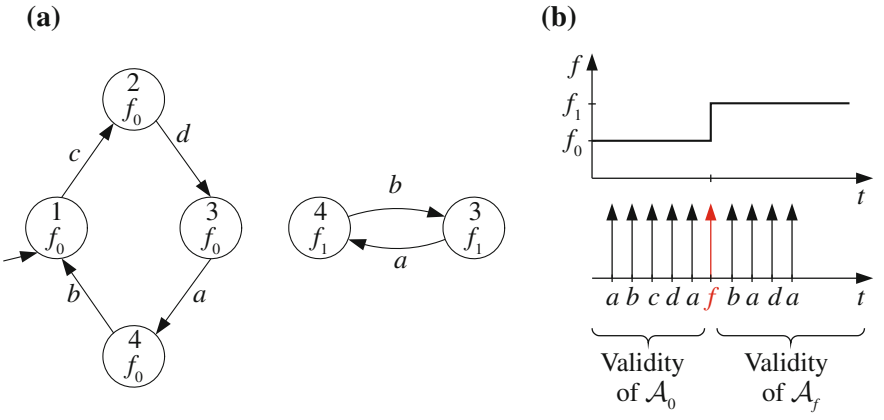


Fig. 11.10 Faults change the system properties

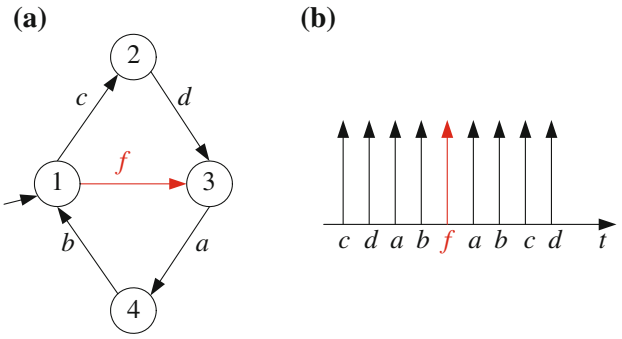


Fig. 11.11 Fault interpreted as an unobservable event

In this situation, the fault has again to be found by comparing the measured I/O pair with the model of the system, but the methods to do so differ from the methods to be developed here because of the different assumptions on the faulty system behaviour.

Both approaches to include faulty events into the model of a system are similar, but have been followed by different groups in literature. The similarity becomes obvious if the fault event shown by the arrow in Fig. 11.11a between the state 1 and state 3 is introduced as a state transition between a state of the model $\mathcal{A}_0$ in Fig. 11.10a and a state of the model $\mathcal{A}_f$. The difference of both approaches lies in the fact that in the first approach the fault brings the state into another "region" of the state set of the automaton that usually has a completely different structure than the state set of the faultless behaviour.

Diagnosis as a model-selection problem. This and the next chapters follow the first approach of fault modelling, in which the fault changes the dynamics and, hence, the structure of the automaton graph of a system. Accordingly, the system is described by a set of models, which is denoted by $\{\mathcal{A}_f,\ f \in \mathcal{F}\}$, where $\mathcal{F}$ is the set of faults considered. Every model has its own behavioural relation L_f, where the index f indicates the fault case. For this modelling approach, the diagnostic problem can be stated as the problem to select the model of the current hehavior out of the model set, (Fig. 11.12). If this model is valid for the fault f, then f is considered as a solution to the diagnostic problem.

As the fault f changes, the dynamics of the system, the state transition function or the behavioural relation of the model used depends upon f. If deterministic automata are used to represent the system, the model set $\{\mathcal{A}_f,\ f \in \mathcal{F}\}$ consists of deterministic automata

$$\mathcal{A}_f = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, G_f, H_f, z_0),$$

where the state transition function G_f and the output function H_f depend upon the fault f. The other components $\mathcal{Z}$, $\mathcal{V}$, $\mathcal{W}$ and z_0 may also depend upon f, but to simplify notation, it is assumed that these elements are the same for all models.

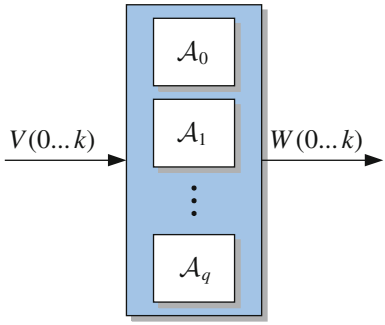


Fig. 11.12 Fault identification as model identification problem

If nondeterministic automata

$$\mathcal{N}_f = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L_{\mathrm{n}f}, z_0)$$

are used to represent the system, the behavioural relation L_{n} depends upon f and these relations of different models are distinguished by the additional index f. For stochastic automata

$$\mathcal{S}_f = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L_f, p_0(z))$$

the state transition relation L changes with the occurrence of the fault f. The fault appears as an additional element in the conditional probability distribution:

$$L_f = \mathrm{Prob}(Z(1) = z', W(0) = w \mid Z(0) = z, V(0) = v, F = f),$$

where F denotes the stochastic variable for the fault. In all three cases, the indices f indicate that the corresponding functions depend upon the fault f considered.

All the given models can be used if the fault f does not change over time. This assumption will be adopted in most parts of this chapter to elaborate the main ideas. It does not mean that the system must be faulty from the beginning, but that the models are set-up for the situation that the fault does not change during the online application of the diagnostic algorithm. Of course, the algorithms developed in this chapter can be used for changing faults, but then not all properties that are proved for them are still valid.

Changing faults. If the diagnostic problem should be considered for changing faults, the fault has to be interpreted as an external signal $f(k)$ that is described by the sequence

$$F(0 \ldots k_{\mathrm{e}}) = (f(0), f(1), \ldots, f(k_{\mathrm{e}})). \tag{11.30}$$

The models have to be extended to cope with the additional input (Fig. 11.13). In the deterministic case, the functions G and H have now two input arguments such that Eqs. (11.6) and (11.7) have to be replaced by

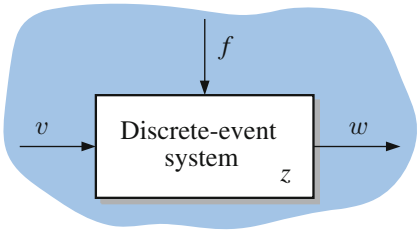


Fig. 11.13 Fault interpreted as an additional input

$$z(k+1) = G(z(k), v(k), f(k)), \quad z(0) = z_0 \tag{11.31}$$

$$w(k) = H(z(k), v(k), f(k)). \tag{11.32}$$

For the nondeterministic automaton, the behavioural relation now describes the behaviour of the system with the additional argument $f(k)$ as follows:

$$L_{\mathrm{n}}(z(k+1), w(k), z(k), v(k), f(k)) = 1.$$

For stochastic automata, in the state transition probability distribution

$$L = \mathrm{Prob}(Z(1) = z', W(0) = w \mid Z(0) = z, V(0) = v, F(k) = f)$$

the fault is now a time-dependent variable.

Fault model. It is usually too general to assume that the fault may follow an arbitrary sequence (11.30), because this would mean that the fault may change with every event occurring. If the fault is considered as an external signal, then the kind of signals to be considered has to be restricted by a *fault model*. Its usefulness should be explained here for the stochastic automaton, but the same idea is applicable for other forms of discrete-event models too.

The fault model

$$\mathcal{S}_{\mathrm{f}} = (\mathcal{F}, G_{\mathrm{f}}, p_{f0}) \tag{11.33}$$

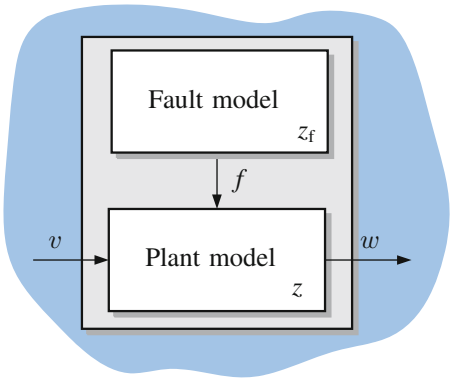


Fig. 11.14 Representation of a faulty system including a fault model

says how the fault $f(k)$ can behave. Hence, it describes a generator of the additional input to the plant model as depicted in Fig. 11.14. Its state set corresponds to the fault set $\mathcal{F}$ and the function $G_\mathrm{f} : \mathcal{F} \times \mathcal{F} \longrightarrow [0, 1]$ describes the conditional probability that the fault changes from f towards f' within one time step:

$$G_\mathrm{f}(f' \mid f) = \mathrm{Prob}(F(1) = f' \mid F(0) = f). \tag{11.34}$$

If the time steps are equidistant (like in discrete-time systems), these probabilities are closely related to the well-known measure of mean time before failure. p_{f0} denotes the probability distribution over the initial fault set.

The combination of the plant model with the fault model depicted in Fig. 11.14 results in a stochastic automaton

$$\tilde{\mathcal{S}} = (\tilde{\mathcal{Z}}, \mathcal{V}, \mathcal{W}, \tilde{L}, \tilde{p}_0) \tag{11.35}$$

whose state set is given by

$$\tilde{\mathcal{Z}} = \mathcal{Z} \times \mathcal{F} \tag{11.36}$$

and whose behavioural relation $\tilde{L}$ is obtained from L and G_f according to

$$\tilde{L}(z', f', w \mid z, f, v) = L(z', w \mid z, f, v) \cdot G_\mathrm{f}(f' \mid f) \tag{11.37}$$

with $z, z' \in \mathcal{Z}$, $v \in \mathcal{V}$, $w \in \mathcal{W}$ and $f, f' \in \mathcal{F}$. If the elements $\tilde{z} \in \tilde{\mathcal{Z}}$ are written as the vector

$$\tilde{z} = \begin{pmatrix} z \\ f \end{pmatrix}, \tag{11.38}$$

the behavioural relation $\tilde{L}$ in Eq. (11.37) can be rewritten as

$$\begin{aligned} &\tilde{L}(\tilde{z}', w \mid \tilde{z}, v) = \mathrm{Prob}(\tilde{Z}(1) = \tilde{z}', W(0) = w \mid \tilde{Z}(0) = \tilde{z}, V(0) = v) \\ &= \mathrm{Prob}\left(\begin{pmatrix} Z(1) \\ F(1) \end{pmatrix} = \begin{pmatrix} z' \\ f' \end{pmatrix}, W(0) = w \;\middle|\; \begin{pmatrix} Z(0) \\ F(0) \end{pmatrix} = \begin{pmatrix} z \\ f \end{pmatrix}, V(0) = v \right) \end{aligned}$$

which gives $\tilde{L}$ the standard form of the behavioural relation of stochastic automaton.

11.3 Diagnostic Problems and Ways of Solution

This sections gives a survey of the diagnostic problems for discrete-event systems and ways of solutions that will be explained in more detail in the remaining part of this chapter.

The diagnostic problem can be stated in a general form as follows (Fig. 11.1):

Diagnostic problem for discrete-event systems

Given:	Model set $\mathcal{A}_f$, $(f \in \mathcal{F})$
	Input sequence $V(0 \ldots k_e)$
	Output sequence $W(0 \ldots k_e)$
Find:	Set of fault candidates.

As the set of fault candidates has been defined with respect to the behaviour $\mathcal{B}_f$ of the model set $\mathcal{A}_f$, $(f \in \mathcal{F})$ and for the three models introduced in Sect. 11.2, the behaviour has been defined in a uniform way by Eqs. (11.11), (11.15) or (11.25), respectively, the solution of the diagnostic problem can be found by checking the consistency of the measured I/O pair with the model set used.

As the diagnostic problem should be solved usually online with increasing time horizon $k_e = 0, 1, \ldots$, an important aspect of all methods developed in this and the following chapter lies in the fact that these methods are formulated in a recursive way in which they use the diagnostic result up to the time horizon k_e to find the result for the extended time horizon $k_e + 1$. As the intermediate result, the last state $z(k_e)$ or a set $\mathcal{Z}(k_e)$ of possible states has to be stored. These intermediate results include all information about the input sequence $V(0 \ldots k_e)$ and the output sequence $W(0 \ldots k_e)$ processed so far that is relevant for the future diagnostic result. Upon arrival of the next measured values $(v(k_e + 1), w(k_e + 1))$ the diagnostic unit checks which models of the set $\{\mathcal{A}_f,\ f \in \mathcal{F}\}$ are still consistent with the extended sequences $V(0 \ldots k_e + 1)$ and $W(0 \ldots k_e + 1)$ and updates the set of fault candidates $\mathcal{F}^*(k_e + 1)$ accordingly.

The main ideas of the diagnostic methods will be explained for constant faults f, but the bibliographical notes include references, in which the extension for time-varying faults are given.

The difficulty of solving the diagnostic problem depends upon the kind of models used and the online information included in the I/O pair. The following classification starts with the simplest problem and shows the increase in complexity of the diagnostic problem if the model becomes more involved.

Diagnosis of deterministic automata with state measurements. In the simplest case, the models $\mathcal{A}_f$, $(f \in \mathcal{F})$ are deterministic automata and the current state $z(k)$ is measurable, which means that the output $w(k)$ coincides with the model-state $z(k)$. Under this strong assumptions, the consistency test of the I/O pair with a model $\mathcal{A}_f$ reduces to check whether the last measured state transition $z(k_e) \rightarrow z(k_e + 1)$ can occur in the model $\mathcal{A}_f$:

$$z(k_e + 1) = G_f(z(k_e), v(k_e)). \tag{11.39}$$

In this relation, the three signal values $z(k_e)$, $v(k_e)$ and $z(k_e + 1)$ are known and it is checked whether these symbols satisfy the state transition function G_f of the automaton $\mathcal{A}_f$.

To formulate the diagnostic method for all classes of systems considered in this chapter in a uniform way, the result of the consistency test is described by the binary variable $p_f(k_e)$, where $p_f(k_e) = 1$ indicates that the I/O pair $(V(0 \dots k_e), W(0 \dots k_e))$ is consistent with the model $\mathcal{A}_f$ describing the system subject to the fault f, whereas $p_f(k_e) = 0$ shows that the I/O pair is not consistent with this model. Consequently, the set of fault candidates is given by

$$\mathcal{F}^*(k_e) = \{f \in \mathcal{F} : p_f(k_e) = 1\}. \tag{11.40}$$

Diagnosis of deterministic automata with output measurements. The diagnostic problem becomes more involved if the initial state z_0 of the system is unknown and if instead of the state z some output w is measured. Then the diagnostic problem includes a state observation problem.

The diagnostic method starts with the assumption that the initial state z_0 belongs to a given set $\mathcal{Z}_0$. After the first I/O pair $(v(0), w(0))$ is known, it is tested for which initial states $z_0 \in \mathcal{Z}_0$ there exists a state transition $z_0 \xrightarrow{v(0)/w(0)} z(1)$ to some state $z(1)$ that is consistent with the state transition function G_f and the output function H_f of the automaton $\mathcal{A}_f$. What "consistent" means can be seen in the automaton graph, in which an edge from z_0 towards $z(1)$ with the label $v(0)/w(0)$ has to exist. Formulated as equations, "consistency" means that the relations

$$\begin{aligned} z(1) &= G_f(z_0, v(0)) \\ w(0) &= H_f(z_0, v(0)) \end{aligned}$$

are valid.

To represent this method in a recursive way, two sets of states are introduced:

$$\begin{aligned} \mathcal{Z}_f(0 \mid -1) &= \mathcal{Z}_0 \\ \mathcal{Z}_f(0 \mid 0) &= \{z_0 \in \mathcal{Z}_0 : w(0) = H_f(z_0, v(0))\}. \end{aligned}$$

The set $\mathcal{Z}_f(0 \mid -1)$ represents the a-priori information about the initial state, which has to be given as input $\mathcal{Z}_0$ to the diagnostic algorithm. After the I/O pair $v(0), w(0))$ is known, the set $\mathcal{Z}_f(0 \mid 0)$ is determined, which includes all elements of $\mathcal{Z}_0$ for which the output $H_f(z_0, v(0))$ generated by the automaton coincides with the measured output $w(0)$. Furthermore the set

$$\mathcal{Z}_f(1 \mid 0) = \{z(1) = G_f(z_0, v(0)) : z_0 \in \mathcal{Z}_f(0 \mid 0)\}$$

can be determined, which includes all states to which the model $\mathcal{A}_f$ can move under the input $v(0)$.

The set $\mathcal{Z}_f(1 \mid 0)$ is used as the starting point of the second recursion step. After the I/O pair $(v(1), w(1))$ has been measured, the two sets

$$\mathcal{Z}_f(1\,|\,1) = \{z(1) \in \mathcal{Z}_f(1\,|\,0) : w(1) = H_f(z(1), v(1))\}$$
$$\mathcal{Z}_f(1\,|\,0) = \{z(2) = G_f(z(1), v(1)) : z(1) \in \mathcal{Z}_f(1\,|\,1)\}$$

are determined in a similar way as the sets $\mathcal{Z}_f(0\,|\,0)$ and $\mathcal{Z}_f(1\,|\,0)$. In general, for each time horizon k_e the following two sets are generated alternately after the I/O pair $(v(k_\mathrm{e}), w(k_\mathrm{e}))$ has been measured:

$$\mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e}) = \{z(k_\mathrm{e}) \in \mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e}-1) : w(k_\mathrm{e}) = H_f(z(k_\mathrm{e}), v(k_\mathrm{e}))\} \tag{11.41}$$
$$\mathcal{Z}_f(k_\mathrm{e}+1\,|\,k_\mathrm{e}) = \{z(k_\mathrm{e}+1) = G_f(z(k_\mathrm{e}), v(k_\mathrm{e})) : z(k_\mathrm{e}) \in \mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e})\}. \tag{11.42}$$

As long as both sets are not empty, there exists a state sequence for the automaton $\mathcal{A}_f$ such that the automaton generates for the measured input sequence the measured output sequence and, consequently, the I/O pair is consistent with the model $\mathcal{A}_f$. Hence, the indicator $p_f(k_\mathrm{e})$ has now to be determined as

$$p_f(k_\mathrm{e}) = \begin{cases} 1 \text{ if } \mathcal{Z}_f(k_\mathrm{e}+1\,|\,k_\mathrm{e}) \neq \emptyset \\ 0 \text{ else.} \end{cases} \tag{11.43}$$

After this indicator is known for the current time horizon k_e for all models of the set $\{\mathcal{A}_f,\ f \in \mathcal{F}\}$, the set of fault candidates is obtained by Eq. (11.40).

Diagnosis of nondeterministic automata. For systems described by nondeterministic automata $\mathcal{N}_f$, $(f \in \mathcal{F})$ the diagnostic problem includes always a state observation problem, because even if the initial state z_0 is unambiguously known, the nondeterminism of the automaton can produce ambiguity with respect to the current state in each state transition. The complexity of the diagnostic problem increases due to the nondeterminism that allows the model to associate with each state z and input v a set of successor states z' rather than a unique state.

For the known I/O pair $(v(k_\mathrm{e}), w(k_\mathrm{e}))$, the sets $\mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e})$ and $\mathcal{Z}_f(k_\mathrm{e}+1\,|\,k_\mathrm{e})$ have to be determined using the behavioural relation $L_{\mathrm{n}f}$ of the nondeterministic automaton $\mathcal{N}_f$ according to the relations

$$\mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e}) = \{z(k_\mathrm{e}) \in \mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e}-1) : \exists z(k_\mathrm{e}+1) : L_{\mathrm{n}f}(z(k_\mathrm{e}+1), w(k_\mathrm{e}), z(k_\mathrm{e}), v(k_\mathrm{e})) = 1\}$$
$$\mathcal{Z}_f(k_\mathrm{e}+1\,|\,k_\mathrm{e}) = \{z(k_\mathrm{e}+1) : \exists z(k_\mathrm{e}) \in \mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e}) : L_{\mathrm{n}f}(z(k_\mathrm{e}+1), w(k_\mathrm{e}), z(k_\mathrm{e}), v(k_\mathrm{e})) = 1\}.$$

Then the indicator $p_f(k_e)$ can be found by Eq. (11.43) and the set of fault candidates by Eq. (11.40).

Diagnosis of stochastic automata. If the system is described by stochastic automata $\mathcal{S}_f$, $(f \in \mathcal{F})$, the ambiguities of the diagnostic result can be reduced by associating with each element of the set of fault candidates the probability

$$p_f(k_e) = \text{Prob}(f \mid V(0 \ldots k_e), W(0 \ldots k_e)),$$

which replaces the binary indicator $p_f(k_e)$ of consistency used for the deterministic or nondeterministic automaton. For the set of fault candidates the relation (11.40) is extended to become

$$\mathcal{F}^*(k_e) = \{f \in \mathcal{F} : p_f(k_e) > 0\}, \tag{11.44}$$

but besides this set the value of $p_f(k_e)$ remains important for the interpretation of the diagnostic result. This indicator shows with which certainty each fault $f \in \mathcal{F}^*(k_e)$ is present.

The basis for determining the probability $p_f(k_e)$ is again the solution of the state observation problem, which now means to calculate the probability

$$\text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e), f)$$

that the system subject to fault f can be in state z after it has obtained the input sequence $V(0 \ldots k_e)$ and generated the output sequence $W(0 \ldots k_e)$. This probability is abbreviated as $p_f(z, k_e)$:

$$p_f(z, k_e) = \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e), f).$$

Similarly, the probability to be in the state z' at time $k_e + 1$ after having received the input sequence $V(0 \ldots k_e)$ and generated the output sequence $W(0 \ldots k_e)$ is denoted by

$$p'_f(z', k_e) = \text{Prob}(Z(k_e + 1) = z' \mid V(0 \ldots k_e), W(0 \ldots k_e), f).$$

In Sect. 11.6 a recursive representation of these two probabilities will be given to complete the consistency test and to make it possible to determine the set of fault candidates together with the mentioned probabilities.

Outline of the diagnostic methods. The diagnostic problem will be solved for the three classes of automata in the next sections in a uniform way. This survey has shown that the diagnostic problem includes a state observation problem unless the state is measurable. Therefore, the state observation problem is solved first and later extended to fault detection and fault identification.

11.4 Diagnosis of Deterministic Automata

11.4.1 Diagnostic Algorithm

As the diagnostic problem is very simple if the state z is measurable (cf. Eq. (11.39)), this section is devoted to deterministic automata with outputs w. A pair $(V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e}))$ of input and output sequences with finite time horizon k_e is called *consistent* with the deterministic automaton $\mathcal{A}_f$ if there exists a state sequence

$$Z(0\ldots k_\mathrm{e}+1) = (z(0), z(1), \ldots, z(k_\mathrm{e}+1)) \tag{11.45}$$

for which the relations

$$z(k+1) = G(z(k), v(k)), \quad z(0) = z_0 \tag{11.46}$$

$$w(k) = H(z(k), v(k)) \tag{11.47}$$

are satisfied for $k = 0, 1, \ldots, k_\mathrm{e}$. If the initial state z_0 is known, the consistency can be tested by simply applying Eq. (11.46) for $k = 0, 1, \ldots, k_\mathrm{e}$ to generate for the measured input sequence (11.1) the state sequence (11.45). Then Eq. (11.47) is used to test whether the outputs $w(k)$ obtained by the model are identical to the elements of the measured output sequence (11.2). If the initial state is only known to belong to a set $\mathcal{Z}_0$, then the sequences of state sets described in Eqs. (11.42) and (11.43) have to be generated and consistency means that none of them is empty.

The following algorithm summarises the diagnostic steps to be performed for deterministic automata. The algorithm starts for $k_\mathrm{e} = 0$ and considers an increasing time horizon. The newest measured I/O pair $(v(k_\mathrm{e}), w(k_\mathrm{e}))$ is denoted by $(\bar{v}, \bar{w})$ in the algorithm and the state sets $\mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e})$ and $\mathcal{Z}_f(k_\mathrm{e}+1\,|\,k_\mathrm{e})$ by $\mathcal{Z}_f$ or $\mathcal{Z}'_f$, respectively, for $f \in \mathcal{F}$.

Algorithm 11.1 *Diagnosis of deterministic automata*

Given: Deterministic automata $\mathcal{A}_f$, $(f \in \mathcal{F})$
Set of initial states $\mathcal{Z}_0$

Initialisation: $\mathcal{Z}'_f = \mathcal{Z}_0$ for all $f \in \mathcal{F}$
$k_\mathrm{e} = 0$

Loop: 1. Measure the next I/O pair $(\bar{v}, \bar{w})$.

2. Determine $\mathcal{Z}_f = \{z \in \mathcal{Z}'_f \,:\, \bar{w} = H_f(z, \bar{v})\}$ for all $f \in \mathcal{F}$.

3. Determine $\mathcal{Z}'_f = \{z' = G_f(z, \bar{v}) \,:\, z \in \mathcal{Z}_f\}$ for all $f \in \mathcal{F}$.

4. Set $p_f(k_\mathrm{e}) = 1$ if $\mathcal{Z}'_f \neq \emptyset$ and $p_f(k_\mathrm{e}) = 0$ otherwise for all $f \in \mathcal{F}$.

5. Determine $\mathcal{F}^*(k_e) = \{f \in \mathcal{F} : p_f(k_e) = 1\}$.
6. $k_e := k_e + 1$
   Continue with Step 1.

Result: Set of fault candidates $\mathcal{F}^*(k_e)$ for increasing time horizon k_e

This algorithm can be applied to a large number of different models $\mathcal{A}_f$ and to I/O sequences of arbitrary length k_e because its complexity is only linear with respect to the number of models and the length of the sequences. The algorithm gives the best possible result, because it determines the set of fault candidates. For every element of the set $\mathcal{F}^*(k_e)$, the I/O pair is consistent with the corresponding model $\mathcal{A}_f$ up to time k_e and there is no fault in the remaining set $\mathcal{F}\backslash\mathcal{F}^*(k_e)$ for which the I/O pair is consistent with the model $\mathcal{A}_f$.

The diagnostic result has the following consequences. If f_0 does not belong to $\mathcal{F}^*(k_e)$, a fault is detected with certainty. For every time horizon k_e, the system may be subject to any fault $f \in \mathcal{F}^*(k_e)$.

Diagnosability of deterministic automata. An important question asks under what conditions the diagnostic method developed so far is able to detect a fault (fault detectability) or to unambiguously identify the fault (fault identifiability). These two properties will be investigated in the following paragraphs. As a basis for these investigations, the next section reviews known results on the equivalence of states of deterministic I/O automata from [211].

11.4.2 Results on Deterministic Automata with Equivalent States

Equivalence of states. This section considers automata with arbitrary initial states z_0. It compares the behaviour of a single automaton if this automaton starts its movement in two different initial states or the behaviour of two automata with different initial states.

Definition 11.1 (*Equivalent states*) Two states z and $\tilde{z}$ of the automaton $\mathcal{A} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, G, H)$ are said to be equivalent if the relation

$$\phi(z, V) = \phi(\tilde{z}, V) \tag{11.48}$$

holds for all $V \in \mathcal{V}^*$. Otherwise they are called *distinguishable*.

Equivalence of the states z and $\tilde{z}$ is indicated by $z \sim \tilde{z}$. An automaton that has no equivalent states is called *minimal*.

State equivalence of two automata. Definition 11.1 can be applied to states of two distinct automata

$$\mathcal{A}_i = (\mathcal{Z}_i, \mathcal{V}, \mathcal{W}, G_i, H_i) \quad \text{and} \quad \mathcal{A}_j = (\mathcal{Z}_j, \mathcal{V}, \mathcal{W}, G_j, H_j).$$

Analogously to Eq. (11.48), the states $z \in \mathcal{Z}_i$ and $\tilde{z} \in \mathcal{Z}_j$ are said to be equivalent if the relation

$$\phi_i(z, V) = \phi_j(\tilde{z}, V) \tag{11.49}$$

is satisfied for all $V \in \mathcal{V}^*$. Equation (11.49) can be written in the form (11.48), in which the same automaton map ϕ occurs on both sides, if the automata $\mathcal{A}_i$ and $\mathcal{A}_j$ are lumped together to get the automaton

$$\bar{\mathcal{A}} = (\bar{\mathcal{Z}}, \mathcal{V}, \mathcal{W}, \bar{G}, \bar{H})$$

with

$$\bar{\mathcal{Z}} = \mathcal{Z}_i \cup \mathcal{Z}_j$$

$$\bar{G}(z, v) = \begin{cases} G_i(z, v) & \text{if } z \in \mathcal{Z}_i \\ G_j(z, v) & \text{if } z \in \mathcal{Z}_j \end{cases} \qquad \bar{H}(z, v) = \begin{cases} H_i(z, v) & \text{if } z \in \mathcal{Z}_i \\ H_j(z, v) & \text{if } z \in \mathcal{Z}_j. \end{cases}$$

Denote the automaton map of $\bar{\mathcal{A}}$ by $\bar{\phi}$. Then the states $z \in \mathcal{Z}_i$ and $\tilde{z} \in \mathcal{Z}_j$ of the two automata $\mathcal{A}_i$ and $\mathcal{A}_j$ are equivalent if the relation

$$\bar{\phi}(z, V) = \bar{\phi}(\tilde{z}, V) \tag{11.50}$$

holds. Consequently, the equivalence test described in the next paragraph for a single automaton can also be applied for testing the equivalence of states of two automata.

Equivalence test. The equivalence test uses the following recursive equivalence definition:

- Two states z and $\tilde{z}$ are said to be 0-equivalent ($z \overset{0}{\sim} \tilde{z}$), if Eq. (11.48) holds for a single input symbol $V = v$. This is true if and only if the relation

  $$H(z, v) = H(\tilde{z}, v) \quad \text{for all} \quad v \in \mathcal{V}$$

  is satisfied. Accordingly, the state set $\mathcal{Z}$ is partitioned into sets $\mathcal{Z}_i^0$, ($i = 1, 2, \ldots, q_0$) such that a state pair $(z, \tilde{z})$ belongs to the same set $\mathcal{Z}_i^0$ if and only if the automaton produces for every input symbol the same output for both states:

$$z, \tilde{z} \in \mathcal{Z}_i^0 \Longleftrightarrow H(z, v) = H(\tilde{z}, v) \quad \text{for all} \quad v \in \mathcal{V}. \tag{11.51}$$

The function

$$H^* : \mathcal{Z}^0 \times \mathcal{V} \to \mathcal{W}$$

with

$$\mathcal{Z}^0 = \{\mathcal{Z}_i^0 \mid i = 1, 2, \ldots, q_0\}$$

is introduced to associate with each state set $\mathcal{Z}_i^0$ and input $v \in \mathcal{V}$ the output $w = H(z, v)$ such that Eq. (11.51) holds:

$$H^*(\mathcal{Z}_i^0, v) = H(z, v) \quad \text{for all} \quad z \in \mathcal{Z}_i^0, \ v \in \mathcal{V}. \tag{11.52}$$

States belonging to two different sets $\mathcal{Z}_i^0$ and $\mathcal{Z}_j^0$, $(i \neq j)$ are 0-distinguishable.

- For $k \geq 0$ two states z and $\tilde{z}$ are said to be $(k+1)$-equivalent ($z \overset{k+1}{\sim} \tilde{z}$), if Eq. (11.48) holds for all input sequences $V(0 \ldots l)$ with time horizon $l \leq k+1$. This is true if and only if the successor states $z' = G(z, v)$ and $\tilde{z}' = G(\tilde{z}, v)$ of z or $\tilde{z}$, respectively, are k-equivalent:

$$G(z, v) \overset{k}{\sim} G(\tilde{z}, v) \quad \text{for all} \quad v \in \mathcal{V}.$$

Hence, the kth partition of $\mathcal{Z}$ into the sets $\mathcal{Z}_i^k$, $i = 1, 2, \ldots, q_k$ is refined to get the $(k+1)$st partition of $\mathcal{Z}$ into the sets $\mathcal{Z}_i^{k+1}$, $(i = 1, 2, \ldots, q_{k+1})$ such that the relation

$$z, \tilde{z} \in \mathcal{Z}_i^{k+1} \Longleftrightarrow \forall v \in \mathcal{V} \ \exists j : G(z, v), G(\tilde{z}, v) \in \mathcal{Z}_j^k \tag{11.53}$$

holds. The function

$$G_k^* : \mathcal{Z}^{k+1} \times \mathcal{V} \to \mathcal{Z}^k$$

with

$$\begin{aligned} \mathcal{Z}^k &= \{\mathcal{Z}_i^k \mid i = 1, 2, \ldots, q_k\} \\ \mathcal{Z}^{k+1} &= \{\mathcal{Z}_i^{k+1} \mid i = 1, 2, \ldots, q_{k+1}\} \end{aligned}$$

associates with each state set $\mathcal{Z}_i^{k+1}$ and each input $v \in \mathcal{V}$ the set $\mathcal{Z}_j^k$ such that Eq. (11.53) holds. Hence, the relation

$$G(z, v) \in G_k^*(\mathcal{Z}_i^{k+1}, v) \quad \text{for all} \quad z \in \mathcal{Z}_i^{k+1}, \ v \in \mathcal{V} \tag{11.54}$$

is valid.

If states are $(N-1)$-equivalent with $N = |\mathcal{Z}|$, they are equivalent according to Definition 11.1. Then the final result of the recursive state partitioning is obtained

and denoted by symbols $\mathcal{Z}_i$ without superscript:

$$\mathcal{Z} = \cup_{i=1}^{q} \mathcal{Z}_i . \tag{11.55}$$

The mapping G^*_{N-2} is also denoted by G^*. Usually, the refinement of the state partitioning finishes before the $(N-1)$st refinement step.

If the states z and $\tilde{z}$ are not k-equivalent, they are called k-distinguishable.

Lemma 11.1 (Uniqueness of the state set partition) [211] *The state set partition (11.55) is unique. Two states $z, \tilde{z}$ belong to the same set $\mathcal{Z}_i$ if and only if they are equivalent.*

The lemma implies that the states $z, \tilde{z}$ are k-equivalent if and only if they belong to the same set $\mathcal{Z}_i^k$ of the kth state partition. Otherwise, they are k-distinguishable.

This test is summarised in the following algorithm:

Algorithm 11.2 *Partitioning of the state set into sets of equivalent states*

Given: Deterministic automaton $\mathcal{A}$

1. Determine the partition $\mathcal{Z}^0$ such that Eq. (11.51) is satisfied.
2. For $k = 0, 1, \ldots, N-2$, determine the partition $\mathcal{Z}^{k+1}$ such that Eq. (11.53) holds
   If $\mathcal{Z}^{k+1} = \mathcal{Z}^k$ holds, finish this step.
3. Denote the final partition obtained by $\mathcal{Z}_i$, $(i = 1, 2, \ldots, q)$.

Result: Partition (11.55) of the state set into sets $\mathcal{Z}_i$ of equivalent states.

This algorithm has the complexity $O(N^2)$, where N is the cardinality of $\mathcal{Z}$.

Properties of automata with equivalent states. Consider the state sequences that an automaton can generate starting in a pair $(z, \tilde{z})$ of equivalent states. An important fact

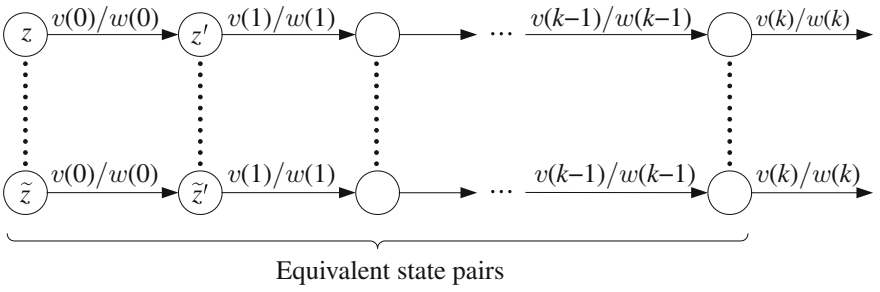


Fig. 11.15 State trajectories over equivalent state pairs

is that for all input sequences $V(0 \ldots k_e)$ with arbitrary time horizon k_e these two state sequences only go over equivalent state pairs $(z, \tilde{z})$, $(z', \tilde{z}')$ etc. (Fig. 11.15, cf. [128], Theorem 3.3). All state transitions involved have the same I/O pair $(v(k), w(k))$, $(k = 0, 1, \ldots, k_e)$.

For k-distinguishable state pairs $(z, \tilde{z})$, there exists an input sequence $V(0 \ldots k)$ for which the output sequences that are generated by the automaton starting in the initial state z or $\tilde{z}$, respectively, are not equal:

$$\phi(z, V(0 \ldots k)) \neq \phi(\tilde{z}, V(0 \ldots k)). \tag{11.56}$$

The input sequence $V(0 \ldots k)$ for which the relation (11.56) holds is called a *distinguishing input sequence* of the state pair $(z, \tilde{z})$.

If the states z and $\tilde{z}$ are distinguishable, distinguishing input sequences have at most $N - 1$ symbols. Hence, for distinguishable states the output sequences are identical for at most $N-2$ symbols and the fact that the states are distinguishable can be identified, for a reasonably chosen input sequence, in finite time. Consequently, all further investigations can be restricted to finite input sequences $V(0 \ldots k_e)$ with $k_e < N - 1$.

If z and $\tilde{z}$ are $(k-1)$-equivalent but k-distinguishable, then the output sequences W are identical up to the element $w(k-1)$ and there exists a distinguishing input sequence $\bar{V}(0 \ldots k)$ such that the output sequences are not completely identical:

$$\begin{aligned} &\phi(z, V) = \phi(\tilde{z}, V) \quad \text{for all} \quad V \in \mathcal{V}^l, \; l = 0, 1, \ldots, k \\ &\exists\, \bar{V} \in \mathcal{V}^{k+1} : \; \phi(z, \bar{V}) \neq \phi(\tilde{z}, \bar{V}). \end{aligned} \tag{11.57}$$

An important fact is that the output sequences

$$\begin{aligned} W(0 \ldots k) &= \phi(z, \bar{V}(0 \ldots k)) = (w(0), w(1), \ldots, w(k)) \\ \tilde{W}(0 \ldots k) &= \phi(\tilde{z}, \bar{V}(0 \ldots k)) = (\tilde{w}(0), \tilde{w}(1), \ldots, \tilde{w}(k)) \end{aligned}$$

differ only in in the last element $w(k)$:

$$\begin{aligned} w(0) &= \tilde{w}(0) \\ w(1) &= \tilde{w}(1) \\ &\vdots \\ w(k-1) &= \tilde{w}(k-1) \\ w(k) &\neq \tilde{w}(k). \end{aligned}$$

Therefore, the state trajectories starting in the states z and $\tilde{z}$ go over state pairs with decreasing equivalence properties (Fig. 11.16).

This fact has a direct interpretation. When starting in the two initial states z or $\tilde{z}$ the first k elements $v(0), \ldots, v(k-1)$ of the input sequence $\bar{V}(0 \ldots k)$ are used to

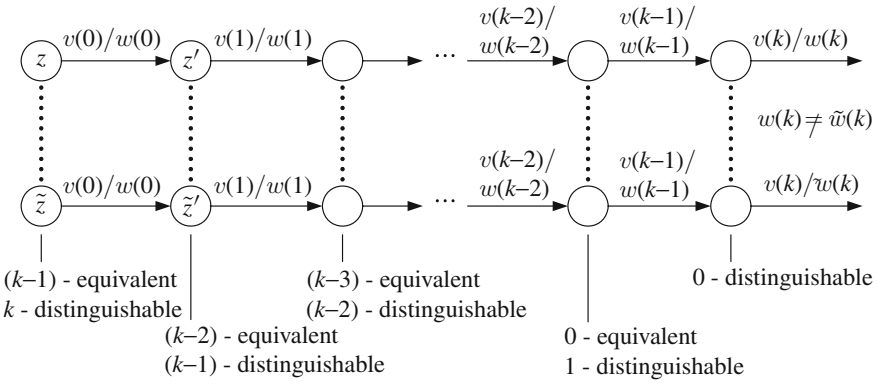


Fig. 11.16 State trajectories generated by a distinguishing input sequence $\bar{V}(0 \ldots k)$ that start in a k-distinguishable, $(k-1)$-equivalent state pair

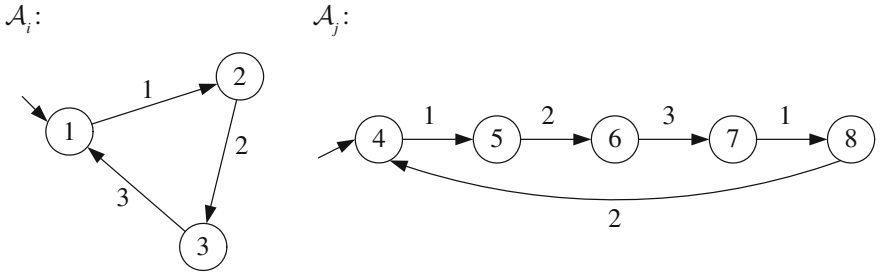


Fig. 11.17 Two automata

bring the system into two states $z(k)$ or $\tilde{z}(k)$, respectively, that are 0-distinguishable. Hence, for these states an input symbol $v(k)$ exists for which the outputs $w(k)$ and $\tilde{w}(k)$ generated in both states are different:

$$w(k) = H(z(k), v(k)) \neq H(\tilde{z}(k), V(k)) = \tilde{w}(k).$$

If the states to be tested belong to different automata with N_i of N_j states, respectively, the maximum length necessary to distinguish these states depends on the cardinality of both state sets. At most $\max_{i,j}(N_i, N_j) + 1$ input symbols are necessary. Figure 11.17 illustrates this fact for two automata with 3 or 5 states. To distinguish the states 1 and 4, an input sequence of length 6 is necessary.

11.4.3 Fault Detectability

Fault detection concerns the question whether or not a fault has occurred in the system. This section deals with the question under what conditions a fault f can be detected.

The notion of fault detectability describes the property of a system to change its behaviour in case of a fault f in such a way that a diagnostic system, knowing the I/O pair and the model $\mathcal{A}_0$ of the faultless system, can detect the fault f. The fault is detectable only if the I/O pair (V, W) generated by the faulty system is not consistent with the behaviour $\mathcal{B}_0$ of the faultless system. Since for the system subject to fault f, the output sequence is given by $W = \phi_f(z_{f0}, V)$, this condition can be represented as

$$(V, \phi_f(z_{f0}, V)) \notin \mathcal{B}_0. \tag{11.58}$$

Note that whether or not the relation (11.58) is satisfied depends upon the input sequence V. There may exist input sequences V such that the I/O pair generated by the faulty system coincides with some I/O pair of the faultless system and, hence, do not give any indication for the diagnostic system to detect the fault, whereas for other input sequences Eq. (11.58) holds.

Hence, fault detectability has to be defined as the chance to find out the presence of the fault. This "chance" exists if there is some input sequence such that the relation (11.58) is satisfied. This fact is the motivation for the following definition, in which, as before, it is assumed that the faultless system is described by the initialised automaton $(\mathcal{A}_0, z_{00})$ with the automaton map ϕ_0 and the system subject to fault f by the initialised automaton $(\mathcal{A}_f, z_{f0})$ with the automaton map ϕ_f.

Definition 11.2 (*Fault detectability*) The fault f is said to be *detectable* if there exists a finite input sequence V such that the relation

$$(V, \phi_f(z_{f0}, V)) \notin \mathcal{B}_0 \tag{11.59}$$

holds, where $\mathcal{B}_0$ denotes the behaviour of the faultless system $\mathcal{A}_0$.

Note that the detectability of a fault f is a property of the system to change its dynamical behaviour with respect to the faultless system $\mathcal{A}_0$. For a system, there may exist detectable and undetectable faults.

Detectability test. A direct consequence of the detectability definition is described in the following lemma:

Lemma 11.2 *A fault f is detectable if and only if the behaviour $\mathcal{B}_f$ of the system subject to fault f is different from the behaviour $\mathcal{B}_0$ of the faultless system*

$$\mathcal{B}_f \neq \mathcal{B}_0. \tag{11.60}$$

Condition (11.60) claims that there exists at least one I/O pair occurring for the faulty system that does not occur for the faultless system. In Fig. 11.18 the I/O pair

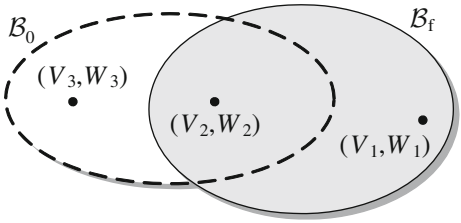


Fig. 11.18 Illustration of the detectability condition (11.60)

(V_1, W_1) satisfies the relation (11.59). It shows that if the input sequence V_1 is applied to the faulty system, the output sequence W_1 occurs and the fault can be detected because the I/O pair (V_1, W_1) is not consistent with the behaviour $\mathcal{B}_0$ of the faultless system. The figure also shows that the fault can only be detected for specific input sequences. If instead of V_1 the input sequence V_2 is applied, the faulty system results in the I/O pair (V_2, W_2) that belongs not only to $\mathcal{B}_f$ but also to $\mathcal{B}_0$ and the diagnostic system does not get any indication for the presence of the fault.

The property of fault detectability can be tested as follows:

Theorem 11.1 (Detectability criterion) *The fault f is detectable if and only if the initial states z_{00} and z_{f0} of the deterministic automata $\mathcal{A}_0$ or $\mathcal{A}_f$, respectively, are distinguishable.*

Proof (Sufficiency:) If the initial states are equivalent, then Eq. (11.49) holds, which for the automata considered here reads as

$$\phi_f(z_{f0}, V) = \phi_0(z_{00}, V) \qquad \text{for all } V \in \mathcal{V}^*.$$

Hence, the behaviours $\mathcal{B}_0$ and $\mathcal{B}_f$ of the automata $\mathcal{A}_0$ and $\mathcal{A}_f$ are identical, which is in contradiction to Eq. (11.60).

(Necessity:) If the initial states are distinguishable, then there exists an input sequence V such that the inequality

$$W_0 = \phi_0(z_{00}, V) \neq W_f = \phi_f(z_{f0}, V)$$

holds. Hence, the I/O pair (V, W_f) belongs to the behaviour $\mathcal{B}_f$ of the faulty system but not to the behaviour $\mathcal{B}_0$ of the faultless system and, due to Eq. (11.59), the fault is detectable. □

Fault detectability test. According to Theorem 11.1, the detectability of a fault f can be tested by applying the equivalence test described in Sect. 11.4.2 to the initial states z_{00} and z_{f0} of the automata $\mathcal{A}_0$ and $\mathcal{A}_f$. The complexity of this test is $O(N^2)$.

Remarks. The degree of distinguishability of the initial states z_0 and z_{f0} is a measure of the length of the input sequence V for which the fault f can be detected. If both states are k-distinguishable, a distinguishing input sequence $V(0 \ldots k)$ of length $k+1$ exists. A method for finding distinguishing input sequences with minimum length will be developed in Sect. 11.4.5.

To solve the fault detection task necessitates only the availability of the model $\mathcal{A}_0$ of the faultless system. However, the information included in this model is neither sufficient for the test of the detectability of the faults $f \in \mathcal{F}$ nor for the determination of a distinguishing input sequence V for which Eq. (11.59) holds. For both steps it has to be known how the behaviour of the faulty system distinguishes from the behaviour of the faultless system. This information is included in the model set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$.

11.4.4 Fault Identifiability

Fault identification is the task to find the fault $f \in \mathcal{F}$ that the system is subject to. This task requires to know the whole model set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$.

This section deals with the important question under what conditions it is possible to identify a fault f. Fault identifiability is a system property that depends upon the automaton maps ϕ_f or, equivalently, upon the state transition functions G_f and output functions H_f of the system for all $f \in \mathcal{F}$, but not on the diagnostic method.

The notion of fault identifiability should describe the situation that a diagnostic unit can be able to identify a fault after a finite number of input symbols. That is, for the I/O pairs generated for a specific input sequence V the relations

$$(V, W_{\tilde{f}}) \in \mathcal{B}_{\tilde{f}} \tag{11.61}$$

$$(V, W_f) \notin \mathcal{B}_f \quad \text{for all} \quad f \neq \tilde{f} \tag{11.62}$$

should hold for some fault $\tilde{f}$. Like fault detectability, the possibility to find the fault depends upon a reasonable choice of the input sequence V.

Figure 11.19 illustrates this situation for a system with the fault set $\mathcal{F} = \{0, 1, 2\}$. For every fault case $f \in \mathcal{F}$, there exists an input sequence denoted by V_f, $(f \in \mathcal{F})$ such that the output sequence

$$W_f = \phi(z_{f0}, V_f)$$

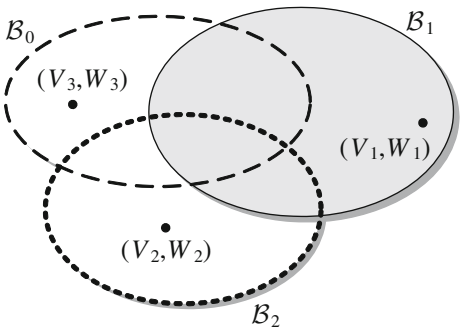


Fig. 11.19 Illustration of fault identification

generated by the system subject to fault f results in an I/O pair (V_f, W_f) that only belongs to the behaviour $\mathcal{B}_f$ relevant to this fault and not to the behaviour of the other fault cases. If the system gets this input sequence, the fault f can be unumbiguously identified.

Definition 11.3 (*Fault identifiability*) Consider a system that is described by a set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$ of deterministic automata. The fault $\tilde{f}$ is called *identifiable*, if there exists an input sequence V such that Eqs. (11.61) and (11.62) hold with $W_f = \phi_f(z_{f0}, V)$ and, hence, the set of fault candidates is a singleton: $\mathcal{F}^*(V, W) = \{\tilde{f}\}$.

Identifiability test. Before stating the identifiability criterion, the fault identification task is reformulated. Fault identification can be considered as the task to decide for a given I/O pair (V, W) which component $\mathcal{A}_f$ of the model set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$ generates for the input sequence V the output sequence W. The overall model

$$\bar{\mathcal{A}} = (\bar{\mathcal{Z}}, \mathcal{V}, \mathcal{W}, \bar{G}, \bar{H}),$$

which includes the behaviour of all models $\mathcal{A}_f$, $(f \in \mathcal{F})$ is obtained as follows:

$$\bar{\mathcal{Z}} = \cup_{f\in\mathcal{F}} \mathcal{Z}_f \tag{11.63}$$

$$\bar{G}(z, v) = G_f(z, v) \quad \text{if } z \in \mathcal{Z}_f \tag{11.64}$$

$$\bar{H}(z, v) = H_f(z, v) \quad \text{if } z \in \mathcal{Z}_f. \tag{11.65}$$

It is initialised with one of the initial states z_{f0}:

$$\bar{z} \in \bar{\mathcal{Z}}_0 = \{z_{f0} \mid f \in \mathcal{F}\}. \tag{11.66}$$

For such an initial state and an input sequence V, the automaton $\bar{\mathcal{A}}$ generates one of the output sequences that the models of the set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$ can generate. Hence, its behaviour $\bar{\mathcal{B}}$ includes the behaviour of all models $\mathcal{A}_f$:

$$\bar{\mathcal{B}} = \cup_{f\in\mathcal{F}} \mathcal{B}_f.$$

Theorem 11.2 (Identifiability criterion) [215] *Assume that all automata $\mathcal{A}_f$, $(f \in \mathcal{F})$ are minimal. Then all faults $f \in \mathcal{F}$ are identifiable if the automaton $\bar{\mathcal{A}}$ is minimal.*

Example 11.2 System with identifiable faults
As an example, consider the model $\mathcal{A}_f$, $(f = 0, 1, 2)$ shown in Fig. 11.20. The absence of equivalent states is proved by means of the state transition function $\bar{G}$ and the output function $\bar{H}$ of the automaton $\bar{\mathcal{A}}$ obtained by Eqs. (11.64), (11.65):

$\bar{G} =$

v \ z	1	2	3	4	5
1	2	1	4	3	5
2	2	1	4	3	5

,

$\bar{H} =$

v \ z	1	2	3	4	5
1	1	2	1	1	1
2	1	2	1	2	1

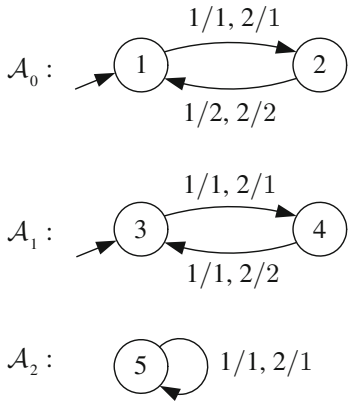


Fig. 11.20 Models of three fault cases

The analysis of the output function $\bar{H}$ results in the function H^*

$H^* =$

v \ z	$\mathcal{Z}_1^0$			$\mathcal{Z}_2^0$	$\mathcal{Z}_3^0$
	1	3	5	2	4
1	1	1	1	2	1
2	1	1	1	2	2

and the sets

$$\mathcal{Z}_1^0 = \{1, 3, 5\}, \quad \mathcal{Z}_2^0 = \{2\}, \quad \mathcal{Z}_3^0 = \{4\}.$$

Note that in the table representing the function H^*, the columns belonging to all states of the same set $\mathcal{Z}_i^0$ are identical. For a given input v, the entry in the corresponding row represents the value $H^*(\mathcal{Z}_i^0, v)$. Furthermore, the function G^* is obtained

$G^* =$

v \ z	$\mathcal{Z}_1^1$	$\mathcal{Z}_2^1$	$\mathcal{Z}_3^1$	$\mathcal{Z}_4^1$	$\mathcal{Z}_5^1$
	1	3	5	2	4
1	$\mathcal{Z}_2^0$	$\mathcal{Z}_3^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$
2	$\mathcal{Z}_2^0$	$\mathcal{Z}_3^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$

which proves that the automaton $\bar{\mathcal{A}}$ is minimal.

To select input sequences for which the faults are identifiable, note that the automata $\mathcal{A}_0$ and $\mathcal{A}_1$ are in the initial states $z_{00} = 1$ or $z_{01} = 3$, respectively, as shown in the figure. With the input sequence

$$V_{\text{det}} = (1, 1)$$

a fault can be detected, because the output sequences

$$W_{0\text{det}} = (1, 2), \quad W_{1\text{det}} = (1, 1), \quad W_{2\text{det}} = (1, 1)$$

are different for the faultless case ($W_{0\text{det}}$) and for the faulty cases ($W_{1\text{det}}$, $W_{2\text{det}}$). With the input sequence

$$V_{\text{id}} = (1, 2)$$

the output sequences of the models $\mathcal{A}_1$ and $\mathcal{A}_2$ are different

$$W_{1\text{id}} = (1, 2), \quad W_{2\text{id}} = (1, 1)$$

and the fault can be identified. In summary, the concatenated input sequence

$$V = V_{\text{det}} \cdot V_{\text{id}} = (1, 1, 1, 2)$$

yields the output sequences

$$\begin{aligned} W_0 &= W_{0\text{det}} \cdot W_{0\text{id}} = (1, 2, 1, 2) \\ W_1 &= W_{1\text{det}} \cdot W_{1\text{id}} = (1, 1, 1, 2) \\ W_2 &= W_{2\text{det}} \cdot W_{2\text{id}} = (1, 1, 1, 1), \end{aligned}$$

which unambiguously identify the fault. □

The example also shows that the length $4 = N_1 + N_2 + N_3 - 1$ of the distinguishing input sequence V depends on the cardinality $N_1 + N_2 + N_3$ of the overall model $\bar{\mathcal{A}}$ rather than on the separate cardinalities N_i of the state sets of the models $\mathcal{A}_i$, $(i = 0, 1, 2)$. However, the length 2 of the inputs V_{det} and V_{id} are due to the cardinality of pairs of submodels ($2 = \max_{i \neq j}(N_i + N_j) - 1$).

11.4.5 Method for Determining Distinguishing Input Sequences

The important result of the investigations of the preceding section is the fact that for identifiable faults there exist distinguishing input sequences V with finite length such that the pair (V, W) belongs to precisely one set $\mathcal{B}_f$, $(f \in \mathcal{F})$. The question considered in this section is how to find this input sequence:

Determination of distinguishing input sequences	
Given:	Deterministic automaton $\mathcal{A}$ with automaton map ϕ
	State pair $(z, \tilde{z})$
Find:	Input sequence V such that $\phi(z, V) \neq \phi(\tilde{z}, V)$

The main problem to be solved concerns the determination of the shortest input sequence with which identifiable faults can be unambiguously identified. It should

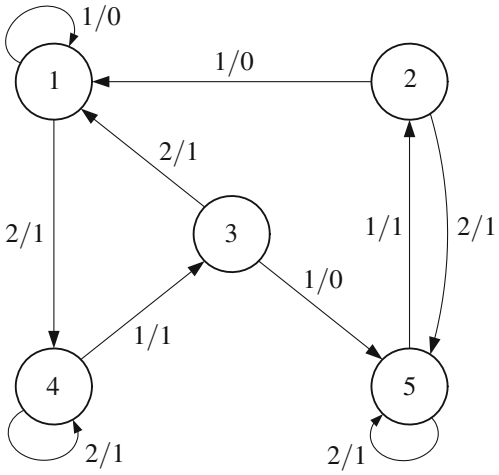


Fig. 11.21 Automaton

become clear after a *minimum* number of input symbols to which behaviour $\mathcal{B}_f$, ($f \in \mathcal{F}$) the I/O pair belongs.

The preceding section also has shown that the identification of a fault f can be reduced to the problem of identifying the state of the automaton $\bar{\mathcal{A}}$. If the result belongs to the state set $\mathcal{Z}_f$, then the fault f has been identified. As a consequence, this section deals with the problem of finding the shortest distinguishing input sequence for identifying the state of the automaton $\bar{\mathcal{A}}$.

Explanation of the method by an example. The solution to the problem stated above will be explained first by considering the example automaton shown in Fig. 11.21.

Example 11.3 Determination of distinguishing inputs
The automaton is described by the following state transition function G and output function H:

$G =$

v \ z	1	2	3	4	5
1	1	1	5	3	2
2	4	5	1	4	5

, $H =$

v \ z	1	2	3	4	5
1	0	0	0	1	1
2	1	1	1	1	1

To determine 0-equivalent states the output function is analysed with the following result:

$H^* =$

	$\mathcal{Z}_1^0$			$\mathcal{Z}_2^0$	
	1	2	3	4	5
$v = 1$	0	0	0	1	1
$v = 2$	1	1	1	1	1

The further decomposition of the state set $\mathcal{Z}$ is obtained by means of the state transition function G:

$G_0^* =$

	$\mathcal{Z}_1^1$		$\mathcal{Z}_2^1$	$\mathcal{Z}_3^1$	
	1	2	3	4	5
$v=1$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_2^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$
$v=2$	$\mathcal{Z}_2^0$	$\mathcal{Z}_2^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_2^0$	$\mathcal{Z}_2^0$

$\downarrow$

$G_1^* =$

	$\mathcal{Z}_1^2$		$\mathcal{Z}_2^2$	$\mathcal{Z}_3^2$	$\mathcal{Z}_4^2$
	1	2	3	4	5
$v=1$	$\mathcal{Z}_1^1$	$\mathcal{Z}_1^1$	$\mathcal{Z}_3^1$	$\mathcal{Z}_2^1$	$\mathcal{Z}_1^1$
$v=2$	$\mathcal{Z}_3^1$	$\mathcal{Z}_3^1$	$\mathcal{Z}_1^1$	$\mathcal{Z}_3^1$	$\mathcal{Z}_3^1$

$\downarrow$

$G_2^* =$

	$\mathcal{Z}_1^3$	$\mathcal{Z}_2^3$	$\mathcal{Z}_3^3$	$\mathcal{Z}_4^3$	$\mathcal{Z}_5^3$
	1	2	3	4	5
$v=1$	$\mathcal{Z}_1^2$	$\mathcal{Z}_1^2$	$\mathcal{Z}_4^2$	$\mathcal{Z}_2^2$	$\mathcal{Z}_1^2$
$v=2$	$\mathcal{Z}_3^2$	$\mathcal{Z}_4^2$	$\mathcal{Z}_1^2$	$\mathcal{Z}_3^2$	$\mathcal{Z}_4^2$

The result shows that the automaton does not possess equivalent states, because all state sets obtained are singletons.

In the following, it will be shown that the sequence of results obtained when testing the existence of equivalent states by means of Algorithm 11.2 can be used to determine distinguishing input sequences. First consider the function H^* defined in Eq. (11.52). The decomposition shows that for the input $v = 1$ the state pairs $(1, 4)$, $(1, 5)$, $(2, 4)$, $(2, 5)$, $(3, 4)$, $(3, 5)$ can be distinguished by the output that the system generates. For all six pairs, the automaton $\mathcal{A}$ gives the output $w = 0$ if it is in the first state and the output $w = 1$ for the second state. The listed state pairs are the elements of the set

$$\mathcal{Z}_1^0 \times \mathcal{Z}_2^0 = \{(1, 4), (1, 5), (2, 4), (2, 5), (3, 4), (3, 5)\}.$$

These state pairs are 0-distinguishable and the distinguishing input is $v = 1$.

The function G_0^* is obtained when partioning the state set $\mathcal{Z}_1^0$ into the sets $\mathcal{Z}_1^1$ and $\mathcal{Z}_2^1$:

$$\mathcal{Z}_1^0 = \mathcal{Z}_1^1 \cup \mathcal{Z}_2^1.$$

It shows how to distinguish between the state pairs

$$\mathcal{Z}_1^1 \times \mathcal{Z}_2^1 = \{(1, 3), (2, 3)\}.$$

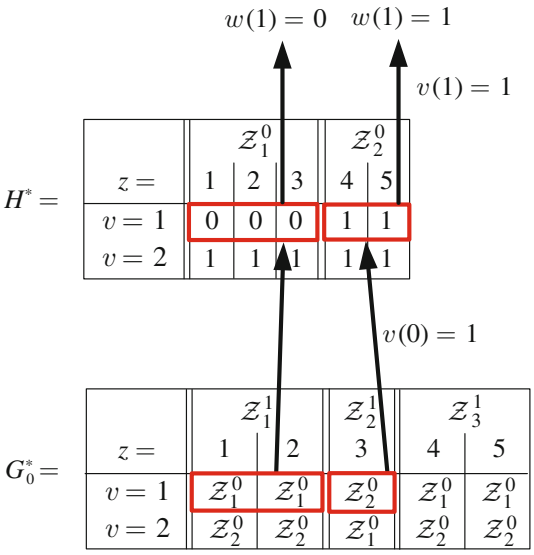


Fig. 11.22 Determination of the distinguishing input sequence of 1-distinguishing state pairs

These state pairs are 1-distinguishable and 0-equivalent, because they belong to the same set $\mathcal{Z}_1^0$ of the first decomposition and, thus, are not distinguishable without any state transition. However, if an input sequence of length 2 is used and, hence, one state transition is caused, these state pairs can be distinguished.

The distinguishing input sequence can be read off the decomposition by looking for a row in the G_0^* table with different entries for the state sets $\mathcal{Z}_1^1$ and $\mathcal{Z}_2^1$. The entries $\mathcal{Z}_1^0$ and $\mathcal{Z}_2^0$ in the first row say that using the input symbol $v(0) = 1$ state transitions occur where the states $z = 1$ and $z = 2$ are moved to one of the states $z' \in \mathcal{Z}_1^0$ whereas from the state $\tilde{z} = 3$ the automaton moves towards one of the states $\tilde{z}' \in \mathcal{Z}_2^0$ (Fig. 11.22). As the successor states belong to different sets of 0-distinguishable states, using the second input symbol $v(1) = 1$ the automaton gives the output $w(1) = 0$ if it has started in the initial state $z = 1$ or $z = 2$; whereas it generates the output $w(1) = 1$ if it has started in the state $\tilde{z} = 3$. Consequently, the state pairs

$$(z, \tilde{z}) \in \{(1, 3), (2, 3)\}$$

are 1-distinguishable and a distinguishing input sequence is

$$V(0 \ldots 1) = (1, 1).$$

As the final example, consider the last decomposition step of the state transition function G, where the set $\mathcal{Z}_1^2$ is partitioned

$$\mathcal{Z}_1^2 = \mathcal{Z}_1^3 \cup \mathcal{Z}_2^3.$$

Hence, the states 1 and 2, which are the only elements of the sets $\mathcal{Z}_1^3$ and $\mathcal{Z}_2^3$ are distinguishable after this decomposition step, which is the third one. Hence, these states are 3-distinguishable and 4 input symbols for a distinguishing input sequence $V(0 \ldots 3)$ have to be found by retracing the decomposition.

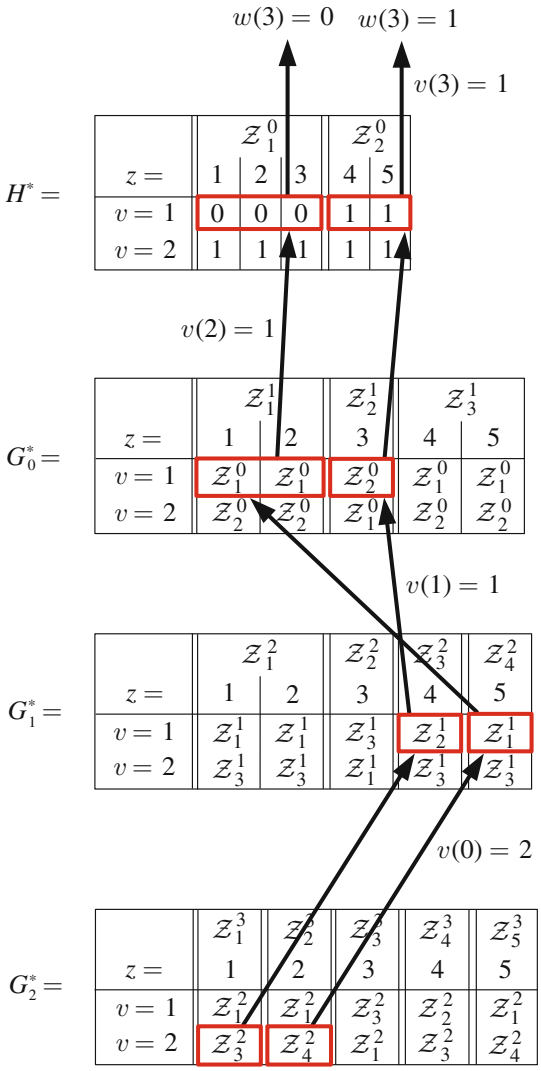


Fig. 11.23 Determination of the distinguishing input sequence of 3-distinguishing state pairs

As Fig. 11.23 shows the input sequence

$$V(0\ldots3) = (2, 1, 1, 1)$$

yields state transitions among the following state sets:

$$1 \xrightarrow{v(0)=2} \mathcal{Z}_3^2 \xrightarrow{v(1)=1} \mathcal{Z}_2^1 \xrightarrow{v(2)=1} \mathcal{Z}_2^0$$
$$2 \xrightarrow{v(0)=2} \mathcal{Z}_4^2 \xrightarrow{v(1)=1} \mathcal{Z}_1^1 \xrightarrow{v(2)=1} \mathcal{Z}_1^0.$$

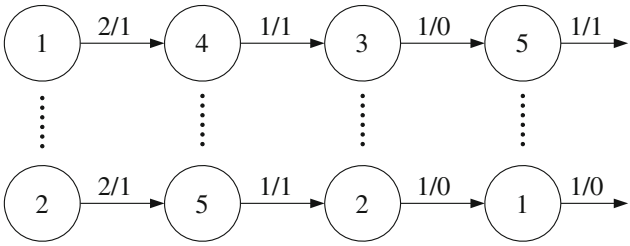


Fig. 11.24 State trajectories for determining whether the automaton is in the initial state 1 or 2

Obviously, the first three input symbols bring the automaton in a pair of 0-distinguishable states. The last input symbol $v(3) = 1$ is used to get two different output symbols $w(3) = 1$ or $w(3) = 0$, where the first symbol indicates that the automaton has started its movement in the state $z(0) = 1$; whereas the second symbol occurs if the automaton has had the initial state $z(0) = 2$.

What the automaton really does for the distinguishing input sequence, can be determined by means of the state transition function G and the output function H. The result is shown in Fig. 11.24. For the first three input symbols, the output symbols generated by the automaton are the same for both initial states. Only the last output symbol can be used as an indicator for determining the initial state in which the automaton has started its movement. This figure has the same structure as Fig. 11.16.

The analysis result can be summarised in the following table, which gives for all state pairs the distinguishing input sequence and shows which last output symbol occurs for both initial states:

z	$\tilde{z}$	$V(0 \ldots k)$	$w(k)$ for $z(0) = z$	$w(k)$ for $z(0) = \tilde{z}$
1	2	(2, 1, 1, 1)	1	0
1	3	(1, 1)	0	1
1	4	1	0	1
1	5	1	0	1
2	3	(1, 1)	0	1
2	4	1	0	1
2	5	1	0	1
3	4	1	0	1
3	5	1	0	1
4	5	(1, 1, 1)	1	0

Note that there may be several distinguishing input sequences, but the sequences shown here have minimum length. □

Algorithm. The method explained in the preceding section, for example automaton, is now formalised to get an algorithm, which proceeds in two main steps:

1. The state set $\mathcal{Z}$ is partitioned into sets of equivalent states:

$$\mathcal{Z} = \cup_{i=1}^{q} \mathcal{Z}_i .$$

As intermediate results, the partitions

$$\mathcal{Z} = \cup_{i=1}^{q_k} \mathcal{Z}_i^k, \quad k = 1, 2, \ldots, N-1$$

are obtained together with the functions H^* and G_k^*, $(k = 0, 1, \ldots, N-3)$ such that Eq. (11.51) or (11.53) and

$$\mathcal{Z}_i = \mathcal{Z}_i^{N-2}, \quad i = 1, 2, \ldots, q$$

hold. If z and $\tilde{z}$ are not equivalent, they belong to different sets $\mathcal{Z}_i$ and $\mathcal{Z}_j$ $(i \neq j)$. Otherwise, no distinguishing input sequence exists.

2. The result of the first step is evaluated backwards in the following way. First, find the number k_e for which the states z and $\tilde{z}$ are elements of a common set $\mathcal{Z}_l^{k_\mathrm{e}-1}$ and of disjoint sets $\mathcal{Z}_i^{k_\mathrm{e}}$ and $\mathcal{Z}_j^{k_\mathrm{e}}$, $(i \neq j)$:

$$k_\mathrm{e}: \; z, \tilde{z} \in \mathcal{Z}_l^{k_\mathrm{e}-1} \quad \text{and} \quad z \in \mathcal{Z}_i^{k_\mathrm{e}}, \; \tilde{z} \in \mathcal{Z}_j^{k_\mathrm{e}} \text{ for some } i \neq j. \tag{11.67}$$

Hence, z and $\tilde{z}$ are k_e-distinguishable. Introduce the new symbols

$$\mathcal{Z}^{k_\mathrm{e}} = \mathcal{Z}_i^{k_\mathrm{e}} \quad \text{and} \quad \tilde{\mathcal{Z}}^{k_\mathrm{e}} = \mathcal{Z}_j^{k_\mathrm{e}}.$$

Second, to find the distinguishing input sequence $V(0 \ldots k_\mathrm{e})$ determine the input $v(k_\mathrm{e} - k) = v_k$ for $k = k_\mathrm{e}, k_\mathrm{e} - 1, \ldots, 1$ such that

$$G_{k-1}^*(\mathcal{Z}^k, v_k) = \mathcal{Z}^{k-1} \neq G_{k-1}^*(\tilde{\mathcal{Z}}^k, v_k) = \tilde{\mathcal{Z}}^{k-1}. \tag{11.68}$$

Finally, choose the input $v(k_\mathrm{e}) = v_0$ such that

$$H^*(\mathcal{Z}^0, v_0) = w \neq H^*(\tilde{\mathcal{Z}}^0, v) = \tilde{w}. \tag{11.69}$$

The result is the distinguishing input sequence

$$V(0 \ldots k_\mathrm{e}) = (v(0), v(1), \ldots, v(k_\mathrm{e})) = (v_{k_\mathrm{e}}, v_{k_\mathrm{e}-1}, \ldots, v_1, v_0). \tag{11.70}$$

These steps can be formally described as the following algorithm:

Algorithm 11.3 *Determination of a distinguishing input sequence*

Given: Deterministic automaton $\mathcal{A}$ and state pair $(z, \tilde{z})$

1. Determine the sets $\mathcal{Z}_i^0$, $(i = 1, \ldots, q_0)$ and the function H^* according to Eq. (11.52).
2. Determine the sets $\mathcal{Z}_i^k$, $(k = 1, 2, \ldots, N-1,\ i = 1, \ldots, q_k)$ and the functions G_k^*, $(k = 1, 2, \ldots, N-2)$ according to Eq. (11.54).

   If z and $\tilde{z}$ belong to the same set $\mathcal{Z}_i^{N-1}$, stop (they are not distinguishable).
3. Determine the length k_e of the distinguishing input sequence according to Eq. (11.67).
4. For $k = k_\mathrm{e}, k_\mathrm{e} - 1, \ldots, 1$ determine v_k such that Eq. (11.68) holds.
5. Determine v_0 by means of Eq. (11.69).

Result: Distinguhishing input sequence (11.70).

Theorem 11.3 (Minimum distinguishing input sequences) *Algorithm 11.3 results in a minimum distinguishing input sequence whenever the states z and $\tilde{z}$ are distinguishable.*

Proof According to Lemma 11.1, the state set partition obtained by Steps 1 and 2 of the algorithm is unique. The states z and $\tilde{z}$ belong to the same set $\mathcal{Z}_i^k$ if and only if they are k-equivalent. Hence, Eq. (11.67) ensures that the length k_e of the input sequence is minimal. Furthermore, the choice of the input symbols according to Eq. (11.68) ensures that the state sequence ends in 0-distinguishing state sets and, finally, the output sequences generated by the automaton $\mathcal{A}$ for the two different initial states z and $\tilde{z}$ distinguish in the last symbol $w(k_\mathrm{e})$. Hence, the input sequence obtained by the algorithm is distinguishing. □

11.5 Diagnosis of Nondeterministic Automata

11.5.1 Method for Testing the Consistency of an I/O Pair with a Nondeterministic Automaton

This section extends the diagnostic method explained in Sect. 11.4 for deterministic automata towards nondeterministic automata

$$\mathcal{N} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L_\mathrm{n}, \mathcal{Z}_0).$$

The main problems to be solved concern the new description of the system dynamics by the behavioural relation L_n, which replaces the functions G and H of the deterministic automaton, and the fact that the nondeterminism of the model brings about ambiguities into the diagnostic result with any state transition.

The main idea is again the test of the consistency of the measured I/O pair $(V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e}))$ with the model $\mathcal{N}_f$ of the system subject to fault f. As this test is independent of the fault case, the index f is left out in the following development of the test method. $\mathcal{Z}_0$ is a given set of states, in which the automaton starts its movement.

According to the definition (11.4), an I/O pair (V, W) is consistent with a model $\mathcal{N}$ if the relation

$$(V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e})) \in \mathcal{B}$$

holds, where $\mathcal{B}$ is the behaviour (11.15) of the model $\mathcal{N}$. In order to show clearly how to carry out this test, the elements of the measured sequences are marked by a bar:

$$V(0\ldots k_\mathrm{e}) = (\bar{v}_0, \bar{v}_2, \ldots, \bar{v}_{k_\mathrm{e}}) \tag{11.71}$$

$$W(0\ldots k_\mathrm{e}) = (\bar{w}_0, \bar{w}_2, \ldots, \bar{w}_{k_\mathrm{e}}). \tag{11.72}$$

For these representations, consistency with the model $\mathcal{N}$ claims that there exists a state sequence

$$Z(0\ldots k_\mathrm{e}+1) = (z_0, z_1, \ldots, z_{k_\mathrm{e}+1})$$

such that the relation

$$L_\mathrm{n}(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) = 1$$

holds for $k = 0, 1, \ldots, k_\mathrm{e}$ and $z_0 \in \mathcal{Z}_0$. Equivalently, the relation

$$\exists Z(0\ldots k_\mathrm{e}+1): \prod_{k=0}^{k_\mathrm{e}} L_\mathrm{n}(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) = 1 \tag{11.73}$$

has to be valid, which can be represented by summing over all possible state sequences:

$$\sum_{z_{k_\mathrm{e}+1}\in\mathcal{Z}} \sum_{z_{k_\mathrm{e}}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_\mathrm{e}} L_\mathrm{n}(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) \geq 1. \tag{11.74}$$

This fact is stated by the following lemma:

Lemma 11.3 *The I/O pair (11.71), (11.72) is consistent with the nondeterministic automaton $\mathcal{N}$ if and only if the condition (11.74) is satisfied.*

Recursive test. The following presents a recursive test of the condition (11.74). The function $p(z', k_e + 1)$ is used as an indicator whether $(p(z', k_e + 1) = 1)$ or not $(p(z', k_e + 1) = 0)$ a state sequence $Z(0 \ldots k_e + 1)$ with the last element $z_{k_e+1} = z'$ exists for which the condition (11.73) is satisfied. This function can be determined recursively as follows:

$$p'(z, 0) = \begin{cases} 1 & \text{if } z \in \mathcal{Z}_0 \\ 0 & \text{else} \end{cases} \tag{11.75}$$

$$p'(z', k+1) = \left\lfloor \sum_{z \in \mathcal{Z}} L_n(z', \bar{w}_k, z, \bar{v}_k) \cdot p'(z, k) \right\rfloor, \quad k = 0, \ldots, k_e, \tag{11.76}$$

where the symbol $\lfloor . \rfloor$ signifies the replacement of any positive integer by 1:

$$\lfloor p \rfloor = \begin{cases} 0 & \text{if } p = 0 \\ 1 & \text{if } p \geq 1. \end{cases}$$

The I/O pair (11.71), (11.72) is consistent with the model $\mathcal{N}$ if and only if the inequality

$$\bar{p} = \left\lfloor \sum_{z' \in \mathcal{Z}} p'(z', k_e + 1) \right\rfloor > 0 \tag{11.77}$$

holds.

As a remark, it should be mentioned that the operation $\lfloor . \rfloor$ reduces the value of the argument to 1 in order to avoid increasing values in the sequence $p(z, k)$, $(k = 0, 1, \ldots, k_e)$. The consistency test only distinguishes between vanishing and non-vanishing values.

Lemma 11.4 *The I/O pair (11.71), (11.72) is consistent with the model $\mathcal{N}$ if and only if the condition (11.77) is satisfied, where $p(z, k_e + 1)$ is obtained by the recursion relation (11.75), (11.76).*

Proof The lemma is proved by showing that the condition (11.77) is satisfied if and only if the condition (11.74) is satisfied, which is necessary and sufficient for the consistency. To do so, define the function $\tilde{p}(z, k)$ as follows:

$$\tilde{p}(z, 0) = \begin{cases} 1 & \text{if } z \in \mathcal{Z}_0 \\ 0 & \text{else} \end{cases} \tag{11.78}$$

$$\tilde{p}(z', k+1) = \sum_{z \in \mathcal{Z}} L_n(z', \bar{w}_k, z, \bar{v}_k) \cdot \tilde{p}(z, k) \quad k = 0, 1, \ldots, k_e. \tag{11.79}$$

Obviously, $p'(z, k) > 0$ holds if and only if $\tilde{p}(z, k) > 0$ is valid. It will be proved now that the equation

$$\tilde{p}(z_{k_e+1}, k_e+1) = \sum_{z_{k_e}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_e} L_n(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) \tag{11.80}$$

holds which proves the lemma.

The proof is done by induction. For $k_e = 0$, Eqs. (11.78) and (11.79) yield

$$\begin{aligned}\tilde{p}(z_1, 1) &= \sum_{z_0\in\mathcal{Z}} L_n(z_1, \bar{w}_0, z_0, \bar{v}_0) \cdot \tilde{p}(z_0, 0) \\ &= \sum_{z_0\in\mathcal{Z}_0} L_n(z_1, \bar{w}_0, z_0, \bar{v}_0),\end{aligned}$$

which is identical to Eq. (11.80) for $k_e = 0$.

Now assume that Eq. (11.80) holds for some $k_e = k_e$

$$\tilde{p}(z_{k_e+1}, k_e+1) = \sum_{z_{k_e}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_e} L_n(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k)$$

and prove that this relation is satisfied for $k_e = k_e + 1$:

$$\tilde{p}(z_{k_e+2}, k_e+2) = \sum_{z_{k_e+1}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_e+1} L_n(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k). \tag{11.81}$$

A reformulation of the right-hand side of this equation results in

$$\begin{aligned}&\sum_{z_{k_e+1}\in\mathcal{Z}} \sum_{z_{k_e}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_e+1} L_n(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) \\ &= \sum_{z_{k_e+1}\in\mathcal{Z}} L_n(z_{k_e+2}, \bar{w}_{k_e+1}, z_{k_e+1}, \bar{v}_{k_e+1}) \\ &\quad \cdot \left(\sum_{z_{k_e}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_e} L_n(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) \right) \\ &= \sum_{z_{k_e+1}\in\mathcal{Z}} L_n(z_{k_e+2}, \bar{w}_{k_e+1}, z_{k_e+1}), \bar{v}_{k_e+1}) \cdot \tilde{p}(z_{k_e+1}, k_e+1) \\ &= \tilde{p}(z_{k_e+2}, k_e+2)\end{aligned}$$

which proves Eq. (11.81). □

Test algorithm. The recursive test leads to the following algorithm:

Algorithm 11.4 *Consistency test for nondeterministic automata*

Given: Nondeterministic automaton $\mathcal{N}$
I/O pair (11.71), (11.72)

1. Determine $p'(z, 0)$ by Eq. (11.75)
2. Apply Eq. (11.76) for $k = 0, 1, \ldots, k_e$ to determine $p'(z', k_e + 1)$
3. Test the condition (11.77).

Result: If and only if the condition (11.77) is satisfied, the I/O pair is consistent with the model $\mathcal{N}$

State observation result. As a byproduct of the consistency test, the set $\mathcal{Z}(k_e + 1)$ of states is obtained in which the automaton $\mathcal{N}$ can recide after it has accepted the input sequence $V(0 \ldots k_e)$ and generated the output sequence $W(0 \ldots k_e)$. This set is given by

$$\mathcal{Z}(k+1) = \{z' \in \mathcal{Z} \mid p'(z', k+1) > 0\}, \quad k = 0, 1, \ldots, k_e. \tag{11.82}$$

Hence, all state sequences $Z(0 \ldots k_e)$ considered in the consistency tests end in a state

$$z(k_e + 1) \in \mathcal{Z}(k_e + 1). \tag{11.83}$$

Corollary 11.1 *The I/O pair* $(V(0 \ldots k_e), W(0 \ldots k_e))$ *is consistent with the model* $\mathcal{N}$ *if and only if* $\mathcal{Z}(k_e + 1) \neq \emptyset$ *holds.*

To present the result of the consistency test in a similar way as for deterministic automata, the indicator $p(k)$ is introduced as follows:

$$p(k_e) = \begin{cases} 1 \text{ if } \mathcal{Z}(k_e + 1) \neq \emptyset \\ 0 \text{ else.} \end{cases}$$

11.5.2 Diagnostic Algorithm

The consistency test developed so far has to be applied to a set $\{\mathcal{N}_f \mid f \in \mathcal{F}\}$ of nondeterministic automata to get a fault identification algorithm. This algorithm works online, where the next measured I/O pair $(\bar{v}, \bar{w})$ is processed to get the sets $\mathcal{Z}$ and $\mathcal{Z}'$ of possible current states z or possible future states z', respectively, to decide about the fault candidates. Instead of $\mathcal{Z}$ and $\mathcal{Z}'$, indicators $p(z)$ and $p'(z')$ for the states $z, z' \in \mathcal{Z}$ to be elements of $\mathcal{Z}$ or $\mathcal{Z}'$, respectively, are determined. A model $\mathcal{N}_f$ is consistent with the I/O pair up to the current time horizon k_e if these sets are

non-empty, which is again represented by the indicator

$$p_f(k_\mathrm{e}) = \begin{cases} 1 \text{ if } \mathcal{Z}_f(k_\mathrm{e}+1) \neq \emptyset \\ 0 \text{ else.} \end{cases}$$

With this information, the diagnostic algorithm for nondeterministic automata can be formulated similarly to Algorithm 11.1 as follows:

Algorithm 11.5 *Diagnosis of nondeterministic automata*

Given: Nondeterministic automata $\mathcal{N}_f$, $(f \in \mathcal{F})$
Set of initial states $\mathcal{Z}_0$

Initialisation: $p'_f(z) = \begin{cases} 1 & \text{if } z \in \mathcal{Z}_0 \\ 0 & \text{else} \end{cases}$ for all $f \in \mathcal{F}$
$k_\mathrm{e} = 0$

Loop: 1. Measure the next I/O pair $(\bar{v}, \bar{w})$.

2. Determine $p_f(z) = \left\lfloor \sum_{z' \in \mathcal{Z}} L_\mathrm{n}(z', \bar{w}, z, \bar{v}) \cdot p'_f(z) \right\rfloor$ for all $f \in \mathcal{F}$.

3. Determine $p'_f(z') = \left\lfloor \sum_{z \in \mathcal{Z}} L_\mathrm{n}(z', \bar{w}, z, \bar{v}) \cdot p_f(z) \right\rfloor$ for all $f \in \mathcal{F}$.

4. Set $p_f(k_\mathrm{e}) = \left\lfloor \sum_{z' \in \mathcal{Z}} p'_f(z') \right\rfloor$ for all $f \in \mathcal{F}$.

5. Determine $\mathcal{F}^*(k_\mathrm{e}) = \{f \in \mathcal{F} : p_f(k_\mathrm{e}) = 1\}$.

6. $k_\mathrm{e} := k_\mathrm{e} + 1$
Continue with Step 1.

Result: Set of fault candidates $\mathcal{F}^*(k_\mathrm{e})$ for increasing time horizon k_e

The sets $\mathcal{Z}_f$ and $\mathcal{Z}'_f$ of current states or future states that are necessary to decide about the consistency can be obtained by

$$\begin{aligned} \mathcal{Z}_f &= \{z \in \mathcal{Z} : p_f(z) = 1\} \\ \mathcal{Z}'_f &= \{z' \in \mathcal{Z} : p'_f(z') = 1\}. \end{aligned}$$

Algorithm 11.5 can be used to solve the following fault diagnostic tasks:

- **Fault detection**: If the algorithm is applied to the single model $\mathcal{N}_0$ describing the faultless system, a fault is detected, if after Step 4 the relation $p_0(k_\mathrm{e}) = 0$ holds.

- **Fault identification**: For a set of models $\mathcal{N}_f$, ($f \in \mathcal{F}$), the algorithm yields the set of fault candidates $\mathcal{F}^*(k_e)$, which is the best possible diagnostic result.

Example 11.4 Fault detection of a nondeterministic automaton
Consider the nondeterministc automata $\mathcal{N}_0$ and $\mathcal{N}_1$ whose automaton graphs are depicted in Fig. 11.25. They describe a system in the faultless case or subject to fault $f = 1$, respectively.

The initial state is assumed to belong to the set

$$\mathcal{Z}_0(0 \mid -1) = \{1, 2, 3, 4\} \quad \text{and} \quad \mathcal{Z}_1(0 \mid -1) = \{5, 6, 7, 8\},$$

which yields the initial value of $\mathcal{Z}_0'$ and $\mathcal{Z}_1'$ in the initialisation step of the algorithm. After the first measurement

$$v(0) = 1 \quad \text{and} \quad w(0) = 1$$

has been obtained, Step 2 yields $p_0(z)$ and $p_1(z)$ for all $z \in \mathcal{Z}$ and, hence, the following sets

$$\mathcal{Z}_0(0 \mid 0) = \{1, 2, 3\} = \mathcal{Z}_0 \quad \text{and} \quad \mathcal{Z}_1(0 \mid 0) = \{5, 6, 7\} = \mathcal{Z}_1,$$

which represent all initial states z_0 for which a state transition exists such that the nondeterministic automaton generates the output $w(0) = 1$ for the input $v(0) = 1$. In the automaton graphs this sets can be found by looking for all states $z \in \mathcal{Z}(0 \mid -1)$ in which an edge starts that is labelled with $v = 1$ and $w = 1$. Step 3 the algorithm determines the sets $\mathcal{Z}_0(1 \mid 0)$ and $\mathcal{Z}_1(1 \mid 0)$ which in the notation used above coincide with the sets

$$\mathcal{Z}_0' = \{1, 2, 3, 4\} \quad \text{and} \quad \mathcal{Z}_1' = \{5, 6, 7, 8\}.$$

These are the sets of states in which the automata can be after they have generated the output $w(0) = 1$ for the input $v(0) = 1$. As both automata are consistent with the measurement obtained so far ($p_0(0)) = 1$, $p_1(0) = 1$), the set of fault candidates includes both faults:

$$\mathcal{F}^*(0) = \{0, 1\}.$$

For the next measurement

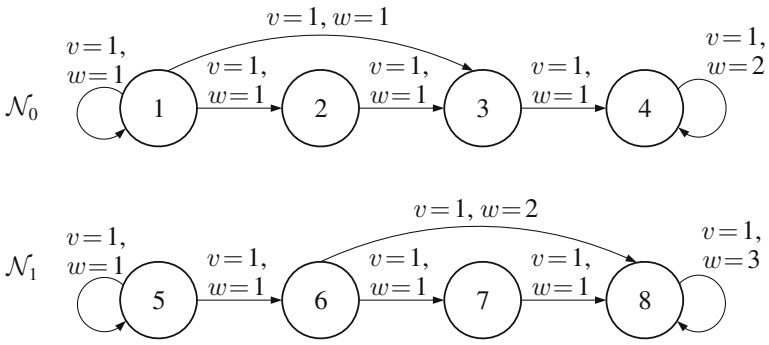


Fig. 11.25 Automaton graph of the example

$$v(1) = 1 \quad \text{and} \quad w(1) = 1$$

the Steps 2 and 3 of the algorithm yield again

$$\mathcal{Z}_0 = \{1, 2, 3\} \quad \text{and} \quad \mathcal{Z}_1 = \{5, 6, 7\}$$

and

$$\mathcal{Z}'_0 = \{1, 2, 3, 4\} \quad \text{and} \quad \mathcal{Z}'_1 = \{5, 6, 7, 8\}.$$

and for

$$v(2) = 1 \quad \text{and} \quad w(2) = 1$$

the same sets. An improvement of the diagnostic result is obtained after the measurement

$$v(3) = 1 \quad \text{and} \quad w(3) = 3$$

has been obtained, because then the results

$$\mathcal{Z}_0 = \emptyset \quad \text{and} \quad \mathcal{Z}_1 = \{8\}$$

show that the measurement sequence is inconsistent with the faultless case and, thus, the set of fault candidates only includes the single fault $f = 1$:

$$\mathcal{F}^*(3) = \{1\}.$$

Hence, the I/O pair (1, 1, 1, 1), (1, 1, 1, 3) yields the diagnostic result that a fault has occurred (the faultless case is inconsistent with the measurements) and that the set of fault candidates only includes, as a single fault, the fault case $f = 1$. □

11.6 State Observation of Stochastic Automata

This and the next section extend the methods developed so far towards stochastic automata

$$\mathcal{S} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L, p_0(z)).$$

The investigations are cut into two parts. In the first part described in this section, the state observation problem is solved, which does not only yield a test for the consistency of the measured I/O pair with a stochastic automaton, but also the probability for each state to be the current state of the automaton after the I/O pair has appeared. In the second part presented in the next section, this method is extended to determine the probability distribution over the set $\mathcal{F}$ of all faults considered to get a diagnostic result.

11.6.1 Method for Testing the Consistency of an I/O Pair with a Stochastic Automaton

The basis for finding fault candidates is the check whether the measured I/O pair belongs to the behaviour $\mathcal{B}$, which is defined for the stochastic automaton in Eq. (11.25). The consistency test should not only answer the question whether the I/O pair (V, W) belongs to the behaviour $\mathcal{B}$, but also with which probability this I/O pair occurs for the automaton $\mathcal{S}$. The probability information included in the behavioural relation L should be used to distinguish between I/O pairs that may occur often and those pairs that appear seldom. If this method is applied to the models of the faulty system, it should be possible to distinguish between faults with higher probability and rarely occurring faults.

This section extends the consistency test from nondeterministic towards stochastic automata. As this test is independent of the fault case, the dependency of the model upon the fault $f \in \mathcal{F}$ is omitted in this section.

The consistency test uses the representation (11.71), (11.72) of the I/O pair, where the bar over the symbols indicate that these symbols are measured and, hence, known in the test. The I/O pair defines the values for the stochastic variables $V(k)$ and $W(k)$, $(k = 0, 1, \ldots, k_\mathrm{e})$ that represent the current value of the input and the output signals.

An I/O pair is consistent with the model $\mathcal{S}$ if a state sequence

$$Z(0 \ldots k_\mathrm{e} + 1) = (z_0, z_1, \ldots, z_{k_\mathrm{e}+1}) \tag{11.84}$$

with positive probability exists, which satisfies the relation

$$\exists Z(0 \ldots k_\mathrm{e} + 1) : \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0) > 0. \tag{11.85}$$

This inequality replaces Eq. (11.73) for the nondeterministic automaton. The last term represents the a-priori probability of the initial state:

$$p_0(z) = \mathrm{Prob}(Z(0) = z).$$

Accordingly, $\mathcal{Z}_0 = \{z \in \mathcal{Z} : p_0(z) > 0\}$ holds.

Analogously to Eq. (11.74) the condition (11.85) can be formulated as

$$\sum_{z_{k_\mathrm{e}+1} \in \mathcal{Z}} \sum_{z_{k_\mathrm{e}} \in \mathcal{Z}} \cdots \sum_{z_0 \in \mathcal{Z}} \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0) \geq 0. \tag{11.86}$$

Lemma 11.5 *The I/O pair (11.71), (11.72) is consistent with the stochastic automaton $\mathcal{S}$ if and only if the condition (11.86) is satisfied.*

Recursive solution of the state observation problem. If the condition (11.86) is satisfied, there exists a state sequence that appears for the I/O pair (V, W) with positive probability. Hence, there is a non-empty set of states in which the automaton $\mathcal{S}$ resides after this I/O pair has appeared. For the solution of the fault diagnostic problem, it is not only necessary to know whether this set is non-empty, but also with which probability the elements of this set appear as the possible state of the automaton. The following recursive solution of the state observation problem produces this probability distribution.

The probability of the state sequence (11.84) for the I/O pair (11.71), (11.72) is denoted by

$$\begin{aligned}&\mathrm{Prob}(Z(k_\mathrm{e}) = z_{k_\mathrm{e}}, \ldots, Z(0) = z_0 \mid \\ &\quad V(k_\mathrm{e}) = \bar{v}_{k_\mathrm{e}}, \ldots, V(0) = \bar{v}_0,\ W(k_\mathrm{e}) = \bar{w}_{k_\mathrm{e}}, \ldots, W(0) = \bar{w}_0).\end{aligned}$$

As this notation is rather complex, this probability will be abbreviated as

$$\mathrm{Prob}(Z(0 \ldots k_\mathrm{e}) \mid V(0 \ldots k_\mathrm{e}), W(0 \ldots k_\mathrm{e}))$$

in the future with similar notations for other probabilities. The following lemma shows how this probability can be determined.

Lemma 11.6 (Probability distribution of the state sequence) *Consider a stochastic automaton with initial state probability distribution $p_0(z)$ and a consistent I/O pair (V, W). Then*

$$\begin{aligned}&\mathrm{Prob}(Z(0 \ldots k_\mathrm{e}) \mid V(0 \ldots k_\mathrm{e}), W(0 \ldots k_\mathrm{e})) \\ &= \frac{\sum\limits_{z_{k_\mathrm{e}+1}} \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0)}{\sum\limits_{Z(0 \ldots k_\mathrm{e}+1)} \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0)}\end{aligned} \tag{11.87}$$

describes the probability that the stochastic automaton has generated the state sequence $Z(0 \ldots k_\mathrm{e})$.

Note that the denominator in Eq. (11.87) does not vanish because the I/O pair is assumed to be consistent and, hence, Eq. (11.86) holds. In the application of Lemma 11.6 first the inequality (11.86) can be checked to find out whether the I/O pair is consistent with the automaton and if this test is successful Eq. (11.87) is applied to determine the probability distribution

$$\mathrm{Prob}(Z(0 \ldots k_\mathrm{e}) \mid V(0 \ldots k_\mathrm{e}), W(0 \ldots k_\mathrm{e}))$$

for all state sequences $Z(0 \ldots k_\mathrm{e}) \in \mathcal{Z}^*$.

Proof According to Bayes' formula, the relation

$$\begin{aligned}&\text{Prob}(Z(0\dots k_\mathrm{e}) \mid V(0\dots k_\mathrm{e}), W(0\dots k_\mathrm{e})) \\ &= \frac{\text{Prob}(Z(0\dots k_\mathrm{e}), V(0\dots k_\mathrm{e}), W(0\dots k_\mathrm{e}))}{\text{Prob}(V(0\dots k_\mathrm{e}), W(0\dots k_\mathrm{e}))}\end{aligned} \tag{11.88}$$

holds, where the inequality $\text{Prob}(V(0\dots k_\mathrm{e}), W(0\dots k_\mathrm{e})) > 0$ holds because the pair (V, W) is assumed to be consistent with the stochastic automaton. Equation (11.88) can be reformulated as

$$\begin{aligned}&\text{Prob}(Z(0\dots k_\mathrm{e}) \mid V(0\dots k_\mathrm{e}), W(0\dots k_\mathrm{e})) \\ &= \frac{\sum\limits_{z_{k_\mathrm{e}+1}} \text{Prob}(z_0, \dots, z_{k_\mathrm{e}+1}, \bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}, \bar{w}_0, \dots, \bar{w}_{k_\mathrm{e}})}{\sum\limits_{Z(0\dots k_\mathrm{e}+1)} \text{Prob}(z_0, \dots, z_{k_\mathrm{e}+1}, \bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}, \bar{w}_0, \dots, \bar{w}_{k_\mathrm{e}})}.\end{aligned} \tag{11.89}$$

The probability distribution that appears in the numerator and denominator of (11.89) can be simplified as follows:

$$\text{Prob}(z_0, \dots, z_{k_\mathrm{e}+1}, \bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}, \bar{w}_0, \dots, \bar{w}_{k_\mathrm{e}}) = \tag{11.90}$$

$$\begin{aligned}&\text{Prob}(z_{k_\mathrm{e}+1}, \bar{w}_{k_\mathrm{e}} \mid z_{k_\mathrm{e}}, \bar{v}_{k_\mathrm{e}}, Z(0\dots k_\mathrm{e}-1), V(0\dots k_\mathrm{e}-1), W(0\dots k_\mathrm{e}-1)) \cdot \\ &\cdot \text{Prob}(z_{k_\mathrm{e}}, \bar{v}_{k_\mathrm{e}}, Z(0\dots k_\mathrm{e}-1), V(0\dots k_\mathrm{e}-1), W(0\dots k_\mathrm{e}-1)) =\end{aligned} \tag{11.91}$$

$$\begin{aligned}&\text{Prob}(z_{k_\mathrm{e}+1}, \bar{w}_{k_\mathrm{e}} \mid z_{k_\mathrm{e}}, \bar{v}_{k_\mathrm{e}}) \cdot \\ &\cdot \text{Prob}(z_0, \dots, z_{k_\mathrm{e}}, \bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}, \bar{w}_0, \dots, \bar{w}_{k_\mathrm{e}-1})\end{aligned} \tag{11.92}$$

$$= \dots =$$

$$\begin{aligned}&\text{Prob}(z_{k_\mathrm{e}+1}, \bar{w}_{k_\mathrm{e}} \mid z_{k_\mathrm{e}}, \bar{v}_{k_\mathrm{e}}) \cdot \\ &\cdot \text{Prob}(z_{k_\mathrm{e}}, \bar{w}_{k_\mathrm{e}-1} \mid z_{k_\mathrm{e}-1}, \bar{v}_{k_\mathrm{e}-1}) \cdot \\ &\cdot \dots \cdot \text{Prob}(z_1, \bar{w}_0 \mid z_0, \bar{v}_0) \cdot \text{Prob}(z_0, \bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}) =\end{aligned} \tag{11.93, 11.94}$$

$$\prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0) \cdot \text{Prob}(\bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}). \tag{11.95}$$

To obtain Eq. (11.91), Bayes' formula was used. The first probability in (11.91) can be reformulated as (11.92) due to the Markov property (11.27). The second probability distribution in (11.91) and (11.92) has a similar form as (11.90). Only the state z and output w with the highest time index have disappeared in the list of arguments. Hence, the same simplification steps can be carried out several times until Eq. (11.94) is obtained. As z_0 is independent of $\bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}$ and

$$\text{Prob}(Z(k+1) = z(k+1), W(k) = \bar{w}_k \mid Z(k) = z_k, V(k) = \bar{v}_k) = L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k)$$

represents the behavioural relation of the stochastic automaton, Eq. (11.95) is obtained. Finally, (11.87) results from inserting (11.95) into (11.89) and simplifying the resulting expression. □

From Lemma 11.6, the solution to the observation problem is obtained by determining the conditional probability distributions

$$\begin{aligned} p(z, k_\mathrm{e}) = \mathrm{Prob}(Z(k_\mathrm{e}) &= z | V(0) = v_0, V(1) = v_1, \ldots, V(k_\mathrm{e}) = v_{k_\mathrm{e}}, \\ &\quad W(0) = w_0, W(1) = w_1, \ldots, W(k_\mathrm{e}) = w_{k_\mathrm{e}}) \\ p'(z', k_\mathrm{e}) = \mathrm{Prob}(Z(k_\mathrm{e}+1) &= z' | V(0) = v_0, V(1) = v_1, \ldots, V(k_\mathrm{e}) = v_{k_\mathrm{e}}, \\ &\quad W(0) = w_0, W(1) = w_1, \ldots, W(k_\mathrm{e}) = w_{k_\mathrm{e}}). \end{aligned}$$

They replace the binary indicators $p(z, k_\mathrm{e})$ and $p'(z', k_\mathrm{e})$ defined in Eq. (11.76), which have been used to show whether or not the state $z, z' \in \mathcal{Z}$ can be assumed by the automaton $\mathcal{N}$ at time k_e or $k_\mathrm{e}+1$ if the automaton received the input sequence $V(0 \ldots k_\mathrm{e})$ and generated the output sequence $W(0 \ldots k_\mathrm{e})$.

The results are summarised in the following theorem, which is a direct consequence of Lemma 11.6:

Theorem 11.4 (Solution to the observation problem) *Consider a stochastic automaton with the initial state probability distribution $p_0(z)$. If the I/O pair (V, W) is consistent with the automaton, the current state probability distribution is given by*

$$p(z_{k_\mathrm{e}}, k_\mathrm{e}) = \frac{\sum\limits_{Z(0\ldots k_\mathrm{e}-1)} \sum\limits_{z_{k_\mathrm{e}+1}} \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0)}{\sum\limits_{Z(0\ldots k_\mathrm{e}+1)} \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0)} \tag{11.96}$$

and the set of current automaton states by

$$\mathcal{Z}(k_\mathrm{e} \mid k_\mathrm{e}) = \{z_{k_\mathrm{e}} \,:\, p(z_{k_\mathrm{e}}, k_\mathrm{e}) > 0\}. \tag{11.97}$$

Note that for time $k_\mathrm{e} = 0$ Eq. (11.96) yields the a-posteriori probability distribution $p(z, 0), (z \in \mathcal{Z})$ that the automaton is in the initial state z and has generated the output

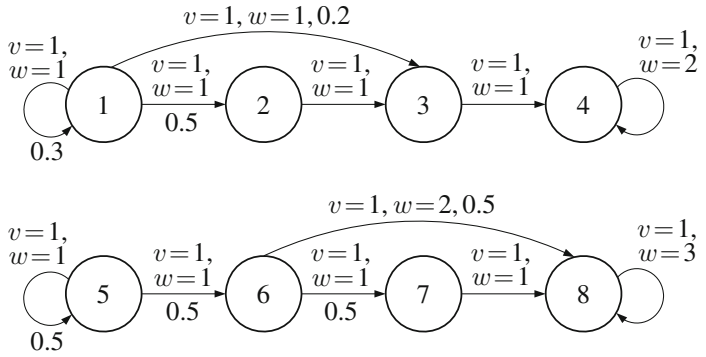


Fig. 11.26 Automaton graph of the example

$\bar{w}_0$ for the input $\bar{v}_0$. This probability is, in general, different from $p_0(z)$, which represents the a-priori probability distribution of the initial state. Hence, the set

$$\mathcal{Z}(0 \mid 0) = \{z \in \mathcal{Z} \,:\, p(z, 0) > 0\}$$

is different from the a-priori information about the initial state represented by

$$\mathcal{Z}(0 \mid -1) = \{z \in \mathcal{Z} \,:\, p_0(z) > 0\} \tag{11.98}$$

and satisfies the relation

$$\mathcal{Z}(0 \mid 0) \subseteq \mathcal{Z}(0 \mid -1).$$

Example 11.5 State observation of a stochastic automaton
Consider the stochastic automaton whose automaton graph is shown in Fig. 11.26 and whose initial state is uniformly distributed:

$$p_0(z) = \frac{1}{8} \quad z = 1, \ldots, 8.$$

The value of the behavioural relation L is indicated at the corresponding edges of the automaton graph unless its value is 1 for deterministic state transitions. The automaton graph

Table 11.1 Probability distribution Prob($Z \mid V, W$) of the example automaton

$k_e = 0$ $V(0...0) = (1)$ $W(0...0) = (1)$		$k_e = 1$ $V(0...1) = (1,1)$ $W(0...1) = (1,1)$		$k_e = 2$ $V(0...2) = (1,1,1)$ $W(0...2) = (1,1,1)$	
$Z(0..0)$	Prob($Z\|V, W$)	$Z(0..1)$	Prob($Z\|V, W$)	$Z(0..2)$	Prob($Z\|V, W$)
(1)	0.1818	(1,1)	0.0923	(1,1,1)	0.0632
(2)	0.1818	(1,2)	0.1538	(1,1,2)	0.1053
(3)	0.1818	(1,3)	0.0615	(1,1,3)	0.0421
(5)	0.1818	(2,3)	0.3077	(1,2,3)	0.3509
(6)	0.0909	(5,5)	0.1538	(5,5,5)	0.1754
(7)	0.1818	(5,6)	0.0769	(5,5,6)	0.0877
		(6,7)	0.1538	(5,6,7)	0.1754
#1	#2	#3	#4	#5	#6

$k_e = 3$ $V(0...3) = (1,1,1,1)$ $W(0...3) = (1,1,1,3)$		$k_e = 4$ $V(0...4) = (1,1,1,1,1)$ $W(0...4) = (1,1,1,3,3)$	
$Z(0...3)$	Prob($Z \mid V, W$)	$Z(0...4)$	Prob($Z \mid V, W$)
(5,6,7,8)	1	(5,6,7,8,8)	1
#7	#8	#9	#10

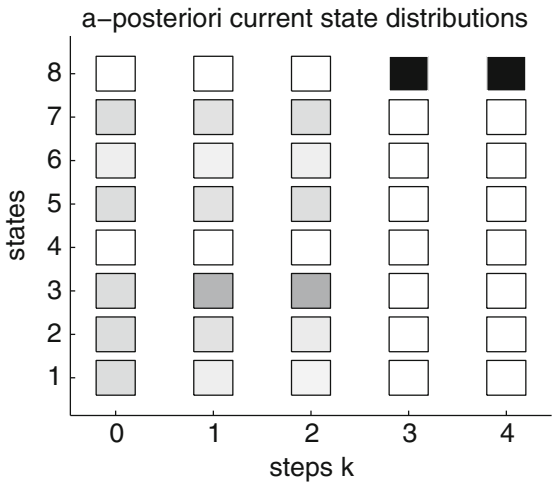


Fig. 11.27 Observation result

consists of two separate parts and it is interesting to see how the probability $p(z, k_e)$ distributes over these two parts for increasing k_e.

All state sequences that occur with non-vanishing probability for the input sequence

$$V(0\ldots4) = (1, 1, 1, 1, 1)$$

and yield the output sequence

$$W(0\ldots4) = (1, 1, 1, 3, 3)$$

are shown in Table 11.1 together with the probabilities

$$\begin{aligned} &\text{Prob}(z_0 \mid \bar{v}_0, \bar{w}_0), \quad \text{Prob}\,(Z(0\ldots1) \mid V(0\ldots1), W(0\ldots1)), \\ &\ldots, \quad \text{Prob}(Z(0\ldots4) \mid V(0\ldots4), W(0\ldots4)), \end{aligned}$$

which are obtained by means of Eq. (11.87). Note that the value of

$$\text{Prob}(Z(0\ldots0) \mid V(0\ldots0), W(0\ldots0)) = \text{Prob}(z_0 \mid \bar{v}_0, \bar{w}_0)$$

which is shown in column #2 differs from the a-priori probability distribution $p_0(z) = \frac{1}{8}$ because this a-posteriori probability includes the information provided by the I/O pair $(\bar{v}_0, \bar{w}_0)$. This is the reason why the states $Z(0) = 4$ and $Z(0) = 8$, both of which are assumed by the automaton with the a-priori probability $p_0(z) = \frac{1}{8}$ do not appear in column #1.

The state probability distribution obtained by means of Eq. (11.96) is shown in Fig. 11.27. The probabilities are depicted in greyscale. Black rectangles symbolise a probability of one, white rectangles a probability of zero. The set $\mathcal{Z}(k_e \mid k_e)$ includes all states z_{k_e} with non-zero probability (grey and black boxes):

$$\begin{aligned}\mathcal{Z}(0\,|\,0) &= \{1,2,3,5,6,7\}\\ \mathcal{Z}(1\,|\,1) &= \{1,2,3,5,6,7\}\\ &\vdots\\ \mathcal{Z}(4\,|\,4) &= \{8\}. \quad \square\end{aligned}$$

Recursive form of the solution. For the application, the elements of the sequences $V(0\ldots k_\mathrm{e})$ and $W(0\ldots k_\mathrm{e})$ appear one after the other for $k_\mathrm{e} = 0, 1, 2, \ldots$ and should be processed in this way. Therefore, the following recursive form of the solution to the state observation problem is important (for a proof cf. [218]). In the representation given, the indicator $p(z, k_\mathrm{e})$ denotes the probability

$$p(z, k_\mathrm{e}) = \mathrm{Prob}(Z(k_\mathrm{e}) = z \,|\, V(0\ldots k_\mathrm{e}),\, W(0\ldots k_\mathrm{e}))$$

that z is the state at time k_e and $p'(z', k_\mathrm{e})$ the probability

$$p'(z', k_\mathrm{e}) = \mathrm{Prob}(Z(k_\mathrm{e}+1) = z' \,|\, V(0\ldots k_\mathrm{e}),\, W(0\ldots k_\mathrm{e}))$$

for z' to be the next state.

Theorem 11.5 (Recursive solution to the state observation problem) *Consider a stochastic automaton with the initial state distribution $p_0(z)$. If the I/O pair (V, W) is consistent with the stochastic automaton, the a-posteriori state probability distribution is given by the recursive relations*

$$p(z, k_\mathrm{e}) = \frac{\sum\limits_{z'} L(z', \bar{w}_{k_\mathrm{e}} \,|\, z, \bar{v}_{k_\mathrm{e}}) \cdot p'(z', k_\mathrm{e}-1)}{\sum\limits_{z',\tilde{z}} L(z', \bar{w}_{k_\mathrm{e}} \,|\, \tilde{z}, \bar{v}_{k_\mathrm{e}}) \cdot p'(z', k_\mathrm{e}-1)}, \quad (k_\mathrm{e} \geq 0) \qquad (11.99)$$

with

$$p'(z', k_\mathrm{e}) = \frac{\sum\limits_{z} L(z', \bar{w}_{k_\mathrm{e}} \,|\, z, \bar{v}_{k_\mathrm{e}}) \cdot p'(z, k_\mathrm{e}-1)}{\sum\limits_{z,\tilde{z}} L(\tilde{z}, \bar{w}_{k_\mathrm{e}} \,|\, z, \bar{v}_{k_\mathrm{e}}) \cdot p'(z, k_\mathrm{e}-1)}, \quad (k_\mathrm{e} > 0) \qquad (11.100)$$

$$p'(z', -1) = p_0(z'), \qquad (k_\mathrm{e} = 0). \quad (11.101)$$

Equation (11.100) is used after the pair $(V(k_\mathrm{e}) = \bar{v}_{k_\mathrm{e}}, W(k_\mathrm{e}) = \bar{w}_{k_\mathrm{e}})$ has been measured. It describes the probability distribution of the future state $Z(k_\mathrm{e}+1) = z'$ by using the information about the movement of the stochastic automaton until time k_e that is included in the I/O pair $(V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e}))$. It is a recursive relation with the initialisation given by Eq. (11.101). Equation (11.100) makes it possible to determine $p'(z', k_\mathrm{e})$, $(z' \in \mathcal{Z})$ for given $p'(z, k_\mathrm{e}-1)$, $(z \in \mathcal{Z})$ and the new measurements $(\bar{v}_{k_\mathrm{e}}, \bar{w}_{k_\mathrm{e}})$. Hence, only the probability distribution $p'(z, k_\mathrm{e}-1)$, $(z \in \mathcal{Z})$ has to be stored in the computer memory, which consists of N values.

Equation (11.99) describes how the prediction from the previous time point has to be corrected after the new measurements $\bar{v}_{k_e}$ and $\bar{w}_{k_e}$ became available. This step can be interpreted as a projection onto the set of those states $Z(k_e) = z$ from which the automaton can have moved when generating the new measurement information $W(k_e) = \bar{w}_{k_e}$. The result of the recursion is the a-posteriori probability distribution $p(z, k_e)$ over all current states $Z(k_e) = z \in \mathcal{Z}$ for the given measurements until time k_e. With this structure, the recursive solution to the state observation problem shows a remarkable similarity to the Kalman filter, which likewise can be decomposed into a prediction and a projection step.

A-priori knowledge about the initial state. In the solution to the state observation problem, the initial state probability distribution $p_0(z)$ is assumed to be known. Since in applications this a-priori knowledge is often not available, $p_0(z)$ is measured or "guessed". The question arises what happens if the a-priori knowledge about z_0 is in conflict with the actual initial state of the stochastic process.

To answer this question, assume that $\hat{p}_0(z)$ denotes the approximate initial state probability distribution and consider the sets

$$\mathcal{Z}_0 = \{z : p_0(z) > 0\} \subseteq \mathcal{Z} \tag{11.102}$$

$$\hat{\mathcal{Z}}_0 = \{z : \hat{p}_0(z) > 0\} \subseteq \mathcal{Z}. \tag{11.103}$$

The stochastic process starts from an initial state $z_0 \in \mathcal{Z}_0$, whereas the observation algorithm assumes that the automaton starts from some state $z_0 \in \hat{\mathcal{Z}}_0$. The a-priori knowledge about the initial state is not in conflict with the real system, if the relation

$$\hat{\mathcal{Z}}_0 \supseteq \mathcal{Z}_0 \tag{11.104}$$

holds true. Then, the solution to the observation problem ensures that the relation

$$Z(k_e) \in \mathcal{Z}(k_e \mid k_e)$$

is valid for all $k_e \geq 0$, i.e. the set of current states determined by the observation algorithm includes the true state of the stochastic process. If the probability distribution $\hat{p}_0(z)$ used in the observation algorithm is different from the real distribution $p_0(z)$, the probability distribution $p(z, k_e)$ obtained by the algorithm is wrong, but it is accepted in practice as solution to the observation problem due to the lack of a better a-priori knowledge.

If, however, condition (11.104) is violated, then it is possible that the probability $\text{Prob}(W(0 \ldots k_e) \mid V(0 \ldots k_e), z_0)$ is zero for all initial states $z_0 \in \hat{\mathcal{Z}}_0$. Consequently, like in the case of an inconsistent I/O pair, the violation of the condition (11.104) makes the denominators in Eqs. (11.99) and (11.100) vanish, which can be used as an indicator to stop the observation algorithm.

This and the preceding remark show that the observation algorithm cannot distinguish between an inconsistent I/O pair and a wrong initial state probability distribution. As a consequence, in an application the set $\hat{\mathcal{Z}}_0$ has to be chosen "large

enough". A secure way is to choose $\hat{p}_0(z)$ so that $\hat{\mathcal{Z}}_0 = \mathcal{Z}$ holds, for example, using the uniform initial state distribution

$$\hat{p}(z) = \text{Prob}(Z(0) = z) = \frac{1}{N} \quad \text{for all} \quad z \in \mathcal{Z}. \tag{11.105}$$

Besides the lack of knowledge of $p_0(z)$ another fact makes it reasonable to use the a-priori probability distribution (11.105). For many stochastic automata, the solution $p(z, k_e)$ of the observation problem is (nearly) independent of $p_0(z)$ for $k_e \geq \bar{k}$ with very small $\bar{k}$. This is particularly true if the set $Z(k_e \mid k_e)$ has only a few elements compared to the cardinality N of the state set $\mathcal{Z}$.

11.6.2 Observation Algorithm

To show how the observation method developed in this section can be applied online, this section presents an observation algorithm, which is based on the recursive solution given in Theorem 11.5. The following symbols are used in the algorithm:

$$\begin{aligned} h(z) &= \sum_{z'} L(z', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}) \cdot p'(z', k_e - 1) \\ p(z) &= \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e)) \\ p'(z') &= \text{Prob}(Z(k_e + 1) = z' \mid V(0 \ldots k_e), W(0 \ldots k_e)) \end{aligned}$$

Algorithm 11.6 *Observation of stochastic automata*

Given: Stochastic automaton $\mathcal{S}$

Initial state probability distribution $p_0(z)$.

Initialisation: $p'(z) = p_0(z)$ for all $z \in \mathcal{Z}$

$k_e = 0$.

Loop:

1. Measure the current input $\bar{v}$ and output $\bar{w}$.
2. Determine $h(z) = \sum_{z'} L(z', \bar{w} \mid z, \bar{v}) \cdot p'(z')$ for all $z \in \mathcal{Z}$
3. If $\sum_z h(z) = 0$ holds, stop the algorithm (inconsistent I/O pair or wrong initial state distribution).
4. Determine $p(z) := \dfrac{h(z)}{\sum_z h(z)}$ for all $z \in \mathcal{Z}$.

5. Determine $p'(z') := \dfrac{\sum_z L(z', \bar{w} \mid z, \bar{v})\, p'(z)}{\sum_z h(z)}$ for all $z \in \mathcal{Z}$.

6. Determine $\mathcal{Z}(k_e \mid k_e)$ according to Eq. (11.97).

7. $k_e := k_e + 1$
   Continue with Step 1.

Result: $p(z) = \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e))$ and $\mathcal{Z}(k_e \mid k_e)$ for increasing time horizon k_e.

In Step 3, the consistency of the I/O pair and of the a-priori probability distribution is tested. Steps 4 and 5 use Eq. (11.99) or (11.100), respectively. The test in Step 3 ensures that the denominators of both equations do not vanish.

Note that in each step of the algorithm very easy calculations have to be carried out, which makes the algorithm applicable under relatively strong real-time constraints. Since the algorithm is based on the recursive solution to the state observation problem, its complexity does *not* increase with the length of the measurement sequences V and W. Only the N values of the functions $h(z)$ and $p'(z)$ have to be stored in the memory.

11.6.3 Observability of Stochastic Automata

In this section, a notion of observability is introduced, which takes into account that for stochastic automata the state generally cannot be determined unambiguously. Even if the initial state is known, the automaton may produce an I/O pair that does not allow to track the state trajectory with certainty.

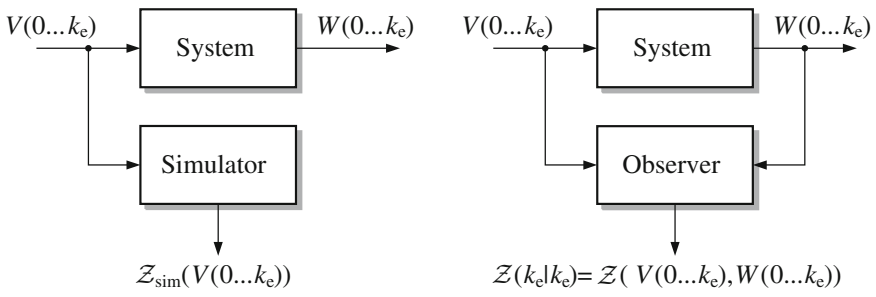


Fig. 11.28 Comparison of simulation and observation

The observability definition is based on a comparison of the results that are obtained by means of simulation and of observation (Fig. 11.28). Roughly speaking, the stochastic automaton is called observable if it is possible to determine the state more precisely by state observation than by simulation. Before defining the observability in this way, simulation and observation have to be compared in more detail.

For both simulation and observation, the initial state distribution $p_0(z)$ and the input sequence V have to be known.

- In **simulation**, the initial state probability distribution is propagated according to the state transition relation G of the stochastic automaton and yields the state probability distribution $\text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e - 1))$, $(z \in \mathcal{Z})$ of the state at the end of the time interval considered. Accordingly, the set of states in which the stochastic automaton recides at time k_e with non-vanishing probability for the given the input sequence $V(0 \ldots k_e)$ is

$$\mathcal{Z}_{\text{sim}}(V(0 \ldots k_e)) = \{z : \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e)) > 0\}. \tag{11.106}$$

- In **state observation** described by Theorem 11.5 the additional information included in the output sequence W is used to determine the probability

$$\text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e)), \quad z \in \mathcal{Z}$$

  according to Eq. (11.96). The set of states $\mathcal{Z}(k_e \mid k_e)$ is obtained by Eq. (11.97).

As the state observation uses the generated output sequence as additional constraint when determining the set of possible states $Z(k_e)$, the relation

$$\mathcal{Z}_{\text{sim}}(V(0 \ldots k_e)) \supseteq \mathcal{Z}(k_e \mid k_e)$$

holds.

Stochastic unobservability. An automaton is called *unobservable* if for all input sequences V the I/O pair (V, W) does not include more information than the input V alone. This definition can be stated more formally as follows:

Definition 11.4 (*Stochastic unobservability*) A stochastic automaton is called stochastically unobservable if the behavioural relation L can be represented as the product of two functions

$$G : \mathcal{Z} \times \mathcal{Z} \times \mathcal{V} \to [0, 1]$$
$$H : \mathcal{W} \times \mathcal{V} \to [0, 1]$$

such that

$$L(z', w \mid z, v) = G(z' \mid z, v) \cdot H(w \mid v) \tag{11.107}$$

holds for all $z', z \in \mathcal{Z}$, $v \in \mathcal{V}$ and $w \in \mathcal{W}$.

As before, the functions $G(z', z, v)$ and $H(w, v)$ are denoted by $G(z' \mid z, v)$ or $H(w \mid v)$, respectively, because they turn out to be conditional probabilities.

Definition 11.4 has an obvious interpretation. Equation (11.107) says that the output w does not depend on z and, hence, does not provide any information about the current automaton state. Furthermore Eq. (11.107) implies that w does not depend on z' and, hence, the output does not provide any information about the successor state either.

Observability test. Before the consequences of Eq. (11.107) will be discussed it should be mentioned how this definition can be used to test whether a stochastic automaton is stochastically unobservable. It is easy to see that the functions G and H used in Eq. (11.107) are the conditional probabilities defined by Eqs. (11.19) and (11.20). From this fact, it is clear that the decomposition of L in the form (11.107) is found (if it exists) by determining G and H according to Eqs. (11.19) and (11.20) and by testing whether Eq. (11.107) holds. Note that the function H obtained from Eq. (11.20) is not allowed to depend upon z, but $H(w \mid z_i, v) = H(w \mid z_j, v)$ has to hold for all $z_i, z_j \in \mathcal{Z}$, $v \in \mathcal{V}$ and $w \in \mathcal{W}$.

Property of unobservable automata. The following lemma says that the definition of unobservability satisfies the aim to call an automaton observable if the current state can be determined more precisely by state observation than by simulation.

Lemma 11.7 [306] *If the stochastic automaton is stochastically unobservable, then for all consistent I/O pairs (V, W) the results of the state observation problem and of the simulation are identical for all $k_e = 0, 1, 2, \ldots$:*

$$\begin{aligned} \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e)) &= \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e)) \\ \mathcal{Z}(k_e \mid k_e) &= \mathcal{Z}_{\text{sim}}(V(0 \ldots k_e)). \end{aligned}$$

Hence, the output sequence W does not include any information about the state trajectory of the automaton that cannot be obtained from the initial state probability $p_0(z)$ and the input sequence V. In this case, the observers described by Theorems 11.4 and 11.5 work as simulators.

Stochastic unobservability of a state set. The stochastic automaton may not satisfy the condition (11.107) for all $z \in \mathcal{Z}$, but only for a subset $\mathcal{G}_z \subseteq \mathcal{Z}$, where $\mathcal{G}_z$ should have at least two elements ($|\mathcal{G}_z| \geq 2$). Then a stochastic automaton is called *stochastically unobservable within a set $\mathcal{G}_z$* if the relation

$$L_{\mathcal{G}_z}(z', w \mid z, v) = G_{\mathcal{G}_z}(z' \mid z, v) \cdot H_{\mathcal{G}_z}(w \mid v) \tag{11.108}$$

holds for all $z, z' \in \mathcal{G}_z$, $v \in \mathcal{V}$, and $w \in \mathcal{W}$, with

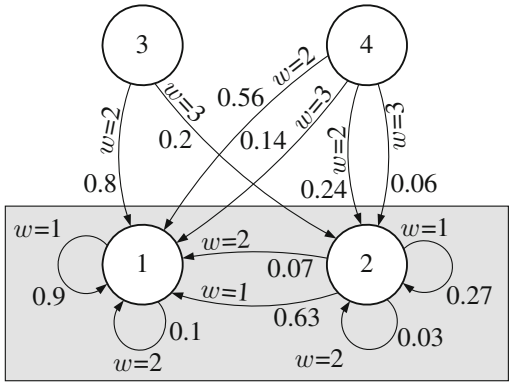


Fig. 11.29 Stochastic automaton with stochastically unobservable set {1, 2}

$$L_{\mathcal{G}_z}(z', w \mid z, v) = \frac{L(z', w \mid z, v)}{\sum_{z' \in \mathcal{G}_z} \sum_{w} L(z', w \mid z, v)} \tag{11.109}$$

$$G_{\mathcal{G}_z}(z' \mid z, v) = \sum_{w} L_{\mathcal{G}_z}(z', w \mid z, v) \tag{11.110}$$

$$H_{\mathcal{G}_z}(w \mid v) = \sum_{z' \in \mathcal{G}_z} L_{\mathcal{G}_z}(z', w \mid z, v). \tag{11.111}$$

In this definition, the functions L, G and H appearing in Definition 11.4 are replaced by $L_{\mathcal{G}_z}$, $G_{\mathcal{G}_z}$ or $H_{\mathcal{G}_z}$, respectively. These functions are normalised as shown in Eqs. (11.109)–(11.111) so as to define conditional probability distributions. For these new functions, Eq. (11.108) is, in principle, the same as Eq. (11.107), but it has only to be satisfied for the states z, z' that belong to the set $\mathcal{G}_z$. Lemma 11.7 holds as long as the automaton remains within the set $\mathcal{G}_z$:

$$z(k) \in \mathcal{G}_z \quad \text{for } k = 0, 1, \ldots, k_e.$$

If the automaton is stochastically unobservable within the set $\mathcal{G}_z$ then the state observer acts as a simulator as long as the state remains in the set $\mathcal{G}_z$.

Example 11.6 Observability of stochastic automata
Consider the stochastic automaton shown in Fig. 11.29, which has the only input $v = 1$. The automaton is not stochastically unobservable according to Definition 11.4. However, the state set $\mathcal{G}_z = \{1, 2\}$ is stochastically unobservable. This can be verified by means of Eq. (11.107) as shown in Table 11.2. Note that $G_{\mathcal{G}_z} \cdot H_{\mathcal{G}_z}$ is identical to $L_{\mathcal{G}_z}$ for all z, z', w and v. □

Stochastic observability. The stochastic observability can now be defined as the property of an automaton not to possess sets of unobservable states:

Definition 11.5 (*Stochastic observability*) A stochastic automaton is called stochastically observable if it does not possess any set $\mathcal{G}_z$ of stochastically unobservable states.

Table 11.2 Test for stochastic unobservability of the state set $\mathcal{G}_z = \{1, 2\}$

		$L_{\mathcal{G}_z}$		$G_{\mathcal{G}_z}$		$H_{\mathcal{G}_z}$		$G_{\mathcal{G}_z} \cdot H_{\mathcal{G}_z}$	
		$z=1$	$z=2$	$z=1$	$z=2$	$z=1$	$z=2$	$z=1$	$z=2$
$z'=1$	$w=1$	0.9	0.63	1.0	0.7	0.9	0.9	0.9	0.63
	$w=2$	0.1	0.07			0.1	0.1	0.1	0.07
	$w=3$	0	0			0	0	0	0
$z'=2$	$w=1$	0	0.27	0	0.3	0.9	0.9	0	0.27
	$w=2$	0	0.03			0.1	0.1	0	0.03
	$w=3$	0	0			0	0	0	0

Consequently, the stochastic automaton is called observable if it is possible to determine every state more precisely by state observation than by simulation. This fact is represented by the following corollary, which follows directly from Lemma 11.7 and Definition 11.5.

Corollary 11.2 *If the stochastic automaton is stochastically observable, the following relation holds:*

$$\mathcal{Z}(k_\mathrm{e} \mid k_\mathrm{e}) \subseteq \mathcal{Z}_\mathrm{sim}(V(0 \ldots k_\mathrm{e})), \quad (k_\mathrm{e} \geq 0). \tag{11.112}$$

Note that the equality sign may hold for some input sequence even if the automaton is observable. The reason for this is given in Sect. 11.6.4.

Remark 11.1 (*Observability test*) The following remarks concern the test of a given stochastic automaton concerning observability. The definition of unobservable state sets implies that if the set $\mathcal{G}_z$ is stochastically unobservable then any subset $\tilde{\mathcal{G}}_z \subset \mathcal{G}_z$ is stochastically unobservable as well, because Eq. (11.108) holds not only for all $z, z' \in \mathcal{G}_z$ but also for all $z, z' \in \tilde{\mathcal{G}}_z$ and the normalisation carried out in Eqs. (11.109)–(11.111) does not influence this result. Hence, the test of a stochastic automaton starts with testing all pairs $z, \bar{z}$ whether or not they are unobservable sets. If no such pair is found, the stochastic automaton does not possess any unobservable set and is, therefore, stochastically observable. If such pairs exist, larger unobservable state sets can be obtained (if they exist) from combinations of such unobservable pairs according to the following corollary. □

Corollary 11.3 *A stochastic automaton is stochastically unobservable within a set $\mathcal{G}_z$ of at least three states ($|\mathcal{G}_z| \geq 3$) if and only if the stochastic automaton is stochastically unobservable within all subsets $\mathcal{G}_z^i \subset \mathcal{G}_z$ of two states ($|\mathcal{G}_z^i| = 2$).*

Therefore, the search for stochastically unobservable sets of states can be reduced to the test of all pairs of states. The stochastic automaton is stochastically observable if no unobservable pair of states is found.

11.6.4 Distinguishing Inputs

In this section, it is investigated under what conditions the observer improves its result by processing the k_eth I/O pair $(\bar{v}_{k_e}, \bar{w}_{k_e})$ in comparison with the result obtained for the I/O pair of length $k_e - 1$. This analysis uses the recursive formulation of the observer given in Theorem 11.5 and compares the observation result with the simulation results obtained by the recursion (11.26). It will be shown that even if the automaton is observable there exist input values v for which the next recursion step of the observer yields the same result as the next recursion step of the simulator. Hence, the movement of the automaton under this input cannot contribute to an improvement of the knowledge about the current state.

To start the analysis, it is assumed that both simulation and observation have obtained the same state probability distribution at time k_e

$$\begin{aligned}&\text{Prob}(Z(k_e)=z \mid V(0\ldots k_e-1), W(0\ldots k_e-1)) = \\ &\quad = \text{Prob}(Z(k_e)=z \mid V(0\ldots k_e-1)) \\ &\quad = p'(z \mid k_e-1),\end{aligned}$$

where both observation and simulation have used the information available until time $k_e - 1$. Consequently, both methods yield the same state set at time k_e:

$$\begin{aligned}\mathcal{Z}(k_e-1 \mid k_e-1)) &= \mathcal{Z}_{\text{sim}}(V(0\ldots k_e-1)) \\ &= \{z : p'(z, k_e-1) > 0\}.\end{aligned}$$

The set of successor states that the automaton can reach at time $k_e + 1$ for the input $\bar{v}_{k_e} = \bar{v}$ is given as follows:

$$\mathcal{Z}_{\text{sim}}(V(0\ldots k_e)) = \{z \mid G(z \mid \tilde{z}, \bar{v}) > 0 \text{ for some } \tilde{z} \in \mathcal{Z}_{\text{sim}}(V(0\ldots k_e-1))\}.$$

In the following, the question should be answered under what condition the k_eth observation step yields a set $\mathcal{Z}(k_e \mid k_e)$ which is a proper subset of $\mathcal{Z}_{\text{sim}}(V(0\ldots k_e))$. The answer can be obtained from the considerations made in Sect. 11.6.3. Lemma 11.7 has shown that state observation yields the same result as simulation if the automaton is unobservable; and hence, the behavioural relation can be decomposed according to Eq. (11.107). This result is applied here for the fixed input $\bar{v}$. If the decomposition (11.107) is possible for the input $\bar{v}_{k_e} = \bar{v}$, simulation and observation lead to the same set of states z_{k_e+1}. This result is summarised in the following corollary. It shows that the equality sign can hold in Eq. (11.112).

Corollary 11.4 *Assume that the state probability distribution $p'(z, k_e-1)$ is known and the stochastic automaton generates the output $\bar{w}_{k_e} = \bar{w}$ for the input $\bar{v}_{k_e} = \bar{v}$. Then simulation and state observation yield the same state probability distribution*

$$\begin{aligned} p'(z, k_\mathrm{e}) &= \mathrm{Prob}(Z(k_\mathrm{e}+1)=z \mid V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e})) \\ &= \mathrm{Prob}(Z(k_\mathrm{e}+1)=z \mid V(0\ldots k_\mathrm{e})) \end{aligned}$$

for all $z \in \mathcal{Z}$ if and only if the relation

$$L(z', \bar{w} \mid z, \bar{v}) = G(z' \mid z, \bar{v}) \cdot H(\bar{w} \mid \bar{v}) \tag{11.113}$$

holds with some constant $H(\bar{w} \mid \bar{v})$ for all $z \in \mathcal{Z}(k_\mathrm{e}-1 \mid k_\mathrm{e}-1)$ and $z' \in \mathcal{Z}_\mathrm{sim}(V(0\ldots k_\mathrm{e}))$.

The condition (11.113) can be tested in each recursion step of the observation algorithm in order to indicate whether the observer works really as an observer or merely as a simulator. If the condition is satisfied, the observer yields the same state probability distribution $\mathrm{Prob}(Z(k_\mathrm{e}+1)=z \mid V(0\ldots k_\mathrm{e}))$ as a simulation step described by Eq. (11.26) (with k_e replacing $k_\mathrm{e}-1$).

This result has interesting consequences concerning the choice of the input. Even if the stochastic automaton is observable according to Definition 11.5, not all individual I/O pairs $(\bar{v}, \bar{w})$ lead to an improvement of the observation result compared to the corresponding simulation step. The reason for this is given by the fact that the observability definition claims that the decomposition of the behavioural relation L according to Eq. (11.107) is impossible for all states $z, z' \in \mathcal{Z}$, all input values $v \in \mathcal{V}$ and all output values $w \in \mathcal{W}$. However, such a decomposition may be possible for the set $\tilde{\mathcal{V}} \subset \mathcal{V}$ of input values even if it is impossible for all $v \in \mathcal{V}$. If the stochastic automaton gets, for some reason, only input values from $\tilde{\mathcal{V}}$ the solution of the observation problem is not better than the simulation result (cf. Lemma 11.7), although the stochastic automaton is observable. Therefore, it is important to know, which input should be used in order to get an improved observation result. These input values are called *distinguishing*. For such input values $\bar{v}$ the decomposition (11.107) is impossible for $v = \bar{v}$ and all z', z and w.

As a consequence, for every given state set $\mathcal{G}_z$ it is possible to partition the input set $\mathcal{V}$ into two sets $\tilde{\mathcal{V}}(\mathcal{G}_z)$ and $\bar{\mathcal{V}}(\mathcal{G}_z)$ such that the decomposition of L is possible for all $v \in \tilde{\mathcal{V}}(\mathcal{G}_z)$:

$$\begin{aligned} L_{\mathcal{G}_z}(z', w \mid z, v) &= G_{\mathcal{G}_z}(z' \mid z, v) \cdot H_{\mathcal{G}_z}(w \mid v) \\ &\qquad \text{for all } \; z', z \in \mathcal{G}_z, \; v \in \tilde{\mathcal{V}}(z), \; w \in \mathcal{W}, \end{aligned} \tag{11.114}$$

whereas such a decomposition of L is impossible for all $v \in \bar{\mathcal{V}}(\mathcal{G}_z)$. From Lemma 11.7, it is obvious that the state observer cannot improve its result at time k_e for states $z_{k_\mathrm{e}}, z(k_\mathrm{e}+1) \in \mathcal{G}_z$ if the stochastic automaton obtains the input $\bar{v}_{k_\mathrm{e}} \in \tilde{\mathcal{V}}(\mathcal{G}_z)$. Hence, $\tilde{\mathcal{V}}(\mathcal{G}_z)$ is the set of distinguishing inputs. Only if an input $v \in \bar{\mathcal{V}}(\mathcal{G}_z)$ is applied, the state observer gets enough information for determining the state probability distribution better than it is determined by simulation.

In an application where the state of the system should be found as quickly as possible, the input has to be chosen at every time instant from the current set of distinguishing inputs. This set has to be determined at time k_e as follows. Select

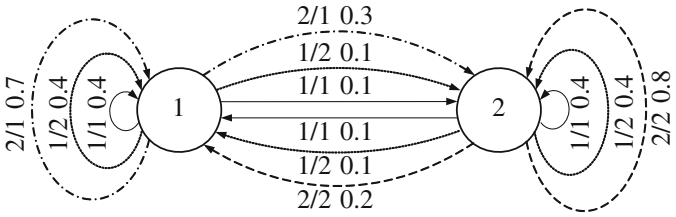


Fig. 11.30 Automaton graph of the example

the set $\mathcal{G}_z$ to include all possible current states $z_{k_e} \in \mathcal{Z}(k_e|k_e - 1)$ and all possible successor states z_{k_e+1}, for which $L(z_{k_e+1}, w|z_{k_e}, v)$ holds for some v and w. Then $\bar{\mathcal{V}}(\mathcal{G}_z)$, which is obtained as described above, is the set of distinguishing inputs form which the current input $V(k_e)$ should be selected. Note that the set $\bar{\mathcal{V}}(\mathcal{G}_z)$ changes from one time step to the next because the set of current state changes. It depends on the practical circumstances whether the input can really be chosen with the aim of state observation or whether the selection of the input has to satisfy other control aims. In any case, the set of distinguishing inputs is the set of preferred input signals as far as state observation is concerned.

If the stochastic automaton is stochastically observable, for every state z there exists at least one distinguishing input v. If, in particular, the decomposition (11.107) is impossible for all $v \in \mathcal{V}$, the stochastic automaton is called *uniformly stochastically observable*. Then, every input is distinguishing and the observation result improves in every recursion step.

Example 11.7 Distinguishing input values
Figure 11.30 shows the automaton graph of a stochastic automaton with two input symbols $v \in \{1, 2\}$ and two output symbols $w \in \{1, 2\}$. Only one output symbol is distinguishing. The edges in Fig. 11.30 are labelled by the I/O pair v/w for which they occur and by the corresponding probability. It can be seen that for $\bar{v} = 1$ the relation (11.113) holds with

$$H(\bar{w}=1 \mid \bar{v}=1) = 0.5 \quad \text{and} \quad H(\bar{w}=2 \mid \bar{v}=1) = 0.5$$

and

$$\begin{aligned} G(z'=1 \mid z=1, \bar{v}=1) &= 0.8, & G(z'=1 \mid z=2, \bar{v}=1) &= 0.2 \\ G(z'=2 \mid z=1, \bar{v}=1) &= 0.2, & G(z'=2 \mid z=2, \bar{v}=1) &= 0.8. \end{aligned}$$

Figures 11.31 and 11.32 show the effect of the non-distinguishing input on the observation result. Starting with a uniform initial state distribution over the whole state set the observation result is shown in Fig. 11.32. It is identical to the simulation result as long as the input is $v = 1$ (time steps $k = 0, 1, \ldots, 7$). Because of the symmetry of $G(z' \mid z, \bar{v})$, the simulation yields uniform distributions for these times steps. Between time steps $k = 8$ and $k = 13$, the distinguishing input $v = 2$ is applied and, hence, the observation result improves immediately. Instead of a uniform probability distribution, Fig. 11.32 shows that the observation results in high probability for the state $z = 2$ at $k = 8, 9$ and for the state $z = 1$ at $k = 10 \ldots 13$.

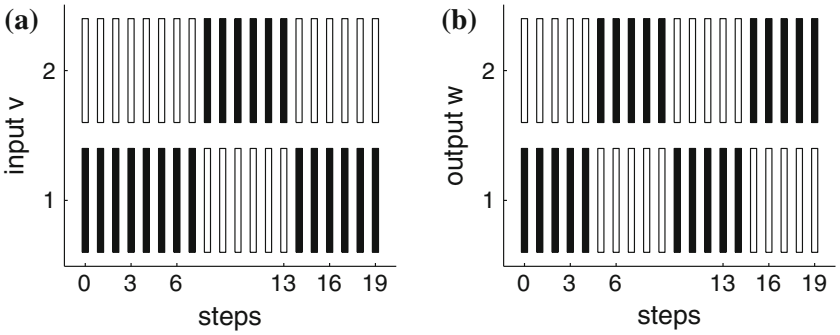
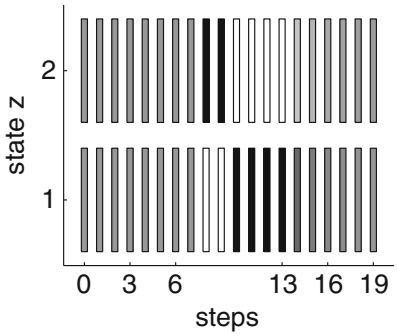


Fig. 11.31 Sequences of input (*left*) and output symbols (*right*)

Fig. 11.32 Observation result



When the input returns to the non-distinguishing input symbol $v = 1$ at time step $k = 14$ the observation falls back into a simulation, quickly loosing all state information. □

11.7 Diagnosis of Stochastic Automata

11.7.1 Principle of Consistency-Based Diagnosis Applied to Stochastic Automata

This section shows how the diagnostic problem can be solved for discrete-event systems described by a set $\{\mathcal{S}_f,\ f \in \mathcal{F}\}$ of stochastic automata. The behaviour of these automata is denoted by $\mathcal{B}_f,\ f \in \mathcal{F}$.

The idea of consistency-based diagnosis is to ask whether the measured I/O pair is consistent with the automaton $\mathcal{S}_f$. If the answer is in the affirmative

$$(V(0 \ldots k_\mathrm{e}), W(0 \ldots k_\mathrm{e})) \in \mathcal{B}_f, \tag{11.115}$$

the fault f may have occurred and is, thus, a fault candidate. In case of a negative answer

$$(V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e})) \notin \mathcal{B}_f, \tag{11.116}$$

the conclusion is that the system is not subjected to the fault f. For stochastic automata, for all fault candidates additional information can be obtained, which is expressed as the probability

$$\mathrm{Prob}(F = f \mid V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e}))$$

that the fault f has occurred if the system has generated the I/O pair $V(0\ldots k_\mathrm{e})$, $W(0\ldots k_\mathrm{e})$. Hence, diagnosing the stochastic automaton means to answer the question:

> With which probability has a fault f occurred if the system has generated the I/O pair $(V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e}))$?

The result is a set $\mathcal{F}^*(k_\mathrm{e})$ of *fault candidates*, which are those faults $f \in \mathcal{F}$ for which the I/O pair with time horizon k_e is consistent with the model $\mathcal{S}_f$:

$$\mathcal{F}^*(k_\mathrm{e}) = \{f \in \mathcal{F} : \mathrm{Prob}(F = f \mid V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e})) > 0\}.$$

11.7.2 Diagnosis of Stochastic Automata with Constant Faults

The diagnostic problem can be solved by observing the internal state for all models $\mathcal{S}_f$ and by selecting all those faults f for which the I/O pair is consistent with the model. Further, the probability of the fault for the measured I/O pair has to be determined. This idea will be explained in this section for constant faults $F = F(k)$ and extended to time-varying faults in the next section.

The idea is first to determine the common probability for the state z to occur at time k_e and the fault f to be present in the system

$$\begin{aligned}\mathrm{Prob}(Z(k_\mathrm{e}) = z, F = f \mid V(0) = \bar{v}_0, V(1) = \bar{v}_1, \ldots, V(k_\mathrm{e}) = \bar{v}_k,\\ W(0) = \bar{w}_0, W(1) = \bar{w}_1, \ldots, W(k_\mathrm{e}) = \bar{w}_k),\end{aligned}$$

where again the measured values of the input and the output are marked by a bar. This probability is abbreviated as $p_f(z, k_\mathrm{e})$. It is used to determine the marginal probability of the fault:

$$\begin{aligned}&\mathrm{Prob}(F = f \mid V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e}))\\ &= \sum_{z\in\mathcal{Z}} \mathrm{Prob}(Z(k_\mathrm{e}) = z, F = f \mid V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e})),\end{aligned}$$

which will be denoted later by $p_f(k_e)$. To extend the state observation algorithm for stochastic automata developed in Sect. 8.6 three probability distributions are necessary:

$$\begin{aligned} p_f(z, k_e) &= \text{Prob}(Z(k_e) = z, F = f \mid V(0 \ldots k_e), W(0 \ldots k_e)) \\ p'_f(z', k_e) &= \text{Prob}(Z(k_e) = z', F = f \mid V(0 \ldots k_e - 1), W(0 \ldots k_e - 1)). \end{aligned}$$

The fault probability is abbreaviated as

$$p_f(k_e) = \text{Prob}(F = f \mid V(0 \ldots k_e), W(0 \ldots k_e))$$

and obtained as marginal probability of $p_f(z, k_e)$. If $p_f(k_e) = 0$, the I/O pair with time horizon k_e is inconsistent with the stochastic automaton $\mathcal{S}_f$. Otherwise, f is a fault candidate and $p_f(k_e)$ is the probability with which this fault has occurred.

The extension of Eq. (11.99) towards faulty systems yields the diagnostic method, which is summarised in the following theorem:

Theorem 11.6 (Recursive solution to the diagnostic problem) *Consider a set* $\{\mathcal{S}_f, f \in \mathcal{F}\}$ *of stochastic automata. The a-posteriori probabilities of the fault* f *together with a state* z *can be recursively determined as follows:*

$$p_f(z, k_e) = \frac{\sum\limits_{z'} L_f(z', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}) \cdot p'_f(z, k_e - 1)}{\sum\limits_{z, z', f} L_f(z', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}) \cdot p'_f(z, k_e - 1)} \tag{11.117}$$

and

$$p'_f(z', k_e) = \frac{\sum\limits_{z} L_f(z', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}) \cdot p'_f(z, k_e - 1)}{\sum\limits_{z, z', f} L_f(z', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}) \cdot p'_f(z, k_e - 1)} \quad (k_e > 0) \tag{11.118}$$

$$p'_f(z', -1) = p_0(z') \cdot \text{Prob}(F = f), \qquad (k_e = 0). \tag{11.119}$$

where $\text{Prob}(F = f)$ *denotes the a-priori fault probability. The diagnostic result is*

$$p_f(k_e) = \sum_{z \in \mathcal{Z}} p_f(f, k_e). \tag{11.120}$$

This result is used now to extend Algorithm 11.6 for the state observation towards the following algorithm for fault diagnosis:

Algorithm 11.7 *Diagnosis of stochastic automata*

Given: Set of stochastic automata $\{\mathcal{S}_f,\ f \in \mathcal{F}\}$.

Initial state probability distribution $p_0(z)$

Initial fault probability distribution $\text{Prob}(F = f)$.

Initialisation: $p'_f(z) = p_0(z) \cdot \text{Prob}(F = f)$ for all $z \in \mathcal{Z}$ and $f \in \mathcal{F}$

$k_e = 0$.

Loop:

1. Measure the current input $\bar{v}$ and output $\bar{w}$.
2. Determine $h_f(z) = \sum_{z'} L_f(z', \bar{w} \mid z, \bar{v}) \cdot p'_f(z')$ for all $z \in \mathcal{Z}$ and $f \in \mathcal{F}$
3. If $\sum_z h_f(z) = 0$ holds, the fault f is not a fault candidate: $p_f(k_e) = 0$.
   If $\sum_{z,f} h_f(z) = 0$, the I/O pair is inconsistent with all models of the set $\{\mathcal{S}_f,\ f \in \mathcal{F}\}$. Stop the algorithm (an unknown fault has occurred).
4. Determine $p_f(z) = \dfrac{h_f(z)}{\sum\limits_{z,f} h_f(z)}$ for all $z \in \mathcal{Z}$ and $f \in \mathcal{F}$.
5. Determine $p'_f(z') = \dfrac{\sum\limits_{z} L(z', \bar{w} \mid z, \bar{v})\ p'(z)}{\sum\limits_{z,f} h_f(z)}$ for all $z \in \mathcal{Z}$ and $f \in \mathcal{F}$.
6. Determine $p_f(k_e) = \sum_{z \in \mathcal{Z}} p_f(f, k_e)$ and $\mathcal{F}^*(k_e) = \{f \in \mathcal{F} : p_f(k_e) > 0\}$.
7. $k_e := k_e + 1$
   Continue with Step 1.

Result: Set of fault candidates $\mathcal{F}^*(k_e)$ together with fault probability $p_f(k_e)$ for increasing time horizon k_e.

The algorithm needs to get the a-priori fault probability distribution $\text{Prob}(F(0) = f)$, $(f \in \mathcal{F})$ as an input. If nothing is known about the faults, a uniform distribution

$$\text{Prob}(F(0) = f) = \frac{1}{q}, \quad f \in \mathcal{F}$$

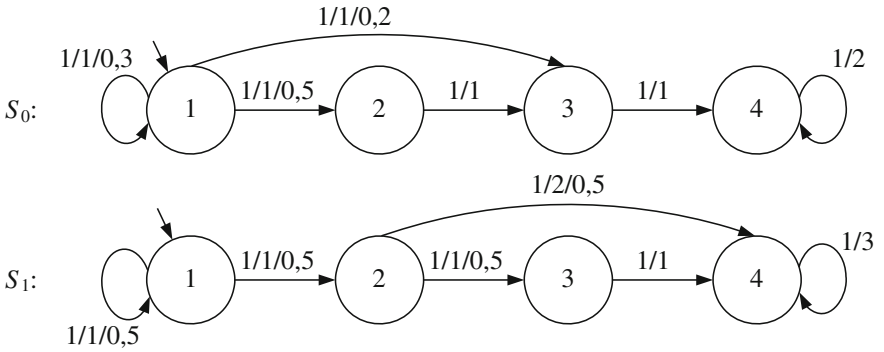


Fig. 11.33 Model of the faultless and the faulty system

may be reasonable, where q is the number of faults considered.

The diagnostic result can be summarised as follows:

- **Fault detection**: If $f_0 \notin \mathcal{F}^*(k_e)$ holds with f_0 denoting the faultless case, then a fault is detected in the system.

- **Fault identification**: The fault f that is present in the system belongs to the set of fault candidates

$$f \in \mathcal{F}^*(k_e)$$

unless it is not considered in the set of faults $\mathcal{F}$. If the set $\mathcal{F}^*(k_e)$ is a singleton, the fault is unambiguously identified. If this set includes more than one element, the probability $p_f(k_e)$, $f \in \mathcal{F}^*(k_e)$ describes with which frequency these faults appear.

Example 11.8 Diagnosis of a stochastic automaton
The automaton in Fig. 11.33 has the state $\tilde{Z} = (z, \quad f)^T$ for the two fault cases $f = 0$ und $f = 1$. Assume that the initial state $z_0 = 1$ is unambiguously known:

$$p_0(z) = \text{Prob}(Z(0) = z) = \begin{cases} 1 & \text{for } z = 1 \\ 0 & \text{else.} \end{cases}$$

For the fault, the a-priori probability distribution is assumed to be uniform:

$$\text{Prob}(F = f) = 0.5 \quad \text{for } f = 0, 1,$$

which means that the diagnostic algorithm starts with the initial probabilities

$$p'_f(z) = \begin{cases} 0.5 & \text{for } z = 1, f = 0, 1 \\ 0 & \text{else.} \end{cases}$$

Table 11.3 shows the diagnostic result for the I/O pair

Table 11.3 Probability distribution of the fault $p_f(k_e)$

	$k_e = 0$ $V(0...0) = (1)$ $W(0...0) = (1)$	$k_e = 1$ $V(0...1) = (1,1)$ $W(0...1) = (1,1)$	$k_e = 2$ $V(0...2) = (1,1,1)$ $W(0...2) = (1,1,1)$
f	$p_f(0)$	$p_f(1)$	$p_f(2)$
0	0.5	0.5714	0.5614
1	0.5	0.4286	0.4386

	$k_e = 3$ $V(0...3) = (1,1,1,1)$ $W(0...3) = (1,1,1,3)$	$k_e = 4$ $V(0...3) = (1,1,1,1,1)$ $W(0...3) = (1,1,1,3,3)$
f	$p_f(3)$	$p_f(4)$
0	0	0
1	1	1

$$V(0\ldots3) = (1,1,1,1,1)$$
$$W(0\ldots3) = (1,1,1,3,3).$$

For the initial state $z_0 = 1$, the first I/O pair ($V(0) = 1, W(0) = 1$) does not give any information about the fault, because both models generate the same output. At time $k_e = 1$, the system can be in one of the following states:

$$\mathcal{Z}_0(1\,|\,0) = \{1,2,3\}$$
$$\mathcal{Z}_1(1\,|\,0) = \{1,2\}$$

The I/O pair ($V(1) = 1, W(1) = 1$) excludes the state transition $2 \to 4$ and, hence, the fault $f = 1$ is less probable than the fault $f = 0$ after the second measurement. After the measurement $W(3) = 3$, the fault $f = 1$ is unambiguously identified. □

11.7.3 Extension to Time-Varying Faults

The diagnostic method developed in the last section for constant faults can be extended to time-varying faults using the model

$$\tilde{\mathcal{S}} = (\tilde{\mathcal{Z}}, \mathcal{V}, \mathcal{W}, \tilde{L}, \text{Prob}(\tilde{z}(0)))$$

given in Eq. (11.35), which includes the information about all fault cases and about the dynamics of the fault $F(k)$, $(k = 0, 1, \ldots, k_e)$. The main idea is to replace the behavioural relation L_f, $(f \in \mathcal{F})$ by the behavioural relation $\tilde{L}$ of the model $\tilde{\mathcal{S}}$.

Then, Eqs. (11.117)–(11.120) have to be replaced by the following relations:

$$p_f(z, k_e) = \frac{\sum_{z', f'} \tilde{L}(z', f', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}, f) \cdot p'_f(z, k_e - 1)}{\sum_{z, z', f, f'} \tilde{L}(z', f', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}, f) \cdot p'_f(z, k_e - 1)} \tag{11.121}$$

$$p'_f(z', k_e) = \frac{\sum_{z, f} \tilde{L}(z', f', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}, f) \cdot p'_f(z, k_e - 1)}{\sum_{z, z', f, f'} \tilde{L}(z', f', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}, f) \cdot p'_f(z, k_e - 1)}, \quad k_e > 0 \tag{11.122}$$

$$p'_f(z', -1) = p_0(z') \cdot \text{Prob}(F = f), \qquad k_e = 0 \tag{11.123}$$

$$p_f(k_e) = \sum_{z \in \mathcal{Z}} p_f(f, k_e). \tag{11.124}$$

Discussion of the diagnostic results. The diagnostic algorithm yields the set $\mathcal{F}^*(k_e)$ of fault candidates for time k_e. Each of the fault separately "explains" why the measured I/O pair occurs. In addition to this set, the probability distribution $p_f(k_e)$ evaluates with which probability each fault candidate represents the real fault.

Under practical circumstances, several heuristic extensions can be made. For example, a threshold s can be fixed and only those faults that occur with a probability higher than s are announced to the human operator in the control room. Then the threshold can be used to adapt the result of the diagnostic algorithm to the certainty with which the behavioural relation of the automaton is known and to the degree of danger that the different faults may have on the system performance. This adaptation of the diagnostic results is analogous to the use of thresholds in the residual evaluation of diagnostic methods for continuous-variable systems described in Chap. 6.

11.7.4 Diagnosability of Stochastic Automata

When solving practical problems, the diagnostic algorithm should provide a set $\mathcal{F}^*(k_e)$ which is a singleton for a possibly small time horizon k_e. Whether or not this is possible, depends on the diagnosability of the stochastic automaton, which is investigated in this section. Note that the diagnosablity is a system property, which depends upon the system dynamics described by the behavioural relation of the automaton and by the measured signals v and w, but does not refer to the diagnostic method applied. Diagnosability claims that the fault f has to be found by *appropriately* using all the information available.

It is not easy to find conditions under which the system is diagnosable, because the question whether a fault can be detected does not only depend on the system dynamics but also on the initial state and on the input sequence. However, the frequent discussions among theoreticans and people from different application fields on "hidden faults" that do not influence the measurement sequence and, thus,

cannot be found by any diagnostic algorithm, and discussions on the fact that different faults have to bring about different effects on the system behaviour if they should be discriminated, show that diagnosability is an important practical issue.

In this section, the results on the observability of stochastic automata presented in Sect. 11.6.3 will be used to define and analyse the diagnosability of automata. Like in Sect. 11.6.3, the starting point is the investigation under what conditions the automaton is not diagnosable.

Definition 11.6 (*Stochastic undiagnosability*) A stochastic automaton $\tilde{\mathcal{S}}$ with behavioural relation

$$\tilde{L}(z', f', w \mid z, v, f) = L(z', w \mid z, v, f) \cdot G_{\mathrm{f}}(f' \mid f)$$

is called stochastically undiagnosable if it satisfies the property

$$L(z', f', w \mid z, v, f) = L(z', w \mid z, v) \tag{11.125}$$

for all $z', z \in \mathcal{Z}$, $w \in \mathcal{W}$, $v \in \mathcal{V}$ and $f \in \mathcal{F}$.

Clearly, under the condition (11.125) the state and output sequences of the stochastic automaton are independent of the fault f, because the fault does no longer appear in L and, hence, the fault cannot be "seen" from the measured I/O pair. In analogy to Lemma 11.7, it can be proved that for undiagnosable automata the diagnostic result coincides with the result obtained by simulation of the faulty behaviour:

Lemma 11.8 *If the stochastic automaton is stochastically undiagnosable, then for all input sequences V and for all output sequences W the diagnostic result is identical to the simulation result:*

$$\mathrm{Prob}(f(k_{\mathrm{e}}) \mid V(0 \ldots k_{\mathrm{e}}), W(0 \ldots k_{\mathrm{e}})) = \mathrm{Prob}(f(k_{\mathrm{e}}) \mid V(0 \ldots k_{\mathrm{e}})). \tag{11.126}$$

The left-hand side of Eq. (11.126) is the result obtained from the diagnostic algorithm. As the fault f does not depend on the input v, the right-hand side of Eq. (11.126) is given by the relation

$$\begin{aligned}
&\mathrm{Prob}(F(k_{\mathrm{e}}) = f \mid V(0 \ldots k_{\mathrm{e}})) \\
&= \sum_{F(0 \ldots k_{\mathrm{e}}-1)} G_{\mathrm{f}}(f \mid f(k_{\mathrm{e}}-1)) \cdot G_{\mathrm{f}}(f(k_{\mathrm{e}}-1) \mid f(k_{\mathrm{e}}-2)) \cdot \ldots \\
&\qquad \cdot G_{\mathrm{f}}(f(1) \mid f(0)) \cdot \mathrm{Prob}(F(0) = f(0)),
\end{aligned}$$

which predicts the state of the fault model $\mathcal{S}_f$. The fault changes with increasing time horizon k_{e} and so does the probability distribution

$$\mathrm{Prob}(F(k_{\mathrm{e}}) = f \mid V(0 \ldots k_{\mathrm{e}})).$$

However, the only information used for simulation is the state transition relation G_f of the fault model. As the diagnostic algorithm uses further information given by the output sequence W it is reasonable to expect that the diagnostic result is better than the simulation result. The lemma says that for stochastically undiagnosable automata this expectation is not met. The diagnostic algorithm cannot improve the simulation result.

A given stochastic automaton is generally not completely stochastically undiagnosable according to Eq. (11.125), but there may exist one or more state sets $\mathcal{G}_z \subset \mathcal{Z}$ and one or more fault sets $\mathcal{G}_f \subset \mathcal{F}$ such that the behaviour within the set $\mathcal{G}_z$ does not depend on the faults $f \in \mathcal{G}_f$. Then Eq. (11.125) does not hold for all z, z' and f, but for all $z, z' \in \mathcal{G}_z$ and all $f \in \mathcal{G}_f$. If a non-empty fault set $\mathcal{F}$ can be found such that Eq. (11.125) is satisfied for all $z, z' \in \mathcal{G}_z$, the set $\mathcal{G}_z$ is called a *stochastically undiagnosable state set*. If the stochastic automaton does not possess such a state set, it is called diagnosable.

Definition 11.7 (*Stochastic diagnosability*) A stochastic automaton is called stochastically diagnosable if it does not possess any stochastically undiagnosable state set.

It is obvious from the investigations above that for stochastically diagnosable systems the diagnostic algorithm yield better results than a simulation of the behaviour of the fault model.

Example 11.9 Diagnosability of stochastic automata
The stochastic automaton depicted in Fig. 11.33 is not stochastically undiagnosable. Nevertheless, the set of faults $\mathcal{F}$ is stochastically undiagnosable within the set of states $\mathcal{G}_z = \{1, 2, 3\}$. Hence, as long as the system is not in state $z = 4$ nor has the possibility to go to this state within one time step, no information about the fault can be obtained. This result can be seen from Example 11.8. The fault $f = 1$ is proved not to exist at time $k_\mathrm{e} = 3$ when the output $w = 3$ occurs, which proves that the automaton is in the state $\tilde{z} = (4, 2)^\mathrm{T}$. □

Example 11.10 Diagnosis of a stochastic automata
As an example, consider the task system to diagnose a fault by means of the automata shown in Figs. 11.34 and 11.35. Both automata together describe the behavioural relation $L_f(z', w \mid z, v)$. The fault is assumed to be constant.

A diagnosability check for $v = 1$ yields the result that the stochastic automaton is not diagnoseable with respect to the set of faults $\mathcal{F} = \{1, 2\}$ within all states $\mathcal{G}_z = \mathcal{Z}$. For $v = 2$ the automaton is diagnosable.

Three experiments are considered. First, the input is fixed at $v = 2$ and the fault is $f = 1$. An experiment with the initial state $z = 6$ yields the output sequence shown in the left part of Fig. 11.36. A second experiment with the same initial state is made for $v = 2$ and $f = 2$ resulting in the output sequence shown in the middle part of Fig. 11.36, and a third experiment with $v = 1$ and $f = 1$ leads to the right part of Fig. 11.36.

The diagnostic results corresponding to the three experiments are shown in Fig. 11.37. It can be seen that the fault is isolated for $v = 2$, but for $v = 1$ the diagnostic result is merely a simulation of the initially known fault distribution, which results for the given automaton in a sequence of uniform distributions. The result is obtained because the automaton is undiagnosable for $v = 1$ in the whole state set $\mathcal{Z}$. □

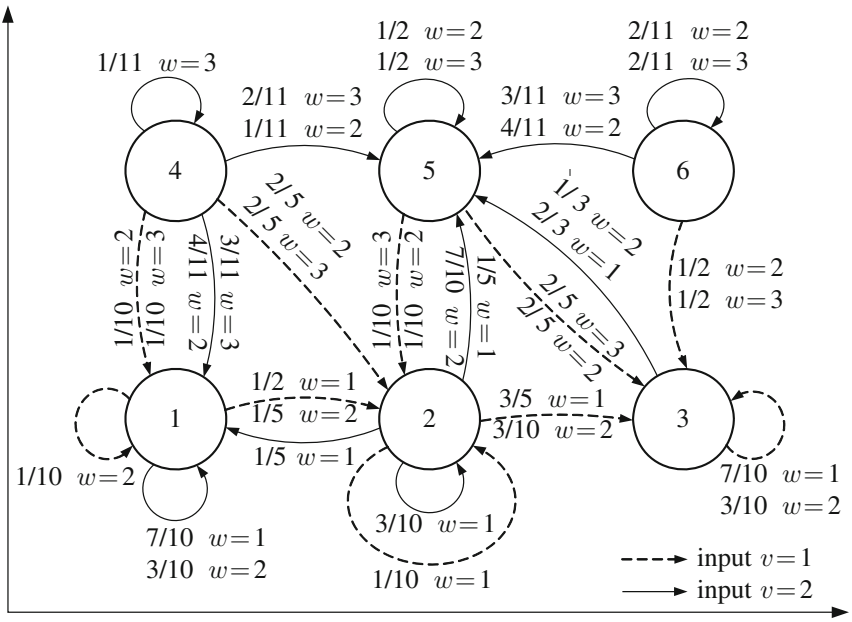


Fig. 11.34 Automaton graph for fault $f = 1$

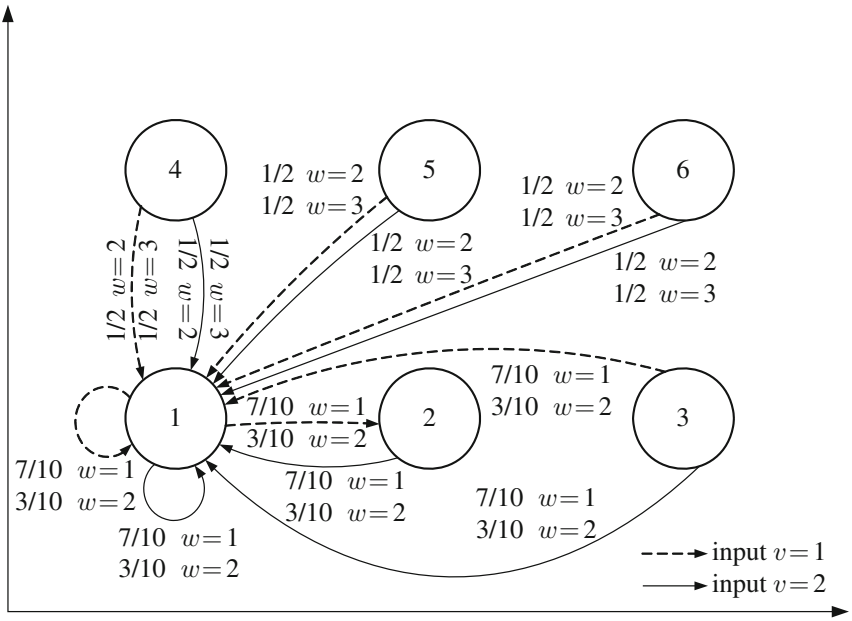


Fig. 11.35 Automaton graph for fault $f = 2$

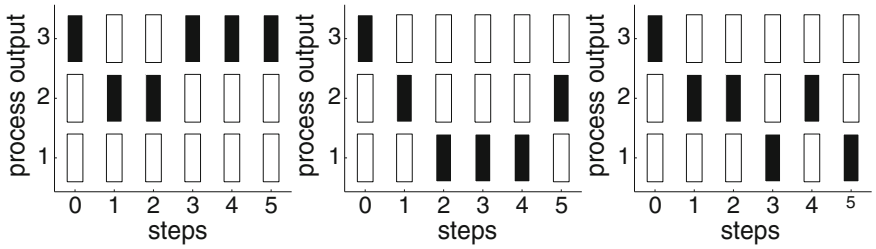


Fig. 11.36 Output sequences for $v = 2$, $f = 1$ (*left*), $v = 2$, $f = 2$ (*middle*) and $v = 1$, $f = 1$ (*right*)

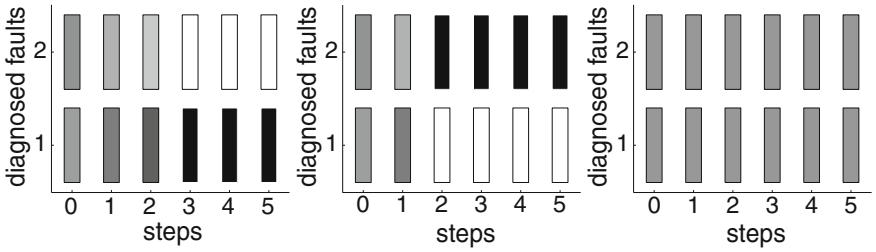


Fig. 11.37 Diagnostic results for the three experiments shown in Fig. 11.36 in the same order

11.8 Exercises

Exercise 11.1 Observability of stochastic automata
Assume that the current state probability distribution is given for time k_e and denoted by $\text{Prob}(z(k_e))$. Prove the following fact: If at time $k_e + 1$ an input $v(k_e + 1)$ and an output $w(k_e + 1)$ occur for which a decomposition (11.107) is possible for the state set

$$\mathcal{Z}(k_e \mid k_e) = \{z : \text{Prob}(Z(k_e) = z) > 0\}$$

then in Steps 4 and 5 of the observation algorithm the same results are obtained as by simulating the automaton behaviour according to Eq. (11.26). □

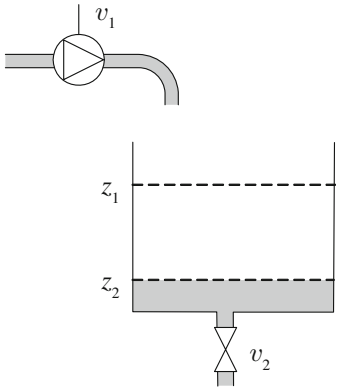
Exercise 11.2 Diagnosis of fixed faults
How can the Algorithm 11.7 be simplified if the fault is known not to change over time? □

Exercise 11.3 Diagnosis of a batch reactor
The reactor shown in Fig. 11.38 is used within a larger batch process, where it is filled and emptied in order to bring a certain amount of liquid into another reactor. For the behaviour of the reactor only the empty and the full state is distinguished, where the liquid level is above the higher or below the lower border shown in the figure. These states are denoted by z_1 and z_2.

To fill the reactor, the pump is switched on (input v_1), to empty the reactor, the input v_2 opens the valve. A security check ensures that the pump is not switched on if the valve is open.

1. Describe the reactor by a deterministic automaton

Fig. 11.38 Batch reactor



2. The fault f_1 breaks the pump. Extend you model in order to describe the reactor for the faultless and the faulty operation mode.
3. The faultless reactor remains faultless with the probability of 99 % in all state transitions that are caused by switching the pump or opening and closing the valve. Extend you model to get a stochastic automaton, which reflects this information. □

Exercise 11.4 Diagnosis of nondeterministic automata
How can the diagnostic method developed in this chapter be simplified if instead of a stochastic automaton a nondeterministic automaton is used to describe the system under consideration? Do the sets of fault candidates obtained by both methods distinguish? □

11.9 Bibliographical Notes

The first results concerning state observation of discrete-event systems occurred in connection with the supervisory control theory developed in [284] where the supervisor has to reconstruct the current state of the system from partially measurable states or events. Reference [198] defined the notion of the observable language and developed results on the existence of the combined supervisory control and the observation problem given. Reference [55] showed that for supervisory control the problem of state observation can be reformulated as an event observation problem.

Observability of discrete-event-systems. The classical observability definition has been given in [56] and was used also, for example, in the textbooks [51, 320]. According to this definition a stochastic automaton is called *semi-deterministic* or observable if for all states z the successor state $z' = \varphi(z, v, w)$ can be unambiguously determined if the current state z, the current input v and current output w are known. This definition is useful only if it can be *assumed* that the automaton state z at some time k is precisely known. Then, the future sequence of states starting in z can be unambiguously determined. However, as long as this assumption is not satisfied, the notion

of observability does not say anything about the solvability of the observation problem. Similar remarks hold true for other observability definitions like the one given in [259] which likewise claim that the automaton state should be unambiguously determined.

Several papers have been published about the connection of state observation and fault diagnosis for discrete-event systems. Some of the aspects discussed are summarised in [95].

Diagnosability and diagnostic methods. The diagnosis of discrete-event systems was the subject of a steadily increasing number of papers in the past with the references [12, 194, 277, 319] as early papers on the diagnostic problem for Petri nets, and [179, 197, 216, 218, 297, 306] for nondeterministic or stochastic automata. Conditions on the automaton under which the fault can be uniquely determined have been derived in [197, 297]. If the input to the automaton should satisfy certain requirements to avoid the situation where the system reaches forbidden states, the diagnosability conditions appear to be stronger as shown in [268, 269]. A combination of Petri net and automata theoretic approaches is described in [52].

In the stochastic setting, there are different definitions of diagnosability. In [366], a stochastic automaton is said to be diagnosable if on all state trajectories the fault can be detected and, hence, the fault is eventually unambiguously identified. In [218], diagnosability means that the fault changes the I/O behaviour and leads to an increase in its probability. Reference [179] describes detectability of a fault as the ability to estimate the current state of a system with increasing certainty. Reference [277] outlined the connections between discrete-event models and logical descriptions, which opens the way to apply diagnostic methods elaborated in the field of artificial intelligence to discrete-event systems (for survey cf. [139] or [212]). However, most of the diagnostic methods developed in artificial intelligence can only be used for static system descriptions, whereas the methods that have been developed here allow to diagnose dynamical systems far from their equilibrium state.

Based on Algorithm 11.7 for the diagnosis of stochastic automata, a specific sensor and actuator supervision system has been developed in [219].

The results on the diagnosability of deterministic automata developed in Sect. 11.4 have been published in [215]. The method for finding distinguishing inputs developed in Sect. 11.4.5 is similar to the one described in [128] but uses an alternative notation and, thus, directly extends a method for determining equivalent states of deterministic automata. Details can be found in [303, 304].

Extensions. The issue of complexity reduction of the diagnosis of automata has been considered in [210]. The basic idea was to lump unobservable states of the model together because during the movement in such sets of states the observation or diagnostic algorithm does not gain any additional information about the system. It has been shown that this method of complexity reduction can be applied for nondeterministic automata. However, for stochastic automata the complexity reduction brings about biased diagnostic results.

The results reported in Sect. 11.5 have been developed in [218]. All proofs of the results given here can be found in this reference.

The extension of the diagnostic method to remote diagnosis, where data loss in the network has to be tolerated, is described in [114, 302].

The methods explained in this chapter can be extended to timed automata as shown in [351].

The introduction to automata theory and the Exercise 11.3 follow the textbook [209].