An Automata Theoretic Approach to Modular Diagnosis of
Discrete-Event Systems
Acknowledgements

It took more than five years to develop the results presented in this book. This work would not have been possible without the support and encouragement of some very special people.

First of all my special thanks go to Prof. Dr.-Ing. Jan Lunze for the possibility to work at his institute, his supervision, and his professional advice. His refusal to settle for mediocrity helped me to understand what science is all about. My special thanks also go to Prof. Dr.-Ing. Alexander Fay for reviewing this work and the fruitful discussions at his institute.

I also thank all my former colleagues at the institute for the professional and private discussions we had (usually with a cup of coffee at hand). The friendly atmosphere and possibility to annoy everyone with seemingly silly questions really contributed to this work. I also have to say a big thank you to everyone who reviewed parts of this work (Christian Falkenberg, Philippe Planchon, Jan Richter, and Florian Wenck). And, of course, thank you for the great time and fun we had after work. You rock!

Five years are a long time and not every day has been a happy day, sometimes making the progress of my work seem to come to an halt. Without great friends and a supporting family I would not have had the power to keep things up and going. I especially thank Philippe for his friendship, continuous support, and sharing an office with a “pfälzer Dickschädel” for such a long time. This work definitely would have ended prematurely without the friendship and support of Christian. Thank you for your trust, having an open ear even at two o’clock in the morning, and simply being there when needed.

My personal thanks go to my family who supported me not only throughout these years, but throughout my whole life. Thank you for giving me hold and for making this possible.

Nürnberg, April 2007

Jörg Neidig
Jörg Neidig

An Automata Theoretic Approach to Modular Diagnosis of Discrete-Event Systems
## Contents

1. Introduction 1
   1.1. Diagnosis of Technological Systems 1
   1.2. Motivation and Aim 2
   1.3. Tasks and Main Results 3
   1.4. Literature Overview 5
   1.5. Document Structure 7

I. Theoretical Foundation 9

2. Elaboration of the Diagnostic Task 11
   2.1. Principles of Diagnosis 11
   2.2. Different Information Structure Concepts 14
   2.3. Diagnosability and Output Indifference 17

3. Introduction to Automata Theory 19
   3.1. Relational Data Representation 19
   3.2. Single Automaton 23
   3.3. Synchronous Automata Network 35

II. Diagnosis of Nondeterministic Automata Networks 47

4. Centralised Diagnosis of Nondeterministic Processes 49
   4.1. Centralised Information Structure 49
   4.2. Diagnosis of a Single Automaton 50
   4.3. Application to Automata Networks 54
   4.4. Computational Enhancements 57
   4.5. Diagnosability and Output Indifference 59
   4.6. State Observation 68
   4.7. Conclusion 71

5. Decentralised Diagnosis of Nondeterministic Automata Networks 73
   5.1. Motivation of Decentralised Diagnosis 73
   5.2. Solution to the Decentralised Diagnostic Problem 74
   5.3. Complexity Considerations 80
   5.4. Impact of the Information Structure Constraints on the Diagnostic Result 82
   5.5. Behavioural View on the Lack of Soundness 87
### Contents

5.6. Diagnosability ........................................... 89  
5.7. Conclusion .............................................. 91  

6. **Coordinated Diagnosis of Nondeterministic Automata Networks** 93  
6.1. Information Structure Revisited .......................... 93  
6.2. Coordination with Unidirectional Information Flow ......... 95  
6.3. Coordination with Bidirectional Information Flow ........... 102  
6.4. Diagnosability ........................................ 108  
6.5. Comparison of the Presented Diagnostic Approaches ......... 109  

III. **Diagnosis of Stochastic Automata Networks** 111  

7. **Centralised Diagnosis of Stochastic Processes** 113  
7.1. Diagnosis of a Single Automaton .......................... 113  
7.2. Application to Automata Networks ......................... 118  
7.3. Diagnosability and Output Indifference .................... 120  

8. **Limitations of the Modularisation of Stochastic Processes** 125  
8.1. The Consequence of Stochastic Dependence ................ 125  
8.2. Alternative Approaches .................................. 131  

IV. **Further Aspects of Interconnected Discrete-Event Systems** 133  

9. **Simulation of Stochastic Automata Networks** 135  
9.1. Introduction .............................................. 135  
9.2. Simulation of the Behaviour of a Single Stochastic Automaton .... 136  
9.3. Straight Forward Approach to Simulation of Networks .......... 138  
9.4. Simulation Using On-line-Composition .................... 139  
9.5. Conclusion .............................................. 141  

10. **Abstraction of Networks of Quantised Systems** 143  
10.1. Motivation .............................................. 143  
10.2. Abstraction of Quantised Systems ......................... 144  
10.3. Networks of Quantised Systems .......................... 148  
10.4. Abstraction of Networks .................................. 149  
10.5. Assuring Completeness .................................. 153  
10.6. Conclusion .............................................. 156
11. Direct Feedback in Automata Networks 157  
11.1. Introduction .................................................. 157  
11.2. Feedback in Interconnected Systems ...................... 158  
11.3. Solution to the Feedback Problem .......................... 160  
11.4. Bypassing the Feedback Problem .......................... 166  
11.5. Conclusion .................................................... 167

12. Application Example 169  
12.1. Centralised Diagnosis of the Air-Path of a Diesel Engine .... 169  
12.2. Diagnosis of a Three-Tank System .......................... 172  
12.3. Simulation of a Three-Tank System .......................... 176

13. Summary 179

Bibliography 181

A. Notation, Symbols, Operators 193
B. Proofs 197
C. Algorithms and Implementation 207
D. Introduction to SQL 211
E. Example Relations 215
1

Introduction

1.1. Diagnosis of Technological Systems

The continuously improving technology has already reached a complexity in which even specialists find it more and more difficult to comprehend the technical connections and to remove occurring faults. Therefore, it is necessary to develop methods that automatically find the faults in technological systems and warn the user timely to avoid harm to man, environment, and machine. This necessity is the background for the present book in which several approaches to diagnosis are developed and their applications discussed.

The developed approaches all base on the concept depicted in Figure 1.1. The technological system under fault is subject to inputs from the environment and produces outputs. In general, the system input/output-behaviour is altered by the fault. The task of the diagnostic system is to find the fault using measurements of the inputs and outputs and mathematical models of the technological system.

The approaches presented in this book are qualitative in the sense that no real-valued signals and models, but abstract qualitative information is used instead as shown in Figure 1.2. This figure includes a real-valued signal (the light coloured trajectory) and an abstraction to qualitative values depicted as the black bars. With only the qualitative information at hand it is known that the signal value lies in the set marked by the bars, but the exact signal value is unknown. Instead, it can only be said that the signal value is e.g. low. In spite of this coarse reduction of information fault diagnosis is still possible in most cases, because, in general, a fault changes the system’s behaviour qualitatively.
1. Introduction

1.2. Motivation and Aim

Qualitative approaches to diagnosis have proven to be sensible and powerful when dealing with complex systems. First of all, these methods disburden the engineer from the need to derive detailed quantitative models by restricting the information about the system to the minimum needed for the desired task. Secondly, the qualitative methods allow for an easy implementation and do not require complex solver for differential equations like many quantitative methods do.

Despite their easy structure all monolithic quantitative models suffer from the common problem of state-space explosion. That is, with an increasing number of signals, the size of the model quickly becomes too large to be handled, even with modern computer equipment. Therefore, complexity reduction is an important goal of current research in diagnosis of discrete-event systems.

In this book multiple approaches are presented to solve the complexity problem by modelling the system using a modular structure, i.e. as a connection of several sub-
systems, and to obtain the advantages of the modular model in diagnosis by dividing the diagnostic task into several subtasks (Fig. 1.3). The type of models used throughout the book are the synchronised nondeterministic and stochastic automata network. The distinctive features of these models are that the state transitions of all automata happen synchronously and that a clock pulse determines the progress of time. Furthermore, the communication between the different automata of the network occurs through unmeasurable coupling signals.

![Diagram of Modular Diagnosis](Figure 1.3: Modular diagnosis)

### 1.3. Tasks and Main Results

The primary aim of this book is to provide an extensive and detailed framework for reducing the computational complexity in qualitative model-based diagnosis of discrete-valued systems. The reduction of complexity is achieved through compositional modelling and modularisation of the diagnostic task. The main features of the developed diagnostic methods are the following:

- **Certainty.** For all developed diagnostic methods and algorithms it will be proven that the result is complete. That is, no fault must be excluded from the result set wrongly. It must not happen that a faulty system is diagnosed as faultless.

- **On-line applicability.** All algorithms are given in a form to allow for on-line diagnosis. The diagnostic results are refined recursively in each diagnostic step using the newest measurements only.

- **Robustness.** The usage of coarse discrete-valued discrete-time signals (cf. Figure 1.2) reduces not only the information to the minimum needed for the diagnostic task, but makes the methods robust against signal disturbances and noise.
1. Introduction

- **Low computational effort.** In fulfillment of the aim of this book the diagnostic algorithms have low computational costs in comparison to other model-based qualitative methods. Furthermore, the algorithms are straightforward to implement which is supported by a number of examples.

The presented diagnostic concepts presented use a differing amount of knowledge about the system and a differing degree of modularisation. They are compared with respect to the following aspects. The **quality** of the diagnostic result can be measured by the number of fault cases in the diagnostic result. A good diagnostic result contains as few fault cases as possible while assuring that the actual fault is included in the result (completeness). **Scalability** characterises the expenditure to transfer the solution of a problem to a larger problem. Scalability is not quantified, but is used to compare approaches qualitatively instead. Good scalability reflects low costs when scaling a problem up. **Reliability** or fail-safe indicate how vulnerable an approach is to partial failures or breakdowns. In general, the reliability increases with the degree of the modularisation. **Simplicity** is a subjective quantity indicating how easy an approach can be realised and comprehended in the field of practise. **Reusability** defines how good the solution of a problem can be transferred to a similar problem. **Distribution** characterises the ability to spread the setup of a task locally and the design of the solution among different personnel.

To achieve the aim of the book the following problems have to be solved:

- **Representation.** The automata network has to be modelled and represented in a way which is convenient for modular diagnostic methods.

- **Diagnosis.** Clearly, the primary task in diagnosis is to find the faults in technological systems. The aim is to return a diagnostic result which contains as few faults as possible while satisfying the completeness demand.

- **Information structure constraints.** Computational costs can be decreased be reducing the amount of information used or utilising approaches which take the system structure into consideration. The effect of different information structure constraints on the diagnostic result is to be investigated.

The main results presented in this book are the solutions to these problems.

- Two different ways of representing automata networks are derived. One approach is based on **characteristic functions** and allows for algebraic calculations in the algorithms. Also problem solutions and proofs can be given as closed algebraic expressions. This approach is advantageous when modelling
stochastic automata networks. The second approach is based on relations from relational algebra. It permits a compact and structured notation of interconnected systems and provides powerful set operations to handle data in tabular form. This approach is mainly used to model nondeterministic automata networks.

- Several approaches to diagnosis are developed for nondeterministic and stochastic automata networks. All methods fulfill the completeness demand. The quality of the diagnostic result depends highly on the information structure chosen for the specific task. The results of the different methods are put in relation with each other.

- The diagnosis of nondeterministic automata networks is investigated for centralised, decentralised and coordinated information structures. It is discovered that the decentralised approach demands the least computational resources, but leads to a degradation of the diagnostic result. The coordinated approach returns the ideal (i.e. best possible) diagnostic result with a distinct computational advantage over the centralised approach. For stochastic automata networks the centralised and decentralised information structure is investigated. It is proven that, in general, a modularisation of the diagnostic task is impossible without reducing the explanatory power of the probabilistic values.

Further results:

- Direct feedback in networks with synchronised signals might render the overall system ill-defined. This feedback problem has been solved and criteria are given to test whether a system with feedback is well-defined or not.

- It is shown that in general it is not possible to decompose a stochastic system into different components which can be treated independently in diagnosis. I.e. the modular approaches developed for the nondeterministic automata network are not transferable to stochastic systems.

- It is shown how the automata network model of a given technical system is derived while assuring the completeness of the model.

1.4. Literature Overview

An overview over different diagnostic methods can e.g. be found in [25, 41]. Qualitative diagnosis using automata models has been investigated in numerous publications. However, most approaches deal with asynchronous discrete-event models rather than
1. Introduction

the models with synchronised signals described in this book [20, 55, 60, 61]. The most prominent monolithic diagnostic method for discrete-event models is described in [89] and is the basis for most approaches in this area. In these approaches the system model and the measurement of a sequence of events is used to detect and isolate faults. It is not distinguished between input and output events. A serious drawback of these approaches is that the fault is interpreted as a single event. I.e. a fault influences a system only punctiform instead of constantly changing the dynamics. In [110] these differences are investigated in depth. A further drawback of these approaches is the inability to use synchronised events, i.e. it is not allowed that multiple events occur simultaneously.

The basis for the present book is the monolithic approach to diagnosis for the single stochastic automaton as introduced in [12, 66, 93]. The used model is discrete-time and all signals are synchronised on a clock. In [93] it is also described in depth how such an automaton model can be derived automatically from the continuous-valued system description of a plant. Because the automaton does not have to be built by hand, complex systems can be modelled and diagnosed instead of using only small academic examples as it is often the case in the approaches mentioned above.

The complexity problem in modelling, diagnosis and control of discrete-event systems has been mentioned in several works and solving this problem is an important task in current research [36, 57, 95, 104, 111]. Decentralised diagnostic methods for networks of discrete-event systems [16] have been published in recent years and are mostly extensions of the approach from [89] to modular models [24, 78, 80, 84]. Alternative approaches (e.g. [95, 97]) use models based on language theory as given by Wonham in [87, 106]. Petri-nets as an equivalent representation of automata are also used for diagnosis, however, without any distinct improvements to the previously mentioned approaches [8, 38, 52]. Hierarchical diagnostic approaches are e.g. published in [37, 100].

Coordinated diagnosis is a fairly new branch of research. In [29] the concept is described in detail for discrete-event systems. In [42] a coordinated method for static systems is published. [38] uses a coordinated approach with a Petri-Net model. Common to all of the above approaches is that the synchronising events, in other words the information coupling the components, are measurable. This effectively decouples the components from a diagnostic perspective. The modelling approach presented in this work allows to include coupling signals which cannot be measured. This is important when modelling existing plants where different components influence each other, but no sensor is available to measure that influence.
Diagnosis of stochastic automata networks is only investigated by a small number of groups [10, 69, 81, 85, 101], because it is impossible to decentralise the diagnosis of stochastic networks without losing information. In [15, 27, 28, 51] dynamic Bayesian networks – a type of model close to the synchronised stochastic automata networks – are used for diagnosis. The complexity problem is solved by splitting the network into several subnets and estimating the error which is made when treating stochastic dependent variables as independent.

1.5. Document Structure

This book is divided into four parts which are described in the following.

- **Part I** conveys the **theoretical basics** needed for the solution of the diagnostic task which is described in detail in Section 2. In Section 3 the automata network is introduced formally.

- **Part II** contains the primary results of this book. Several approaches for **diagnosis of nondeterministic automata networks** are presented here. In Section 4 the centralised diagnosis of single nondeterministic automata and of automata networks is given as the basis for the further investigations. The diagnostic approach is modularised in Section 5 to a fully decentralised diagnostic approach. In Section 6 an approach to coordinated diagnosis based on the decentralised methods is presented. The reduced computational complexity together with the outstanding quality of the diagnostic result marks the major result of this work.

- **Part III** investigates the **diagnosis of stochastic automata networks**. In Section 7 methods for the centralised diagnosis of these networks are given analogously to Section 4. In Section 8 it is described why the modularisation poses problems in the stochastic case. Criteria which have to be fulfilled for a successful decentralisation are given.

- **Part IV** contains further information about the handling of the synchronised automata network. The simulation of the behaviour of an automata network is investigated in Section 9. How a qualitative composite model can be gained is discussed in Section 10. In Section 11 the problem of direct feedback in automata networks with synchronised signals is dealt with. Finally, Section 12 contains application examples to demonstrate the applicability of the approaches presented in this paper.

The book closes with a summary in Section 13. The appendix contains among the proofs and algorithms a short introduction to SQL.
1. Introduction

**Reading the book.** Part I lays the theoretical foundation needed for comprehending the remainder of this work and can therefore be considered as obligatory for the reader who is unfamiliar with the subject. Parts II to IV are fairly independent of each other. Although reading them in the given order is recommended, they are comprehensible on their own. As written above, Part II contains the major results of this paper and should be seen as compulsory. The chapters in this part depend highly on each other and should be read in the given order. The diagnosis of stochastic automata networks in Part III is not done to the same extend as for the nondeterministic automata network and can be seen as an extension for the interested reader. Part IV is intended for the interested reader only as it contributes merely indirectly to solving the diagnostic task.
Part I.

Theoretical Foundation
Elaboration of the Diagnostic Task

The aim of diagnosis is to detect if a system is operating abnormally (fault detection) and possibly to identify the cause for the abnormal behaviour (fault identification). In this chapter the principles of model-based diagnosis are explained in Section 2.1 and the notions of centralised, decentralised and coordinated diagnosis, which are the quintessence of this book, are presented in Section 2.2. Finally, the notions of diagnosability and output indifference are introduced in Section 2.3.

2.1. Principles of Diagnosis

Throughout the book the abbreviations

\[ V(0\ldots k)=(v(0),v(1),\ldots,v(k)) \] and
\[ W(0\ldots k)=(w(0),w(1),\ldots,w(k)) \]

are used to denote sequences of inputs \( v \) and outputs \( w \) (Figure 2.1).

![Figure 2.1: Discrete-time input \( v(k) \) and output \( w(k) \) of a system](image)

The \textit{behaviour} \( B \) of a system is defined as the set of all I/O-sequences of arbitrary length which can be generated by the system:

\[ B \subseteq (\mathcal{N}_v \times \mathcal{N}_w) \cup (\mathcal{N}_v \times \mathcal{N}_w)^2 \cup \cdots, \]
where $\mathcal{N}_v$ and $\mathcal{N}_w$ denote the domains of $v$ and $w$, respectively. The consistency-based diagnosis is then performed by comparing the measurement

$$B(k) = (V(0\ldots k), W(0\ldots k))$$

to the behaviour $B(f)$ of models $A^f$ which describe the system under the influence of a given fault $f \in \mathcal{N}_f$ as depicted in Figure 2.2 on the next page (multi-model approach). That is, the behaviour $B(f)$ includes all input/output-sequences modelled by $A^f$.

**Definition 2.1 (Consistency):** A model $A^f$ is consistent with the measurement if

$$B(k) \in B(f)$$

holds.

The ideal result of a diagnosis is a set $\mathcal{F}^\star(k) \subseteq \mathcal{N}_f$ which includes exactly those faults for which the measurements up to time $k$ are consistent with the model:

$$\mathcal{F}^\star(k) := \{f | B(k) \in B(f)\}.$$

(2.2)

This can be written analogously as

$$f \in \mathcal{F}^\star(k) \iff B(k) \in B(f).$$

(2.3)

Because the behaviours $B(f)$ of the different fault models are non-exclusive, $B(k)$ might be consistent with several models such that $\mathcal{F}^\star(k)$ includes more faults in addition to the actual fault. The consistency test (2.1) is the core of a diagnostic algorithm. One major engineering task is to develop an algorithm which realises this test such that the diagnostic result is as close as possible to the ideal set $\mathcal{F}^\star(k)$.

The faults $f \in \mathcal{F}^\star(k)$ are called fault candidates. Because faults are only excluded from $\mathcal{F}^\star(k)$ if the model is inconsistent with the measurement, it is assured that no faults are excluded wrongly, i.e. no fault is missed. The diagnostic result is said to be complete and sound.

**Definition 2.2 (Completeness):** A diagnostic result $\mathcal{F} \subseteq \mathcal{N}_f$ is complete iff all fault cases $f \in \mathcal{N}_f$ for which the measurements are consistent with the respective behaviour $B(f)$ are contained in the set $\mathcal{F}$, i.e. the following relation holds

$$f \in \mathcal{F}(k) \iff B(k) \in B(f).$$

(2.3)
2.1. Principles of Diagnosis

**Definition 2.3 (Soundness):** A diagnostic result $F \subseteq N_f$ is *sound* iff for all fault cases $f$ in the set $F$ the measurements are consistent with the respective behaviour $B(f)$, i.e. the following relation holds

$$f \in F(k) \Rightarrow B(k) \subseteq B(f).$$

For a diagnostic result which is complete, but not sound the relation $F(k) \supseteq F(k)^\star$ holds. I.e. the set $F(k)$ might include additional fault cases. These additional faults are called *spurious solutions*. If a diagnostic result is sound, but not complete the relation $F \subseteq F^\star$ holds. I.e. the set $F$ might not include all fault candidates.

Whereas the complete result might cause false alarms, it is guaranteed that no broken system is diagnosed as faultless. Because this guarantee is very important, the completeness property is required for all diagnostic methods presented in this work.

For notational convenience the faultless case is included in the set of faults to be identified and will be marked by $f = 1$. With this convention a fault is detected whenever the faultless case is not in the diagnostic result set and a fault $f = i$ is identified if it is the only fault case in the result:

$$1 \notin F(k) \quad \text{fault detection}$$

$$\{i\} = F(k) \quad \text{fault identification}.$$ 

A system is said to operate faultlessly if the fault case $f = 1$ has been identified. This assumes that all possible fault cases have been considered during diagnosis *(closed*...
2. Elaboration of the Diagnostic Task

 worldly assumption).

In this book only process models in form of automata are used for diagnosis. The different fault models \( \mathcal{A}^f \) are aggregated to a single model \( \mathcal{A} \) which depends on the immeasurable signal \( f \). Furthermore, it is assumed that the model is identical to the process. While this may not be a realistic assumption, it can be assumed without loss of generality that it is always possible to design the model such that its behaviour includes the behaviour of the process. This in turn ensures that no process behaviour is missed, i.e. not modelled [68]. The modelling process is investigated in detail in Chapter 10.

With this convention the diagnoser \( \mathcal{D} \) is defined by the tupel

\[
\mathcal{D} = \{ \mathcal{A}, \mathcal{B}, \mathcal{I}, \mathcal{A}, \mathcal{R} \},
\]

where

- \( \mathcal{A} \) is the automaton model of the process,
- \( \mathcal{B} \) is the measurement of the system’s input and output signals,
- \( \mathcal{I} \) is the a-priori information about the system’s state and faults (initial condition),
- \( \mathcal{A} \) is the algorithm used to calculate the diagnostic result, and
- \( \mathcal{R} \) is the diagnostic result. It defines what the aim of the algorithm is.

The diagnostic result is either a set of faults \( \mathcal{F} \) or a discrete probability distribution \( p(f) \) over the faults. In some cases the notation \( \mathcal{R}(\bullet) \) will be used to clarify that the result concerns the signal \( \bullet \).

The diagnosis of static systems or short static diagnosis is a special case of diagnosis where the measurements and the behaviours only include one time-step \( (k = 0) \) and the system is only evaluated in its steady-state. I.e. no information about the system’s dynamics is used.

2.2. Different Information Structure Concepts

A diagnostic setup consists of two components, the system which is to be diagnosed and the diagnoser. Both components may again be split up into several interconnected subsystems. E.g. the process may be modelled as an automata network and the
2.2. Different Information Structure Concepts

diagnostic task may be broken down into several smaller tasks. The information structure specifies the type and quantity of information which is transmitted between these components. It answers the question “Who knows what?” [46]. Depending on the nature of the setup different constraints may apply to the information structure. Whereas in control theory it is common to have a “everybody knows everything”-concept, in diagnosis it is assumed that a principle restriction applies such that the diagnoser has no access to the fault signals (cf. Fig. 2.2).

Generally, the information structure constraints may be imposed by the given operating structure of the system or may be introduced during the design of the system’s automation concepts. Restricting the information will in many cases degrade the system’s output. However, a suboptimal output might be acceptable whenever notions such as simplicity of application, flexibility or computational restraints are to be met. In the following different information structure concepts in diagnosis will be discussed [37, 64, 83, 95, 100, 107].

Centralised diagnosis: In centralised diagnosis one plant is diagnosed by one diagnoser (cf. Fig. 2.3a). In general, the diagnoser has access to all measurements and a full model of the plant. However, restrictions may apply concerning the model accuracy and the available measurements. This approach poses the least restrictions on the information structure. The centralised approach is investigated in Chapter 4 for the nondeterministic and in Chapter 7 for the stochastic automaton.

Decentralised diagnosis: In decentralised diagnosis multiple local diagnosers diagnose multiple plants or multiple components of one plant. Every diagnoser has only access to a subset of the available measurements and only a part of the plant’s model. The diagnosers do not exchange information, but instead operate completely independent of each other (cf. Fig. 2.3b). Because every diagnoser has only very little information about the whole system, this approach poses the strongest restrictions on the information structure. Decentralised diagnosis is a special case of distributed diagnosis described below. The decentralised approach is considered in Chapter 5 for nondeterministic and in Chapter 8 for stochastic automata networks.

Distributed diagnosis: In distributed diagnosis multiple local diagnosers diagnose multiple components of one plant. Every diagnoser has only access to a subset of the available measurements and only a part of the plant’s model, but the diagnosers may exchange (intermediate) diagnostic results (cf. Fig. 2.3c). Distributed diagnosis is decentralised diagnosis with communication. Restrictions may apply to the quantity of the exchanged information and to the interconnection of the diagnosers. This approach is not further investigated.

Coordinated diagnosis is a slight modification of distributed diagnosis. Here, no direct communication between the local diagnosers is allowed. Instead they commu-
2. Elaboration of the Diagnostic Task

a) Centralised

b) Decentralised

c) Distributed

d) Coordinated

e) Hierarchical

Figure 2.3.: Information structure in diagnosis
cate through an additional component, the coordinator (cf. Fig. 2.3d). The coordinator has only structural knowledge of the plant and has no access to any measurements. The coordinated approach is addressed in Section 6 for the nondeterministic automata network only. Coordinated diagnosis is not a hierarchical approach. Instead all components, including the coordinator, work simultaneously on the task of identifying the system faults.

Hierarchical diagnosis divides the diagnostic process into two or more steps (multi-layer approach) in which an increasing amount of information is used to solve different diagnostic tasks of varying priority (cf. Fig. 2.3e). Generally, in the initial step very little information is used to perform fault detection. If a fault is detected additional information is used to identify the fault. Different diagnostic algorithms may be used for each step. Hierarchical diagnosis is not subject of this book, but the presented approaches can easily be adapted to such a setup.

2.3. Diagnosability and Output Indifference

Diagnosability is a system property which specifies the prospect of success of diagnosing a system fault. Since it is a system property it does not depend on the actual diagnostic algorithm nor the used system model.

Different notions of diagnosability have been proposed for different types of models of discrete-event systems. The most prominent definition of diagnosability has been proposed in [89,90] for nondeterministic discrete-event systems. It requires that every fault event leads to observations distinct enough to enable a unique identification of the fault with a finite delay. I.e. a system is diagnosable if it is guaranteed that every fault is identified within a fixed amount of time. This definition is relaxed to the so called I-diagnosability which only requires a guaranteed fault isolation within a fixed amount of time after the occurrence of certain indicator events associated with the fault cases. Different adaptions of these definitions to distributed nondeterministic discrete-event systems have been proposed [23,79,95].

A different notion of diagnosability has been proposed in [67,93] for clocked systems which requires the system to contain information about the occurrence of faults. I.e. it requires that the behaviours of any two fault models must not be identical. For clocked systems it is not sensible to require a certain detection time, in such cases when it is not possible to state any postulations on the system inputs (e.g. the system can stay idle for some time). The advantage of this definition is its applicability to stochastic systems. It is therefore the basis for all criteria for testing diagnosability in
this work and is explained in more detail in Sections 4.5 and 7.3.

**Definition 2.4:** A process is *not diagnosable* if there does not exists an input sequence \( V \) such that \((V, W) \notin B(f), \forall f \in \mathcal{N}_f\) holds.

In other words, a process is not diagnosable if it is not possible to exclude a fault \( f \) from the set of faults \( \mathcal{F} \) through inconsistency.

In addition to diagnosability the notion of output indifference is used. It is a system property which specifies if the diagnostic result differs from a mere simulation or in other words, if the output \( w \) contains information about the fault.

**Definition 2.5:** A process is *output indifferent* there does not exists an input sequence \( V \) such that for the behaviours \( B(f = i) \neq B(f = j) \) for all \( i, j \in \mathcal{N}_f \) with \( i \neq j \) holds.

In other words, a process is output indifferent if the influence of a fault \( f \) is not visible in the measurable process output \( w \) for all faults \( f \in \mathcal{N}_f \) at all times \( k \). Although, strictly speaking, a fault can be detected despite such an indifference, it is reasonable to state that a diagnostic approach as depicted in Fig. 2.2 is not necessary.

The diagnosability of the system is a necessary condition for a successful diagnostic task. However, the outcome of the diagnosis depends on additional factors. Restricting the information available for diagnosis does impair the prospect of success of diagnosing a system fault. Additionally, the applied diagnostic algorithm might not take advantage of the available information. Therefore, a second type of diagnosability – the A-diagnosability – is defined later in this book, which depends on the diagnostic algorithm and the applied information constraints, to be able to compare the different diagnostic concepts.

---

1. Please note, that for this and all further definitions it is assumed that all fault cases have been modelled *(closed world assumption)* as stated in Section 2.1.
2. Fault detection by simulation can for example be performed in (trivial) cases where a certain input will always cause a fault.
“He who loves practice without theory is like the sailor who boards ship without a rudder and compass and never knows where he may cast.”

Leonardo da Vinci (Italian Painter and Engineer, 1452–1519)

Introduction to Automata Theory

The automaton with inputs and outputs and the automaton network are the discrete-event models used throughout this book for fault diagnosis, state observation, and simulation. In this chapter they are introduced using two different formalisms, characteristic functions and relational data models, and some basic characteristics are discussed. Therefore, Section 3.1 starts with a short introduction to relational algebra. Afterwards the single automaton is defined in Section 3.2. This definition is extended to automata networks in Section 3.3.

3.1. Relational Data Representation

3.1.1. Relational Data Model

In this section the relational data model first published by E. Codd in 1970 is introduced formally. In relational data models the information is stored in relations where each variable is identified by its unique attribute. This representation is used later in this book to represent the automata network in a very compact way. For more detailed information on the relational data model confer [21, 22, 54, 56, 74].

Given is a set \( \mathcal{N}_d \) of \( n \) finite not necessarily distinct domains \( \mathcal{N}_{a_1}, \mathcal{N}_{a_2}, \ldots, \mathcal{N}_{a_n} \), where each domain may only hold atomic and unstructured values. Each domain is assigned to a signal \( q \) contained in the set of signals \( \mathbf{q} \), where each signal is identified by its unique attribute \( a_i \in \mathbf{a} \). That is, there exists a mapping \( \text{dom} : \mathbf{a} \rightarrow \mathcal{N}_a \).\(^1\) A

\(^1\)For notational convenience it will be assumed that every signal \( \bullet \) has its domain \( \mathcal{N}_\bullet \). Furtheron, the attribute for each signal is set to be identical with the symbol identifying the signal, e.g. \( \text{dom}(v^1) = \mathcal{N}_{v^1} \) holds, where \( v^1 \) is the attribute identifying the signal \( v^1 \). Attributes can be interpreted as the names of the signals.
relation $M(q)$ is then defined as a subset of the cartesian product $K(q)$ of these $n$ domains:

$$M(q) \subseteq K(q) = \mathcal{N}_{a_1} \times \mathcal{N}_{a_2} \times \ldots \times \mathcal{N}_{a_n},$$

(3.1)

Alternatively, the relation can be defined as

$$M(q) \subseteq \text{dom}(a_1) \times \text{dom}(a_2) \times \ldots \times \text{dom}(a_n)$$

One element $M \in M$ is called a tuple. The schema $\text{sch}(M) = \{a_1, a_2, \ldots, a_n\}$ collects all attributes of the relation. All attributes of a relation must be named unambiguously, but several relations may share an attribute. The values of an attribute $a$ of a specific relation $M$ are addressed by $M.a$.

Relations are usually depicted as tables where each signal marks a column and each row a tuple. The respective attributes are denoted on the first row (cf. Figure 3.1). The definition of relations results in three properties:

1. All tuples of a relation are distinct.
2. The sorting of the rows is insignificant.
3. The order of the attributes is insignificant.

![Figure 3.1: Relational model](image)

**Example 1:**

In this example a telephone-book will be modelled using the relational representation. Consider the following relation

$$\text{telephonebook} \subseteq \text{string}^{20} \times \text{string}^{20} \times \text{int}^{16},$$

where $\text{string}^{20}$ denotes a domain consisting out of all strings with a maximum length of 20
3.1. Relational Data Representation

letters and int\textsubscript{16} a domain consisting of the integer numbers up to 2\textsuperscript{16}. The schema of the relation is given by

\[
sch(\text{telephonebook}) = \{\text{name, street, telephone number}\},
\]

with

\[
dom(\text{name}) = \text{string20}, \quad dom(\text{street}) = \text{string20}, \quad dom(\text{telephone number}) = \text{int16}.
\]

Then the telephone-book can be depicted with three exemplary tuples as in Table 3.1. This relation can be realised in MySQL with the code stated below.\(^2\)

MySQL-Code (creating the relation of Example 1):

\[
> \text{CREATE TABLE telephonebook (name char(20),} \\
> \text{street char(20), telephone_number mediumint unsigned);}
\]

<table>
<thead>
<tr>
<th>Table 3.1.: Relational representation of a telephone book</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Smith</td>
</tr>
<tr>
<td>Doe</td>
</tr>
<tr>
<td>Miller</td>
</tr>
</tbody>
</table>

3.1.2. Relational Algebra

The operators of relational algebra can be applied to one or multiple relations and return a relation. In the following, three fundamental relational operations are described, where \(\tilde{q} \subseteq q\) is a subset of signals and \((M_1, M_2) = M\) a division of the tuple \(M \in K\) (cf. [43, 44]).

**Projection.** The *projection* of \(M(q)\) onto \(\tilde{q} \subseteq q\) is defined by

\[
\pi_{\tilde{q}}(M) := \{M_1 \in K(\tilde{q}) \mid \exists (M_1, M_2) \in M(q)\}.
\]

The projection removes all signals not included in \(\tilde{q}\), hence reduces the number of columns in the table. Please note that the notation of the operator denotes all remaining signals as the index of \(\pi\). The operator is clarified by the following example and MySQL-code.

\(^2\)A short introduction to SQL is given in Appendix D. Detailed language references can be found e.g. in [31, 70, 75].
Example 2:
Consider the relation \( L \) given in Figure 3.3 on Page 32. The projection

\[ L_2 = \pi_{z,f}(L) \]

results in the relation given in Table 3.2b. A straight-forward elimination of the attributes \( z', v, w \) would result in Table 3.2a, however, the definition of relations does not allow identical tuples, which therefore must be removed.

MySQL-code (Projection):
> /* syntax */
> SELECT DISTINCTROW attributes FROM relation;
> /* Example 2 */
> SELECT DISTINCTROW z, f FROM L;

Selection. A selection removes all tuples \( M \) from the relation \( M \) which do not satisfy a constraint \( C \)

\[ \sigma_C(M) := \{ M \in M \mid C(M) \} \]

The constraint \( C \) is a logical expression which evaluates to true or false. That is, the result of the selection operation contains all tuples of \( M \) for which the expression \( C \) evaluates to true. This operation reduces the number of rows in the table. The constraint \( C \) can be stated for any subset of \( \text{sch}(M) \). Please note that the constraint is denoted as an index of \( \sigma \) analogously to the projection operator. An example and the MySQL-code for this operation are given below.

Example 3:
Again, consider the relation \( L \) given in Figure 3.3. The selection

\[ L_3 = \sigma_{z=2,f=1}(L) \]

removes all tuples from \( L \) for which \( z = 2 \) and \( f = 1 \) do not hold. The result of the operation is given in Table 3.3.

MySQL-Code (Selection):
> SELECT * FROM relation WHERE condition /* syntax */;
> SELECT * FROM L WHERE z=2 AND f=1 /* Example 3*/;

Analogously to the abbreviated notation of stochastic variables (e.g. \( P(z) := P(z_p = z) \)) the selection of tuples with an attribute \( a \) of a specific, but not explicitly given value is denoted by \( \sigma_a \).
Natural join. The natural join combines two relations $M_1(q_1)$, $M_2(q_2)$ as follows:

$$M_1 \bowtie M_2 := \pi_{q_1 \cup q_2} \left( \sigma_{M_1(q_1 \cap q_2) = M_2(q_1 \cap q_2)} (M_1 \times M_2) \right).$$

From the cartesian product of $M_1$ and $M_2$ the tuples are selected which have identical values for $M_1$ and $M_2$ in $q_1 \cap q_2$. These tuples are then projected upon $q_1 \cup q_2$. Two examples and the MySQL-code for this operation are given below.

Example 4:
Consider the Tables 3.4a and 3.4b. The cartesian product of both tables is given in Table 3.4c and their natural join results in the Table 3.4d. For the natural join the values of the signal $s$, which is common to both tables, are compared. All rows in which the two occurrences of $s$ have not identical values are removed from the cartesian product. Finally, the superfluous column of $s$ is removed. □

Example 5:
Consider the Tables 3.5a and 3.5b. They are identical to the tables of the previous example, with the exception that they do not share the signal $s$. Therefore, the natural join of the two tables as given in Table 3.5c is identical with their cartesian product as no rows are deleted. □

| MySQL-Code (Natural join):
> SELECT * FROM T1 NATURAL JOIN T2;

The natural join operation fulfils the commutative and associative laws. This means that multiple tables can be joined by applying the operator repeatedly. For the result it is insignificant in which order the tables are processed, however the size of the intermediate results does depend on the ordering.

3.2. Single Automaton

3.2.1. Overview

The automaton is a representation of a discrete-event system. In general, automata are defined by a set of discrete automaton states, some alphabet which the automaton uses to communicate with its environment, and some formalism which defines the automaton’s transitions. In addition to that, the initialised automaton has information about its initial state. The expressions automaton and state machine are used synonymously. An automaton with a finite state set is called finite automaton or finite state machine [48, 62].
3. Introduction to Automata Theory

Table 3.2.: Relation $L_2$ from Example 2

<table>
<thead>
<tr>
<th>$z$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$z$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3.3.: Relation $L_3$ from Example 3

<table>
<thead>
<tr>
<th>$z'$</th>
<th>$w$</th>
<th>$z$</th>
<th>$v$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.4.: Tables for demonstrating the natural join operation

<table>
<thead>
<tr>
<th>$f_1$</th>
<th>$s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$f_2$</th>
<th>$s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$f_1$</th>
<th>$s$</th>
<th>$f_2$</th>
<th>$s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
3.2. Single Automaton

Table 3.5.: Tables for demonstrating the natural join operation

<table>
<thead>
<tr>
<th></th>
<th>(T_1)</th>
<th></th>
<th>(T_2)</th>
<th></th>
<th>(T_1 \bowtie T_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(f_1)</td>
<td>(s_1)</td>
<td>(f_2)</td>
<td>(s_2)</td>
<td>(f_1)</td>
</tr>
<tr>
<td>a)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>c)</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

In this work automata are not discussed in their most general form. Instead, the representation has been chosen to suit the considered task, namely diagnosis of discrete-event systems. For this, it is distinguished between the input, output, and fault alphabet, whereby the fault is interpreted as an unmeasurable input signal (cf. Figure 3.2a). The automaton’s state transitions are synchronised on a fixed clock pulse symbolised by the metronome in the figure. That is, the time-line is discretised into equally spaced intervals of the length \(T\) and the passing of time determines the progress of the automaton’s evolution. Such an automaton is also called time-driven automaton or discrete-time automaton [20]. The integer number \(k\) counts the number of elapsed time intervals since a given reference point \((k = 0)\). The time point of the \(k\)-th state transition is denoted by \(t_k\).

In this paper two different automata representations are used interchangeably. In Sections 3.2.2 to 3.2.10 the single automaton is covered using a representation where the automaton’s transitions are defined by a characteristic function \(L\) which maps the automaton’s signal domains onto the sets \([0, 1]\) or \(\{0, 1\}\) (cf. [91]). In Section 3.2.11 the automaton is specified using relational algebra where all possible transitions are collected as tuples in the relation \(L\). Functions and relations are equivalent representations of a discrete-event system. Whereas the representation using functions can be found frequently in literature the relational representation is novel. The advantages and disadvantages of both representations are discussed in Section 3.2.12.
3. Introduction to Automata Theory

a) Fault as an unmeasurable input signal

b) Fault interpreted as a state

Figure 3.2.: Different interpretations of the fault signal

### 3.2.2. Deterministic Automaton

The deterministic automaton (DA) is a representation of a *discrete-valued deterministic process*. It is given by the tuple

\[ \mathcal{A}^d = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f, L^d, z(0), f(0)) \]  

with the finite sets

\[ \mathcal{N}_z = \{1, \ldots, N\} \quad \text{of automaton states}, \]
\[ \mathcal{N}_v = \{1, \ldots, M\} \quad \text{of input symbols}, \]
\[ \mathcal{N}_w = \{1, \ldots, R\} \quad \text{of output symbols}, \]
\[ \mathcal{N}_f = \{1, \ldots, S\} \quad \text{of fault symbols}. \]

The state of the automaton is denoted by \( z(k) \in \mathcal{N}_z \), the input by \( v(k) \in \mathcal{N}_v \), the output by \( w(k) \in \mathcal{N}_w \), and the fault by \( f(k) \in \mathcal{N}_f \). The initial state and fault is given by \( z(0) \) or \( f(0) \), respectively. The dynamics of the automaton are defined by the characteristic function

\[ L^d : \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \times \mathcal{N}_f \rightarrow \{0, 1\} \]

where \( L^d(z(k + 1), w(k), z(k), v(k), f(k)) = 1 \) iff for the input \( v(k) \) the state changes from \( z(k) \) to \( z(k + 1) \) and the automaton produces the output \( w(k) \) under the influence of fault \( f(k) \). Otherwise \( L^d(z(k + 1), w(k), z(k), v(k), f(k)) = 0 \) holds [91]. Whenever the actual time \( k \) is not of interest the relation will be abbreviated by \( L^d(z', w, z, v, f) \). Since the automaton is deterministic

\[ \sum_{z' = 1}^{N} \sum_{w = 1}^{R} L^d(z', w, z, v, f) = 1, \quad \forall z \in \mathcal{N}_z, v \in \mathcal{N}_v, f \in \mathcal{N}_f \]  

The superscript \( d \) indicates that the automaton is deterministic.
3.2. Single Automaton

holds. That is, for every combination of the current state, input, and fault exactly one transition is possible.

The behaviour of a DA is given by

\[ B(f) = \{(V(0\ldots k_h), W(0\ldots k_h)) | \exists Z(0\ldots k_h + 1) \} \]

\[ \mathcal{L}^d(z(k+1), w(k), z(k), v(k), f(k)) = 1 \text{ for } k = 0, 1, \ldots, k_h - 1 \]

and \( \text{Poss}(z_p(0) = z(0), f_p(0) = f(0)) = 1, \forall k_h \in \mathbb{N} \}. \] (3.6)

whereby \( Z(0\ldots k_h + 1) \) abbreviates the state sequence

\[ (z(0), z(1), \ldots, z(k_h)). \]

That is, the behaviour includes all I/O-sequences of arbitrary length for which the automaton can generate a state sequence for a given initial state and fault [12].

3.2.3. Nondeterministic Automaton

The nondeterministic automaton (NA) is a representation of a discrete-valued nondeterministic process. As opposed to the deterministic automaton the movement of the nondeterministic automaton

\[ A^n = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f, \mathcal{L}^n, z(0), f(0)) \] (3.7)

is not unambiguous. That is, for every current state, input, and fault combination multiple transitions are possible. The sets \( \mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f \) are defined as in the deterministic case. The characteristic function

\[ \mathcal{L}^n : \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \times \mathcal{N}_f \rightarrow \{0, 1\}, \]

\[ \mathcal{L}^n(z(k+1), w(k), z(k), v(k), f(k)) = \text{Poss}(z(k+1), w(k), z(k), v(k), f(k)) \] (3.9)

assumes the value \( \mathcal{L}^n(z', w, z, v, f) = 1 \) for all possible transitions. Otherwise \( \mathcal{L}^n(z', w, z, v, f) = 0 \) holds. The automaton is assumed to be live, meaning at least one transition is possible for every state, input, and fault combination:

\[ \bigvee_{z' = 1}^{N} \bigvee_{w = 1}^{R} \mathcal{L}^n(z', w, z, v, f) = 1, \forall z \in \mathcal{N}_z, v \in \mathcal{N}_v, f \in \mathcal{N}_f. \] (3.10)

\(^4\)The superscript \( n \) indicates that the automaton is nondeterministic.
The symbol $\lor$ denotes the boolean OR-operation. The behaviour of the nondeterministic automaton can be given analogously to the deterministic automaton.

### 3.2.4. Stochastic Automaton

The stochastic automaton (SA) is a representation of a *discrete-valued stochastic process*. With the sets $\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f$ defined as above it is given by the tuple\(^5\)

$$\mathcal{A}^s = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f, \mathcal{L}^s, \mathbf{p}(z(0)), \mathbf{p}(f(0))).$$  \hspace{1cm} (3.11)

The characteristic function\(^6\) $\mathcal{L}^s$ provides not only the information about which transitions are possible, but gives the *conditional probability* $P$ for the respective state transitions [19]:

$$\mathcal{L}^s : \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \times \mathcal{N}_f \rightarrow [0, 1]$$  \hspace{1cm} (3.12)

$$\mathcal{L}^s(z(k+1), w(k) \mid z(k), v(k), f(k)) = P(z_p(k+1) = z(k+1), w_p(k) = w(k) \mid z_p(k) = z(k), v_p(k) = v(k), f_p(k) = f(k)).$$  \hspace{1cm} (3.13)

Whenever possible, the function will be abbreviated by $\mathcal{L}^s(z', w \mid z, v, f)$. The sum of the probabilities of all outcomes of an experiment has to be equal to one. Therefore, for the characteristic function

$$\sum_{z'=1}^N \sum_{w=1}^R \mathcal{L}^s(z', w \mid z, v, f) = 1, \forall z \in \mathcal{N}_z, v \in \mathcal{N}_v, f \in \mathcal{N}_f$$  \hspace{1cm} (3.14)

holds. The initial state and fault are given as discrete probability distributions

$$\mathbf{p}(z(0)) = \begin{pmatrix} P(z_p(0) = 1) \\ P(z_p(0) = 2) \\ \vdots \\ P(z_p(0) = N) \end{pmatrix}, \quad \mathbf{p}(f(0)) = \begin{pmatrix} P(f_p(0) = 1) \\ P(f_p(0) = 2) \\ \vdots \\ P(f_p(0) = S) \end{pmatrix}.$$

---

\(^5\)The superscript $s$ indicates that the automaton is stochastic.

\(^6\)The term characteristic function of the stochastic automaton is used analogously to the deterministic and nondeterministic automaton for reasons of consistency. It is not to be confused with its usage in probabilistic theory namely as the Fourier transformation of the distribution function of a stochastic variable.
3.2. Single Automaton

The behaviour of a SA is given by

\[ B(f) = \{(V(0\ldots k_h), W(0\ldots k_h)) \mid \exists Z(0\ldots k_h + 1) \} \]

\[ L^d(z(k + 1), w(k), z(k), v(k), f(k)) > 0 \text{ for } k = 0, 1, \ldots, k_h - 1 \]

and \( P(z_p(0) = z(0), f_p(0) = f(0)) > 0, \forall k_h \in \mathbb{N} \). \ (3.15)

The behaviour includes all I/O-sequences of arbitrary length for which the automaton can generate at least one possible state sequence for a given initial state and fault.

The nondeterministic automaton gained by applying

\[ L^s(z', w|z, v, f) > 0 \Rightarrow L^n(z', w, z, v, f) = 1 \quad \text{and} \]

\[ L^s(z', w|z, v, f) = 0 \Rightarrow L^n(z', w, z, v, f) = 0 \]

is said to be the embedded nondeterministic automaton of the SA.

3.2.5. Autonomous Automaton

A simplified form of the automaton defined in the previous sections is the autonomous automaton, which has neither input nor output signals and therefore does not communicate with its environment. This deterministic, nondeterministic and stochastic autonomous automaton is given by

\[ A^{d,n} = (\mathcal{N}_z, L^{d,n}, z(0)), \quad A^s = (\mathcal{N}_z, L^s, P(z(0))) \]

with the characteristic function

\[ L^{n,d} : \mathcal{N}_z \times \mathcal{N}_z \rightarrow \{0, 1\}, \quad L^{n,d}(z(k + 1), z(k)) = \text{Poss}(z(k + 1), z(k)) \]

\[ L^s : \mathcal{N}_z \times \mathcal{N}_z \rightarrow [0, 1], \quad L^s(z(k + 1)|z(k)) = P(z_p(k + 1) = z(k + 1)|z_p(k) = z(k)). \]

For the characteristic function the relations hold:

\[ \sum_{z'}^N L^d(z', z) = 1, \forall z \in \mathcal{N}_z \] \quad (3.16)

\[ \sum_{z'}^N L^n(z', z) = 1, \forall z \in \mathcal{N}_z \] \quad (3.17)

\[ \sum_{z'}^N L^s(z'|z) = 1, \forall z \in \mathcal{N}_z. \] \quad (3.18)
3. Introduction to Automata Theory

3.2.6. State Transition and Output Function

The state transition function and the output function are extracts of the characteristic function, i.e. they contain only partial information about an automaton’s movement. The state transition function $G$ contains information about the next state $z'$, the output function $H$ about the current output $w$. For the stochastic automaton they are given by

$$
G^s: \mathcal{N}_z \times \mathcal{N}_v \times \mathcal{N}_f \rightarrow [0, 1] \\
G^s(z(k + 1) | z(k), v(k), f(k)) = P(z_p(k + 1) = z(k + 1) | z_p(k) = z(k), v_p(k) = v(k), f_p(k) = f(k))
$$

and

$$
H^s: \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \times \mathcal{N}_f \rightarrow [0, 1] \\
H^s(w(k) | z(k), v(k), f(k)) = P(w_p(k) = w(k) | z_p(k) = z(k), v_p(k) = v(k), f_p(k) = f(k)).
$$

The functions $G$ and $H$ can be derived from the characteristic function $L$ by

$$
G^s(z'|z, v, f) = \sum_{w=1}^{R} L^s(z', w|z, v, f) \quad (3.19) \\
H^s(w|z, v, f) = \sum_{z'=1}^{N} L^s(z', w|z, v, f). \quad (3.20)
$$

The state transition and the output function for the deterministic and nondeterministic case are defined analogously using

$$
G^{d,n}(z', z, v, f) = \bigvee_{w=1}^{R} L^{d,n}(z', w, z, v, f) \quad (3.21) \\
H^{d,n}(w, z, v, f) = \bigvee_{z'=1}^{N} L^{d,n}(z', w, z, v, f). \quad (3.22)
$$

3.2.7. Modelling the Dynamics of the Fault

In the previous sections the fault $f(k)$ has been interpreted as an unmeasurable input signal as depicted in Figure 3.2a and the process generating the fault signal has been unaccounted for. If the dynamics of the fault is of interest the characteristic function
3.2. Single Automaton

can be extended to include a dynamic fault model (cf. Figure 3.2b). For the stochastic automaton the extended characteristic function amounts to

\[ L^s : \mathcal{N}_z \times \mathcal{N}_f \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \times \mathcal{N}_f \rightarrow [0, 1] \] (3.23)

\[ L^s(z(k+1), f(k+1), w(k) | z(k), v(k), f(k)) = \\
\quad \quad P(z_p(k+1) = z(k+1), v_p(k+1) = f(k+1), \\
\quad \quad w_p(k) = w(k) | z_p(k) = z(k), v_p(k) = v(k), f_p(k) = f(k)). \] (3.24)

Under the assumption that the faults occur randomly a fault transition function \( F \) with

\[ F^s : \mathcal{N}_f \times \mathcal{N}_f \rightarrow [0, 1] \]

\[ F^s(f(k+1) | f(k)) = P(f(k+1) | f(k)) \]

can be introduced to model non-constant faults:

\[ L^s(z', f', w | z, v, f) = L^s(z', w | z, v, f)F^s(f' | f). \] (3.25)

For the deterministic and nondeterministic automaton the dynamic fault model is denoted analogously.

Dynamic fault models can be used in diagnosis to detect and isolate time-variant faults as for example increasing or strain dependent faults. For notational convenience dynamic faults will not be considered in the remainder of this book bearing in mind that all presented methods and algorithms can easily be extended to dynamic faults using the above notation. The latter approach given in Equation (3.25) is implemented in the MATLAB-toolbox DiamondQ and has been applied successfully to a number of practical applications, e.g. for diagnosing the air-path of a diesel engine as described in Chapter 12 (cf. also [98, 99, 112]).

3.2.8. Automaton Graph and Automaton Table

An automaton can be represented equivalently by an automaton graph or an automaton table. An automaton graph is a graphical representation of an automaton, where the states are visualised by nodes and the transitions by directed arcs which connect the nodes. For the considered automaton with fault the arcs are labelled with the input/output/fault-triples for the respective transitions. Additionally, in case of a stochastic automaton the respective probabilities are denoted along the arcs. Only possible transitions (\( L^d = 1, L^n = 1, \) or \( L^s > 0 \)) are depicted. An example for an automaton graph of a nondeterministic automaton is given in Figure 3.3a.
3. Introduction to Automata Theory

The listing of all possible transitions in tabular form is called automaton table. Thereby, the columns of the table list all variables of the characteristic function. Since only possible transitions are included, the value of the characteristic function is neglected in the deterministic and nondeterministic case. For the stochastic automaton an additional column lists the probability for the respective transitions. The ordering of the rows is insignificant. An example for an automaton table is given in Figure 3.3b. This tabular notation suggests the use of relational algebra and database theory to describe the automata model. This is described in Section 3.2.11.

<table>
<thead>
<tr>
<th>z'</th>
<th>w</th>
<th>z</th>
<th>v</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3.3.: Exemplary automaton graph and table for a nondeterministic automaton

### 3.2.9. Markov Property

The stochastic automaton as defined in Section 3.2.4 possesses the Markov property [19]. That is, the automaton’s future state and output only depend on the current state, input and fault and are independent of any previous signal values:

\[
P(z_p(k+1) = z(k+1), w_p(k) = w(k) \mid z_p(0) = z(0), \ldots, z_p(k) = z(k),
\]

\[
v_p(0) = v(0), \ldots, v_p(k) = v(k), f_p(0) = f(0), \ldots, f_p(k) = f(k)) =
\]

\[
P(z_p(k+1) = z(k+1), w_p(k) = w(k) \mid z_p(k) = z(k), v_p(k) = v(k), f_p(k) = f(k)).
\]

Such a system is also called a Markov chain. Additionally, for the stochastic automaton the relation

\[
P(z_p(k+1) = z', w_p(k) = w \mid z_p(k) = z, v_p(k) = v, f_p(k) = f) =
\]

\[
P(z_p(1) = z', w_p(0) = w \mid z_p(0) = z, v_p(0) = v, f_p(0) = f).
\]
holds, which means that the automaton’s movement does not depend on the absolute time point \( k \). Markov chains fulfilling the above relation are called \textit{homogeneous}.

Strictly speaking, the Markov property is only defined for stochastic processes. However, for the DA and NA analog relations hold:

\[
\text{Poss}(z(k+1), w(k), z(0), \ldots, z(k), v(0), \ldots, v(k), f(0), \ldots, f(k)) = \\
\text{Poss}(z(k+1), w(k), z(k), v(k), f(k))
\]

\[
\text{Poss}(z_p(k+1) = z', w_p(k) = w, z_p(k) = z, v_p(k) = v, f_p(k) = f) = \\
\text{Poss}(z_p(1) = z', w_p(0) = w, z_p(0) = z, v_p(0) = v, f_p(0) = f).
\]

That is, the future state and output of the DA and NA only depend on the current state, input and fault and their movement does not depend on the absolute time point \( k \). In the remainder of this book this will also be referred to as the Markov property.

### 3.2.10. Special Classes of Automata

Two special classes of automata are of interest in this book, namely the Mealy and the Moore automaton.

**Definition 3.1 (Mealy automaton):** An automaton \( \mathcal{A} \) is a Mealy automaton if its characteristic function \( L \) can be decomposed into a state transition function \( G \) and output function \( H \) without losing any information [19, 66, 71]:

\[
\text{Poss}(z', w, z, v, f) = \text{Poss}(z', z, v, f) \wedge \text{Poss}(w, z, v, f) \text{ or } \\
P(z', w|z, v, f) = P(z'|z, v, f) \cdot P(w|z, v, f), \quad \forall z', z \in \mathcal{N}_z, v \in \mathcal{N}_v, w \in \mathcal{N}_w, f \in \mathcal{N}_f.
\]

**Definition 3.2 (Moore automaton):** An automaton \( \mathcal{A} \) is a Moore automaton if its output function \( H \) does not depend on the input \( v \) [66, 73]:

\[
\text{Poss}(w, z, v, f) = \text{Poss}(w, z, f) \text{ or analogously } \\
P(w|z, v, f) = P(w|z, f), \quad \forall z', z \in \mathcal{N}_z, v \in \mathcal{N}_v, w \in \mathcal{N}_w, f \in \mathcal{N}_f.
\]

Because the fault signal \( f \) is included in the above relations the definition is in contradiction to some books, which state that the output function of a Moore automaton may only depend on the current state \( z \). Please note, that in some textbooks as e.g. in [19] it is stated wrongly, that the output function of the Moore automaton depends on the future state \( z' \).
3. Introduction to Automata Theory

3.2.11. Relational Representation of Single Automata

An automaton can easily be represented using a relational data model. Whereas such a representation might have no apparent advantage when applied to a single automaton, the description of an automaton network, introduced in the following sections, becomes more concise.

The relation of the nondeterministic automaton is given by

\[ L^n \subseteq N_z \times N_w \times N_z \times N_v \times N_f \]

with the schema

\[ \text{sch}(L^n) = \{ z(k+1), w(k), z(k), v(k), f(k) \} \]

and the domains

\[
\begin{align*}
\text{dom}(z(k+1)) &= N_z, \\
\text{dom}(w(k)) &= N_w, \\
\text{dom}(z(k)) &= N_z, \\
\text{dom}(v(k)) &= N_v, \\
\text{dom}(f(k)) &= N_f.
\end{align*}
\]

Each tuple \( L \in L^n \) describes a possible transition from state \( z(k) \) to \( z(k+1) \) for the given input \( v(k) \) while producing the output \( w(k) \) under the influence of fault \( f(k) \). An example relation is depicted in Table 3.6a. Note, that the sorting of the attributes in the schema is insignificant.

Analogously, the relation of a stochastic automaton is given by

\[ L^s \subseteq N_z \times N_w \times N_z \times N_v \times N_f \times (0, 1] \]

with the schema

\[ \text{sch}(L^s) = \{ z(k+1), w(k), z(k), v(k), f(k), \text{Prob} \} \]

with \( \text{dom(Prob)} = (0, 1] \) and the remaining domains defined as above. For the attribute \( \text{Prob} \) the following holds:

\[ \pi_{\text{Prob}} \sigma_{z(k+1), w(k), z(k), v(k), f(k)}(L^s) := P(z(k+1), w(k)|z(k), v(k), f(k)). \]

Only possible transitions, i.e. tuples with \( \text{Prob} > 0 \), are contained in \( L^s \). An example relation is depicted in Table 3.6b.
3.2.12. Comparison of the Functional and the Relational Representations

The main advantage of representing the automaton using characteristic functions is the ability to use algebraic calculations in the simulation, observation, and diagnostic algorithms. This is especially apparent in the case of stochastic automata where calculations like \( L^3(z' = 1, w = 2 | z = 1, v = 3) \) and \( L^3(z' = 2, w = 2 | z = 1, v = 3) \) can easily be performed using mathematical tools like MATLAB®. Problem solutions and proofs can be given as closed algebraic expression. Its disadvantage is a more complicated automata network description and the lack of advanced search and data handling procedures. Also, normal numerical tools have only very limited abilities for handling distributed information as found in networks. The advantage of the relational representation is an easy and structured automata network representation and the possibility to use operations like selection, projection, and join provided by modern data-bases as MySQL®. Its disadvantage lies in the restriction to set operations.

3.3. Synchronous Automata Network

Interconnecting several automata as shown in Figure 3.4 results in an automata network. All networks in this paper are assumed to be homogeneous, i.e. all automata of the network must be of the same kind (deterministic, nondeterministic or stochastic). The transitions of the automata of the network are synchronised on the same pulse, which is indicated by the metronome in the same figure, i.e. all state transitions of all automata happen simultaneously. Although, in principle this pulse could be generated in different kind of ways, e.g. by an input event, it is assumed that the pulse is generated from an external source.

Synchronised signals in combination with feedback connections may result in con-
3. Introduction to Automata Theory

Conflicts. The feedback problem is investigated in depth in Chapter 11. In the remaining chapters it will be assumed that all networks are free of conflicts [115, 125].

The network is formally introduced in Sections 3.3.1 to 3.3.2 using the relational data model and in Section 3.3.3 using characteristic functions. In Section 3.3.4 it is investigated how a network can be composed to a single automaton using composition rules. The complexity of the monolithic and distributed system descriptions are compared in Section 3.3.5.

![Example network](image)

Figure 3.4.: Example network

3.3.1. Generalisation of the Single Automaton

The smallest possible automata network consists of only one automaton with multiple input, output, and fault signals (cf. Fig. 3.5a). In this section the notation of such a network is given which will be extended to a “genuine” network consisting of several automata in the next section.

The nondeterministic automaton with multiple input, output, and fault signals is given by

\[ A^n = (N_z, N_v, N_w, N_f, L^n, z(0), f(0)) \]  

(3.26)

with the finite set \( N_z \) of automaton states with \( z \in N_z \). The sets \( N_v = \{N_{v1}, \ldots, N_{v\mu}\} \) and \( N_w = \{N_{w1}, \ldots, N_{w\rho}\} \) contain the domains of the automaton’s input and output with \( v = \{v^1, \ldots, v^\mu\} \in N_v \) and \( w = \{w^1, \ldots, w^\rho\} \in N_w \). The set \( N_f = \{N_{f1}, \ldots, N_{f\sigma}\} \) contains the domains of the fault signals with \( f = \{f^1, \ldots, f^\sigma\} \in N_f \). In detail, the notation \( v \in N_v \) means that the set \( v \) contains one element of each domain in \( N_v \). The
3.3. Synchronous Automata Network

initial state and fault are given by \( z(0) \) and \( f(0) \). The dynamics of the automaton are defined by the relation

\[
\mathcal{L}^n \subseteq \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \times \mathcal{N}_f
\]

\[
= \mathcal{N}_z \times \mathcal{N}_{w_1} \times \cdots \times \mathcal{N}_{w_\rho} \times \mathcal{N}_z \times \mathcal{N}_{v_1} \times \cdots \times \mathcal{N}_{v_\mu} \times \mathcal{N}_{f_1} \times \cdots \times \mathcal{N}_{f_\sigma}
\]  

(3.27)

with the schema

\[
sch(\mathcal{L}^n) = \{z(k+1), w(k), z(k), v(k), f(k)\}
\]

\[
= \{z(k+1), w^1(k), \ldots, w^\rho(k), z(k), v^1(k), \ldots, v^\mu(k), f^1(k), \ldots, f^\sigma(k)\}
\]

(3.28)

Each tuple \( L \in \mathcal{L}^n \) describes a possible transition from state \( z(k) \) to \( z(k+1) \) for the given inputs \( v(k) \) while producing the outputs \( w(k) \) under the influence of faults \( f(k) \). As defined in Section 3.1 every one of these domains is assigned uniquely to a specific signal by the maps

\[
\text{dom}_v : v \rightarrow \mathcal{N}_v,
\]

\[
\text{dom}_w : w \rightarrow \mathcal{N}_w,
\]

\[
\text{dom}_f : f \rightarrow \mathcal{N}_f.
\]

The stochastic automaton is extended analogously to

\[
\mathcal{A}^s = (\mathcal{N}_z, \mathcal{N}_y, \mathcal{N}_w, \mathcal{N}_f, \mathcal{L}^s, p(z(0)), p(f(0)))
\]

(3.29)

with the relation

\[
\mathcal{L}^s \subseteq \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \times \mathcal{N}_f \times (0, 1]
\]

(3.29)

\[
sch(\mathcal{L}^s) = \{z(k+1), w(k), z(k), v(k), f(k), \text{Prob}\}
\]

(3.30)

with \( \text{dom}(\text{Prob}) = (0, 1] \) and \( \text{Prob} = P(z(k+1), w(k) | z(k), v(k), f(k)) \).

![Figure 3.5. Automata and network structure](image-url)
3.3.2. Network Description Using the Relational data model

An automata network

\[ \mathcal{A}_\gamma = (\mathcal{A}_1, \ldots, \mathcal{A}_\gamma, \mathbf{v}, \mathbf{w}, \mathbf{s}, \mathbf{f}) \]  

(3.31)

consists of \( \gamma \) automata connected through the signals identified by the attributes in the sets \( \mathbf{v}, \mathbf{w}, \mathbf{s}, \mathbf{f} \). These automata are also called network components. The attributes of the network’s signals are collected in

\[ \mathbf{v} = \{v^1, \ldots, v^\mu\} \quad \text{input signals} \]

\[ \mathbf{w} = \{w^1, \ldots, w^\rho\} \quad \text{output signals} \]

\[ \mathbf{s} = \{s^1, \ldots, s^\kappa\} \quad \text{coupling signals} \]

\[ \mathbf{f} = \{f^1, \ldots, f^\sigma\} \quad \text{fault signals} \]

(cf. Fig. 3.5b). Coupling signals

\[ s^i \in \mathcal{N}_s = \{1, \ldots, Q^i\}, i = 1, \ldots, \kappa \quad \text{with} \quad \text{dom}_s: \mathbf{s} \rightarrow \mathcal{N}_s \]

are internal network signals which are not measurable and connect at least two of the network’s automata. With the notational convention stated above, the domains of the signals are given as \( v^1 \in \mathcal{N}_{v^1}, w^1 \in \mathcal{N}_{w^1}, s^1 \in \mathcal{N}_{s^1}, f^1 \in \mathcal{N}_{f^1} \) and so on. The network state consists of the states of all \( \gamma \) automata of the network: \( \mathbf{z} = \{z_1, z_2, \ldots, z_\gamma\} \).

In case of a nondeterministic automata network (NAN) the \( i \)-th automaton \((i \in \{1, \ldots, \gamma\})\) of the network is denoted by

\[ \mathcal{A}_i^n = (\mathcal{N}_{z_i^n}, \mathcal{N}_{v_i^n}, \mathcal{N}_{w_i^n}, \mathcal{N}_{f_i^n}, \mathbf{L}_i^n, \mathbf{z}_i(0), \mathbf{f}_i(0)) \]  

(3.32)

with

\[ \mathbf{L}_i^n \subseteq \mathcal{N}_{z_i^n} \times \mathcal{N}_{v_i^n} \times \mathcal{N}_{w_i^n} \times \mathcal{N}_{f_i^n} \]

\[ \text{sch}(\mathbf{L}_i^n) = \{z_i^n, w_i, z_i, v_i, f_i\} \]

Its attributes are collected in the sets

\[ \mathbf{v}_i \in \mathcal{N}_{v_i} \quad \text{with} \quad \mathbf{v}_i \subseteq \mathbf{v} \cup \mathbf{w} \cup \mathbf{s}, \]

\[ \mathbf{w}_i \in \mathcal{N}_{w_i} \quad \text{with} \quad \mathbf{w}_i \subseteq \mathbf{w} \cup \mathbf{s}, \]

\[ \mathbf{f}_i \in \mathcal{N}_{f_i} \quad \text{with} \quad \mathbf{f}_i \subseteq \mathbf{f}, \]

\[ \mathbf{z}_i \in \mathcal{N}_{z_i}. \]
As every automaton has only one state signal, the convention $z_i = z'$ will be used in the remaining sections. The automata in case of a stochastic automata network (SAN) are defined analogously.

Two automata $A_i, A_j$ are coupled if they share a signal with the same attribute:

$$(v_i \cap w_j) \cup (w_i \cap v_j) \cup (v_i \cap v_j) \neq \emptyset.$$ 

The benefit of the representation using relational data models is that the network topology is implicitly defined by the sets $v_i, w_i$ of all automata of the network by means of the unique signal attributes. This is illustrated in the example below.

Example 6:
The automata network $A_{ij} = (A_1, A_2, A_3, v, w, s, f)$ in Fig. 3.4 on page 36 consists of three automata and has two input signals $v = \{v^1, v^2\}$, one output signal $w = \{w^1\}$, three coupling signals $s = \{s^1, s^2, s^3\}$, and three fault signals $f = \{f^1, f^2, f^3\}$. Automaton $A_1$ is given by

$$A_1 = (N_{z_1}, N_{v_1}, N_{w_1}, N_{f_1}, L_1, z_1(0), f_1(0))$$

with the sets $v_1 = \{v^1, s^2\}$, $w_1 = \{w^1, s^1\}$, and $f_1 = \{f^1\}$ with the domains $N_{v_1} = \{N_{v^1}, N_{s^2}\}$, $N_{w_1} = \{N_{w^1}, N_{s^1}\}$, $N_{f_1} = \{N_{f^1}\}$. $A_2$ is given by

$$A_2 = (N_{z_2}, N_{v_2}, N_{w_2}, N_{f_2}, L_2, z_2(0), f_2(0))$$

with $v_2 = \{s^1\}$, $w_2 = \{s^3\}$, and $f_2 = \{f^2\}$ and the domains $N_{v_2} = \{N_{s^1}\}$, $N_{w_2} = \{N_{s^3}\}$, and $N_{f_2} = \{N_{f^2}\}$. $A_3$ is given by

$$A_3 = (N_{z_3}, N_{v_3}, N_{w_3}, N_{f_3}, L_3, z_3(0), f_3(0))$$

with $v_3 = \{v^2, s^3\}$, $w_3 = \{s^2\}$, $f_3 = \{f^3\}$, $N_{v_3} = \{N_{v^2}, N_{s^3}\}$, $N_{w_3} = \{N_{s^2}\}$, and $N_{f_3} = \{N_{f^3}\}$. The automata $A_1$ and $A_2$ are coupled because they share the signal with the attribute $s^1$. $A_1$ and $A_3$ share $s^2$, and $A_2$ and $A_3$ share $s^3$.

Note, that the sets of component input and output signals $v_i, w_i$ may also include coupling signals $s$ as shown in Figure 3.4. To distinguish between component signals which are network I/O-signals and internal coupling signals the set of local coupling signals is given as

$$s_i = (v_i(k) \cap s(k)) \cup (w_i(k) \cap s(k)).$$

The remaining signals are network I/O-signals and are denoted by

$$\tilde{v}_i(k) = v_i(k) \cap v(k) \quad \text{and} \quad \tilde{w}_i(k) = w_i(k) \cap w(k).$$
3. Introduction to Automata Theory

3.3.3. Network Description Using Characteristic Functions

The network description using characteristic functions is similar to the description introduced in the previous sections. The network is identically defined by the set

\[ \mathcal{A}_x = (\mathcal{A}_1, \ldots, \mathcal{A}_\gamma, \textbf{v}, \textbf{w}, \textbf{s}, \textbf{f}). \]

The network topology is also defined implicitly by means of the unique signal names and the network signals are collected in the sets \( \textbf{v}, \textbf{w}, \textbf{s}, \) and \( \textbf{f}. \) The difference is in the description of the components, more specifically their characteristic functions.

In case of nondeterministic networks the \( i \)-th automaton \( (i \in \{1, \ldots, \gamma\}) \) of the network is denoted by

\[ A^n_i = (\mathcal{N}_{i'_i}, \mathcal{N}_{w_i}, \mathcal{N}_{f_i}, L^n_i, z_i(0), f_i(0)) \]  \hspace{1cm} (3.33)

with the characteristic function

\[ L^n_i (z'_i, w_i, z_i, v_i, f_i) = \text{Poss}(z'_i, w_i, z_i, v_i, f_i). \]

Analogously, the \( i \)-th automaton of a stochastic automaton network is given by

\[ A^n_i = (\mathcal{N}_{i'_i}, \mathcal{N}_{w_i}, \mathcal{N}_{f_i}, L^n_i, p(z_i(0)), p(f_i(0))) \]  \hspace{1cm} (3.34)

with

\[ L^n_i (z'_i, w_i | z_i, v_i, f_i) = P(z'_i, w_i | z_i, v_i, f_i). \]

3.3.4. Composition Rules

Composing an automata network means to create a single automaton with the same input/output behaviour and set of possible state sequences as the original network. The principal approach to composition can be found e.g. in [62, 66]. The rules given in this section are based on the results in [35, 45, 93]. In the following, the rules for composing a network are first stated for the three basic connection types parallel, serial, and feedback as depicted in Figure 3.6. Then these rules will be combined to form the general composition rule. The composition rules are stated for both network representations, namely using characteristic functions and relational algebra.

**Definition 3.3 (Equivalence):** Two discrete-event systems are **equivalent** if their behaviours and sets of possible state sequences are identical.
3.3. Synchronous Automata Network

![Diagrams of automata connections]

Figure 3.6.: Basic connection types

Composition of Networks Described by Characteristic Functions

**Parallel composition**: Given is an automata network $\mathcal{A}_z$ consisting out of two automata in a parallel connection as depicted in Figure 3.6a. A parallel connection is characterised by the absence of any signal paths from one automaton to another. The overall network for the deterministic, nondeterministic, and stochastic case can be represented by

$$
\hat{\mathcal{A}}^d = (\mathcal{N}_z, \mathcal{N}_\hat{v}, \mathcal{N}_\hat{w}, \mathcal{N}_\hat{f}, \hat{L}^d, (z_1(0), z_2(0)), (f_1(0), f_2(0)))
$$

$$
\hat{\mathcal{A}}^n = (\mathcal{N}_z, \mathcal{N}_\hat{v}, \mathcal{N}_\hat{w}, \mathcal{N}_\hat{f}, \hat{L}^n, (z_1(0), z_2(0)), (f_1(0), f_2(0)))
$$

$$
\hat{\mathcal{A}}^s = (\mathcal{N}_z, \mathcal{N}_\hat{v}, \mathcal{N}_\hat{w}, \mathcal{N}_\hat{f}, \hat{L}^s, p(z_1(0), z_2(0)), p(f_1(0), f_2(0)))
$$

with the sets $\mathcal{N}_z = \{N_{z_1}, N_{z_2}\}$, $\mathcal{N}_\hat{v} = \{N_{v_1}, N_{v_2}, N_{v_3}\}$, $\mathcal{N}_\hat{w} = \{N_{w_1}, N_{w_2}\}$, and $\mathcal{N}_\hat{f} = \{N_{f_1}, N_{f_2}\}$. The characteristic functions are determined by

$$
\hat{L}^d(z', \hat{w}, \hat{z}, \hat{v}, \hat{f}) = L^d_1(z'_1, w_1, z_1, \{v_1, v_3\}, f_1) \cdot L^d_2(z'_2, w_2, z_2, \{v_2, v_3\}, f_2)
$$

$$
\hat{L}^n(z', \hat{w}, \hat{z}, \hat{v}, \hat{f}) = L^n_1(z'_1, w_1, z_1, \{v_1, v_3\}, f_1) \wedge L^n_2(z'_2, w_2, z_2, \{v_2, v_3\}, f_2) \quad (3.35)
$$

$$
\hat{L}^s(z', \hat{w}|\hat{z}, \hat{v}, \hat{f}) = L^s_1(z'_1, w_1|z_1, \{v_1, v_3\}, f_1) \cdot L^s_2(z'_2, w_2|z_2, \{v_2, v_3\}, f_2),
$$

and the sets $\hat{z}' = \{z'_1, z'_2\}$, $\hat{z} = \{z_1, z_2\}$, $\hat{v} = \{v_1, v_2, v_3\}$, $\hat{w} = \{w_1, w_2\}$, and $\hat{f} = \{f_1, f_2\}$. The parallel composition of two automata is denoted by $\mathcal{A}_1||\mathcal{A}_2$.

**Lemma 3.1**: The network $\mathcal{A}_z$ and the automaton $\hat{\mathcal{A}}$ gained by applying the rule for parallel composition (3.35) are equivalent.

**Proof**: See Appendix B.1.1.
3. Introduction to Automata Theory

**Serial composition:** Given is an automata network consisting out of two automata in a serial (or cascade) connection as depicted in Figure 3.6b. In a serial connection there exists a signal path from one to another automaton. The overall network for the deterministic, nondeterministic, and stochastic case can be represented by

\[
\hat{A}^d = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f, \hat{L}^d, (z_1(0), z_2(0)), (f^1(0), f^2(0)))
\]

\[
\hat{A}^n = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f, \hat{L}^n, (z_1(0), z_2(0)), (f^1(0), f^2(0)))
\]

\[
\hat{A}^s = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f, \hat{L}^s, p(z_1(0), z_2(0)), p(f^1(0), f^2(0)))
\]

with the sets \( \mathcal{N}_z = \{\mathcal{N}_{z_1}, \mathcal{N}_{z_2}\} \), \( \mathcal{N}_v = \{\mathcal{N}_{v_1}, \mathcal{N}_{v_2}\} \), \( \mathcal{N}_w = \{\mathcal{N}_{w_1}, \mathcal{N}_{w_2}\} \), and \( \mathcal{N}_f = \{\mathcal{N}_{f_1}, \mathcal{N}_{f_2}\} \). The characteristic functions are determined by

\[
\hat{L}^d(\hat{z}', \hat{w}, \hat{z}, \hat{\nu}, \hat{f}) = \sum_{s^1} \hat{L}_1^d(\hat{z}'_1, \{w^1, s^1\}, z_1, v^1, f^1) \cdot \hat{L}_2^d(\hat{z}'_2, w^2, z_2, \{v^2, s^1\}, f^2)
\]

\[
\hat{L}^n(\hat{z}', \hat{w}, \hat{z}, \hat{\nu}, \hat{f}) = \bigvee_{s^1} \hat{L}_1^n(\hat{z}'_1, \{w^1, s^1\}, z_1, v^1, f^1) \land \hat{L}_2^n(\hat{z}'_2, w^2, z_2, \{v^2, s^1\}, f^2)
\]

\[
\hat{L}^s(\hat{z}', \hat{w}|\hat{z}, \hat{\nu}, \hat{f}) = \sum_{s^1} \hat{L}_1^s(\hat{z}'_1, \{w^1, s^1\}|z_1, v^1, f^1) \cdot \hat{L}_2^s(\hat{z}'_2, w^2|z_2, \{v^2, s^1\}, f^2)
\]

and the sets \( \hat{z}' = \{\hat{z}'_1, \hat{z}'_2\} \), \( \hat{z} = \{z_1, z_2\} \), \( \hat{\nu} = \{v^1, v^2\} \), \( \hat{w} = \{w^1, w^2\} \), and \( \hat{f} = \{f^1, f^2\} \).

Because the coupling signal \( s^1 \) is not visible from outside the network, it is eliminated from the characteristic function \( \hat{L} \). The sum and boolean OR operations remove the explicit notation of the influence of the coupling signal while retaining its effects on the behaviour. The serial composition of two automata is denoted by \( \hat{A}_1 \preceq \hat{A}_2 \).

**Lemma 3.2:** The network \( \hat{A}_z \) and the automaton \( \hat{A} \) gained by applying the rule for serial composition (3.36) are equivalent.

**Proof:**
See Appendix B.1.2.

**Feedback composition:** Given is an automata network consisting out of one automaton with a feedback connection as shown in Fig. 3.6c. In a feedback connection an output signal of an automaton is fed back as one of its input signals. The overall network for the deterministic, nondeterministic, and stochastic case can be given as

\[
\hat{A}^d = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f, \hat{L}^d, z_1(0), f^1(0))
\]

\[
\hat{A}^n = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f, \hat{L}^n, z_1(0), f^1(0))
\]

\[
\hat{A}^s = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f, \hat{L}^s, p(z_1(0)), p(f^1(0)))
\]
3.3. Synchronous Automata Network

with

\[ \hat{L}^d (z', w^1, z, v^1, f^1) = \sum_{s^1} L^d_s (z'_1, \{w^1, s^1\}, z_1, \{v^1, s^1\}, f^1) \]
\[ \hat{L}^n (z', w^1, z, v^1, f^1) = \bigvee_{s^1} L^n_s (z'_1, \{w^1, s^1\}, z_1, \{v^1, s^1\}, f^1) \]  
(3.37)
\[ \hat{L}^s (z', w^1 | z, v^1, f^1) = \sum_{s^1} L^s_s (z'_1, \{w^1, s^1\} | z_1, \{v^1, s^1\}, f^1). \]

Analogously to the serial composition the coupling signal \( s^1 \) is eliminated from the characteristic function \( \hat{L} \), because it is not visible from outside the network. The feedback composition of an automaton is denoted by \( \sqcap_A \).

**Lemma 3.3:** The network \( A_\sharp \) and the automaton \( \hat{A} \) gained by applying the rule for feedback composition (3.37) are equivalent.

**Proof:**

See Appendix B.1.3.

These composition rules can be combined to a general composition rule:

\[ \hat{L}^d = \sum_{j=1}^{\kappa} \sum_{s^j \in N^s_j} \bigwedge_{i=1}^\gamma L^d_i = \sum_{s^1} \bigwedge_{i=1}^\gamma L^d_i, \]  
(3.38)
\[ \hat{L}^n = \bigvee_{j=1}^{\kappa} \bigvee_{s^j \in N^s_j} \bigwedge_{i=1}^\gamma L^n_i = \bigvee_{s^1} \bigwedge_{i=1}^\gamma L^n_i, \]  
(3.39)
\[ \hat{L}^s = \sum_{j=1}^{\kappa} \sum_{s^j \in N^s_j} \bigwedge_{i=1}^\gamma L^s_i = \sum_{s^1} \bigwedge_{i=1}^\gamma L^s_i. \]  
(3.40)

For the composition rules the commutative and associative laws hold. This means that multiple automata can be joined by applying the rules repeatedly. The order in which the automata are processed is insignificant [93]. The above operators have been implemented in the toolbox \textsc{DiAMONDQ} for \textsc{MATLAB}. Because the equivalence of the network and the single automaton generated by applying the composition rules has been proven, this automaton is called the \textit{equivalent} automaton of the network.

**Composition of Networks Given as a Relational data model**

**Parallel composition:** Given is an automata network consisting of two automata in relational representation in a parallel connection as depicted in Figure 3.6a. The over-
3. Introduction to Automata Theory

all network is denoted as in the previous section with the behavioural relation for the deterministic and nondeterministic case given by

$$\hat{L}^d(\hat{z}', \hat{w}, \hat{z}, \hat{v}, \hat{f}) = L^d(z'_1, w^1, z_1, \{v^1, v^3\}, f^1) \triangleright L^d(z'_2, w^2, z_2, \{v^2, v^3\}, f^2)$$
$$\hat{L}^n(\hat{z}', \hat{w}, \hat{z}, \hat{v}, \hat{f}) = L^n(z'_1, w^1, z_1, \{v^1, v^3\}, f^1) \triangleright L^n(z'_2, w^2, z_2, \{v^2, v^3\}, f^2).$$

with the sets $\hat{z}' = \{z'_1, z'_2\}$, $\hat{z} = \{z_1, z_2\}$, $\hat{v} = \{v^1, v^2, v^3\}$, $\hat{w} = \{w^1, w^2\}$, and $\hat{f} = \{f^1, f^2\}$. The composition of a SAN cannot be given as a closed expression, because it additionally contains algebraic operations to multiply the probabilities of the respective tuples and is, therefore, stated as MySQL-code instead.

MySQL-Code (Parallel Composition):\(^7\)

```sql
> /* deterministic and nondeterministic case */
> SELECT * FROM L1 NATURAL JOIN L2;
> /* stochastic case */
> SELECT DISTINCTROW zz1,zz2,v1,v2,v3,z1,z2,w1,w2,f1,f2,Prob1*Prob2
> AS Prob FROM L1 NATURAL JOIN L2 ORDER BY (f1,f2);
```

Serial composition: Given is an automata network consisting out of two automata in a serial (or cascade) connection as depicted in Figure 3.6b. The composition is calculated analogously to the previous case with the code given below.

MySQL-Code (Serial Composition):

```sql
> /* deterministic and nondeterministic case */
> SELECT DISTINCTROW zz1,zz2,v1,v2,z1,z2,w1,w2,f1,f2
> AS Prob FROM L1 NATURAL JOIN L2;
> /* stochastic case */
> SELECT zz1,zz2,v1,v2,z1,z2,w1,w2,f1,f2,SUM(Prob1*Prob2)
> AS Prob FROM L1 NATURAL JOIN L2 GROUP BY zz1,zz2,v1,v2,z1,z2,w1,w2,f1,f2;
```

Feedback composition: Given is an automata network consisting out of one automaton with a feedback connection as depicted in Figure 3.6c. The overall network is denoted as in the previous section with the behavioural relations given by

$$\hat{L}^d(z'_1, w^1, z, v^1, f^1) = \pi(z'_1, w^1, z, v^1, f^1) \sigma_{s^{1\prime} = s^{1\prime\prime}}(L^d(z'_1, \{w^1, s^{1\prime\prime}\}, z_1, \{v^1, s^{1\prime\prime}\}, f^1))$$
$$\hat{L}^n(z'_1, w^1, z, v^1, f^1) = \pi(z'_1, w^1, z, v^1, f^1) \sigma_{s^{1\prime} = s^{1\prime\prime}}(L^n(z'_1, \{w^1, s^{1\prime\prime}\}, z_1, \{v^1, s^{1\prime\prime}\}, f^1)).$$

Because all attributes of a relation must be named unambiguously, it is necessary to differentiate between $s^1 = s^{1\prime\prime}$ used as an input and $s^1 = s^{1\prime}$ generated as an output.

\(^7\)Because in MySQL attributes may not be named using special characters, the future state will be named $zz$ instead of $z'$.  

---

44
### 3.3. Synchronous Automata Network

**MySQL-Code (Feedback Composition):**

```sql
> /* deterministic and nondeterministic case */
> SELECT DISTINCTROW zz,w1,z,v1,f1 FROM L1 WHERE s1out=s1in;
> /* stochastic case */
> SELECT zz,w1,z,v1,f1,SUM(Prob1) AS Prob FROM L1
  WHERE s1out=s1in GROUP BY zz,w1,z,v1,f1;
```

It is apparent that the main advantage of the relational representation is the compact notation of the composition operations in case of deterministic and nondeterministic automata networks. However, for SAN it is not possible to give a closed expression. All composition operations can be performed using common database functions.

### 3.3.5. Complexity Considerations

It is known that all types of discrete-event models – including automata – suffer from the problem of *state-space explosion* [60, 96]. The reason lies in the necessity to denote all possible transitions of the DES explicitly (e.g. in tables). In general, it is not possible to give the characteristic function as a closed expression. For large and complex systems, i.e. systems with a large number of signals and states, a discrete-event model might become too large to be handled even with modern computing equipment. One way to counteract is to describe the behaviour of the system in terms of the behaviour of its components (*compositional modelling*). In the case of automata models the result is an automata network whose representation is distinctly less memory consuming\(^8\) than the representation as a single automaton. This advantage results from the structural information about the system which is contained in the network model. Whereas in a single automaton all signal dependencies and independencies have to be included explicitly in the relation \(L\), they are included implicitly in the topology of the network. This will be demonstrated in the following example.

![Exemplary automaton network](image)

**Figure 3.7.:** Exemplary automaton network

---

\(^8\)The memory consumption of a discrete event system is given in the number of possible transitions. This is equal to the number of rows in an automaton table or arcs in an automaton graph. The true memory consumption in bits depends on the implementation and is therefore not investigated in this book.
3. Introduction to Automata Theory

Example 7 (Complexity):
Consider a network consisting of ten automata in a parallel connection as depicted in Fig. 3.7a. Every automaton’s input and output alphabet and state set holds ten elements ($M_i = 10$, $R_i = 10$, $N_i = 10$). No fault cases are considered. In the worst case the relations $L_i$ of all automata are fully occupied, that is all transitions are possible. The memory consumption of the network amounts to the sum of the memory consumption of the ten automata:

$$\sum_{i=1}^{10} N_i M_i R_i = 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 = 1 \cdot 10^5.$$  

Applying the parallel composition $A_1 || A_2 || \cdots || A_{10}$ results in the equivalent automaton whose relation contains

$$\prod_{i=1}^{10} N_i M_i R_i = \prod_{i=1}^{10} 1 \cdot 10^4 = 1 \cdot 10^{40}$$

possible transitions.\(^9\) A serial connection of the same ten automata (cf. Figure 3.7b) results in a network with the identical memory consumption as above, but the relation of the composed automaton $\hat{A} = A_1 \sim A_2 \sim \cdots \sim A_{10}$ includes

$$M_1 R_{10} \prod_{i=1}^{10} N_i N_i = 100 \prod_{i=1}^{10} 100 = 1 \cdot 10^{22}$$

possible transitions. Although this number is significantly smaller than in the case of a parallel connection, it is still too large to be realisable. \[\square\]

The example makes two points abundantly clear. Firstly, the low memory consumption of the network representation is a distinct advantage. It allows to model large and complex processes as a discrete-event system where the representation as a single automaton might be impossible. Secondly, it becomes apparent that the structure of the network greatly influences the number of possible transitions.

Interestingly, the composed relation contains the more possible transitions the more independent the automata are. Couplings between the automata lead to restrictions in their movement, which again is reflected by less possible transitions in the composed relation. This is quite opposite to the behaviour found in linear continuous system theory, where independencies are reflected by the zero entries in the state space model. As a consequence the large number of system simplification procedures and order reduction techniques for linear continuous systems as e.g. found in [50] cannot be transferred to automata networks.

\(^9\)For comparison, currently, the internet search engine Google stores about $10^{15}$ bytes of information. The number of grains of sand on earth is $10^{24}$. It is clear that the amount of memory necessary to store the composed network will not be available in future times.
Part II.

Diagnosis of Nondeterministic Automata Networks
Centralised Diagnosis of Nondeterministic Processes

The centralised diagnosis is the classical approach of finding a fault in a system whereby the full measurement and model information is used. The presented approach is based upon the results in [12, 93] which introduce a centralised diagnostic method for a single stochastic automaton with scalar inputs and outputs and a concept to extend this approach to stochastic automata networks.

In Section 4.2 the approach for the single automaton is transferred to nondeterministic automata with multiple inputs and outputs using characteristic functions and relational algebra. This is done analogously in Section 4.3 for the mentioned extension to networks. In Section 4.4 a computational enhancement of the diagnostic approach is introduced. A feasibility analyses of the different approaches shows that despite the enhancements large problems cannot be solved using the centralised methods. After an analysis of diagnosability and output indifference in Section 4.5 the chapter closes with a conclusion.

4.1. Centralised Information Structure

In centralised diagnosis the least constraints on the information structure are posed. Here a single diagnoser $D_{centr}$ has access to the full system model and to all input and output signals as depicted in Fig. 4.1a (cf. Section 2.2). The diagnosers task is then to gain the set of faults $F_{centr}(k)$ which includes all faults for which the measurements of all I/O-signals are consistent with the global model (cf. Section 2.1). The diagnostic result should be as close as possible to the ideal result $F^\star$ while still being complete.

1The index $centr$ denotes that the diagnostic result has been obtained using a centralised approach.
4. Centralised Diagnosis of Nondeterministic Processes

It is intuitively clear that any further restrictions on the available information during diagnosis can only lead to a degradation of the result. It is therefore to be expected that the result of the centralised approach is closer to the ideal diagnostic result than the decentralised approaches discussed later.

Figure 4.1.: Centralised information structure

4.2. Diagnosis of a Single Automaton

4.2.1. Approach for Automata on Characteristic Functions

In the centralised diagnosis of a single automaton the process model is already given in monolithic form. This model and all measurements \((V, W)\) of the automaton are available to the single diagnoser \(D_{centr}\) (Figure 4.1b). The aim is to gain the set of faults \(F_{centr}(k)\) which includes all faults for which the measurements are consistent with the single automaton model. The diagnostic algorithm is initialised with the a-priori initial condition \(Poss(z(0), f)\) which includes all possible initial state and fault combinations. For notational convenience the fault \(f\) is assumed to be constant throughout the remainder of this work. However, all methods can be transferred to time-varying faults by extending the characteristic function to include a dynamic fault model \(L(z', w, f', z, v, f)\) as described in Section 3.2.7. With this convention the centralised diagnostic problem can be stated as follows:

<table>
<thead>
<tr>
<th>Centralised fault diagnostic problem for nondeterministic automata</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Given:</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Find:</strong></td>
</tr>
</tbody>
</table>
4.2. Diagnosis of a Single Automaton

The signals $v$ and $w$ have to be measurable to allow for on-line diagnosis, i.e. the set of faults is returned for each time step $k$. This problem is solved in a way to allow for on-line diagnosis, i.e. new measurements are used to refine the diagnostic result of the last step. The solution to the diagnostic problem is therefore given in a recursive form and is divided into the two parts projection and prediction. The measurement sequences are abbreviated by

$$V(0\ldots k) = (v(0), v(1), \ldots, v(k))$$
$$W(0\ldots k) = (w(0), w(1), \ldots, w(k)).$$

The prediction step

$$\text{Poss}(z(k+1), f, V(0\ldots k), W(0\ldots k)) = \bigvee_{z(k)} L^n(z(k+1), w(k), z(k), v(k), f) \land \text{Poss}(z(k), f, V(0\ldots k-1), W(0\ldots k-1))$$

(4.1)

returns the possibility of $z(k+1)$ being the next state and $f$ the fault under the condition that the sequences $V$ and $W$ have been measured. It is calculated using the result of the previous prediction step or in case of the first diagnostic step ($k = 0$) the initial condition:

$$\text{Poss}(z(1), f|v(0), w(0)) = \bigvee_{z(0)} L^n(z(1), w(0), z(0), v(0), f) \land \text{Poss}(z(0), f).$$

(4.2)

That is, the knowledge about the automaton’s current state and fault is used to observe its future state and fault. The projection step

$$\text{Poss}(f, V(0\ldots k), W(0\ldots k)) = \bigvee_{z(k+1)} \bigvee_{z(k)} L^n(z(k+1), w(k), z(k), v(k), f) \land \text{Poss}(z(k), f, V(0\ldots k-1), W(0\ldots k-1)) = \bigvee_{z(k+1)} \text{Poss}(z(k+1), f, V(0\ldots k), W(0\ldots k))$$

(4.3)

projects the result of the prediction step onto the fault space. The diagnostic result $F_{centr}(k)$ includes all possible faults $f$ according to the projection step:

$$F_{centr}(k) = \{f|\text{Poss}(f, V(0\ldots k), W(0\ldots k)) = 1\}.$$  

(4.4)
4. Centralised Diagnosis of Nondeterministic Processes

The diagnostic algorithm is then given as follows:

**Algorithm 4.1** (Centralised diagnosis of nondeterministic automata)

 Initialise:  

 1. Measure \( v(k) \) and \( w(k) \)
 2. Apply Equations (4.1) - (4.2) (prediction) for all \( z', f \)
 3. Apply Equation (4.3) (projection) for all \( z', f \)
 4. Stop on user demand, else \( k := k + 1 \)
 5. Repeat from Step 1

Result: \( F_{centr}(k) \) from Equation (4.4)

The diagnostic Algorithm 4.1 is also given using pseudo-code in Appendix C.1.

**Remark:** The increment \( k := k + 1 \) implies that \( z(k + 1) \) becomes \( z(k) \) in the next step.

**Remark:** In implementations of the algorithm it has proven to be sensible to add a test for the “total inconsistency” after Step 2:

\[ \text{If } Poss(z', f, V, W) = 0, \forall z', f \text{ restart with initial condition.} \]

The condition \( Poss(z', f, V, W) = 0, \forall z', f \) indicates an inconsistency between the measured behaviour and the behaviour modelled in \( L \), i.e. a fault which has not been modelled has occurred or in other words the closed world assumption has been violated. This causes the diagnostic method to return an empty set \( F_{centr}(k) = \emptyset \) for all further time-steps. Because such an inconsistency can also be caused by a single mismeasurement, it is recommended to report this total inconsistency to the operator and to restart the algorithm.

**Theorem 4.1 (Complete and sound diagnostic result):**

The diagnostic result \( F_{centr}(k) \) obtained through Algorithm 4.1 is complete and sound, i.e. the following relation holds:

\[ F_{centr}(k) = F^\star(k). \]

**Proof:**

See Appendix B.2.2.

In other words, the given algorithm yields the ideal diagnostic result for the given measurements and the model. The set \( F_{centr} \) does not include spurious solutions.
4.2. Diagnosis of a Single Automaton

while on the other hand no possible fault is overseen. Further improvements of the
result, i.e. a reduction of the set of fault candidates, cannot be achieved by altering the
diagnostic algorithm, but only by supplying additional measurement information or a
more detailed model.

According to Section 2.1 the diagnoser solving the above diagnostic problem is then
fully defined by

\[ D_{centr} = \{ A^n, (V, W), Poss(z(0), f), Algorithm 4.1, F^\star \} . \]

4.2.2. Approach for Automata on Relations

Alternatively, the diagnostic task can be solved using a relational representation of the
automaton model and relational algebra. The aim is then to gain the relation \( F_{centr}(k) \)
with the schema \( sch(F_{centr}(k)) = \{ f \} \) which includes all faults for which the mea-
surements are consistent with the model.\(^2\) The a-priori initial condition \( Pre(−1) \) with
\( sch(Pre(k)) = \{ z(k + 1), f \} \) includes all possible initial state and fault combinations.
With this convention the centralised diagnostic problem presented in the previous
section can be reformulated as follows:

<table>
<thead>
<tr>
<th>Centralised fault diagnostic problem for nondeterministic automata</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given:</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Find:</td>
</tr>
</tbody>
</table>

This problem is solved analogously to the previous realisation using relational al-
gebra. The prediction step

\[ Pre(k) = \pi_{z(k + 1), f}(\sigma(z(k), f) \in Pre(k − 1), v(k), w(k)L^n) \quad (4.5) \]

calculates all possible pairs of faults \( f \) and future states \( z(k + 1) \). Thereby, the selection
operation

\[ \sigma_{z(k), f} \in Pre(k − 1), v(k), w(k) \]

\(^2\)In contradiction to the usual notation of relations \( F \) is denoted using a calligraphic letter as in the above
realisation to avoid confusion.
4. Centralised Diagnosis of Nondeterministic Processes

selects all tuples of the relation \( L \) for which the attributes in \( v(k) \) and \( w(k) \) are identical to the measurements\(^3\) and for which the pairs \((z(k), f)\) are included in the previous diagnostic step \( \text{Pre}(k-1) \). The tuples are then projected by \( \pi_{z(k+1), f} \) upon the attributes \( z(k+1) \) and \( f \) to form the intermediate result \( \text{Pre}(k) \) which in turn will be refined in the next time step \( k+1 \). The set of possible faults is gained by projecting \( \text{Pre}(k) \) upon the attribute \( f \):

\[
F_{\text{centr}}(k) = \pi_f \text{Pre}(k) . \tag{4.6}
\]

The diagnostic algorithm is then given as follows:

Algorithm 4.1R (Centralised diagnosis of NA using relations)

Initialise: \( k = 0 \)

1. Measure \( v(k) \) and \( w(k) \)
2. Apply Equation (4.5)
3. Apply Equation (4.6)
4. Stop on user demand, else \( k := k + 1 \)
5. Repeat from Step 1

Result: \( F_{\text{centr}}(k) \)

Remark: Analogously to the previous algorithm it is recommended to implement a test for the total inconsistency and to restart the algorithm with the initial condition whenever \( \text{Pre}(k) = \emptyset \) holds.

Algorithm 4.1R is also given in MySQL-code in Appendix C.2. As this algorithm only provides an alternative way to Algorithm 4.1 of computing the identical diagnostic result, the same remarks apply, i.e. \( F_{\text{centr}}(k) = F^{\star}(k) \) holds.

4.3. Application to Automata Networks

In this section it is investigated how the centralised approach to diagnosis presented in the previous section can be applied to nondeterministic automata networks. Clearly, if the network is to be diagnosed using a centralised approach a single diagnoser \( D_{\text{centr}} \)

\[^3\] As previously mentioned, the notation \( \sigma_{v(k), w(k)} \) is an abbreviation of the correct notation \( \sigma_{v(k) = v^m(k), w(k) = w^m(k)} \). That is, the condition of the selection operation holds true if \( v(k) \) is identical to the measured input \( v^m(k) \) and \( w(k) \) is identical to the measured output \( w^m(k) \).
4.3. Application to Automata Networks

has to be found which diagnoses the distributed plant using a monolithic model (Figure 4.2). Analogously to the previous section the aim is to gain the set

\[ F_{\text{centr}} = \{ (f^1, \ldots, f^\sigma) | \text{Poss}(f^1, \ldots, f^\sigma, V(0 \ldots k), W(0 \ldots k)) = 1 \} \]  

(4.7)

which contains all fault candidates of the network. The initial conditions are given for all \( \gamma \) automata \( A_i \) separately as \( \text{Poss}(z_i(0), f_i) \). The problem for the centralised diagnosis of nondeterministic automata networks can then stated as follows:

**Centralised fault diagnostic problem for nondet. automata networks**

Given:  
- Nondeterministic automata network \( A_{\sharp} = \{ A_{\sharp 1}, A_{\sharp 2}, \ldots, A_{\sharp \gamma}, v, w, s, f \} \)
- \( v \) and \( w \) are measurable
- A-priori initial conditions \( \text{Poss}(z_i(0), f_i) \) of all \( i \) automata

Find:  
- Set of faults \( F_{\text{centr}}(k) \)

Figure 4.2.: Block diagram of the centralised diagnosis of automata networks

Because the solutions to the diagnostic problems given in the previous section have been defined for single automata models only, they cannot be applied directly to automata networks. The straightforward approach proposed in [69] is to apply the composition rules as given in Section 3.3.4 to gain the equivalent automaton of the network. After the composition the diagnostic solutions as stated in the previous section can be applied using the single automaton (confer Figure 4.2).

To reduce the amount of memory used to represent the composed network it was suggested to calculate the composition on-line. Then, for each time step \( k \) only that part of the network which fits to the current measurements \( v(k) \) and \( w(k) \) is considered. This is realised in the following algorithm.
4. Centralised Diagnosis of Nondeterministic Processes

Algorithm 4.2 (Centralised diagnosis of nondeterministic automata networks)
Initialise: $k = 0$
1. Measure $\mathbf{v}(k)$ and $\mathbf{w}(k)$
2. Compute $\hat{L}$ by applying (3.39) for measurements $\mathbf{v}(k)$ and $\mathbf{w}(k)$
3. Apply Equations (4.1)-(4.2) for all $\mathbf{z}', \mathbf{f}$ using $\hat{L}$
4. Apply Equation (4.3) for all $\mathbf{z}', \mathbf{f}$ using $\hat{L}$
5. Stop on user demand, else $k := k + 1$
6. Repeat from Step 1
Result: $\mathcal{F}_{\text{centr}}(k)$ from Equation (4.7)

The algorithm is given in pseudo-code in Appendix C.3 and in MySQL-code in Appendix C.4.

Theorem 4.2 (Complete and sound diagnostic result):
The diagnostic result $\mathcal{F}_{\text{centr}}(k)$ obtained through Algorithm 4.2 is complete and sound, i.e. the following relation holds:

$$\mathcal{F}_{\text{centr}}(k) = \mathcal{F}^{\star}(k).$$

Proof:
In Section 3.3.4 it has been proven that the behaviour $\mathcal{B}_{\mathbf{A}^{\sharp}}$ of an automaton network $\mathbf{A}^{\sharp}$ and the behaviour $\mathcal{B}_{\hat{\mathbf{A}}}$ of the equivalent automaton $\hat{\mathbf{A}}$, generated by applying the composition rules to this network, are identical: $\mathcal{B}_{\mathbf{A}^{\sharp}} = \mathcal{B}_{\hat{\mathbf{A}}}$. The completeness and soundness of the result of Algorithm 4.1 has been proven in Proofs B.2.1 and B.2.2. By applying this equality to (2.3) for a measurement sequence $\mathbf{B}(k)$ it follows that set of fault candidates is identical for both type of models:

$$f \in \mathcal{F}(k) \iff \mathbf{B}(k) \in \mathcal{B}_{\hat{\mathbf{A}}}^f$$
$$f \in \mathcal{F}(k) \iff \mathbf{B}(k) \in \mathcal{B}_{\mathbf{A}^{\sharp}}^f.$$

This proves that the diagnostic result of Algorithm 4.2 is identical to the result of Algorithm 4.1 and therefore also complete and sound.

The diagnoser solving the above diagnostic problem is then fully defined by

$$\mathcal{D}_{\text{centr}}^{\sharp} = \{ \mathbf{A}^{\sharp}, (\mathbf{V}, \mathbf{W}), \text{Poss}(z_i(0), \mathbf{f}_i) \forall i, \text{Algorithm 4.2}, \mathcal{F}^{\star} \}.$$
### Complexity considerations

Because the network is composed for a specific input and output, this approach seemingly reduces the memory consumption of the model by the factor

$$\Gamma_{v,w} = |N_v| \cdot |N_w|$$

in comparison to the diagnostic approach using the full behavioural function. Nevertheless, the advantages of the compositional model – low memory and structural information – are virtually lost. From considering the serial connection in Example 7 on Page 46 it becomes clear that the proposed approach is not be applicable for large systems. In the example the characteristic function of the composed network for a specific input and output includes

$$\prod_{i=1}^{10} N_i N_i = \prod_{i=1}^{10} 100 = 1 \cdot 10^{20}$$

possible transitions, which is by a factor $\Gamma_{v,w} = 100$ smaller than the full function, but still too large for a reasonable handling with modern computing equipment. Practically, the memory advantage of this approach is even less pronounced, because the model used for diagnosis has to be stored in memory in addition to the original network model as it is computed on-line.

### 4.4. Computational Enhancements

The complexity analysis in the previous section demonstrated that an automata network must not be represented as a single automaton. To retain the network’s memory advantage the diagnostic algorithm must operate directly with the network model (confer Figure 4.3). The idea behind the enhancement is to include the composition operation directly into the diagnostic algorithm. The resulting algorithm does not store parts of the composed network in memory, but uses the composition rule to access and combine the relevant information for the current diagnostic step from the network. Such an enhancement of the centralised diagnostic approach was presented in [114].

With the abbreviation

$$L_i^{(n)}(k) = L_i^{(n)}(z_i(k+1), w_i(k), z_i(k), v_i(k), f_i)$$  \hspace{1cm} (4.8)
4. Centralised Diagnosis of Nondeterministic Processes

The prediction step of the diagnostic algorithm is given by

\[
Poss(z(k+1), f, V(0\ldots k), W(0\ldots k)) = \bigvee_{z(k)} \left( \bigvee_{s(k)} \bigwedge_{i=1}^{\gamma} L_i^n(k) \right) Poss(z(k), f|V(0\ldots k-1), W(0\ldots k-1)). \quad (4.9)
\]

The projection step is given analogously to Equation (4.3) by

\[
Poss(f, V(0\ldots k), W(0\ldots k)) = \bigvee_{z(k+1)} Poss(z(k+1), f, V(0\ldots k), W(0\ldots k)). \quad (4.10)
\]

**Algorithm 4.3** (Centralised diagnosis of nondeterministic automata networks)

Initialise: \( k = 0 \)

1. Measure \( v(k) \) and \( w(k) \)
2. Apply Equation (4.9) for all \( z_i', f_i \)
3. Apply Equation (4.10) for all \( z_i', f_i \)
4. Stop on user demand, else \( k := k + 1 \)
5. Repeat from Step 1

Result: \( \mathcal{F}_{centr}(k) \) from Equation (4.7)

Note, that the diagnostic result is identical to the results gained by the previously presented solutions and this algorithm provides only an alternative way of calculating
the result set. Because all operations in the given solution are scalar, the full memory advantage of the network is retained. As a result several practical diagnostic problems could be solved where the original algorithm was not applicable. The drawback of this approach is the high number of mathematical operations which need to be performed.

Although in general, a high number of calculations poses less problems than a high memory consumption, the “size” of the problem has not been tackled, but only re-located. No structural information contained in the network has been used in the solution. As expected, applying the presented approach to large systems may take too long to compute to be applicable. Additionally, the implementation of this approach in efficient code has emerged to be rather complicated.

4.5. Diagnosability and Output Indifference

4.5.1. Diagnosability

Before designing or operating a diagnostic system it is essential to check if it is actually possible to detect or isolate the faults of interest. The prospect of success of diagnosing a system fault is specified by the system property diagnosability. However, because in general the sufficient condition for diagnosability is more difficult to find than necessary conditions, it is at first investigated under which condition a system is not diagnosable as defined in Definition 2.4. In the remains of this section several criteria for testing if a system is diagnosable or not are derived. At first this will be done for the single nondeterministic process, then these criteria will be extended to apply to automata networks.

Diagnosability in case of the single automaton

Lemma 4.1: A nondeterministic process is not diagnosable as of Definition 2.4 on page 18 if

\[ \text{Poss}(z', w, z, v, f) = \text{Poss}(z', w, z, v) \]

holds for all \( z', z \in \mathcal{N}_z, v \in \mathcal{N}_v, w \in \mathcal{N}_w, f \in \mathcal{N}_f \). \( \diamond \)

This condition states that it is not possible to draw a conclusion about the fault if the model contains no information about it or rather that the fault does not change the system’s behaviour. It is clear that no fault can be excluded from the set of fault candidates by inconsistency if the fault models are identical.
4. Centralised Diagnosis of Nondeterministic Processes

PROOF:
According to (2.3) a fault is a fault candidate iff
\[ f \in \mathcal{F}(k) \iff B(k) \in B(f) \]
holds. If the process’ behaviour does not differ for the fault cases \( B(f = 1) = B(f = 2) = \cdots = B(f = S) \) holds. Then either all fault cases are fault candidates or none of them are. The exclusion of all fault cases (\( \mathcal{F}(k) = \emptyset \)) is ruled out per definition (closed world assumption).

Even if the above corollary is violated, i.e. some faults do change the system’s behaviour, this might not be the case for all faults. Therefore, a stricter definition is stated below, which states that a system is also not diagnosable if there exists a subset of faults which cannot be differentiated.

Definition 4.1: A nondeterministic process is partially not diagnosable if there exists a subset of faults \( \tilde{\mathcal{N}}_f \subseteq \mathcal{N}_f \) for which there does not exists an input sequence \( V \) such that \( (V,W) \notin B(f), \forall f \in \tilde{\mathcal{N}}_f \) holds.

Corollary 4.1: A nondeterministic process is partially not diagnosable iff
\[ \text{Poss}(z',w,z,v,f) = \text{Poss}(z',w,z,v) \]
holds for all \( z',z \in \mathcal{N}_z, v \in \mathcal{N}_v, w \in \mathcal{N}_w \) and a subset of faults \( f \in \tilde{\mathcal{N}}_f \) with \( \tilde{\mathcal{N}}_f \subseteq \mathcal{N}_f \) and \( |\tilde{\mathcal{N}}_f| > 1 \).

Without proof it is clear that any faults with identical fault models cannot be distinguished. The definition of diagnosability follows directly from the above statements.

Definition 4.2 (Diagnosability of the single nondeterministic process): A nondeterministic process is called diagnosable if there does not exist a set \( \tilde{\mathcal{N}}_f \) for which the process is not partially diagnosable.

Diagnosability does not guarantee that a specific diagnostic system will always identify the fault in a diagnosable system. The difference between the different fault models might be marginal and only visible for certain I/O-pairs. Diagnosability only states that it is theoretically possible to identify a fault, whereas for not diagnosable systems it is abundantly clear that this is not possible.

Extension to the automaton with multiple I/O-signals
In the following the above statements are extended to be applicable to nondeterministic processes with multiple input and output signals, including unmeasurable signals.
The measurable subsets of $v_i, w_i$ are denoted by $\bar{v}_i, \bar{w}_i$ as defined on page 39. For a direct extension to automata the process is treated as a subprocess of a network and the attributes are therefore marked with an index $i$.

**Lemma 4.2:** A nondeterministic process with unmeasurable signals is *not diagnosable* if

$$\text{Poss}(z'_i, \bar{w}_i, z_i, \bar{v}_i, f_i) = \text{Poss}(z'_i, \bar{w}_i, z_i, \bar{v}_i)$$

holds for all $z'_i, z_i \in \mathcal{N}_{z_i}, f_i \in \mathcal{N}_{f_i}, \bar{v}_i \in \mathcal{N}_{v_i} \setminus \mathcal{N}_{s_i}, \bar{w}_i \in \mathcal{N}_{w_i} \setminus \mathcal{N}_{s_i}$.

**Proof:**

As per definition the behaviour contains only measurable signals. With the measurement $B_i(k) = (\bar{V}_i(0 \ldots k), \bar{W}_i(0 \ldots k))$ and the behaviour

$$B_i(f_i) \subseteq ((\mathcal{N}_{v_i} \setminus \mathcal{N}_{s_i}) \times (\mathcal{N}_{w_i} \setminus \mathcal{N}_{s_i})) \times ((\mathcal{N}_{v_i} \setminus \mathcal{N}_{s_i}) \times (\mathcal{N}_{w_i} \setminus \mathcal{N}_{s_i}))^2 \times \cdots$$

the proof of Lemma 4.1 applies accordingly.

The extension to partial diagnosability is given analogously without proof.

**Corollary 4.2:** A nondeterministic process with unmeasurable signals is *partially not diagnosable* if

$$\text{Poss}(z'_i, \bar{w}_i, z_i, \bar{v}_i, f_i) = \text{Poss}(z'_i, \bar{w}_i, z_i, \bar{v}_i)$$

holds for all $z'_i, z_i \in \mathcal{N}_{z_i}, f_i \in \tilde{\mathcal{N}}_{f_i}$, with subset of faults $f_i \in \tilde{\mathcal{N}}_{f_i}$ with $|\tilde{\mathcal{N}}_{f_i}| > 1$.

**Diagnosability in case of the automata network**

In the following, the diagnosability of the automata network is put into relation with the diagnosability of its components.

**Lemma 4.3 (Diagnosability of nondeterministic automata networks):** A nondeterministic process represented by an automata network $\mathcal{A}_\sharp$ is *diagnosable* if and only if the equivalent single automaton $\hat{\mathcal{A}}$, generated by applying the composition rules to this network, is diagnosable.
4. Centralised Diagnosis of Nondeterministic Processes

**Proof:**
Applying the composition rule (3.39) to a nondeterministic process represented by an automata network \( \mathcal{A}_x \), results in

\[
L^n(z', w, z, v, f) = \text{Poss}(z', w, z, v, f) = \bigvee_{s=1}^{\gamma} \text{Poss}(z'_i, w_i, z_i, v_i, f_i).
\]

This implies that the nondeterministic process can be transferred to the scalar case

\[
\text{Poss}(\tilde{z}', \tilde{w}, \tilde{z}, \tilde{v}, \tilde{f}).
\]

According to Corollary 4.1 this process is not partially diagnosable if

\[
\text{Poss}(\tilde{z}', \tilde{w}, \tilde{z}, \tilde{v}, \tilde{f}) = \text{Poss}(\tilde{z}', \tilde{w}, \tilde{z}, \tilde{v})
\]

holds for all \( \tilde{z}', \tilde{z} \in \mathcal{N}_{\tilde{z}} \), \( \tilde{v} \in \mathcal{N}_{\tilde{v}} \), \( \tilde{w} \in \mathcal{N}_{\tilde{w}} \) and a subset of faults \( \tilde{f} \in \mathcal{N}_{\tilde{f}} \). This proves that a NAN is exactly then not partially diagnosable if the equivalent automaton is not partially diagnosable. The diagnosability of the NAN follows directly from that.

**Theorem 4.3 (Not diagnosable subprocesses):**
A nondeterministic process represented by an automata network \( \mathcal{A}_x \) is not diagnosable if all automata of the network are not diagnosable.

**Proof:**
With the relation

\[
\text{Poss}(z', \bar{w}, z, v, f) = \text{Poss}(z', \bar{w}, z, \bar{v})
\]

\[
\iff
\bigvee_s \text{Poss}(z', w, z, v, f) = \bigvee_s \text{Poss}(z', w, z, v)
\]

and the application of the composition rules the following derivation puts the network
4.5. Diagnosability and Output Indifference

into relation with the equivalent automaton:

\[
L^n(z', \bar{w}, z, \bar{v}, f) = Poss(z', \bar{w}, z, \bar{v}, f) \\
= \bigvee_{s} \bigwedge_{i} Poss(z'_i, w_i, z_i, v_i, f_i) \quad \text{(composition rule)} \\
= \bigvee_{s} \bigwedge_{i} Poss(z'_i, w_i, z_i, v_i) \quad \text{(all NA are not diagnosable)} \\
= Poss(z', \bar{w}, z, \bar{v}) \quad \text{(composition rule)}
\]

This proves that the overall nondeterministic process is not diagnosable if all automata of the network are not diagnosable. ■

Theorem 4.4 (Influence of an automaton on the network diagnosability):

An automaton \( A^n_i \) being not diagnosable is not a sufficient condition for the network to be partially not diagnosable concerning the faults \( f_i \in \mathcal{N}_{f_i} \). □

In other words, it might be impossible to diagnose the faults \( f_i \) using the measurable signals of \( A^n_i \), although the fault influences the unmeasurable signals \( s_i \). But as long as the fault influences the behaviour of the overall network it is possible to diagnose it indirectly.

PROOF:
The theorem is proven by contradiction (reductio ad absurdum). Propose that a network is never partially diagnosable concerning a fault \( f_i \) when the associated automaton \( A^n_i \) is not diagnosable. Consider now the following academic example depicted in Figure 4.4 consisting out of two automata in a serial connection. The automata’s characteristic functions are given in Table 4.1. Both automata are static systems, i.e. their state does not change and only the left automaton \( A^n_1 \) is influenced by a fault \( f^1 \). Clearly, this automaton is not diagnosable, because no output signal is measurable. However, the fault can be identified directly using the right automaton’s output signal, because \( w^1 = f^1 \) holds. This contradicts the proposition, which proves the theorem. ■

![Figure 4.4: Network with not diagnosable subautomata](image-url)
4. Centralised Diagnosis of Nondeterministic Processes

Table 4.1.: Automaton tables of the automata depicted in Figure 4.4

<table>
<thead>
<tr>
<th></th>
<th>$z_1$</th>
<th>$s^1$</th>
<th>$z_1$</th>
<th>$v^1$</th>
<th>$f^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$z_2$</td>
<td>$w^1$</td>
<td>$z_2$</td>
<td>$s^1$</td>
<td></td>
</tr>
<tr>
<td>b)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Detectability and Identifiability

In Section 2.1 the common terms of fault detection and identification have been given to differentiate between diagnostic tasks of different elaborateness. In the following, the above definitions of diagnosability are applied directly to these notions.

Definition 4.3: A nondeterministic process is *not detectable* with respect to a fault case $f = i$ iff it is not possible for any given measurement sequence $(V, W)$ to distinguish between this fault and the faultless case $f = 1$.

Corollary 4.3: A nondeterministic process is *not detectable* with respect to a fault case $f = i, i \in \mathcal{N}_f$ iff the process is not local diagnosable for the set $\tilde{\mathcal{N}}_f = \{1, i\}$.

Definition 4.4: A nondeterministic process is *not identifiable* with respect to a fault case $f = i$ iff it is not possible for any given measurement sequence $(V, W)$ to distinguish between this and all other fault cases.

Corollary 4.4: A nondeterministic process is *not identifiable* with respect to a fault case $f = i, i \in \mathcal{N}_f$ iff the process is not local diagnosable for any of the sets $\tilde{\mathcal{N}}_f = \{i, j\}, \forall j \in \mathcal{N}_f, i \neq j$.

Definition 4.5: A nondeterministic process is *detectable/identifiable* with respect to a fault case $f = i$ iff it is not not detectable/identifiable with respect to this fault case.

The extension to automata networks is done analogously to Theorem 4.3. The proof is omitted, because of its similarity to the proof of that theorem.

Theorem 4.5 (Detectability/Identifiability of nondet. automata networks):

A nondeterministic process represented by an automata network $A^*_{\sharp}$ is detectable/identifiable with respect to a fault case $f = i$ if and only if the equivalent single automaton, generated by applying the composition rules to this network, is detectable/identifiable with respect to this fault case.
4.5. Diagnosability and Output Indifference

4.5.2. Output Indifference

The notion of output indifference was introduced in Section 2.3 in Definition 2.5. Output indifference means that the output does not contain any information about the fault or, in other words, that the result of a diagnosis does not differ from a mere simulation. In the following, criteria will be presented to test the indifference according to Definition 2.5.

Output indifference in case of the single automaton

**Lemma 4.4:** A nondeterministic process is output indifferent iff

\[ \text{Poss}(z', w, z, v, f) = \text{Poss}(z', z, v, f) \land \text{Poss}(w, v) \] 

(4.11)

holds for all \( z', z \in \mathcal{N}_z, v \in \mathcal{N}_v, w \in \mathcal{N}_w, f \in \mathcal{N}_f \).

This lemma is given without proof, because (4.11) simply states the independence of the output signal from the state and fault signals. Thereby it is assumed that the fault does not influence the input signal. The claim that in such a case the result of a diagnosis does not differ from a mere simulation is formalised by the following corollary.

**Corollary 4.5:** If a nondeterministic process represented by an automaton is output indifferent the following relation holds for all consistent I/O-pairs:

\[ \text{Poss}(f, V(0\ldots k), W(0\ldots k)) = \text{Poss}(f, V(0\ldots k)) = \text{Poss}(f). \]

**Proof:**

With Equations (4.3), (4.11) and \( \text{Poss}(w, v) = 1 \), because of the required consistency, the relation

\[ \text{Poss}(f, V, W) = \bigvee_{z(k+1)} \bigvee_{z(k)} L(k) \land \bigvee_{z(k-1)} L(k-1) \land \cdots \land \bigvee_{z(0)} L(0) \land \text{Poss}(z(0), f) \]
Centralised Diagnosis of Nondeterministic Processes

can be simplified to

\[
\text{Poss}(f, V, W) = \bigvee_{z(k+1), z(k)} \text{Poss}(z(k+1), z(k), v(k), f) \\
\land \cdots \land \bigvee_{z(0)} \text{Poss}(z(1), z(0), v(0), f) \land \text{Poss}(z(0), f) \\
= \text{Poss}(v(k), f) \land \cdots \land \text{Poss}(v(0), f) \land \text{Poss}(f) \\
= \text{Poss}(f, V) \\
= \text{Poss}(f),
\]

because the fault is constant and independent of the input signal. This proves the corollary. ■

A criterium for testing a nondeterministic automaton for output indifference is given below.

**Lemma 4.5:** A nondeterministic process is output indifferent if the nondeterministic automaton \(\mathcal{A}^n\) representing the system is a Mealy-Automaton according to Definition 3.1 and there exists a function \(\tilde{H}(w, v)\) with \(\tilde{H}(w, v) = H(w, z, v, f)\) for all \(z \in \mathcal{N}_z, v \in \mathcal{N}_v, w \in \mathcal{N}_w, f \in \mathcal{N}_f\).

**Proof:**

According to Definition 3.1 the relation

\[
\text{Poss}(z', w, z, v, f) = \text{Poss}(z', z, v, f) \land \text{Poss}(w, z, v, f).
\]

holds for a Mealy-Automaton. Applying \(\tilde{H}(w, v) = H(w, z, v, f)\) returns

\[
\text{Poss}(z', w, z, v, f) = \text{Poss}(z', z, v, f) \land \text{Poss}(w, v)
\]

which is equal to Equation (4.11) in Lemma 4.4. ■

**Extension to automata with multiple I/O-signals**

Again, the above statements are extended to be applicable to nondeterministic processes with multiple input and output signals, including unmeasurable signals. The measurable subsets of \(v_i, w_i\) are denoted by \(\bar{v}_i, \bar{w}_i\) as defined on page 39.

**Lemma 4.6:** A nondeterministic process with unmeasurable signals is output indifferent if

\[
\text{Poss}(z'_{i_1}, w_{i_1}, z_{i_1}, v_{i_1}, f_{i_1}) = \text{Poss}(z'_{i_1}, z_{i_1}, v_{i_1}, f_{i_1}) \land \text{Poss}(\bar{w}_{i_1}, \bar{v}_{i_1}) \quad (4.12)
\]
holds for all $z'_i, z_i \in \mathcal{N}_{z_i}, v_i \in \mathcal{N}_{v_i}, w_i \in \mathcal{N}_{w_i}, f_i \in \mathcal{N}_{f_i}$. 

The dependence of the fault on any unmeasurable signals is irrelevant as long as these are independent of the measurable signals. Corollary 4.5 applies accordingly as is proven in Appendix B.3.1.

**Output indifference in case of the automata network**

Again, the extension to automata networks can be done analogously to Theorem 4.3.

**Theorem 4.6 (Output indifference of nondeterministic automata networks):**

A nondeterministic process represented by an automata network $A$ is output indifferent if and only if the equivalent single automaton $\hat{A}$, generated by applying the composition rules to this network, is output indifferent. □

**Proof:**

As stated in the proof of Theorem 4.3 the nondeterministic process represented by an automata network can be transformed to the scalar case

$$\text{Poss}(\tilde{z}', \tilde{w}, \tilde{z}, \tilde{v}, \tilde{f}).$$

by application of the composition rules and the introduction of bijective mappings. According to Lemma 4.4 this process is output indifferent if

$$\text{Poss}(\tilde{z}', \tilde{w}, \tilde{z}, \tilde{v}, \tilde{f}) = \text{Poss}(\tilde{z}', \tilde{z}, \tilde{v}, \tilde{f}) \land \text{Poss}(\tilde{w}, \tilde{v})$$

(4.13)

holds for all $\tilde{z}', \tilde{z} \in \mathcal{N}_{\tilde{z}}, \tilde{v} \in \mathcal{N}_{\tilde{v}}, \tilde{w} \in \mathcal{N}_{\tilde{w}}, \tilde{f} \in \mathcal{N}_{\tilde{f}}$. This proves that a NAN is exactly then output indifferent if the equivalent automaton is output indifferent. □

**Theorem 4.7 (Output indifferent subprocesses):**

A nondeterministic process represented by an automata network $A$ is output indifferent if all automata of the network are output indifferent. □

**Proof:**

Given is a nondeterministic automata network $A = (A_1, \ldots, A_\gamma, v, w, s, f)$. As a reminder, the sets $v, w$ include the measurable I/O-signals of the network. In the following derivation the composition rules are applied to generate the equivalent automaton. For all subautomata of the network output indifference is assumed according to
Lemma 4.6. The last step shows that the whole nondeterministic process is output indifferent.

\[
L^n(z',w,z,v,f) = \text{Poss}(z',w,z,v,f) = \bigvee_s \bigwedge_{i=1}^\gamma \text{Poss}(z'_i,w_iz_iv_if_i)
= \bigvee_s \left[ \bigwedge_{i=1}^\gamma \text{Poss}(z'_i,z_iz_iv_if_i) \right] \land \bigwedge_{i=1}^\gamma \text{Poss}(\tilde{w}_i,\tilde{v}_i)
= \text{Poss}(z',z,v,f) \land \text{Poss}(\tilde{w},\tilde{v})
\]

4.6. State Observation

An observation is the process of calculating the unmeasurable system state by means of a system model and measurements as depicted in Figure 4.5. Note, that this notion is given in accordance with the practice in continuous system theory, but in contrast to the practice in the discrete-event community where an observation is a sensor measurement.

Observation is an important part of process supervision, since frequently process values cannot be measured because of various technical or financial reasons. In model-based fault diagnosis it is also necessary to estimate the current system state to be able to distinguish between the nominal and the faulty behaviour. In other words, state observation is always an integral part of model-based fault diagnosis. Therefore, this work’s focus is on solving the diagnostic problem, whereas the observation problem is only broached as a simplification of the diagnostic problem in this section.

Analogously to the diagnostic task an observation usually does not return a singleton but a set \(Z\) of possible system states. The ideal observation result includes exactly those states for which the measured I/O-sequence is consistent with the model:

\[
Z^\star (k) = \{z(k) | B(k) \in B \text{ and there } \exists z(k+1) \in N_z \text{ and } f \in N_f \text{ with } L^n(z(k+1),w(k),z(k),v(k),f) = 1\}. \tag{4.14}
\]

The notions of completeness and soundness apply correspondingly. The centralised observation problem can be stated analogously to the centralised diagnostic problem.
4.6. State Observation

as follows:

Centralised state observation problem for nondeterministic automata

Given:
- Automaton \( A^n \) with charact. function \( L^n(z(k+1), w(k), z(k), v(k), f) \)
- \( v \) and \( w \) are measurable
- A-priori initial condition \( Poss(z(0), f) \)

Find:
- Set of states \( Z_{centr}(k) \)

The prediction step in observation is identical to the prediction step (4.15) in the centralised diagnosis

\[
Poss(z(k+1), f, V(0...k), W(0...k)) = \\
\bigvee_{z(k)} L^n(z(k+1), w(k), z(k), v(k), f) \land Poss(z(k), f, V(0...k-1), W(0...k-1)).
\]

(4.15)

It returns the possibility of \( z(k+1) \) being the next state and \( f \) the fault under the condition that the sequences \( V \) and \( W \) have been measured and is calculated using the result of the previous prediction step or in case of the first diagnostic step (\( k = 0 \)) the initial condition:

\[
Poss(z(1), f | v(0), w(0)) = \bigvee_{z(0)} L^n(z(1), w(0), z(0), v(0), f) \land Poss(z(0), f). \]

(4.16)
The projection step

\[
\text{Poss}(z(k), V(0 \ldots k), W(0 \ldots k)) = \\
\bigvee_{z(k+1)} \bigvee_{f} L^n(z(k+1), w(k), z(k), v(k), f) \wedge \\
\text{Poss}(z(k), f, V(0 \ldots k-1), W(0 \ldots k-1))
\]  

projects the result of the prediction step onto the state space. The result of the observation includes all possible current states \(z(k)\) according to the projection step:

\[
Z_{\text{centr}}(k) = \{z(k) | \text{Poss}(z(k), V(0 \ldots k), W(0 \ldots k)) = 1\}. 
\]  

The observation algorithm is then given as follows (cf. also [93]):

**Algorithm 4.4** (Centralised observation of nondeterministic automata)

Initialise: \(k = 0\)

1. Measure \(v(k)\) and \(w(k)\)
2. Apply Equations (4.15) - (4.16) (prediction) for all \(z', f\)
3. Apply Equation (4.17) (projection) for all \(z\)
4. Stop on user demand, else \(k := k + 1\)
5. Repeat from Step 1

Result: \(Z_{\text{centr}}(k)\) from Equation (4.18)

For the result of this algorithm the following theorem holds:

**Theorem 4.8 (Complete and sound state observation result):**

The state observation result \(Z_{\text{centr}}(k)\) obtained through Algorithm 4.4 is complete and sound, i.e. the following relation holds:

\[
Z_{\text{centr}}(k) = Z^*(k).
\]

**Proof:**

See Appendix B.2.3.

Thus, just as the diagnostic algorithm the state observation algorithm yields the ideal result for the given measurements and the model. The state set \(Z_{\text{centr}}\) does not
include spurious solutions while on the other hand no possible fault is overseen.

If a system without fault signals is considered the automaton model simplifies to

$$A^n = (N_N, N_V, N_W, L^n, z(0))$$  \hspace{1cm} (4.19)

with the characteristic function

$$L^n : N_N \times N_W \times N_N \times N_V \rightarrow \{0, 1\},$$ \hspace{1cm} (4.20)

$$L^n(z(k+1), w(k), z(k), v(k)) = Poss(z(k+1), w(k), z(k), v(k)).$$  \hspace{1cm} (4.21)

Then the prediction and observation step can be given as follows:

$$Poss(z(k+1), V(0\ldots k), W(0\ldots k)) =$$

$$\bigvee_{z(k)} L^n(z(k+1), w(k), z(k), v(k)) \land Poss(z(k), V(0\ldots k-1), W(0\ldots k-1)).$$  \hspace{1cm} (4.22)

$$Poss(z(k), V(0\ldots k), W(0\ldots k)) =$$

$$\bigvee_{z(k+1)} L^n(z(k+1), w(k), z(k), v(k)) \land Poss(z(k), V(0\ldots k-1), W(0\ldots k-1)).$$  \hspace{1cm} (4.23)

The observation algorithm given above applies correspondingly.

This section showed how the solution of a diagnostic problem can be transformed to solve the corresponding observation problem while retaining the properties of the original diagnostic solution such as completeness and soundness. Even more generally, this approach can be used to observe every unknown signal. Because of this property the remainder of this book focuses on the fault diagnostic problem only, keeping in mind that all approaches can easily be transformed to state observation.

4.7. Conclusion

In this chapter an approach for the diagnosis of a single nondeterministic automaton and two centralised approaches for diagnosing a nondeterministic automata network have been presented. The first approach for networks applies the composition rules to generate the equivalent automaton of the network and uses the diagnostic method
for the single automaton to generate the result. The second approach avoids the
generation of the equivalent automaton, but performs the same amount of calculations
to receive the diagnostic result.

The outstanding fact is that all approaches yield the **ideal diagnostic result** $F^{\star}$
which is the smallest possible result set satisfying the completeness demand. No algo-
rithm can yield a smaller but complete set given the same measurements and model.
However, the **scalability** of the presented approaches is poor, because the number of
calculations increases exponentially with the size of the automaton or network and
may quickly become too large to be handled even with modern computing equipment.
The **reliability** is low, because of the centralised structure. If the diagnoser fails the
whole diagnostic system fails. The **reusability** of the two network approaches is higher
than of the approach using a single automaton. If the plant is altered only the respec-
tive components in the network model have to be adapted. Because of the centralised
structure the design and realisation of the diagnostic system cannot be **distributed**. The
impossibility to diagnose large systems using a centralised approach necessitates the
development of alternative methods.
Decentralised Diagnosis of Nondeterministic Automata Networks

In the previous chapter it has become apparent that the complexity of the diagnostic problem cannot be reduced as long as a classical centralised information structure is used. In this chapter it will be shown that the complexity can be reduced by exploiting the systems structure and applying a decentralised approach. A short motivation is given in Section 5.1. In Sections 5.2 and 5.2.3 the decentralised diagnostic approach is presented. The impact of the decentralisation on the diagnostic result is discussed in Sections 5.4 and 5.5. Diagnosability is investigated in Section 5.6. The chapter closes with a conclusion in Section 5.7.

5.1. Motivation of Decentralised Diagnosis

In decentralised diagnosis the diagnostic task is broken down into several smaller tasks and is performed by a number of independent local diagnosers $D_i$, whereby each diagnoser has only access to the measurements and the model of the $i$-th component:

$$D_i = \{A_i^n, (V_i \cap V, W_i \cap W), \text{Poss}(z_i(0), f_i), A, F_i(k)\}. \quad (5.1)$$

The sets $V_i \cap V$ and $W_i \cap W$ denote that only the input and output signals of the $i$-th component are available which are input and output signals of the network and therefore measurable. This excludes all coupling and fault signals (see also [114]). Although it is not a necessity, it is assumed that the algorithm $A$ is identical for all diagnosers for a given diagnostic task.

Decentralised diagnosis as depicted in Figure 5.1 poses the hardest constraints on the information structure, i.e. each local diagnoser has only very limited information.
5. Decentralised Diagnosis of Nondeterministic Automata Networks

about the overall process. The advantages of decentralised diagnosis are good scalability, reliability, simplicity, reusability, and distribution. However, it is to be expected that the computational advantages of the decentralised approach imply the degradation of the diagnostic result. The aim is therefore to gain a complete set of faults \( \mathcal{F}_{dec}(k) \supseteq \mathcal{F}^\star(k) \) which is as close to the ideal diagnostic result as possible.

![Block diagram of the decentralised diagnostic approach](image)

Figure 5.1.: Block diagram of the decentralised diagnostic approach

5.2. Solution to the Decentralised Diagnostic Problem

5.2.1. Approach for Automata on Characteristic Functions

The decentralised diagnostic problem is solved in two steps. At first it is investigated how a component of a network can be diagnosed independently of the remaining parts of the network using a local diagnoser \( D_i \). The decentralised diagnosis of \( \gamma \) components results in \( \gamma \) diagnostic results \( F_1(k), \ldots, F_\gamma(k) \). How all local results can be combined to form a global diagnostic result is investigated afterwards.

The problem for the diagnosis of a component of a NAN is given below:

**Local diagnostic problem for nondeterministic automata**

Given:
- Automaton \( A_i^n \) with ch. function \( L_i^n(z_i(k+1), w_i(k), z_i(k), v_i(k), f_i) \)
- \( \bar{v}_i \subseteq v_i \) and \( \bar{w}_i \subseteq w_i \) are measurable
- A-priori initial condition \( \text{Poss}(z_i(0), f_i) \)

Find:
- Set of faults \( \mathcal{F}_i(k) \)
5.2. Solution to the Decentralised Diagnostic Problem

As stated above, the diagnoser $D_i$ of the $i$-th automaton has only limited information about the behaviour of the component, because not all signals in $v_i$ and $w_i$ are measurable, but may include not measurable coupling signals (see also Example 6 on Page 39). As the values of the coupling signals are unknown, the influence of the remaining network on the component is unknown, which leads to additional uncertainties in the diagnostic process.

Different approaches for dealing with the unknown values of the coupling signals are possible. For example, one could choose random values or the most probable ones. But only if all possible values of the coupling signals are considered, it is ensured that no fault case is overlooked and that the diagnostic result is complete.

Example 8:
Given is an automaton $A^n_1$ of a NAN. Table 5.1 lists all tuples which are consistent with the measurement. Projecting these tuples onto the fault space without any restrictions on the coupling signal $s_1$ results in the set $F_1 = \{1, 2, 3\}$. Because $s_1 = f_1$ holds, selecting tuples for a specific coupling signal misses fault cases and the diagnostic result is not complete.

<table>
<thead>
<tr>
<th>$z'_1$</th>
<th>$z_1$</th>
<th>$s_1$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5.1.: Section of an automaton table

To assure the completeness of the diagnostic result Algorithm 4.1 is altered to consider all values of the coupling signals in $s_i$. For this the calculation of the prediction and projection step is performed for all values of the coupling signals and the results are connected via disjunctions. The idea behind this approach is to include all faults in the diagnostic result which are consistent with the local measurements:

$$ F_i(k) = \{ f_i | (V_i(0...k), W_i(0...k)) \in B_i(f_i) \} $$

with the abbreviations

$$ V_i(0...k) = (v_i(0), \ldots, v_i(k)) \quad \text{and} \quad W_i(0...k) = (w_i(0), \ldots, w_i(k)). $$
With this convention the prediction step amounts to

\[
\text{Poss}(z_i(k+1), f_i, \overline{V}_i(0\ldots k), \overline{W}_i(0\ldots k)) = \\
\bigvee_{z_i(k)} \bigwedge_{s_i(k)} (L^n_i(z_i(k+1), w_i(k), z_i(k), v_i(k), f_i) \\
\land \text{Poss}(z_i(k), f_i, \overline{V}_i(0\ldots k-1), \overline{W}_i(0\ldots k-1))). \tag{5.2}
\]

It yields the possibility of \(z_i(k+1)\) being the next state and \(f_i\) the fault if the sequences \(\overline{V}_i\) and \(\overline{W}_i\) have been measured and is calculated using the result of the previous prediction step or in case of the first diagnostic step \((k=0)\) the initial condition

\[
\text{Poss}(z_i(1), f_i, \overline{V}_i(0), \overline{W}_i(0)) = \\
\bigvee_{z_i(0)} \bigwedge_{s_i(0)} (L^n_i(z_i(1), w_i(0), z_i(0), v_i(0), f_i)) \land \text{Poss}(z_i(0), f_i). \tag{5.3}
\]

The projection step projects the result onto the fault space:

\[
\text{Poss}(f_i, \overline{V}_i(0\ldots k), \overline{W}_i(0\ldots k)) = \\
\bigvee_{z_i(k+1)z_i(k)} \bigwedge_{s_i(k)} (L^n_i(z_i(k+1), w_i(k), z_i(k), v_i(k), f_i) \\
\land \text{Poss}(z_i(k), f_i, \overline{V}_i(0\ldots k-1), \overline{W}_i(0\ldots k-1))) = \\
\bigvee_{z_i(k+1)} \text{Poss}(z_i(k+1), f_i, \overline{V}_i(0\ldots k), \overline{W}_i(0\ldots k)). \tag{5.4}
\]

The diagnostic result \(F_i(k)\) includes all possible faults \(f_i\) according to Eqn. (5.4):

\[
F_i(k) = \{f_i | \text{Poss}(f_i, \overline{V}_i(0\ldots k), \overline{W}_i(0\ldots k)) = 1\}. \tag{5.5}
\]

The diagnostic algorithm is then given as follows:

**Algorithm 5.1** (Local diagnosis of nondeterministic automata)

Initialise: \(k = 0\)

1. Measure \(\overline{v}_i(k)\) and \(\overline{w}_i(k)\)
2. Apply Equations (5.2) - (5.3) (prediction) for all \(z_i', f_i\)
3. Apply Equation (5.4) (projection) for all \(z_i', f_i\)
4. Stop on user demand, else \(k := k + 1\) and repeat from Step 1

Result: \(F_i(k)\) from Equation (5.5)
The local diagnosis returns the best possible result given the restricted information. This is formalised by the following theorem.

**Theorem 5.1 (Complete and sound diagnostic result):**
The diagnostic result $\mathcal{F}_i(k)$ obtained by Algorithm 5.1 is complete and sound, i.e. the following holds:
$$
\mathcal{F}_i(k) = \mathcal{F}_i^\star(k).
$$

**Proof:**
See Appendix B.2.4. ■

As indicated above, the local diagnosis of $\gamma$ components of the network results in the same number of independent diagnostic results. The problem of gaining a global set of fault candidates $\mathcal{F}_{dec}(k)$ can be stated as:

**Decentralised diagnostic problem for nondeterministic automata networks**

**Given:**
- Local sets of faults $\mathcal{F}_i(k), \forall i \in \{1, \ldots, \gamma\}$

**Find:**
- Global set of faults $\mathcal{F}_{dec}(k)$

The index $\text{dec}$ denotes that the diagnostic result has been obtained using a decentralised approach. The straightforward approach for solving the problem is to construct the Cartesian product of all local fault sets

$$
\mathcal{F}_{dec}(k) = \mathcal{F}_1(k) \times \mathcal{F}_2(k) \times \cdots \times \mathcal{F}_\gamma(k).
$$

(5.6)

The global set of faults $\mathcal{F}_{dec}(k)$ then contains the tuples of all fault cases which can have occurred simultaneously. The properties of the diagnostic result $\mathcal{F}_{dec}(k)$ are investigated further in Section 5.4.

**Remark:** The straight-forward approach to build the union of the local fault sets

$$
\mathcal{F}_{dec}(k) = \bigcup_{i=1}^{\gamma} \mathcal{F}_i(k)
$$

is not correct. It also includes all fault candidates of the local diagnostic results, but does no longer hold the information about the possible fault combinations. Nevertheless, this approach is common practice and can be interpreted as the display of all local fault candidates without further processing. Picture a control room in which all fault messages from different processes are received and immediately displayed on a computer screen. ■
5. Decentralised Diagnosis of Nondeterministic Automata Networks

5.2.2. Approach for Automata on Behavioural Relations

Alternatively, the problem for the diagnosis of a component of a nondeterministic automata network can be stated using relational algebra.

Local diagnostic problem for nondeterministic automata

Given:
- Automaton $A_i^n$ with relation $L_i^n(z_i(k+1), w_i(k), z_i(k), v_i(k), f_i)$
- $\bar{v}_i \subseteq v_i$ and $\bar{w}_i \subseteq w_i$ are measurable
- A-priori initial condition $Pre_i(-1)$

Find:
- Set of faults $F_i(k)$

The a-priori initial condition $Pre_i(-1)$ with $sch(Pre_i(k)) = \{z_i(k+1), f_i\}$ includes all possible initial state and fault combinations. The problem is solved analogously to the previous realisation using relational algebra. The prediction step

$$Pre_i(k) = \pi_{z_i(k+1), f_i} \left( \sigma_{(z_i(k), f_i)} Pre_i(k-1), v_i(k), \bar{w}_i(k) \right) L_i^n$$  \hspace{1cm} (5.7)

calculates all possible pairs of faults $f_i$ and future states $z_i(k+1)$. Please note, that the selection operation $\sigma_C$ returns all tuples of the relation $L_i^n$ which fulfill the condition

$$C = (\langle z_i(k), f_i \rangle \in Pre_i(k-1), \bar{v}_i(k), \bar{w}_i(k)),$$

which does not specify any constraints on the unmeasurable coupling signals, but only on the measurable signals $\bar{v}_i(k)$ and $\bar{w}_i(k)$. Thus, the result of the selection operation contains the tuples for all values of the coupling signals. The set of possible faults is gained by applying

$$F_i(k) = \pi_{f_i} Pre_i(k).$$  \hspace{1cm} (5.8)

The diagnostic algorithm is then given as follows:

Algorithm 5.1R (Local diagnosis of nondeterministic automata using relations)

Initialise: $k = 0$

1. Measure $\bar{v}_i(k)$ and $\bar{w}_i(k)$

2. Apply Equation (5.7)

3. Apply Equation (5.8)

4. Stop on user demand, else $k := k + 1$ and repeat from Step 1

Result: $F_i(k)$
5.2. Solution to the Decentralised Diagnostic Problem

As this algorithm only provides an alternative way to Algorithm 5.1 for computing the identical diagnostic result, the same remarks apply. The global diagnostic result can be calculated by joining the local diagnostic results by the cartesian product as given in Equation (5.6).

5.2.3. Off-line Preprocessing of the Model

To further decrease the computational costs during the on-line phase of the diagnostic process it is possible to preprocess the automata models off-line. The aim of this preprocessing is to eliminate the coupling signals from the characteristic functions \( L_i^n \). This reduces the amount of memory needed to store the model and the amount of calculations performed during diagnosis, because the computational costly operation \( \bigvee_{s_i} L_i^n \) in Equation (5.2) can be omitted.

The coupling signals \( s_i \) can be eliminated analogously to the composition of a network by applying

\[
\tilde{L}_i^n(z_i', \bar{w}_i, z_i, \bar{v}_i, f_i) = \bigvee_{s_i} L_i^n(z_i', w_i, z_i, v_i, f_i) \tag{5.9}
\]

for automata on characteristic functions or

\[
\tilde{L}_i^n(z_i', \bar{w}_i, z_i, \bar{v}_i, f_i) = \pi_{z_i', \bar{w}_i, z_i, \bar{v}_i, f_i} L_i^n(z_i', w_i, z_i, v_i, f_i)
\]

for automata on relations. The behaviours \( B_i \) of the respective automata remain unchanged by this operation, because the relation still includes all possible transitions. Only the effect of the coupling signals is not explicitly visible any more. The prediction step of a local diagnoser then amounts to

\[
\text{Poss}(z_i(k+1), f_i, \bar{V}_i(0\ldots k), \bar{W}_i(0\ldots k)) = \\
\bigvee_{z_i(k)} \tilde{L}_i^n \land \text{Poss}(z_i(k), f_i, \bar{V}_i(0\ldots k-1), \bar{W}_i(0\ldots k-1)). \tag{5.10}
\]

Clearly, the replacement of the original relation with the processed relation \( \tilde{L}_i^n \) does not change the diagnostic result as the component behaviour is unaltered.
5. Decentralised Diagnosis of Nondeterministic Automata Networks

5.3. Complexity Considerations

**Memory consumption:** As derived in Section 3.3.5 the relation of the equivalent automaton of a network contains a maximum of

\[
\prod_{i=1}^{\gamma} N_i^2 \cdot \prod_{j=1}^{\mu} M_j \cdot \prod_{h=1}^{\rho} R_h \cdot \prod_{\ell=1}^{\sigma} S_{\ell} \tag{5.11}
\]

transitions, where \(N_i = |\mathcal{N}_{z_i}|\), \(M_j = |\mathcal{N}_{\nu_j}|\), \(R_h = |\mathcal{N}_{\nu_h}|\), and \(S_{\ell} = |\mathcal{N}_{f_{\ell}}|\) hold. The combined memory consumption of the condensed relations \(\bar{L}_i^n\) is maximal

\[
\sum_{i=1}^{\gamma} (N_i^2 \cdot M_i \cdot R_i \cdot S_i), \tag{5.12}
\]

where \(M_i = |K(\bar{v}_i)|\), \(R_i = |K(\bar{w}_i)|\), and \(S_i = |K(\bar{f}_i)|\) hold. \(K(q)\) denotes the cartesian product of the domains of the signals in \(q\).

Using the Landau symbol \(O(\bullet)\) an upper bound for the memory consumption can be given [39]. Therefore, the size of the respective signal domains is approximated through an upper bound denoted by \(N, M, R, S\). Then order of magnitude of the memory consumption is

\[
O(N^2 \gamma M^\mu R^\rho S^\sigma) \quad \text{for the equivalent automaton and} \quad O(\gamma N^2 MRS) \quad \text{for the network.}
\]

As expected the memory consumption of the network is distinctly lower than of the equivalent automaton, because it grows linearly with the number of network components as opposed to the exponential growth of the equivalent automaton.

**Computational costs:** In the centralised diagnosis of a single automaton \(N - 1\) disjunctions and \(N\) conjunctions are calculated for each state and fault case in the prediction step. For the projection step additional \(N - 1\) disjunctions are calculated per fault case. For one step of the centralised diagnostic algorithm this amounts to

\[
(N + N - 1)NS + (N - 1)S \tag{5.13}
\]

arithmetic operations, which is of the order \(O(N^2S)\).

In decentralised diagnosis the diagnostic algorithm is of the same order. More
specifically each local diagnoser has to solve a problem of the order $O(N^2S)$ relating to the local model. These local problems can be solved in parallel. Additionally, the decentralised diagnosis necessitates preprocessing as given in Equation (5.9) and a combination of the local results as given in Equation (5.6). However, the costs of the preprocessing and postprocessing are of linear order and therefore negligible in comparison to the costs of the diagnostic algorithm.

Now the valid question arises why the decentralised diagnosis is more computationally efficient than the centralised approach despite the fact that they solve a problem of the same order. The answer lies in the different arguments for both methods. In the centralised approach the size $N$ of the equivalent automaton’s state space is very large in comparison to the state space of a single network component. Even if it is possible to realise a monolithic model of a complex plant, an algorithm of quadratic order will not finish in acceptable time for large $N$.

**Example 9:**
Consider an automata network consisting of two automata in a serial connection as depicted in Figure 5.2. The signal domains are given as follows:

\[
\mathcal{N}_{z_1} = \mathcal{N}_{z_2} = \{1, 2, 3\}, \quad \mathcal{N}_{v_1} = \mathcal{N}_{v_2} = \mathcal{N}_{w_1} = \mathcal{N}_{w_2} = \{1, 2\}, \quad \mathcal{N}_{f_1} = \mathcal{N}_{f_2} = \{1, 2\}.
\]

The automata’s behavioural relations $L_1$ and $L_2$ are given in Table E.2 on Page 216. Both relations contain 24 tuples, whereby the maximum number of possible tuples per relation is

\[
|\mathcal{N}_{z_i}|^2|\mathcal{N}_{v_i}| |\mathcal{N}_{w_i}| |\mathcal{N}_{f_i}| = 144 \quad \text{for } i \in \{1, 2\}.
\]

For the network description a sum of only 48 transitions has to be stored. For the decentralised diagnosis the coupling signal $s^1$ can be eliminated from the relations which reduces the number of tuples for the whole network to 24. The equivalent automaton of the network $\hat{L} = L_1 \bowtie L_2$ contains 285 tuples, which is only about 5.5% of the maximum number of tuples of 5184, but still substantially larger than the network representation.

![Automata network for Example 9](image)

Consider now that $v_1(0) = v_2(0) = w_1(0) = w_2(0) = 1$ has been measured. In the local
5. Decentralised Diagnosis of Nondeterministic Automata Networks

diagnosis of Automaton $\mathcal{A}_1$

\[
\text{Poss}(z'_1 = 1, f_1 = 1, v_1 = 1, w_1 = 1) = \\
L_1(z_1 = 1, w_1 = 1, z_1 = 1, v_1 = 1, f_1 = 1) \land \text{Poss}(z_1 = 1, f_1 = 1) \lor \\
L_1(z_1 = 1, w_1 = 1, z_1 = 2, v_1 = 1, f_1 = 1) \land \text{Poss}(z_1 = 2, f_1 = 1) \lor \\
L_1(z_1 = 1, w_1 = 1, z_1 = 3, v_1 = 1, f_1 = 1) \land \text{Poss}(z_1 = 3, f_1 = 1)
\]

has to be calculated for every combination of $f_1$ and $z'_1$. This amounts to 5 operations per fault-state-combination or 30 operations for the full prediction step. The projection step

\[
\text{Poss}(f_1 = 1, v_1 = 1, w_1 = 1) = \text{Poss}(z'_1 = 1, f_1 = 1, v_1 = 1, w_1 = 1) \lor \\
\text{Poss}(z'_1 = 2, f_1 = 1, v_1 = 1, w_1 = 1) \lor \text{Poss}(z'_1 = 3, f_1 = 1, v_1 = 1, w_1 = 1)
\]

adds 2 operations per fault case or 4 operations for the full projection step. The same number of operations arise for the second automaton $\mathcal{A}_2$. The cartesian product of the local results adds 4 operations. The total sum of operations per diagnostic step amounts then to 72. The number of operations needed for one step of the diagnosis of the equivalent automaton is determined analogously to 644 which is already larger by one order of magnitude than for the decentralised diagnosis. Clearly, for more complex networks the difference between the two diagnostic approaches becomes more distinct.

5.4. Impact of the Information Structure Constraints on the Diagnostic Result

In the following, the result of the decentralised diagnosis will be further analysed by investigating the effects of the information structure constraints. Possible sources for information loss will be examined and it will be determined how the loss of information influences the diagnostic result. The possible sources are the internal coupling error and the fault coupling error.

5.4.1. Error Due to Lack of Information About Internal Coupling

Restricting the information structure such that the automata are treated as independent units may lead to an increased number of fault candidates as compared to the centralised diagnostic approach. This effect can already be seen on a simple setup consisting out of two nondeterministic automata in a serial connection as depicted in Figure 5.3. The difference of the results between the centralised and decentralised diagnostic approaches for this network is derived in the following. In the centralised
5.4. Impact of the Information Structure Constraints on the Diagnostic Result

Figure 5.3.: Decentralised diagnosis of two automata in a serial connection

Approach the initial projection-step \((k = 0)\) of the diagnostic algorithm as in Equation (4.10) is given by

\[
\text{Poss}(f_1, f_2, v_1(0), v_2(0), w_1(0), w_2(0)) = \\
\bigvee_{z_1(0), z_2(0)} \bigvee_{z_1(0)} \bigvee_{z_2(1)} L_1^n(z_1(1), w_1(0), s(0), z_1(0), v_1(0), f_1) \land \\
L_2^n(z_2(1), w_2(0), z_2(0), v_2(0), s(0), f_2) \land \text{Poss}(z_1(0), f_1) \land \text{Poss}(z_2(0), f_2),
\]

where \(L_1^n\) and \(L_2^n\) are the characteristic functions of the two automata of the network. The boolean OR-operations for all current and future states can be applied separately for the respective automata. However, since the value of the signal \(s\) has to be identical for both automata, operations involving this signal cannot be separated:

\[
\text{Poss}(f_1, f_2, v_1(0), v_2(0), w_1(0), w_2(0)) = \\
\bigvee_{s(0)} \left[ \bigvee_{z_1(0), z_1(1)} L_1^n(z_1(1), w_1(0), s(0), z_1(0), v_1(0), f_1) \land \text{Poss}(z_1(0), f_1) \right] \land \\
\left[ \bigvee_{z_2(0), z_2(1)} L_2^n(z_2(1), w_2(0), z_2(0), v_2(0), s(0), f_2) \land \text{Poss}(z_2(0), f_2) \right].
\] (5.14)
With the information structure constraints given in decentralised diagnosis the coupling signal $s$ is regarded as an independent signal by each local diagnoser:

\[
\text{Poss}(f_1, f_2, v_1(0), w_1(0), v_2(0), w_2(0)) \neq \text{Poss}(f_1, v_1(0), w_1(0)) \land \text{Poss}(f_2, v_2(0), w_2(0)) = \\
\left( \bigvee_{z_1(0), z_1(1), s(0)} L_1^n(z_1(1), w_1(0), s(0), z_1(0), v_1(0), f_1) \land \text{Poss}(z_1(0), f_1) \right) \land \\
\left( \bigvee_{z_2(0), z_2(1), s(0)} L_2^n(z_2(1), w_2(0), z_2(0), v_2(0), s(0), f_2) \land \text{Poss}(z_2(0), f_2) \right).
\] (5.15)

The inequality in (5.15) holds, because additional signal combinations are considered during diagnosis which are not physically possible, namely that the signal $s$ has different values for the automata which it connects. The difference between the left and right side of the inequality in (5.15) is

\[
\text{Poss}(f_1, v_1(0), w_1(0)) \land \text{Poss}(f_2, v_2(0), w_2(0)) \land \\
\neg \text{Poss}(f_1, f_2, v_1(0), w_1(0), v_2(0), w_2(0)) = \\
\bigvee_{s^*(0), s^{**}(0)} \left( \bigvee_{z_1(0), z_1(1)} L_1^n(z_1(1), w_1(0), s^*(0), z_1(0), v_1(0), f_1) \land \text{Poss}(z_1(0), f_1) \land \\
L_2^n(z_2(1), w_2(0), z_2(0), v_2(0), s^{**}(0), f_2) \land \text{Poss}(z_2(0), f_2) \right),
\] (5.16)

where $s^* \neq s^{**}$. The superscripts ($*$) and ($**$) indicate that the value of the signal $s$ is assigned independently for the respective Diagnosers 1 and 2.

**Theorem 5.2 (Increased set of fault candidates):**

The diagnostic result (5.6) of the decentralised diagnosis is a superset of the result (4.7) of the centralised approach:

\[
\mathcal{F}_{\text{dec}}(k) \supseteq \mathcal{F}_{\text{centr}}(k) = \mathcal{F}^\star(k).
\]

**PROOF:**

See Appendix B.2.5. 

\[\Box\]
5.4. Impact of the Information Structure Constraints on the Diagnostic Result

**Corollary 5.1:** The diagnostic result $F_{\text{dec}}(k)$ according to Equation (5.6) is complete, but not sound. ⋄

**Definition 5.1 (Spurious solutions):** The solutions in $F_{\text{dec}}(k) \setminus F^*(k)$ are called *spurious solutions*. ⋄

The spurious solutions are a direct consequence of treating the automata as independent units in spite of their coupling and the demand for completeness of the diagnostic result. They are generated by any two independent diagnosers $D_i$ and $D_j$ with $i \neq j$ whenever they are connected through a coupling signal, i.e. $s_i \cap s_j = s^* \neq \emptyset$ holds. Only those tuples are possible for which $s^*$ holds the same value for all automata it connects. However, the independence of the diagnosers results unavoidably in additional tuples in which different values for $s^*$ occur for the connected automata. This generation of spurious solutions is illustrated in the following example.

**Example 10:**
Given is the network depicted in Fig. 5.3. The Tables 5.2(a) and (b) contain the automata relations after the selection of the measurements $v$ and $w$. Although $A_1$ can only generate the value $s^1 = 1$, this is unknown to the diagnoser of $A_2$, which has to assume that both values $s^1 = 1$ and $s^1 = 2$ are possible. The diagnostic result of $D_2$ is $F_2 = \{1,2\}$. If the information about $s^1 = 1$ had been available this result would be $F_2 = \{1\}$.

<table>
<thead>
<tr>
<th>Table 5.2.: Tables for Example 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) $A_1$</td>
</tr>
<tr>
<td>$z_1(k+1)$ &amp; $z_1(k)$ &amp; $s^1(k)$ &amp; $f_1$</td>
</tr>
<tr>
<td>1 &amp; 1 &amp; 1 &amp; 1</td>
</tr>
<tr>
<td>2 &amp; 1 &amp; 1 &amp; 1</td>
</tr>
<tr>
<td>2 &amp; 2 &amp; 1 &amp; 1</td>
</tr>
</tbody>
</table>

**5.4.2. Fault Coupling Error**

Consider that a fault $f$ is influencing multiple components simultaneously as depicted in Fig. 5.4. For example both components can be powered by the same power supply whose failure is denoted by this fault $f$. If a fault case is excluded by diagnoser $D_i$, but not by diagnoser $D_j$, this fault case is still included in the global fault set as a result of the Cartesian product (5.6). However, because a fault case is only excluded from the local diagnostic result if it is absolutely certain that the fault cannot have
occurred, it should also be excluded from the global diagnostic result.

**Definition 5.2 (Coupling fault):** A fault signal $f^i$ whose attribute occurs in the schema of the relations of at least two components is called *coupling fault*. $\diamond$

![Figure 5.4: One fault influences two components](image)

The solution is to calculate the diagnostic result by applying

$$F_{\text{dec}}(k) = F_1(k) \Join F_2(k) \Join \cdots \Join F_\gamma(k).$$

(5.17)

As opposed to Equation (5.6) this ensures that fault cases which are excluded locally are also excluded globally.

**Lemma 5.1:** If a coupling fault has been excluded from at least one local result set $F_i$, it is excluded from the global diagnostic result obtained by applying Equation (5.17). $\diamond$

**Proof:**
Because the natural join fulfills the commutative and associative laws, the proof can be restricted without loss of generality to a network with two components. For a larger network the operator is applied repeatedly. The natural join operation amounts to

$$F_{\text{dec}}(k) = F_1 \Join F_2$$

$$= \pi_f \sigma_{F_1.f = F_2.f} F_1 \times F_2.$$

The selection operation with the constraint $F_1.f^1 = F_2.f^1$ removes all tuples from the Cartesian product for which the fault $f^1$ has different values for the two result sets. Thus, if a fault case has been excluded in one result set, all tuples with this fault case are removed from the Cartesian product and, hence, from the global result. $\blacksquare$

Hence, the application of the natural join operation reduces the number of spurious solutions by considering the coupling through the fault signals. This conclusion is
applied in the next chapter in the coordinated diagnostic approach with respect to the coupling through the signals in $s$ to further refine the diagnostic result.

**Example 11:**
Consider the example system depicted in Figure 5.5 consisting of two parallel components which are both influenced by the fault $f$. The behavioural relations of the components are given in Table 5.3. At the beginning of the diagnosis no information about the fault or the system’s state is known. In the first step the following measurements

\[ v^1 = 1, \quad w^1 = 1, \quad v^2 = 1, \quad w^2 = 1. \]

yield the local diagnostic result $F_1 = \{ f = 1 \}$. The fault case $f = 2$ is excluded and the result of component $A_2$ amounts to $F_2 = \{ f = 1, f = 2 \}$. The cartesian product of the two result sets amounts to

\[ F_{\text{dec}} = F_1 \times F_2 = \{(f = 1, f = 1), (f = 1, f = 2)\}. \]

The natural join resolves the conflict caused by contradictory results by removing the tuple $(f = 1, f = 2)$ and the redundant attribute:

\[ F'_{\text{dec}} = F_1 \bowtie F_2 = \{ f = 1 \}. \]

\[ \square \]

![Diagram](image)

**Figure 5.5.:** Example system with a coupling fault

### 5.5. Behavioural View on the Lack of Soundness

The quality reduction of the diagnostic result in decentralised diagnosis has been pointed out in several publications [20, 59, 114]. In this section the formal conclusions of the previous sections, namely that the global result of the decentralised diagnosis is not sound, whereas the results of the local diagnosers do possess the soundness property, is interpreted by means of the system behaviour.

Consider a network $A_\sharp$ depicted in Figure 5.6a with the behaviour given in Figure 5.6b. Clearly, the measured sequence $B$ is only consistent with the behaviour


5. Decentralised Diagnosis of Nondeterministic Automata Networks

Table 5.3.: Tables for Example 2

<table>
<thead>
<tr>
<th></th>
<th>$z'_1$</th>
<th>$w^1$</th>
<th>$z_1$</th>
<th>$v^1$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$z'_2$</th>
<th>$w^2$</th>
<th>$z_2$</th>
<th>$v^2$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

$B(f = 2)$. Thus, the ideal result of the diagnosis amounts to $\mathcal{F}^* = \{2\}$. In case of the measurement sequence $A$ the diagnostic result is the empty set, because the measurement is inconsistent with both behaviours.

In decentralised diagnosis the behaviours of the local components are used for diagnosis. These behaviours, obtained by projecting the behaviour from Fig. 5.6 onto the local signal sequences $W^1 \times V$ and $W^2 \times V$, are depicted in Figure 5.7. Each local diagnoser returns the best possible diagnostic result given the local behaviour and measurements, i.e. the local results are complete and sound. The same remarks as for the diagnosis of the single automaton apply.

However, by treating the components as independent units, the dependence of the signals $w^1$ and $w^2$ visible in Figure 5.7c is ignored. Thus, the resulting behaviour of the overall system used in decentralised diagnosis is given by the cartesian product of the local behaviours: $\tilde{B} = B_1 \times B_2$. This is depicted in Figure 5.8. It can be seen that the measurement sequence $A$ is now consistent with the behaviour $\tilde{B}(f = 1)$. The set of faults in the result of the decentralised diagnosis increases to $\{1, 2\}$. Formally, the relation

$$\{ f | B(k) \in B(f) \} \subseteq \{ f | B(k) \in \tilde{B}(f) \} \quad (5.18)$$

holds, because $B \subseteq \tilde{B}$ holds.
5.6. Diagnosability

Since it has been stated in Theorem 5.2 that the diagnostic results of the centralised and the decentralised approach are not identical, it is self-evident that the prospect of success of diagnosing a system fault is also not identical for both approaches. However, this insight is in contradiction with treating diagnosability as a system property which is independent of the diagnostic approach. Therefore, a new type of diagnosability, the \textit{A-diagnosability}\textsuperscript{1}, is defined to be able to compare the different diagnostic approaches regarding diagnosability. The A-diagnosability indicates the prospect of success of diagnosing a system fault with a certain diagnostic approach and a given information structure.

\textsuperscript{1}The “A” in A-diagnosability marks its dependence on an algorithm
5. Decentralised Diagnosis of Nondeterministic Automata Networks

Definition 5.3: A process is not a-diagnosable if there does not exist an input sequence $V$ such that $(V, W) \notin B(f), \forall f \in \mathcal{N}_f$ holds using a given diagnostic algorithm. 

From this definition the following lemma can be stated directly without proof.

Lemma 5.2: If a process is not diagnosable it is also not a-diagnosable for any diagnostic algorithm.

The above definition can be extended analogously to Definition 4.1 on Page 60:

Definition 5.4: A process is not partially a-diagnosable if there exists a subset of faults $\tilde{\mathcal{N}}_f \subseteq \mathcal{N}_f$ for which there does not exist an input sequence $V$ such that $(V, W) \notin B(f), \forall f \in \tilde{\mathcal{N}}_f$ holds for a given diagnostic algorithm.

As stated in Section 5.2 the decentralised diagnosis is performed in two steps. In the first step all $i$ components of the network are diagnosed independently by the local diagnosers $D_i$ returning the local diagnostic result $F_i$. In the second step the global diagnostic result $F_{dec}$ is obtained by calculating the Cartesian product (Equation (5.6)). For the first step (the result of Algorithm 5.1) Lemma 4.2 and Corollary 4.2 apply directly.

Theorem 5.3:
A nondeterministic process represented by an automata network $A_\sharp$ is not partially a-diagnosable with respect to the decentralised diagnostic approach if at least one automaton of the network is not (partially) diagnosable.

Figure 5.8.: The network behaviour resulting from the cartesian product of the behaviours of the two components depicted in Figure 5.7a and b
5.7. Conclusion

PROOF:
If the \( i \)-th automaton of the network is not diagnosable, then
\[
\text{Poss}(z'_i, \bar{w}_i, z_i, \bar{v}_i, f_i) = \text{Poss}(z'_i, \bar{w}_i, z_i, \bar{v}_i)
\]
holds according to Lemma 4.2 and as a consequence \( \mathcal{F}_i(k) = \mathcal{N}_{f_i} \) holds for all \( k \).
Applying Equation (5.6) yields
\[
\mathcal{F}_{\text{dec}}(k) = \mathcal{F}_1(k) \times \mathcal{F}_2(k) \times \cdots \times \mathcal{N}_{f_i} \times \cdots \times \mathcal{F}_\gamma(k).
\]
This proves that using the decentralised approach it cannot be distinguished between the different fault cases in \( \mathcal{N}_{f_i} \). The argumentation can be done analogously for not partially diagnosable automata.

As opposed to the centralised approach a single not diagnosable subprocess renders the full diagnostic system not partially diagnosable. Since no information is exchanged between the diagnosers the lack of information in one part of the network cannot be compensated through information from another part.

5.7. Conclusion

In decentralised diagnosis the memory consumption and the computational costs are greatly reduced in comparison with the centralised approach. The reduction of the computational costs is a direct consequence of the modularisation and the exploitation of the system’s structure. The order of computational complexity of the decentralised diagnostic algorithm is identical to that of the centralised algorithm. But because the size of the automaton model is far smaller than the size of the equivalent automaton the decentralised diagnostic task is cheaper to solve. The approach is very reliable, because the failure of one local diagnoser or the outfall of a signal measurement do not cause the breakdown of the full diagnostic system. If a component changes only this component has to be remodelled as opposed to the centralised approach where the full model has to be rebuilt. Hence, the reusability of the decentralised approach is higher. An additional advantage is the possibility to distribute the diagnostic task among computer equipment and personnel.

The drawback of the decentralised approach is the lower quality of the diagnostic result, i.e. the loss of the soundness property. In general, the result set of the decentralised diagnosis includes additional fault cases in comparison to the result of the centralised approach: \( \mathcal{F}_{\text{dec}} \supseteq \mathcal{F}_{\text{centr}} \). As the completeness property of the result is retained the user of this approach is on the safe side, because a system is never diagnosed
5. Decentralised Diagnosis of Nondeterministic Automata Networks

as faultless if a fault has occurred. The aim of the following section is therefore to develop a method which combines the advantages of both approaches: the quality of the centralised approach and the computational costs of the decentralised approach.
The aim of this chapter is to develop a diagnostic method which combines the advantages of the approaches presented in the two previous chapters: the computational advantage of the decentralised approach and the quality of the centralised approach. This is achieved by introducing an additional component – the coordinator – which postprocesses the local diagnostic results as described in Section 6.1. In Sections 6.2 and 6.3 the realisations of the coordinator are presented. Diagnosability is investigated in Section 6.4. The chapter closes with a comparison of the different diagnostic approaches presented so far.

6.1. Information Structure Revisited

In the previous chapter it was shown that the result of a decentralised diagnosis is a superset of the centralised diagnostic result:

\[ \mathcal{F}_{\text{dec}}(k) \supseteq \mathcal{F}_{\text{centr}}(k). \]

It is important to note that this relation is a direct consequence of the information structure constraints and neither specific to the chosen type of model nor the presented algorithm. For a better diagnostic result an alternative information structure has to be used.

Several approaches can be found in literature which tackle this problem by introducing communication links between the local diagnosers or by allowing each diagnoser to incorporate the behaviour of the neighbouring systems (e.g. [13, 60, 95]). The approach investigated in this section introduces an additional component to
coordinate the operation of the local diagnosers. Different coordinators have been proposed for finite state machines and Petri-nets in [29, 38, 44]. The here presented approach differs from the referred approaches in several points. Firstly, the underlying models differ. Whereas in literature solely asynchronous discrete-event systems are investigated the presented approach concerns synchronous subsystems where a global clock determines the progress of time and all state transitions of all subsystems occur simultaneously. Additionally, faults are interpreted as unmeasurable signals which influence the system’s behaviour, whereas in the mentioned approaches faults are interpreted as single unmeasurable events. Secondly, in the presented approach the coupling signals between the subsystems may be not measurable. In the referenced approaches it is assumed that all coupling (or common) events are measurable. Thirdly, the model representation and the diagnostic approach presented in this chapter are based on relational algebra and database theory.

As opposed to the decentralised diagnostic method presented in the previous chapter the values of the coupling signals \( s \) are now not generally treated as possible. Instead, the objective of the introduced coordinator is to estimate the set of possible values of the coupling signals to reduce the number of fault candidates. This is achieved in two steps.

1. All local diagnosers \( D_{ci} \) observe the local unmeasurable signals \( s_i \) in addition to the fault signals \( f_i \). The sets of possible pairs \( R_i = (s_i, f_i) \) are send to the coordinator.

2. The coordinator compares the observations and removes contradictory results.

In the following, two realisations of this approach are derived for nondeterministic automata networks. In the unidirectional coordination approach (Section 6.2) the local results are sent to a coordinator for postprocessing while in the bidirectional coordination approach (Section 6.3) the refined diagnostic results are additionally sent back to the local diagnosers [113, 116]. It is clear that the result of the coordinated diagnosis can only be as good as that of the centralised approach, which has access to the full system information. The aim is to get as close as possible to the result of the ideal diagnostic result \( F^\star \). In the course of the next sections it is shown that the result of a decentralised diagnosis of a NAN can be improved greatly by a coordinator which performs the natural join operation as introduced in Section 3.1 on the local observations. Because both approaches are based on relational algebra, only automata based on relational data models are discussed in this chapter.
6.2. Coordination with Unidirectional Information Flow

The approach to coordinated diagnosis with unidirectional information flow – short unidirectional coordination – is depicted in Figure 6.1. As described in the beginning of this chapter the local diagnosers \( D_{ci} \) calculate the possibility of the pairs \( (s_i, f_i) \). These sets \( \mathcal{R}_i \) of possible pairs are sent to the coordinator, which then performs the natural join operation and projects the result onto the fault space. The result is a refined set of faults \( \mathcal{F}_{uni}(k) \).

![Diagram](image)

Figure 6.1.: Coordination of the local diagnostic results. The flow of information is unidirectional as indicated by the arrow on the right-hand side.

Because of this setup the diagnostic problem can be solved in two separate steps. In the first step the local diagnostic problems are solved, i.e. the sets \( \mathcal{R}_i \) are determined by the local diagnosers. This signal observation problem can be formulated as follows:

### Signal observation problem for nondeterministic automata

**Given:**
- Automaton \( A^n_i \) with relation \( L_i^n(z_{i}(k+1), w_i(k), z_i(k), v_i(k), f_i) \)
- \( \tilde{v}_i \subseteq v_i \) and \( \tilde{w}_i \subseteq w_i \) are measurable
- A-priori initial condition \( Pre_i(-1) \)

**Find:**
- Set \( \mathcal{R}_i(k) \) of possible pairs \( (s_i(k), f_i) \)

The a-priori initial condition \( Pre_i(-1) \) with \( sch(Pre_i(k)) = \{z_{i}(k+1), f_i\} \) includes all possible initial state and fault combinations. This problem can be solved by applying Algorithm 5.1 with an altered projection step (5.8) in which the coupling signals are not eliminated by the projection operator \( \pi \). That is, it is necessary to change the
local diagnostic method such that the schema of the results of the local diagnosers are given by \( \text{sch}(\mathcal{R}_i(k)) = \{ f_i, s_i(k) \} \). The new projection step is then given by

\[
\mathcal{R}_i(k) = \pi_{f_i,s_i(k)}(\sigma_{(z_i(k), f_i) \in \mathcal{P}_{\text{ref}}(k-1), \mathcal{P}_{\text{ref}}(k), \mathcal{P}_{\text{ref}}(k)} L_i)
\]

with

\[
\mathcal{P}_{\text{ref}}(k) = \pi_{z_i(k+1), f_i}(\sigma_{(z_i(k), f_i) \in \mathcal{P}_{\text{ref}}(k-1), \mathcal{P}_{\text{ref}}(k), \mathcal{P}_{\text{ref}}(k)} L_i^n)
\]

as given Equation (5.7).

Without proof it is clear that exactly those faults \( f_i \) are included in the set \( \mathcal{R}_i \) which are included in the result \( \mathcal{F}_i \) of the local diagnosis (5.7)–(5.8). Therefore, the same remarks apply. Additionally, the set \( \mathcal{R}_i \) includes the information for which values of the coupling signals \( s_i \) the faults could have occurred. The aim of the coordination is to determine a complete set of faults \( \mathcal{F}_{\text{uni}}(k) \) which contains less spurious solutions than the result of the decentralised approach \( \mathcal{F}_{\text{dec}}(k) \). This problem can be stated as follows:

**Unidirectional coordination problem**

Given: - Sets of possible pairs \( \mathcal{R}_i(s_i(k), f_i), \forall i \in \{1, \ldots, \gamma\} \)

Find: - Set of faults \( \mathcal{F}_{\text{uni}}(k) \)

This problem is solved by introducing a coordinator which refines the diagnostic results by performing the natural join operation

\[
\mathcal{F}_{\text{uni}}(k) = \pi_f (\mathcal{R}_1(k) \bowtie \mathcal{R}_2(k) \bowtie \cdots \bowtie \mathcal{R}_\gamma(k)).
\]

This operation joins two relations and eliminates exactly those tuples for which the attributes, which exist in both relations, have different values. In other words, contradicting local diagnostic results are removed. This is further clarified in the example on Page 99. The algorithm solving both problems is given below.
6.2. Coordination with Unidirectional Information Flow

**Algorithm 6.1** (Unidirectional coordinated diagnosis of a NAN)

Initialise: $k = 0$

1. For all $i$ automata do *(local diagnosis)*
   a) Measure $\bar{v}_i(k)$ and $\bar{w}_i(k)$
   b) Apply Equations (5.7) and (6.1)
   c) Send result $\mathcal{R}_i(k)$ to coordinator

2. Apply Equation (6.2) *(coordinator)*

3. Stop on user demand, else $k := k + 1$

4. Repeat from Step 1

Result: $\mathcal{F}_{uni}(k)$

It is now investigated if this approach fulfills the completeness and soundness demand.

**Theorem 6.1 (Diagnosis of static systems):**

In case of the diagnosis of static systems the result of Algorithm 6.1 is identical to the result of the centralised diagnosis, i.e.

$$
\mathcal{F}_{uni}(0) = \mathcal{F}_{centr}(0) = \mathcal{F}^\star(0)
$$

(6.3)

holds.

PROOF:

Because the natural join fulfils the commutative and associative laws, the proof can be denoted without loss of generality for two coupled automata (Fig. 6.2). With the abbreviation $T_i := (z_i(0), f_i) \in \text{Pre}_i(-1)$ the cartesian product of $\mathcal{R}_1$ and $\mathcal{R}_2$ amounts to:

$$
\mathcal{R}_1 \times \mathcal{R}_2 = (\pi_{f_1, s_1}(\sigma_{T_1, \bar{v}_1, \bar{w}_1, L_1})) \times (\pi_{f_2, s_2}(\sigma_{T_2, \bar{v}_2, \bar{w}_2, L_2}))
$$

$$
= \pi_{f_1, f_2, \mathcal{R}_1, s_1, \mathcal{R}_2, s_2}(\sigma_{T_1, \bar{v}_1, \bar{w}_1, \bar{w}_2, \bar{L}_1 \times L_2})
$$

$$
= \pi_{f_1, f_2, \mathcal{R}_1, s_1, \mathcal{R}_2, s_2}(\sigma_{T_1, T_2, \bar{f}, v, \bar{w}, \bar{L}_1 \times L_2}).
$$

(6.4)

As a reminder, the values of an attribute $a$ of a specific relation $M$ are addressed by $M.a$. The result of the coordinated diagnosis is then calculated by applying the natural
join operator on the local results:

\[ F_{\text{uni}}(0) = \pi_{f_1,f_2}(R_1(0) \bowtie R_2(0)) \]
\[ = \pi_{f_1,f_2}(\sigma_{R_1.s_1(0)=R_2.s_1(0)}(R_1(0) \times R_2(0))) \]
\[ = \pi_{f_1,f_2}(\sigma_{R_1.s_1(0)=R_2.s_1(0)}(\pi_{f_1,f_2,R_1(0),s_1,R_2(0),s_1}(\sigma_{T_1.T_2.v.w}(L_1 \times L_2)))) \]
\[ = \pi_{f_1,f_2}(\sigma_{T_1.T_2.v.w,R_1.s_1(0)=R_2.s_1(0)}(L_1 \times L_2)). \tag{6.4} \]

With the composition as described in Section 3.3.4 and the initial condition \( \text{Pre}_{\text{full}}(-1) = \text{Pre}_1(-1) \times \text{Pre}_2(-1) \) of the equivalent automaton of the network and its relation \( L_{\text{full}} \) the result of the centralised diagnosis is given as

\[ F_{\text{centr}}(0) = \pi_{f_1,f_2}(\sigma_{z_1(0),z_2(0),f_1,f_2} \in \text{Pre}_{\text{full}}(-1),v,w,L_{\text{full}}) \]
\[ = \pi_{f_1,f_2}(\sigma_{T_1,T_2,v.w,R_1.s_1(0)=R_2.s_1(0)}(L_1 \times L_2)) \]
\[ = F_{\text{uni}}(0), \]
which proves the theorem. \( \blacksquare \)

Figure 6.2.: Unidirectional coordinated diagnosis of two automata in a serial connection

**Theorem 6.2 (Diagnosis of dynamic systems):**

*In case of the diagnosis of dynamic systems the result of Algorithm 6.1 is a superset of the result of the centralised diagnosis, i.e.*

\[ F_{\text{uni}}(k) \supseteq F_{\text{centr}}(k) = F^*(k) \tag{6.5} \]

*holds for all \( k > 0 \). \( \square \)
6.2. Coordination with Unidirectional Information Flow

**Proof:**
This proof will be held analogously to the previous proof. For $k = 1$ the result of the centralised diagnosis is given by

$$F_{\text{centr}}(1) = \pi_{f_1 . f_2} \left( \sigma_{z_1 (1), z_2 (1), f_1 , f_2} \in \text{Pre}_{\text{full}}(0), v, w, R_1, s_1 (1) = R_2, s_1 (1) (L_1 \times L_2) \right)$$

with

$$\text{Pre}_{\text{full}}(0) = \pi_{z_1 (1), z_2 (1), f_1 , f_2} \left( \sigma_{\mathcal{T}_1, \mathcal{T}_2, v, w, R_1, s_1 (0) = R_2, s_1 (0) (L_1 \times L_2)} \right). \quad (6.6)$$

Analogously, the coordinated diagnosis yields

$$F_{\text{uni}}(1) = \pi_{f_1 . f_2} \left( \sigma_{z_1 (1), f_1} \in \text{Pre}_1(0), (z_2 (1), f_2) \in \text{Pre}_2(0), v, w, R_1, s_1 (1) = R_2, s_1 (1) (L_1 \times L_2) \right)$$

with

$$\text{Pre}_i(0) = \pi_{z_i (1), f_i} \left( \sigma_{\mathcal{T}_i, v, w_i} (L_i) \right). \quad (6.7)$$

The selection condition in (6.7) is less restrictive than in (6.6), because no restriction $R_1, s_1 (0) = R_2, s_1 (0)$ is made. Therefore,

$$\text{Pre}_1(0) \times \text{Pre}_2(0) \supseteq \text{Pre}_{\text{full}}(0)$$

holds, which directly yields $F_{\text{uni}}(1) \supseteq F_{\text{centr}}(1)$. Because of the Markov property of the NAN, this result can be generalised to all $k > 0$.  

**Corollary 6.1:** The result $F_{\text{uni}}(k)$ of Algorithm 6.1 is complete, but not sound. \(\Diamond\)

This follows directly from the completeness and soundness of $F_{\text{centr}}(k)$ together with Theorems 6.1 and 6.2.

**Corollary 6.2:** For the results of the different diagnostic approaches the relation holds

$$F_{\text{dec}}(k) \supseteq F_{\text{uni}}(k) \supseteq F_{\text{centr}}(k) = F^\star(k)$$

for all $k$. \(\Diamond\)

It is clear from the proofs of Theorems 6.1 and 6.2 that the coordination reduces the number of spurious solutions compared to the decentralised diagnostic approach. With this it has been shown that the above algorithm fulfills the set objectives. The above theorems will be illustrated by the following example.
6. Coordinated Diagnosis of Nondeterministic Automata Networks

Example 12:
Consider the network depicted in Fig. 6.2 with the relations given in Tables 6.1(a)-(b). Without loss of generality it is assumed that the measurements are constant and the respective part of the relations has already been selected. With the initial condition

\[ \text{Pre}_1(-1) = \{(z_1(0) = 1, f_1 = 1), (z_1(0) = 1, f_1 = 2)\} \]
\[ \text{Pre}_2(-1) = \{(z_2(0) = 1, f_2 = 1), (z_2(0) = 1, f_2 = 2)\} \]

the first two steps of the coordinated diagnosis yield

\[ \text{Pre}_1(0) = \{(z_1(1) = 1, f_1 = 1), (z_1(1) = 1, f_1 = 2)\} \]
\[ \text{Pre}_2(0) = \{(z_2(1) = 2, f_2 = 1), (z_2(1) = 1, f_2 = 2)\} \]
\[ \mathcal{R}_1(0) = \{(s^1(0) = 1, f_1 = 1), (s^1(0) = 2, f_1 = 2)\} \]
\[ \mathcal{R}_2(0) = \{(s^1(0) = 1, f_2 = 1), (s^1(0) = 1, f_2 = 2)\} \]

The cartesian product of the local results is given below.

<table>
<thead>
<tr>
<th>(\mathcal{R}_1 \times s^1)</th>
<th>(f_1)</th>
<th>(\mathcal{R}_2 \times s^1)</th>
<th>(f_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

The natural join \(\mathcal{R}_1 \bowtie \mathcal{R}_2\) removes the last two lines, because \(s^1\) has different values for the two relations. The result of the coordination is therefore

\[ F_{uni}(0) = \{(f_1 = 1, f_2 = 1), (f_1 = 1, f_2 = 2)\} \]

The next diagnostic step is calculated analogously:

\[ \mathcal{R}_1(1) = \mathcal{R}_1(0) \]
\[ \mathcal{R}_2(1) = \{(s^1(1) = 1, f_2 = 2), (s^1(1) = 2, f_2 = 1)\} \]
\[ F_{uni}(1) = \{(f_1 = 1, f_2 = 2), (f_1 = 2, f_2 = 1)\} \]

To compare this result to the centralised diagnosis the equivalent automaton is now derived by applying the composition rules on the network. Its relation is given Table 6.1(c). With the initial condition \(\text{Pre}_{full}(-1) = \text{Pre}_1(-1) \times \text{Pre}_2(-1)\) the first two steps of the centralised diagnostic algorithm yield

\[ F_{cent}(0) = \{(f_1 = 1, f_2 = 1), (f_1 = 1, f_2 = 2)\} \]
\[ F_{cent}(1) = \{(f_1 = 1, f_2 = 2)\} \]

100
Therefore, $\mathcal{F}_{uni}(0) = \mathcal{F}_{centr}(0)$ and $\mathcal{F}_{uni}(1) \supset \mathcal{F}_{centr}(1)$ hold, which is in accordance with Theorems 6.1 and 6.2.

Table 6.1.: Tables for Example 12

<table>
<thead>
<tr>
<th></th>
<th>(A_1^n)</th>
<th></th>
<th>(A_2^n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(z_1(k+1))</td>
<td>(z_1(k))</td>
<td>(s^1(k))</td>
<td>(f_1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\(\mathcal{R}_i(f_i, s_i(0), s_i(1), \ldots, s_i(k)). \tag{6.8}\)

Both approaches have the drawback that more and more values have to be included into the calculation at every time step. To use the information implicitly means to use the improvement obtained by the postprocessing to refine the next local diagnostic steps. In other words, the information gained by the coordinator must be put to the benefit of the diagnosers. This is realised in the next section.

**Complexity considerations.**

**Memory consumption:** The memory consumption of the network model stays unchanged. The memory consumption of the coordinator is very low, because it has to store no model. The structural information needed for the natural join operation is fully defined through the attributes sent by the local diagnosers. During the natural
join operation the memory advantage is also retained. The operation combines the relations containing the local fault/coupling-pairs, which are far smaller relations than the behavioural relations of the local models. In the worst case the cartesian product of these relations contains only

\[ |K(f)| \cdot |K(s)| \]

entries which is of the order \(O(S^\sigma Q^\kappa)\). However, modern database programs provide memory efficient implementations of the join operation which do not build the full cartesian product, but perform the operation for each tuple separately which reduces the order to \(O(S^\sigma)\). Hence, the memory consumption of the coordinator is negligible in comparison to the costs of storing the model.

**Computational costs:** As opposed to the decentralised diagnosis the coupling signals cannot be eliminated from the behavioural relations, because they play an essential role in the coordination. This makes the local diagnosis more expensive. Each local diagnoser has to perform

\[
(NQ + NQ - 1)NS + (NN + NN - 1)QS
\]

arithmetic operations, which is of the order \(O(N^2SQ)\) as opposed to \(O(N^2S)\) in the decentralised case. For both methods, however, the dominating argument is \(N^2\). An upper bound for the additional costs caused by the coordinator is the order of the cartesian product: \(O(S^\sigma Q^\kappa)\), but many database programs provide methods with a distinctly lower number of computations [4]. In general, the result sets \(R_i\) of the local diagnosis are small, especially in comparison with the full behavioural relations of the automata, and cause only little computational costs in the coordination. Clearly, the centralised diagnosis of the equivalent automaton is more costly, because the cartesian product of the behavioural relations is larger, but it is apparent that especially a high number of coupling signals cause high computational costs in the coordinated diagnostic approach.

### 6.3. Coordination with Bidirectional Information Flow

The aim of the coordination with bidirectional information flow – short bidirectional coordination – is to improve the diagnostic result by sharing the information gained by the coordinator with the local diagnosers. As will be shown in the remainder of this section this bidirectional communication link leads to a better diagnostic result compared to the coordinated approach with an unidirectional information flow. The
structure of the diagnostic approach is depicted in Figure 6.3. The diagnostic problem for the bidirectional coordination can then be formulated as follows:

![Diagram of Coordination of the local diagnostic results. The flow of information is bidirectional as indicated by the arrow on the right-hand side.](image)

**Diagnostic problem for nondeterministic automata networks**

Given:
- Automata $A_i^n$ for all $i \in \{1, \ldots, \gamma\}$ with relations $L_i^t(z_i(k+1), w_i(k), z_i(k), v_i(k), f_i)$
- $\tilde{v}_i \subseteq v_i$ and $\tilde{w}_i \subseteq w_i$ are measurable
- A-priori initial conditions $Pre_i(-1)$

Find:
- Set $F_{bi}(k)$ of faults using local diagnosers $D_{ci}$ for the automata

The a-priori initial condition $Pre_i(-1)$ with $sch(Pre_i(k)) = \{z_i(k+1), f_i\}$ includes all possible initial state and fault combinations. This problem is solved analogously to the unidirectional approach. The diagnostic results $R_i$ of the local diagnosers with

$$R_i(k) = \pi_{f_i,s_i(k)}(\sigma_{(z_i(k), f_i)} \in Pre_i(k-1), v_i(k), w_i(k), L_i)$$

as given in Equation (6.1) are refined by the coordinator which performs the natural join operation:

$$J(k) = R_1(k) \bowtie R_2(k) \bowtie \cdots \bowtie R_{\gamma}(k). \quad (6.10)$$

The global diagnostic result is obtained by projecting the result of the coordination $J(k)$ onto the fault space:

$$F_{bi}(k) = \pi_f J(k). \quad (6.11)$$
Coordinated Diagnosis of Nondeterministic Automata Networks

As opposed to the unidirectional diagnostic approach the diagnostic result is not only output to the user, but is used to refine the “prediction”-step of the local diagnosers:

$$\mathcal{P}_{\text{re}}(k) = \pi_{z_i(k+1), f_i} (\mathcal{J}(k), (z_i(k), f_i) \in \mathcal{P}_{\text{re}}(k-1), \mathcal{V}_i(k), \mathcal{W}_i(k)(L_i)). \quad (6.12)$$

Thereby, the selection condition $$(s_i(k), f_i) \in \mathcal{J}(k)$$ ensures that only tuples included in the coordination result (6.10) are used in the next diagnostic step. In other words, all pairs which have been excluded by the coordinator will not be used in the following diagnostic step. The algorithm solving the diagnostic problem is given below.

**Algorithm 6.2** (Bidirectional coordinated diagnosis of a NAN)

1. **Initialise:** $k = 0$
   1. For all $i$ automata do *(Prediction)*
      a) Measure $\mathcal{V}_i(k)$ and $\mathcal{W}_i(k)$
      b) Apply Equation (6.1)
      c) Send result $\mathcal{R}_i(k)$ to coordinator
   2. **Coordination**
      a) Apply Equation (6.10)
      b) $\mathcal{F}_i(k) = \pi_{f_j} \mathcal{J}(k)$
      c) Send $\mathcal{J}(k)$ to local diagnosers
   3. For all $i$ automata apply Equation (6.12) *(Projection)*
   4. Stop on user demand, else $k := k + 1$
   5. Repeat from Step 1

**Result:** $\mathcal{F}_i(k)$

**Remark:** In this algorithm the calculations performed by the diagnosers and the coordinator are interweaved in such a way that the coordinator is the core of the diagnostic system. If the coordinator itself or the communication links to and from the coordinator fail, the full diagnostic process will come to a halt. To allow the local diagnosers to continue to function the following item can be inserted in the projection step: “if $\mathcal{J}(k)$ is unknown, set $\mathcal{J}(k) := \mathcal{N}_{s_i} \times \mathcal{N}_{f_i}$.” Analogously, in the case of a failure of a local diagnoser $\mathcal{D}_{ci}$ the local result must be set to $\mathcal{R}_i = \mathcal{N}_{s_i} \times \mathcal{N}_{f_i}$. □

**Remark:** The communication traffic between the coordinator and the local diagnosers can be reduced by sending only the relevant parts of $\mathcal{J}(k)$ to each local diagnoser: $\tilde{\mathcal{R}}_i = \pi_{s_i, f_i} \mathcal{J}$. □
In the following it is proven that this approach fulfills the stated demands.

**Theorem 6.3 (Equivalence of diagnostic results):**
The result of Algorithm 6.2 is identical to the result of the centralised diagnosis, i.e.

\[
\mathcal{F}_{bi}(k) = \mathcal{F}_{centr}(k) = \mathcal{F}^\star(k)
\]

holds for all \( k \geq 0 \).

\[\square\]

**Proof:** Because the natural join fulfills the commutative and associative laws, the proof can be restricted to two coupled automata as in Fig. 6.2 without loss of generality. With the abbreviations \( T_i(k) := (z_i(k), f_i) \in \text{Pre}_i(k - 1) \) the Cartesian product of \( R_1 \) and \( R_2 \) amounts to:

\[
R_1 \times R_2 = (\pi_{f_1,s^1}(\sigma_{T_1,v_1,w_1,L_1})) \times (\pi_{f_2,s^2}(\sigma_{T_2,v_2,w_2,L_2}))
\]

\[
= \pi_{f_1,f_2,R_1,s^1,R_2,s^2}(\sigma_{T_1,v_1,w_1,v_2,w_2}(L_1 \times L_2))
\]

(6.14)

The result of Algorithm 6.2 for \( k = 1 \) is given by

\[
\mathcal{F}_{bi}(1) = \pi_{f_1,f_2}(R_1(1) \bowtie R_2(1))
\]

\[
= \pi_{f_1,f_2}(\sigma_{R_1,s^1(1),R_2,s^2(1)}(R_1(1) \times R_2(1)))
\]

\[
= \pi_{f_1,f_2}(\sigma_{(T_1,1),T_2(1),v,w,R_1,s^1(1),R_2,s^2(1)}(L_1 \times L_2)).
\]

With the composition as described in Section 3.3.4 the result of the centralised diagnosis is given by

\[
\mathcal{F}_{centr}(1) = \pi_{f_1,f_2}(\sigma_{(z_1(1),z_2(1),f_1,f_2) \in \text{Pre}_{full}(0),v,w,L_{full}})
\]

\[
= \pi_{f_1,f_2}(\sigma_{T_{full}(1),v,w,R_1,s^1(1),R_2,s^2(1)}(L_1 \times L_2)).
\]

Both results differ only in the usage of the relation \( \text{Pre} \), i.e. it has to be proven that

\[
\sigma_{T_{full}(1),T_1(1),T_2(1)}(L_1 \times L_2)
\]

(6.15)
holds, which is investigated below. For the coordinated diagnosis

\[
\text{Pre}_i(0) = \pi_{c_i(1)} f_i \left( \sigma(s^i(0), f_i) \in J_i(0), T_i(0), \overline{v}_i, \overline{w}_i(L_i) \right) \\
= \pi_{c_i(1)} f_i \left( \sigma_{T_1(0), T_2(0), \nu, \omega, R_{1.s^i(0)} = R_{2.s^i(0)}(L_1 \times L_2) \right)
\]

(6.16)

holds for \(i = 1, 2\). For the centralised diagnosis

\[
\text{Pre}_{full}(0) = \pi_{c_1(1)} c_2(1), f_1, f_2 \left( \sigma_{T_{full}(0), \nu, \omega, R_{1.s^i(0)} = R_{2.s^i(0)}(L_1 \times L_2) \right)
\]

(6.17)

holds. Because the initial condition for the centralised diagnosis is set to

\[
\text{Pre}_{full}(−1) = \text{Pre}_1(−1) \times \text{Pre}_2(−1)
\]

the equations

\[
T_{full}(0) = T_1(0) \times T_2(0)
\]

\[
\sigma_{T_{full}(0)}(L_1 \times L_2) = \sigma_{T_1(0)}(L_1 \times L_2)
\]

hold. With (6.16)-(6.17) it follows that (6.15) holds, which proves the equivalence \(F_{bi}(1) = F_{centr}(1)\). Because of the Markov property of the NAN the proof can be extended to all time steps \(k\).

**Corollary 6.3:** The result \(F_{bi}\) of the bidirectional coordinated diagnosis is complete and sound.

This follows directly from the completeness and soundness of \(F_{centr}\) together with Theorem 6.3.

---

Figure 6.4: Bidirectional coordinated diagnosis of two automata in a serial connection
6.3. Coordination with Bidirectional Information Flow

Example 13:
Analogously to Example 12 the network depicted in Fig. 6.4 with the relations of the two automata given in Tables 6.1(a)-(b) is considered. Without restriction of generality it is assumed that the measurements are constant and the respective part of the relations has already been selected. With the initial condition

\[ Pre_1(-1) = \{(z_1(0) = 1, f_1 = 1), (z_1(0) = 1, f_1 = 2)\} \]
\[ Pre_2(-1) = \{(z_2(0) = 1, f_2 = 1), (z_2(0) = 1, f_2 = 2)\} \]

the first two steps of the bidirectional coordinated diagnosis yield

\[ R_1(0) = \{ (s_1(0) = 1, f_1 = 1), (s_1(0) = 2, f_1 = 2) \} \]
\[ R_2(0) = \{ (s_1(0) = 1, f_2 = 1), (s_1(0) = 1, f_2 = 2) \} \]
\[ F_{bi}(0) = \{ (f_1 = 1, f_2 = 1), (f_1 = 1, f_2 = 2) \} \]
\[ R_1(1) = \{ (s_1(1) = 1, f_1 = 1) \} \]
\[ R_2(1) = \{ (s_1(1) = 1, f_2 = 2), (s_1(1) = 2, f_2 = 1) \} \]
\[ F_{bi}(1) = \{ (f_1 = 1, f_2 = 2) \} . \]

The equivalent automaton is now derived by applying the composition rules on the network and its relation is given Table 6.1(c). With the the initial condition \( Pre_{full}(-1) = Pre_1(-1) \times Pre_2(-1) \) the first two steps of the centralised diagnostic algorithm yield

\[ F_{centr}(0) = \{ (f_1 = 1, f_2 = 1), (f_1 = 1, f_2 = 2) \} \]
\[ F_{centr}(1) = \{ (f_1 = 1, f_2 = 2) \} . \]

Therefore, \( F_{bi}(k) = F_{centr}(k) \) holds for all \( k \in \{0, 1\} \), which is in accordance with Theorem 6.3. □

Complexity considerations.

In comparison to the unidirectional coordinated approach the order of the memory consumption and computational costs stay unchanged. However, the bidirectional approach necessitates higher communication costs.

Discussion.

The coordinated diagnostic approach with bidirectional communication combines the advantages of the centralised and decentralised approaches. It returns the ideal diagnostic result \( F^\bullet \) while retaining the low computational costs of the decentralised approach. Thus, the primary goal of this work has been fulfilled. The main reason
for this intriguing result is the use of structural knowledge about the system. As opposed to the classical centralised diagnosis, which uses a “brute force” approach simply to test all possible signal combinations, the modular approaches exploit signal independencies to achieve a far more efficient calculation.

The disadvantage of the coordination is the introduction of a centralised component. That is, the ability to spread the diagnostic task locally is reduced (cf. Section 5.1). The reliability of this approach is as high as in the decentralised approach, because the diagnosis can continue to function as a decentralised diagnosis after the failure of the coordinator (see remarks to Algorithm 6.2).

6.4. Diagnosability

The A-diagnosability of automata networks with respect to the two coordinated diagnostic approaches can be derived directly from the diagnosability analysis for the centralised and decentralised cases.

**Theorem 6.4 (A-diagnosability for the unidirectional coordinated approach):**

A nondeterministic process represented by an automata network $A$ is

- not a-diagnosable with respect to the unidirectional coordinated diagnostic approach if the process is not diagnosable
- a-diagnosable with respect to the unidirectional coordinated diagnostic approach if the process is a-diagnosable with respect to the decentralised approach.

The first item is a direct consequence of Lemma 5.2. The second item follows from Corollary 6.2 which states that $F_{\text{dec}}(k) \supseteq F_{\text{uni}}(k)$ holds for all $k \geq 0$. That is, if a fault case is excluded in a decentralised diagnosis, because of inconsistency, it is also excluded in the coordinated approach. Hence, if a fault case can be identified in the decentralised approach it can also be identified in the coordinated approach.

**Theorem 6.5 (A-diagnosability for the bidirectional coordinated approach):**

A nondeterministic process represented by an automata network $A$ is a-diagnosable with respect to the bidirectional coordinated diagnostic approach if and only if the process is diagnosable.
6.5. Comparison of the Presented Diagnostic Approaches

This theorem follows directly from the equality of the diagnostic results $\mathcal{F}_{bi}(k) = \mathcal{F}^\star(k)$ for all $k \geq 0$ with the same argumentation as above.

**Lemma 6.1:** The diagnosability of the automata $A^n_i$ of the automata network $A^\#_i$ is not a necessary condition for the $A$-diagnosability of the network with respect to a coordinated diagnostic approach.

The proof is omitted, because it follows directly from the proof of Theorem 4.4 on Page 63 together with $\mathcal{F}_{bi}(k) = \mathcal{F}^\star(k)$.

### 6.5. Comparison of the Presented Diagnostic Approaches

In the following, the centralised, decentralised and coordinated diagnostic approaches are compared and a suggestion is given in which cases the different approaches should be applied. The objective criteria such as computational costs and memory requirements have been discussed in the respective chapters and are summarised in Table 6.2.

<table>
<thead>
<tr>
<th></th>
<th>Central</th>
<th>Decentral</th>
<th>Coordinated (bid.)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Computations</strong></td>
<td>$O(N^2 S)$</td>
<td>$O(N^2 S)$</td>
<td>$O(N^2 SQ) + O(S^\sigma Q^\kappa)$</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>$O(N^2 M^\mu R^\rho S^\sigma)$</td>
<td>$O(\gamma N^2 MRS)$</td>
<td>$O(\gamma N^2 MRS) + O(S^\sigma)$</td>
</tr>
<tr>
<td><strong>Quality of result</strong></td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td><strong>Completeness</strong></td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td><strong>Soundness</strong></td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td>bad scalability, not distributable, low reusability, low reliability</td>
<td>very good scalability, highly distributable, good reusability, good reliability</td>
<td>good scalability distributable good reusability good reliability</td>
</tr>
</tbody>
</table>

The table seems to conclude that the centralised approach returns a good result, but is too computational expensive and inflexible. The other two approaches have comparable low computational costs, whereby the coordinated approach returns a good diagnostic result. But can the coordinated approach always be recommended or might in some cases a compositional approach be more expensive than a centralised approach? This question is answered in the following.
Consider a system which has been decentralised (fragmented) to such an extent that the network contains a very high number of atomic components which are connected through a very high number of coupling signals. In such a case the information about the system’s behaviour is almost entirely contained in the coupling.\(^1\) Clearly, in case of the centralised diagnosis the result and the computation is unaffected by the appearance of the coupling network as its effect is included in the equivalent automaton.

In case of the decentralised diagnosis a high number of very small local diagnosers is used to gain the result. Clearly, if the size of the components is minimised then the size of the local diagnosers is also minimised. The same holds true for the computational costs. These costs are minimised, because the majority of the information is stored in the coupling network which is ignored in decentralised diagnosis. The drawback of the minimised costs is the degradation of the diagnostic result. As described in Chapter 5 all values of the coupling signals are considered as possible. The higher the number of these signals is, the less restrictive the condition in the selection operation (5.7) can be. As a guideline, the less information is available the less faults can be excluded.

As opposed to that, the memory requirements and the computational costs are not minimised in case of the coordinated diagnosis. As can be seen from Table 6.2 the number of coupling signals \(Q\) is included in the approximation. Clearly, the task of the coordinator is to reconstruct the values of these coupling signals. If a great amount of values have to be reconstructed, as in this thought experiment, the coordination is very computational expensive. In fact, the high effort of the coordination might render the coordinated diagnostic approach unusable.

Although it has not been investigated in this work, these observations suggest that there must be an optimum degree of decentralisation with respect to the different diagnostic approaches. If the optimum is known it can be taken into consideration right during the modelling process. However, the finding of such an optimum is beyond the scope of this book. Experiments have shown that a general guideline for the decentralisation seems to be “more must be known than unknown”. If the unmeasurable signals are the dominant factor in the network it is advisable to use a centralised diagnostic approach.

\(^1\)If the number of signals rises, at some point the organisation of the data structure has to be taken into consideration when approximating the computational costs. Especially the storage of the signal attributes causes additional memory requirements. These costs have not been considered in any of the given approximations.
Part III.

Diagnosis of Stochastic Automata Networks
Centralised Diagnosis of Stochastic Processes

In this chapter the approaches to centralised diagnosis of nondeterministic automata networks presented in Chapter 4 are extended to stochastic automata networks. It is shown that the previously developed principles can be directly applied to a stochastic process. In Section 7.1 the centralised diagnosis of a single stochastic automaton introduced in [93] is modified to be applicable to automata with multiple inputs and outputs and to relational algebra. In Section 7.2 this approach is extended to stochastic automata networks. Diagnosability is investigated in Section 7.3.

7.1. Diagnosis of a Single Automaton

7.1.1. Approach for Automata on Characteristic Functions

The aim of the centralised diagnosis of stochastic automata (SA) is to gain the fault probability distribution \( p(f|V(0\ldots k), W(0\ldots k)) \), where \( V(0\ldots k) = (v(0), \ldots, v(k)) \) and \( W(0\ldots k) = (w(0), \ldots, w(k)) \) are the input and output sequences. That is, in contrast to the diagnosis of nondeterministic automata the result not only states which fault cases are possible and which are not, but additionally returns the probability for each fault case (cf. Fig. 7.1).

As defined in Section 2.1 the ideal diagnostic result \( \mathcal{F}^\star \) of a diagnosis includes exactly those fault cases for which the measurements up to time \( k \) are consistent with the model:

\[
\mathcal{F}^\star(k) := \{ f | B(k) \in B(f) \}. \tag{7.1}
\]
Since the relation

$$P(f|V(0\ldots k),W(0\ldots k)) > 0 \iff B(k) \in \mathcal{B}(f)$$  \tag{7.2}$$
holds, the diagnostic result can be defined analogously given as

$$\mathcal{F}(k) = \{f|P(f|V(0\ldots k),W(0\ldots k)) > 0\},$$  \tag{7.3}$$
i.e. it includes exactly those faults which have a probability greater than zero. The diagnostic algorithm is initialised with the a-priori probability distribution of initial state $p(z(0))$ and fault $p(f)$. The stochastic variables of the distributions are assumed to be stochastically independent such that

$$P(z_p(0) = z(0), f_p = f) = P(z_p(0) = z(0)) \cdot P(f_p = f) \quad \forall z(0), f$$

holds. The diagnostic problem can then be stated as follows:

**Centralised fault diagnostic problem for stochastic automata**

*Given:*  - Automaton $\mathcal{A}$ with charact. function $L^i(z(k+1),w(k)|z(k),v(k),f)$
  - $v$ and $w$ are measurable
  - A-priori probability distrib. of initial state $p(z(0))$ and fault $p(f)$

*Find:*  - Fault probability distribution $p(f|V(0\ldots k),W(0\ldots k))$

This problem is solved analogously to Algorithm 4.1 in two steps projection and projection.
7.1. Diagnosis of a Single Automaton

The diagnostic algorithm is then given as follows:

The prediction step

\[
P(z(k+1), f|V(0\ldots k), \mathbf{W}(0\ldots k)) = \frac{\sum_{z(k)} L^z(z(k+1), \mathbf{w}(k)|z(k), \mathbf{v}(k), f)P(z(k), f|V(0\ldots k-1), \mathbf{W}(0\ldots k-1))}{\sum_{z(k+1)} \sum_{z(k)} \sum_{f} L^z(z(k+1), \mathbf{w}(k)|z(k), \mathbf{v}(k), f)P(z(k), f|V(0\ldots k-1), \mathbf{W}(0\ldots k-1))}
\]  

(7.4)

yields the joined probability of \(z(k+1)\) being the next state and \(f\) the fault under the condition that the sequences \(V\) and \(W\) have been measured until time \(k\). It is calculated using the result of the previous prediction step or in case of the first diagnostic step \((k = 0)\) the initial condition:

\[
P(z(1), f|v(0), w(0)) = \frac{\sum_{z(0)} L^z(z(1), w(0)|z(0), v(0), f)P(z(0), f)}{\sum_{z(0)} \sum_{f} L^z(z(0), w(0)|z(0), v(0), f)P(z(0), f)}.
\]

(7.5)

The projection step

\[
P(f|V(0\ldots k), W(0\ldots k)) = \sum_{z(k+1)} P(z(k+1), f|V(0\ldots k), W(0\ldots k))
\]

(7.6)

projects the result of the prediction step onto the fault space. The diagnostic result is the probability distribution

\[
p(f|V(0\ldots k), W(0\ldots k)) = \begin{pmatrix}
P(f^1_p = 1, f^2_p = 1, \ldots, f^\sigma_p = 1|V(0\ldots k), W(0\ldots k)) \\
P(f^1_p = 2, f^2_p = 1, \ldots, f^\sigma_p = 1|V(0\ldots k), W(0\ldots k)) \\
\vdots \\
P(f^1_p = S^1, f^2_p = 1, \ldots, f^\sigma_p = 1|V(0\ldots k), W(0\ldots k)) \\
P(f^1_p = 1, f^2_p = 2, \ldots, f^\sigma_p = 1|V(0\ldots k), W(0\ldots k)) \\
\vdots \\
P(f^1_p = S^1, f^2_p = S^2, \ldots, f^\sigma_p = S^\sigma|V(0\ldots k), W(0\ldots k))
\end{pmatrix}
\]

(7.7)

The diagnostic algorithm is then given as follows:
7. Centralised Diagnosis of Stochastic Processes

Algorithm 7.1 (Centralised diagnosis of stochastic automata)

Initialise: $k = 0$

1. Measure $v(k)$ and $w(k)$
2. Apply Equations (7.4) - (7.5) (prediction) for all $z', f$
3. Apply Equation (7.6) (projection) for all $z', f$
4. Stop on user demand, else $k := k + 1$
5. Repeat from Step 1

Result: Distribution $p(f|V(0\ldots k), W(0\ldots k))$ from Equation (7.7)

Remark: Analogously to the previous diagnostic algorithms, in an implementation a test for the total inconsistency should be inserted after Step 2. This total inconsistency between the measured behaviour and the behaviour modelled in $L'$ is given whenever the denominator in Equation (7.4) is zero. Because such an inconsistency can be caused by a single mismeasurement, it has proven to be sensible to report this total inconsistency to the operator and to restart the algorithm with the initial condition.

Theorem 7.1 (Complete and sound diagnostic result):
The set of faults $F_{centr}(k) = \{f|P(f|V(0\ldots k), W(0\ldots k)) > 0\}$ obtained through Algorithm 7.1 is complete and sound, i.e. the following relation holds:

$$F_{centr}(k) = F^{\star}(k).$$

Proof:
See Appendix B.2.1.

That is, the algorithm fulfills the set objective. The following corollary follow directly from that proof together with Proof B.2.2.

Corollary 7.1: The centralised diagnosis of a stochastic automaton and the diagnosis of the nondeterministic automaton embedded in the SA result in the identical set of fault candidates $F_{centr}$.

According to Section 2.1 the diagnoser solving the above diagnostic problem is then fully defined by

$$D_{centr} = \{A^{\delta}, (V, W), P(z(0), f), \text{Algorithm 7.1, } p(f|V(0\ldots k), W(0\ldots k))\}.$$
7.1. Approach for Automata on Relations

Alternatively, the centralised diagnostic problem for stochastic automata can be stated using relational algebra:

Centralised fault diagnostic problem for stochastic automata
Given:  
- Automaton $A$ with relation $L^s(z(k+1), w(k), z(k), v(k), f, \text{Prob})$
- $v$ and $w$ are measurable
- A-priori initial condition $Pre(−1)$

Find:  
- Relation $F_{centr}(k)$ with the schema $\text{sch}(F_{centr}(k)) = \{f, \text{Prob}\}$, where $\text{Prob} = P(f|V(0...k), W(0...k))$

The a-priori initial condition $Pre(−1)$ with the schema $\text{sch}(Pre(k)) = \{z(k + 1), f, \text{Prob}\}$ includes the probabilities of the possible initial state and fault combinations. As the solution to this problem involves algebraic operations it cannot be given as a closed expression using relational algebra. Instead the algorithm is given exemplary using MySQL-Code. In the $k$-th projection step, analogously to Eqn. (7.4), the tuples are selected from $L^s$ which match the measured input and output signals. The probability of these tuples are multiplied with the respective probabilities of the previous projection step $Pre(k − 1)$. The summation over all states $z(k)$ is performed through the command “group by” in combination with the command “sum” [75].

**Prediction step:**

```sql
> SELECT zz, f, SUM(L.Prob * pre.Prob) AS Prob FROM L, pre WHERE v=... and w=... and L.zz=pre.zz and L.f=pre.f GROUP BY zz, f;
```

Again, in the above code the state $z(k+1)$ is denoted by $zz$. The three dots in $v=...$ and $w=...$ have to be replaced with the actual measurement, e.g. by using SQL-variables. The result of the $k$-th prediction step is stored in the relation $Pre(k)$. The diagnostic result is obtained by projecting $Pre(k)$ onto the fault space, i.e. the summation over all states $z(k+1)$ through the group by command in combination with the sum command.

**Projection step:**

```sql
> SELECT f, SUM(pre.Prob) AS Prob FROM pre GROUP BY f;
```

The full algorithm which solves the diagnostic problem is not given here, because of its similarity to the algorithm provided in the previous section.
7. Centralised Diagnosis of Stochastic Processes

7.2. Application to Automata Networks

Analogously to Sections 4.3 and 4.4, the centralised diagnostic approach for stochastic automata can be applied directly to stochastic automata networks (SAN) by utilising the composition rules introduced in Section 3.3.4 (cf. [69]). The diagnostic problem

**Centralised fault diagnostic problem for stochastic automata networks**

*Given:*  
- Stochastic automata network $\mathcal{A}^s = \{\mathcal{A}^s_1, \mathcal{A}^s_2, \ldots, \mathcal{A}^s_\gamma, v, w, s, f\}$  
- $v$ and $w$ are measurable  
- A-priori initial distributions $p(z_i(0))$ and $p(f_i)$ of all $i$ automata

*Find:*  
- Fault probability distribution $p(f_1, \ldots, f_\gamma|V(0\ldots k), W(0\ldots k))$

is then solved as follows. For every new measurement the extract of the equivalent automaton which fits to these measurements is generated by applying the composition rule. Then the method for diagnosing a single stochastic automaton is applied. The algorithm is given as follows:

**Algorithm 7.2** (Centralised diagnosis of stochastic automata networks)

*Initialise:* $k = 0$

1. Measure $v(k)$ and $w(k)$
2. Compute $\hat{L}^s$ by applying (3.40) for measurements $v(k)$ and $w(k)$
3. Apply Equations (7.4) - (7.5) for all $\mathbf{z'}$ using $\hat{L}^s$  
4. Apply Equation (7.6) for all $\mathbf{z'}, f$
5. Stop on user demand, else $k := k + 1$
6. Repeat from Step 1

*Result:* Distribution $p(f|V(0\ldots k), W(0\ldots k))$

**Theorem 7.2** (Complete and sound diagnostic result):

*The set of faults $\mathcal{F}_{centr}(k) = \{f|P(f|V(0\ldots k), W(0\ldots k)) > 0\}$ obtained by Algorithm 7.2 is complete and sound, i.e. the following relation holds:

$$\mathcal{F}_{centr}(k) = \mathcal{F}^\star(k).$$

**Proof:**

The equivalence of the diagnostic results of Algorithms 7.1 and 7.2 is proven in [69].

118
7.2. Application to Automata Networks

The algorithm is also provided in pseudo-code in Appendix C.6. The diagnoser solving the above diagnostic problem is then fully defined by

$$D_{\text{centr}}^\# = \{ A_0^\#, (V, W), P(z_i(0), f_i) \; i = 1, \ldots, \gamma, \text{Alg. 7.2.}, p(f|V(0 \ldots k), W(0 \ldots k)) \}.$$

Analogously to the centralised diagnosis of NAN it stands to reason that the above approach is not feasible, because the network must never be represented as a single automaton. The network’s memory advantage can be retained by operating directly with the network model and including the composition operation directly into the diagnostic algorithm. That way, the equivalent automaton of the network has not to be stored in memory [114].

With the abbreviation

$$L_i^\#(k) = L_i^\#(z_i(k), w_i(k)|z_i(k), v_i(k), f_i)$$

the prediction step of the diagnostic algorithm with “on-line composition” is then given by

$$P(z(k+1), f|V(0 \ldots k), W(0 \ldots k)) = \frac{\sum z(k) \sum s(k) \prod_{i=1}^\gamma L_i^\#(k)P(z(k), f|V(0 \ldots k-1), W(0 \ldots k-1))}{\sum z(k+1) \sum z(k) \sum f \sum s(k) \prod_{i=1}^\gamma L_i^\#(k)P(z(k), f|V(0 \ldots k-1), W(0 \ldots k-1))}. \quad (7.9)$$

As has been stated in previous chapters the set of coupling signals $s$ is a subset of the union of the sets of component input and output signals $v_i, w_i$. The projection step yields the probability of the faults:

$$P(f|V(0 \ldots k), W(0 \ldots k)) = \sum_{z(k+1)} P(z(k+1), f|V(0 \ldots k), W(0 \ldots k)). \quad (7.10)$$

The algorithm solving the diagnostic problem with “on-line composition” is supplied in Appendix C.7. As this approach only provides an alternative way of calculating the identical diagnostic result as Algorithm 7.2, the same remarks apply. Because all operations in the given solution are scalar, the full memory advantage of the network is retained. As a result several practical diagnostic problems could be solved where the original Algorithms 7.1 and 7.2 were not applicable. However, analogously to the nondeterministic case the drawback of this approach is the high number of mathematical operations which need to be performed.

119
7. Centralised Diagnosis of Stochastic Processes

7.3. Diagnosability and Output Indifference

7.3.1. Diagnosability

In the following several criteria for testing if a system is diagnosable or not are derived analogously to Section 4.5. At first this will be done for the single stochastic process, then these criteria will be extended to apply to stochastic automata networks.

Diagnosability in case of the single automaton

**Lemma 7.1:** A stochastic automaton is not diagnosable as of Definition 2.4 if

\[ P(z', w|z, v, f) = P(z', w|z, v) \]

holds for all \( z', z \in \mathcal{N}_z, v \in \mathcal{N}_v, w \in \mathcal{N}_w, f \in \mathcal{N}_f \).

This condition states that it is not possible to draw a conclusion about the fault if the model contains no information about it.

**Definition 7.1:** A stochastic automaton is not partially diagnosable if there exists a subset of faults \( \tilde{\mathcal{N}}_f \subseteq \mathcal{N}_f \) for which there does not exists an input sequence \( V \) such that \( (V, W) \not\in B(f), \forall f \in \tilde{\mathcal{N}}_f \) holds.

**Corollary 7.2:** A stochastic automaton is not partially diagnosable iff

\[ P(z', w|z, v, f) = P(z', w|z, v) \]

holds \( \forall z', z \in \mathcal{N}_z, v \in \mathcal{N}_v, w \in \mathcal{N}_w \) and a subset \( f \in \tilde{\mathcal{N}}_f \) with \( \tilde{\mathcal{N}}_f \subseteq \mathcal{N}_f, |\tilde{\mathcal{N}}_f| > 1 \).

The definition of diagnosability follows directly from that.

**Definition 7.2 (Diagnosability of the single stochastic automaton):** A stochastic automaton is called diagnosable iff there does not exist a set \( \tilde{\mathcal{N}}_f \) for which the automaton is not partially diagnosable.

Again, it is not guaranteed that a diagnostic system will always identify the fault in a diagnosable system. It only states that in diagnosable systems it is possible to identify the fault, whereas for not diagnosable systems it is not.

**Extension to the automaton with multiple I/O-signals**

In the following, the above statements are extended to be applicable to stochastic processes with multiple input and output signals, including unmeasurable signals. The measurable subsets of \( v_i, w_i \) are denoted by \( \tilde{v}_i, \tilde{w}_i \) as defined on page 39.
7.3. Diagnosability and Output Indifference

Lemma 7.2: A stochastic automaton with unmeasurable signals is not diagnosable if

\[ P(z'_i, \bar{w}_i | z_i, \bar{v}_i, f_i) = P(z'_i, \bar{w}_i | z_i, \bar{v}_i) \]

holds for all \( z'_i, z_i \in \mathcal{N}_{z_i}, f_i \in \mathcal{N}_{f_i}, \bar{v}_i \in \mathcal{N}_{v_i} \setminus \mathcal{N}_{s_i}, \bar{w}_i \in \mathcal{N}_{w_i} \setminus \mathcal{N}_{s_i} \).

Corollary 7.3: A stochastic automaton with unmeasurable signals is not partially diagnosable if

\[ P(z'_i, \bar{w}_i | z_i, \bar{v}_i, f_i) = P(z'_i, \bar{w}_i | z_i, \bar{v}_i) \]

holds for all \( z'_i, z_i \in \mathcal{N}_{z_i}, f_i \in \tilde{\mathcal{N}}_{f_i}, \bar{v}_i \in \mathcal{N}_{v_i}, \bar{w}_i \in \mathcal{N}_{w_i} \setminus \mathcal{N}_{s_i} \) and a subset of faults \( f_i \in \tilde{\mathcal{N}}_{f_i} \) with \( |\tilde{\mathcal{N}}_{f_i}| > 1 \)

The proofs for the above lemmas and corollaries are omitted, because they can be given analogously to the proofs in Section 4.5.1.

Diagnosability in case of the automata network

The following theorems on diagnosability are given analogously to the nondeterministic case and, therefore, without any further discussion.

Lemma 7.3 (Diagnosability of stochastic automata networks): A stochastic automata network \( A^\#_s \) is diagnosable if and only if the equivalent single automaton \( \hat{A}^s \), generated by applying the composition rules to this network, is diagnosable.

Proof:
See Appendix B.3.2.

Theorem 7.3 (Not diagnosable subprocesses):
A stochastic automata network \( A^\#_s \) is not diagnosable if all automata of the network are not diagnosable.

Proof:
See Appendix B.3.3.

Theorem 7.4 (Influence of a single automaton on the network diagnosability):
A stochastic automaton \( A^\#_s \) being not diagnosable is not a sufficient condition for the network to be not partially diagnosable concerning the faults \( f_i \in \mathcal{N}_{f_i} \).

121
This theorem states that it might be possible to diagnose the faults $f_i$ indirectly through their influence on other components of the network.

**Proof:**
See Appendix B.3.4.

### Detectability and Identifiability

In the following, detectability and identifiability are defined for stochastic processes.

**Definition 7.3:** A fault case $f = i$ of a stochastic automaton is *not detectable* iff it is not possible for any given measurement sequence $(V, W)$ to distinguish between this fault and the faultless case $f = 1$.

**Corollary 7.4:** A fault case $f = i, i \in \mathcal{N}_f$ of a stochastic automaton is *not detectable* iff the automaton is not partially diagnosable for the set $\tilde{\mathcal{N}}_f = \{1, i\}$.

**Definition 7.4:** A fault case $f = i$ of a stochastic automaton is *not identifiable* iff it is not possible for any given measurement sequence $(V, W)$ to distinguish between this and all other fault cases.

**Corollary 7.5:** A fault case $f = i, i \in \mathcal{N}_f$ of a stochastic automaton is *not identifiable* iff the automaton is not partially diagnosable for any of the sets $\tilde{\mathcal{N}}_f = \{i, j\}, \forall j \in \mathcal{N}_f, i \neq j$.

**Definition 7.5:** A fault case $f = i$ of a stochastic automaton is detectable/identifiable iff it is not not detectable/identifiable with respect to this fault case.

The extension to automata networks is done analogously to Theorem 7.3.

**Theorem 7.5 (Detectability/Identifiability of stochastic automata networks):**
A fault case $f = i$ of a stochastic automata network $\mathcal{A}_z^\delta$ is detectable/identifiable if and only if it is detectable/identifiable for the equivalent single automaton, generated by applying the composition rules to this network.

### 7.3.2. Output Indifference

In the following criteria will be presented to test the output indifference according to Definition 2.5.
Output indifference in case of the single automaton

**Lemma 7.4:** A stochastic automaton is *output indifferent* as of Definition 2.5 iff

\[ P(z', w|z, v, f) = P(z'|z, v, f) \cdot P(w|v) \]  \hspace{1cm} (7.11)

holds for all \( z', z \in N_z, v \in N_v, w \in N_w, f \in N_f \).

The claim that in such a case the result of a diagnosis does not differ from a mere simulation is formalised by the following corollary.

**Corollary 7.6:** If a stochastic automaton is output indifferent the following relation holds for all consistent I/O-pairs:

\[ P(f|V(0\ldots k), W(0\ldots k)) = P(f|V(0\ldots k)) = P(f). \]

**PROOF:**

See Appendix B.3.5.

A criterium for testing a stochastic automaton for output indifference is given below.

**Lemma 7.5:** A stochastic automaton \( \mathcal{A}^s \) is output indifferent if it is a Mealy-Automaton according to Definition 3.1 and there exists a function \( \hat{G}(w|v) \) with

\[ \hat{G}(w|v) = G(w|z, v, f) \]

for all \( z \in N_z, v \in N_v, w \in N_w, f \in N_f \).

**PROOF:**

See Appendix B.3.6.

Extension to the automaton with multiple I/O-signals

In the following the above statements are extended to be applicable to stochastic processes with multiple input and output signals, including unmeasurable signals. The measurable subsets of \( \mathbf{v}_i, \mathbf{w}_i \) are denoted by \( \bar{\mathbf{v}}_i, \bar{\mathbf{w}}_i \) as defined on page 39.

**Lemma 7.6:** A SA with unmeasurable signals is *output indifferent* if

\[ P(z'_i, w_i|z_i, \mathbf{v}_i, \mathbf{f}_i) = P(z'_i|z_i, \mathbf{v}_i, \mathbf{f}_i) \cdot P(w_i|\mathbf{v}_i) \]  \hspace{1cm} (7.12)

holds for all \( z'_i, z_i \in N_{z_i}, \mathbf{v}_i \in N_{\mathbf{v}_i}, w_i \in N_{\mathbf{w}_i}, \mathbf{f}_i \in N_{\mathbf{f}_i} \).

The dependence of the fault on any unmeasurable signals is irrelevant as long as these are independent of the measurable signals. Corollary 7.6 applies accordingly.
Output indifference in case of the automata network

Again, the extension to automata networks can be done analogously to Theorem 7.3.

**Theorem 7.6 (Output indifference of stochastic automata networks):**
A SAN $A_s^\#$ is output indifferent iff the equivalent single automaton $\hat{A}^\#$, generated by applying the composition rules to this network, is output indifferent. □

**Proof:**
As stated in the proof of Theorem 7.3 the SAN can be transformed to the scalar case

$$P(z'|w|z',\bar{v},\bar{f})$$

by application of the composition rules and the introduction of bijective mappings. According to Lemma 7.4 this automaton is output indifferent if

$$P(z'|w|z',\bar{v},\bar{f}) = P(z'|z',\bar{v},\bar{f}) \cdot P(w|\bar{v})$$ (7.13)

holds for all $z',z \in \mathcal{N}_z, \bar{v} \in \mathcal{N}_v, w \in \mathcal{N}_w, f \in \mathcal{N}_f$. This proves that a SAN is exactly then output indifferent if the equivalent automaton is output indifferent. ■

**Theorem 7.7:**
A stochastic automata network $A^\#_s$ is output indifferent if all automata of the network are output indifferent. □

**Proof:**
Given is a SAN $A^\#_s = (A^1_s, \ldots, A^\gamma_s, v, w, s, f)$, where the sets $v, w$ include the measurable I/O-signals of the network. All network components are output indifferent. In the following the composition rules are applied to generate the equivalent automaton. The last step shows that the whole stochastic automaton is output indifferent.

$$L^s(z', w|z, v, f) = P(z', w|z, v, f) = \sum_{s} \prod_{i=1}^{\gamma} P(z'_i, w_i|z_i, v_i, f_i)$$

$$= \sum_{s} \left[ \prod_{i=1}^{\gamma} P(z'_i|z_i, v_i, f_i) \cdot P(w_i|\bar{v}_i) \right] = \sum_{s} \left[ \prod_{i=1}^{\gamma} P(z'_i|z_i, v_i, f_i) \right] \cdot \prod_{i=1}^{\gamma} P(w_i|\bar{v}_i) = P(z'|z, v, f) \cdot P(w|\bar{v})$$ ■
“It’s not worth doing something unless you were doing something that someone, somewhere, would much rather you weren’t doing.”

Terry Pratchett (British novelist, *1948)

8

Limitations of the Modularisation of Stochastic Processes

The high computational costs in the centralised diagnosis of the stochastic automata network make a modularisation of the diagnosis analogously to the decentralised diagnosis of the nondeterministic automata network desirable. Unfortunately, the results obtained in Section 5 for NAN cannot be transferred directly to stochastic automata networks. In this section the reason for this deficiency as well as possible solutions or workarounds are presented (cf. also [114]).

8.1. The Consequence of Stochastic Dependence

8.1.1. The Separation Problem

The intention of decentralised diagnosis of stochastic automata networks is to break down the diagnostic task described in the previous chapter into several independent diagnostic tasks. Thereby, the local diagnosers should determine the probability distributions of the local faults under the condition the sequences \( \overline{V}_i(0...k), \overline{W}_i(0...k) \) have been measured. Again, note that \( \overline{v}_i \subseteq v_i, \overline{w}_i \subseteq w_i \) denote the measurable subsets of the I/O-signals of Automaton \( A_i \). Which signals are measurable and which are not is specified by the information structure. That in general such a modularisation of the diagnosis of stochastic processes is not possible is shown in the following.

Formally, the problem for the diagnosis of a component of a stochastic automata network can be stated as follows:

---

\(^1\)The breaking down of the diagnostic task is also termed the decentralisation of the centralised fault diagnostic problem for stochastic automata or in short “decentralisation”.

125
8. Limitations of the Modularisation of Stochastic Processes

![Diagram of a stochastic automata network consisting of two automata in a serial connection]

Figure 8.1.: Decentralised diagnosis of a stochastic automata network consisting of two automata in a serial connection

**Local diagnostic problem for stochastic automata**

Given:
- Automaton $\mathcal{A}_i^s$ with ch. function $L_i^s(z_i(k+1), w_i(k) \mid z_i(k), v_i(k), f_i)$
- $\bar{v}_i \subseteq v_i$ and $\bar{w}_i \subseteq w_i$ are measurable
- A-priori prob. distributions of initial states $p(z_i(0))$ and faults $p(f_i)$

Find:
- Fault probability distribution $p(f_i \mid V_i(0 \ldots k), W_i(0 \ldots k))$

Analogously to the decentralised diagnosis of NAN in Section 5 it is clear that to ensure completeness the diagnostic result has to be calculated for all possible values of the coupling signals. Therefore, the equation for the projection step of the local diagnosis of a stochastic automaton $\mathcal{A}_i^s$ must be transformed to

\[
P(f_i \mid V_i(0 \ldots k), W_i(0 \ldots k)) = \sum_{z_i(k+1)} \sum_{z_i(k)} \sum_{s_i(k)} L_i^s(k) \cdot P(z_i(k), f_i \mid V_i(0 \ldots k-1), W_i(0 \ldots k-1))
\]

which includes an additional sum over all coupling signals $s_i(k)$. However, in general the different stochastic processes of the network do not operate independently, but their stochastic variables depend on each other via the coupling. It is apparent that the neglect of this dependence might lead to wrong results. This will be investigated in the following by disassembling the centralised diagnosis of a network step by step into a decentralised diagnosis, i.e. it is to be investigated if the following holds:

\[
P(f \mid V, W) = P(f_1 \mid V_1, W_1) \cdot P(f_2 \mid V_2, W_2) \cdots P(f_\gamma \mid V_\gamma, W_\gamma).
\]

Consider a simple network consisting of two SA in a series connection (Fig. 8.1) w.l.o.g. since every network can be composed to a single automaton using the composition rules. Because the automata possess the Markov property the analysis can be restricted to the initial diagnostic step. The result of a centralised diagnosis ac-
cording to Equations (7.5)-(7.6) of the network’s equivalent automaton \( \hat{A}^s \) with the characteristic function \( \hat{L}^s \) for step \( k = 0 \) is given by

\[
P(f | v(0), w(0)) = P(f^1, f^2 | v^1(0), v^2(0), w^1(0), w^2(0))
\]

\[
= \frac{\sum_{z(1)} \sum_{z(0)} \hat{L}^s(0) \cdot P(z(0), f)}{\sum_{z(1)} \sum_{z(0)} \sum_f \hat{L}^s(0) \cdot P(z(0), f)}. 
\]

Applying the composition rules yields

\[
P(f | v(0), w(0)) = \frac{\sum_{z(1)} \sum_{z(0)} \sum_{s^1(0)} L_1^s(0) L_2^s(0) \cdot P(z(0), f)}{\sum_{z(1)} \sum_{z(0)} \sum_f \sum_{s^1(0)} L_1^s(0) L_2^s(0) \cdot P(z(0), f)}
\]

\[
= \frac{\sum_{z(1), z(2)} \sum_{z(1), z(2)} \sum_{s^1(0)} L_1^s(0) L_2^s(0) \cdot P(z(1)(0), f^1) P(z(2)(0), f^2)}{\sum_{z(1), z(2)} \sum_{z(1), z(2)} \sum_{s^1(0)} L_1^s(0) L_2^s(0) \cdot P(z(1)(0), f^1) P(z(2)(0), f^2)}.
\]

The summation over the different states and faults can be separated as follows:

\[
P(f | v(0), w(0)) = \frac{\sum_{s^1(0)} \left[ \sum_{z(1), z(2)} L_1^s(0) P(z(1)(0), f^1) \sum_{z(2), z(2)} L_2^s(0) P(z(2)(0), f^2) \right]}{\sum_{s^1(0)} \left[ \sum_{z(1), z(2)} \sum_{s^1(0)} L_1^s(0) P(z(1)(0), f^1) \sum_{z(2), z(2)} \sum_{s^1(0)} L_2^s(0) P(z(2)(0), f^2) \right]}
\]

(8.2)

However in general, after separating the summation over the coupling signal \( s^1 \) as proposed in (8.1), the equality sign does no longer hold:

\[
P(f | v(0), w(0)) \neq \frac{\sum_{z(1), z(2)} \sum_{s^1(0)} L_1^s(0) P(z(1)(0), f^1)}{\sum_{z(1), z(2)} \sum_{s^1(0)} L_1^s(0) P(z(1)(0), f^1)} \cdot \frac{\sum_{z(1), z(2)} \sum_{s^2(0)} L_2^s(0) P(z(2)(0), f^2)}{\sum_{z(1), z(2)} \sum_{s^2(0)} L_2^s(0) P(z(2)(0), f^2)}
\]

\[
= \frac{\sum_{z(1), z(2)} \sum_{s^1(0)} L_1^s(0) P(z(1)(0), f^1)}{\sum_{z(1), z(2)} \sum_{s^1(0)} L_1^s(0) P(z(1)(0), f^1)} \cdot \frac{\sum_{z(1), z(2)} \sum_{s^2(0)} L_2^s(0) P(z(2)(0), f^2)}{\sum_{z(1), z(2)} \sum_{s^2(0)} L_2^s(0) P(z(2)(0), f^2)}
\]

\[
= P(f^1 | v^1(0), w^1(0)) \cdot P(f^2 | v^2(0), w^2(0)).
\]

The argumentation for this follows the argumentation in Section 5.4 for the decen-
8. Limitations of the Modularisation of Stochastic Processes

Centralised diagnosis of nondeterministic automata networks. The equality is violated, whenever signal combinations of \( s^1 \) are considered during diagnosis which are not possible regarding the dynamics of the overall system, namely that the signal has different values for the automata which it connects. Whereas in the nondeterministic case the result could be estimated through an overset relation, this is not possible in the stochastic case. The relative error made when treating dependent stochastic variables as independent can be unbounded. The conclusion can summarised as follows:

**Decentralisation of the diagnosis of stochastic automata networks**

*In general, the diagnostic problem for stochastic automata networks cannot be solved using a decentralised information structure.*

**Proof:**

The above statement is proven by contradiction (reductio ad absurdum). Assume that the diagnostic problem for stochastic automata can always be decentralised. Consider now the following example.

Given is the network depicted in Fig. 8.1 and the characteristic functions given in Tables 8.1(a)-(b). As in previous examples the tables only show the part of the characteristic function that fits to the measured I/O-signals. The a-priori initial conditions are equally distributed. It can be easily seen in Tables 8.1(a)-(b) that for both automata both fault cases are equally probable, i.e. \( \sum_{z_i} \sum_{s^1_i} L^i = c_i \) for \( f^i = 1 \) and \( f^i = 2 \).

Therefore, the decentralised diagnosis results in

\[
\begin{align*}
P(f^1 = 1|v^1(0), w^1(0)) &= 0.5, \\
P(f^1 = 2|v^1(0), w^1(0)) &= 0.5, \\
P(f^2 = 1|v^2(0), w^2(0)) &= 0.5, \\
P(f^2 = 2|v^2(0), w^2(0)) &= 0.5.
\end{align*}
\]

However, the centralised diagnosis using the equivalent automaton with the characteristic function given in Table E.1 in Appendix E on Page 215 returns

\[
\begin{align*}
P(f^1 = 1, f^2 = 1|v^1(0), w^1(0), v^2(0), w^2(0)) &= 0.45 \neq \not= 0.25 \\
P(f^1 = 1|v^1(0), w^1(0)) \cdot P(f^2 = 1|v^2(0), w^2(0)) &= 0.25.
\end{align*}
\]

This contradicts the proposition which proves the statement.

However, it is possible to decentralise the diagnostic problem in some cases. These cases are identified in the next section.

**8.1.2. Stochastic Independence**

It has been stated in the previous section that the source for the inequality of the results of the centralised and the decentralised diagnostic approach is the coupling through

\[
128
\]
the unmeasurable signals \( s \). In the following, three criteria concerning the coupling signals are presented, which indicate that the diagnostic task can be decentralised.

### Measuring the signals

Without proof it is clear that the different stochastic processes of the network can be decoupled by measuring the signals that connect the automata. In other words, the more signals are measured, the more the diagnosis can be decentralised. The more the diagnostic task is decentralised, the smaller the computational costs. If all signals of the network are measurable, i.e. \( s = \emptyset \) holds, the network can be fully decentralised. It is the engineers task to trade off between the costs for additional measurements and for computational resources.

### Calculable coupling signals

A coupling signal is *calculable* if its value \( s(k) \) is *unambiguously* defined by means of the current measurements \( v(k) \) and \( w(k) \) for all \( k \). Analogously to the previous criterium, the stochastic processes of the network are independent of each other if the coupling between them is known.

**Lemma 8.1:** The diagnostic task can be decentralised, if the coupling signals of the stochastic network are calculable.
8. Limitations of the Modularisation of Stochastic Processes

PROOF:
Again, consider the simple network depicted in Fig. 8.1 without loss of generality. Let

\[ S_1(v^1, w^1) = \{ s^1 \mid \exists L_1(z'_{p1} = z'_1, s^1_p = s^1, w^1_p = w^1|z_{p1} = z_1, v^1_p = v^1, f^1_p = f^1) > 0, \\
\quad z'_1, z_1 \in N_{s1}, s^1 \in N_{s^1}, f^1 \in N_{f^1} \} \]

be the set of all values of the signal \( s^1 \) the automaton \( A^s_1 \) can generate for a given I/O-pair \((v^1, w^1)\). Analogously

\[ S_2(v^2, w^2) = \{ s^1 \mid \exists L_2(z'_{p2} = z'_2, w^2_p = w^2|z_{p2} = z_2, v^2_p = v^2, s^1_p = s^1, f^2_p = f^2) > 0, \\
\quad z'_2, z_2 \in N_{s2}, s^1 \in N_{s^1}, f^2 \in N_{f^2} \} \]

denotes the set of all values of the signal \( s^1 \) the automaton \( A^s_2 \) can have received if the I/O-pair \((v^2, w^2)\) has been measured. If

\[ \text{card}(S_1 \cap S_2) = 1, \text{ for all } v^1 \in N_{v^1}, v^2 \in N_{v^2}, w^1 \in N_{w^1}, w^2 \in N_{w^2} \]

holds, the signal \( s^1 \) is unambiguously defined. Then the sum \( \sum_{s^1} \) in Equation (8.2) can be omitted, because it only concerns one value. Then

\[
P(f|v(0), w(0)) = \frac{\sum_{s^1}(0) \left[ \sum_{z_1(1), z_1(0)} L_1(z_1(0))P(z_1(0), f^1) \sum_{z_2(1), z_2(0)} L_2(z_2(0))P(z_2(0), f^2) \right]}{\sum_{s^1}(0) \left[ \sum_{z_1(1), z_1(0)} \sum_{f^1} L_1(z_1(0))P(z_1(0), f^1) \sum_{z_2(1), z_2(0)} \sum_{f^2} L_2(z_2(0))P(z_2(0), f^2) \right]}
\]

\[
= \frac{\sum_{z_1(1), z_1(0)} L_1(z_1(0))P(z_1(0), f^1) \cdot \sum_{z_2(1), z_2(0)} L_2(z_2(0))P(z_2(0), f^2)}{\sum_{z_1(1), z_1(0)} \sum_{f^1} L_1(z_1(0))P(z_1(0), f^1) \cdot \sum_{z_2(1), z_2(0)} \sum_{f^2} L_2(z_2(0))P(z_2(0), f^2)}
\]

\[
= \frac{\sum_{z_1(1), z_1(0)} L_1(z_1(0))P(z_1(0), f^1) \cdot \sum_{z_2(1), z_2(0)} L_2(z_2(0))P(z_2(0), f^2)}{\sum_{z_1(1), z_1(0)} \sum_{f^1} L_1(z_1(0))P(z_1(0), f^1) \cdot \sum_{z_2(1), z_2(0)} \sum_{f^2} L_2(z_2(0))P(z_2(0), f^2)}
\]

holds, which proves the lemma.

Superfluous (uninformative) coupling signals

A coupling signal is uninformative if either its value does not influence the value of any other signal or its value is not influenced by any other signal, i.e. if the signal is independent of the other signals in the network. As opposed to the previous cases,
the value of the coupling signal might be unknown, but is totally irrelevant for the network’s behaviour.

**Lemma 8.2:** The diagnostic task can be decentralised, if the coupling signals of the stochastic network are uninformative.

**Proof:**
See Appendix B.4.1.

**Result**
The above statements are formalised in the following theorem.

**Theorem 8.1 (Condition for decentralisation):**
There exists a decentralised diagnoser of a component of a stochastic automata network if one of the following conditions holds:

1. \( s = \emptyset \)
2. All coupling signals are either calculable or uninformative.

**Definition 8.1 (Stochastic independence):** Two stochastic automata for which at least one of the conditions in Theorem 8.1 holds are called *stochastically independent*.

The conditions in Theorem 8.1 are very hard and, hence, not fulfilled by the majority of the SAN. In general, these networks cannot be diagnosed using a decentralised information structure as long as exact probabilistic diagnostic results are the aim. Alternative approaches with a more relaxed aim are given in the following section.

**8.2. Alternative Approaches**

Several approaches for tackling the problem of decentralising coupled stochastic process have been published. Although all these publications deal with models with asynchronous rather than synchronous signals, their course of action might be partially applicable to synchronised automata networks. The two most prominent approaches are shortly described in this section.

In the previous section it has been shown that probability theory poses hard restrictions on the decentralisation of stochastic processes. The probability concerning
stochastic depended variables must never be calculated separately, else the result is wrong. In [11, 17, 40] it has therefore been suggested to violate the assumptions of the classical probabilistic theory and to model the process using the more general measure theory. For such measures the mentioned restriction of probability theory does not necessarily apply, but the significance of the diagnostic result is reduced. In other words, the calculated value is no longer the probability of the fault, but a mere indicator for its likelihood. It can be questioned if it is worth the effort to calculate these indicators when their explanatory power is limited.

In [15, 27, 28, 51] methods for representing stochastic processes as dynamic Bayesian networks – a type of model close to the synchronised stochastic automata networks – are described. To solve the complexity problems the authors suggest to split the network into several almost independent subnets. The reasoning behind this is as follows:

When simulating the behaviour of an automaton with the characteristic function \( L \) for two initial state distributions \( p(z(0)) \) and \( \tilde{p}(z(0)) \), the distributions will draw nearer to each other with increasing time horizon \( k \). In other words, the characteristic function reduces the distance

\[
\| p(z(k)) - \tilde{p}(z(k)) \|
\]

for any given norm. The cited authors have proven that for the relative entropy

\[
D(p(z(k))|\tilde{p}(z(k))) = \sum_{z=1}^{N} p(z(k)) \log_2 \frac{p(z(k))}{\tilde{p}(z(k))}
\]

that contracting factor can be given as

\[
D(p(z(k+1))|\tilde{p}(z(k+1))) \leq (1 - \Lambda)D(p(z(k))|\tilde{p}(z(k)))
\]

with

\[
\Lambda := \min_{i,h} \sum_{j=i}^{N} \min(P(z' = j|z = i), P(z' = j|z = h)).
\]

I.e. for subsystems with a \( \Lambda \) close to one it is assured that the neglect of stochastic dependencies has only very limited influence on the state distribution. An upper border for this error can be given. The unsolved problem is how to partition a network such that the above demand is fulfilled. Furthermore, it has not yet been investigated to what extend this approach is transferable to stochastic automata networks.
Part IV.

Further Aspects of Interconnected Discrete-Event Systems
Simulation of Stochastic Automata Networks

In this chapter it is shown how the behaviour of an automata network can be simulated while retaining the memory advantage of the network representation. After a short introduction in Section 9.1 it is first shown in Section 9.2 how the behaviour of a single stochastic automaton can be simulated. Section 9.3 demonstrates that the extension of this approach to automata networks is not straightforward. An approach to simulation using on-line-composition is suggested in Section 9.4.

9.1. Introduction

Simulation is the task to calculate a system’s output and state sequence given the system model, an initial state, and an input sequence. The information flow in simulation is straight from input to output as opposed to the information flow in diagnosis in which input and output are both used to calculate the diagnostic result (Figure 9.1). In the following sections this general concept is applied to stochastic automata networks. In case of stochastic systems instead of the outcome of a random experiment the probability distribution over output and state is given as the distribution holds more information and a random state and output can be calculated using these distributions. It is shown that the simulation cannot be done on a component basis, i.e. the simulation task cannot be split up into several smaller tasks. Instead, the simulation of the behaviour of the full network has to be performed for all components simultaneously. The aim is to derive a simulation method which uses only little computational resources.
9. Simulation of Stochastic Automata Networks

a) In diagnosis the measured input and output signals are used to calculate the diagnostic result

b) In simulation the output is calculated using the input signal

Figure 9.1.: Information flow in diagnosis and simulation

9.2. Simulation of the Behaviour of a Single Stochastic Automaton

In the following the simulation of the behaviour of a single stochastic automaton is derived (cf. also [63, 66, 122]). Given an autonomous system

\[ A^s = (\mathcal{N}_z, \mathbf{p}(z(0))) \]

with the characteristic function

\[ L^s : \mathcal{N}_z \times \mathcal{N}_z \rightarrow [0, 1] \]

\[ L^s(z(k+1) | z(k)) = P(z_p(k+1) = z(k+1) | z(k) = z_p(k)) \]

the probability \( P(z(k+1) \) can be calculated by applying the theorem of total probability [18, 63]:

\[ P(z_p(k+1) = z(k+1) = \sum_{z(k) \in \mathcal{N}_z} L^s(z(k+1) | z(k)) \cdot P(z_p(k) = z(k)) \quad (9.1) \]

For an easy implementation this calculation can be brought into vectorial form with the probability distribution

\[ \mathbf{p}(z(k)) = \begin{pmatrix} P(z_p(k) = 1) \\ \vdots \\ P(z_p(k) = N) \end{pmatrix} \]
9.2. Simulation of the Behaviour of a Single Stochastic Automaton

and the matrix
\[ L_M^s = \begin{pmatrix} P(z_p(k+1) = 1 | z_p(k) = 1) & \cdots & P(z_p(k+1) = 1 | z_p(k) = N) \\ \vdots & \ddots & \vdots \\ P(z_p(k+1) = N | z_p(k) = 1) & \cdots & P(z_p(k+1) = N | z_p(k) = N) \end{pmatrix}. \]

(9.2)

The simulation algorithm can then be given as follows:

**Algorithm 9.1** (Simulation of the behaviour of a single autonomous automaton)

Initialise: \( k = 0, p(z(0)) \) is given

1. Apply \( p(z(k+1)) = L_M^s \cdot p(z(k)) \)
2. Stop on user demand, else \( k := k + 1 \)
3. Repeat from Step 1

Result: \( p(z(k+1)) \)

This approach can be extended to the general stochastic automaton as defined in Section 3.2.4:

\[ P(z_p(k+1) = z(k+1), w_p(k) = w(k)) = \sum_{z(k) \in \mathcal{N}_z} \sum_{v(k) \in \mathcal{N}_v} L^s(z(k+1), w(k) | z(k), v(k)) \cdot P(z_p(k) = z(k), v_p(k) = v(k)). \]

(9.3)

As in the previous chapters it is assumed that the input is independent of the system state, i.e. \( P(z(k), v(k)) = P(z(k)) \cdot P(v(k)) \) holds. Analogously to the above algorithm the calculation can be brought into vectorial form with the probability distributions \( p(z(k), v(k)) = p(z(k)) \times p(v(k)) \) and \( p(z(k+1), w(k)) = p(z(k+1)) \times p(w(k)) \) and the matrix

\[ L_M^s = \begin{pmatrix} L_w = 1 | v_p = 1 & \cdots & L_w = 1 | v_p = M \\ \vdots & \ddots & \vdots \\ L_w = R | v_p = 1 & \cdots & L_w = R | v_p = M \end{pmatrix}. \]

This matrix consists of \( M \cdot R \) blocks, whereby each block includes the probabilities of the state transitions as in Equation (9.2) for the respective input/output pair. The

---

1 The fault signal \( f \) is omitted in this chapter, because it can be treated just as an input signal \( v \) and does not introduce any additional aspects.
simulation algorithm can then be extended as follows:

**Algorithm 9.2** (Simulation of the behaviour of a single automaton)
Initialise: \( k = 0, p(z(0)) \) is given

1. Get \( p(v(k)) \) from measurement
2. Apply \( p(z(k), v(k)) = p(z(k)) \times p(v(k)) \)
3. Apply \( p(z(k+1), w(k)) = L_M \cdot p(z(k), v(k)) \)
4. Apply \( p(z(k+1)) = \sum_{w(k) \in \mathcal{N}_w} p(z(k+1), w(k)) \)
5. Apply \( p(w(k)) = \sum_{z(k+1) \in \mathcal{N}_w} p(z(k+1), w(k)) \)
6. Stop on user demand, else \( k := k + 1 \)
7. Repeat from Step 1

Result: \( p(w(k)) \)

### 9.3. Straight Forward Approach to Simulation of Networks

Clearly, the simulation algorithm 9.2 can be applied directly to an automata network \( \mathcal{A}_s^\sharp \) if it is composed to a single stochastic automaton \( \hat{\mathcal{A}}_s^\sharp \) using the composition rules as described in Section 3.3.4 beforehand (off-line-composition). The advantage of this approach is the ability to use existing simulation software. The DIAMOND\textsuperscript{Q} toolbox for MATLAB\textsuperscript{®}, for example, includes functions for the simulation of stochastic automata using the Simulink\textsuperscript{®} environment. The drawback is the loss of the memory advantage of the network representation. In general, the equivalent automaton of the network is too large and therefore too memory consuming for this approach to be practicable.

The second obvious approach to the simulation of the network behaviour is to use Algorithm 9.2 to simulate the behaviour of the different components separately and to propagate the signal distributions through the network. This is clarified by the following example.

**Example 14:**
Given is a stochastic automata network consisting out of two automata in a serial connection.
9.4. Simulation Using On-line-Composition

Figure 9.2.: Network consisting out of two automata in a serial connection

as depicted in Figure 9.2. The simulation of the behaviour of automaton $A_1^s$ results in the joint probability

$$P(z_1(k+1), s(k)) = \sum_{z_1(k), v^1(k)} P(z_1(k+1), s(k)|z_1(k), v^1(k)) \cdot P(z_1(k)) \cdot P(v^1(k)).$$

The projection onto the domain of the coupling signal results in

$$P(s(k)) = \sum_{z_1(k+1)} P(z_1(k+1), s(k))$$

which is then used for the simulation of the behaviour of the second automaton $A_2^w$:

$$P(z_2(k+1), w^1(k)) = \sum_{z_2(k), s(k)} P(z_2(k+1), w^1(k)|z_2(k), s(k)) \cdot P(z_2(k)) \cdot P(s(k)).$$

As reasonable as this approach might sound it is not applicable. In [122] it has been proven that the result obtained by this sequential simulation is not correct and, therefore, differs from the result of the composition approach. As in Chapter 8 the reason for the wrong result in case of the separate calculation is the stochastic dependence of the different components. Because the two automata in the above example are connected through the signal $s$ they are not stochastically independent and, therefore, the probabilities concerning their signals must not be calculated separately.

The result of this section is that both presented approaches are not suitable for the simulation of the behaviour of stochastic automata networks. Storing the equivalent automaton of the network uses too much memory, but the simulation results are correct. Splitting the simulation task in several smaller tasks uses very little memory, but returns wrong results. The aim of the next section is to find an applicable method which combines the advantages of both methods.

9.4. Simulation Using On-line-Composition

Some simulation approaches for stochastic automata networks with synchronised signals have been proposed the most notable being [82, 85]. However, these approaches rely on building the full cartesian product of the component relations and focus on
9. Simulation of Stochastic Automata Networks

reducing the computational costs for this operation. In [15] the simulation of dynamic Bayesian networks – a type of model close to stochastic automata networks – is addressed. The approach returns an approximation of the result and is explained in more detail in Section 8.2.

The idea behind the simulation approach using on-line-composition presented in this section is similar to the diagnostic approaches using on-line-composition described in Sections 4.4 and 7.2. To gain the correct simulation result the behaviour of the full network must be used without building the full equivalent automaton beforehand. Instead, the simulation is performed by acquiring the necessary information directly from the characteristic functions of the network components and connecting them using the composition rules. This clarified by the following example using the serial connection depicted in Figure 9.2.

Example 15:
Applying the composition rule (3.36) to the network results in the equivalent automaton \( \hat{A} \) with the characteristic function

\[
\hat{L}^s((z_1', z_2', w^1 | (z_1, z_2), v^1)) = \sum_s L_1^s(z_1', s | z_1, v^1) \cdot L_2^s(z_2', w^1 | z_2, s).
\]

This characteristic function could now be transformed into the matrix representation \( \hat{L}_M^s \) as described in Section 9.2 and used in Algorithm 9.2 to gain the simulation result. The simulation result could also be gained using the scalar representation

\[
P((z_1(k + 1), z_2(k + 1)), w^1(k)) = \sum_{z_1(k)} \sum_{z_2(k)} \sum_{v^1(k)} \hat{L}^s((z_1', z_2', w^1 | (z_1, z_2), v^1)) \cdot P((z_1(k), z_2(k)), v^1(k)).
\]

As opposed to that the simulation approach with on-line-composition inserts the composition rules directly into the scalar calculation:

\[
P((z_1(k + 1), z_2(k + 1)), w^1(k)) = \\
\sum_{z_1(k)} \sum_{z_2(k)} \sum_{v^1(k)} \left( \sum_s L_1^s(z_1', s | z_1, v^1) \cdot L_2^s(z_2', w^1 | z_2, s) \right) \cdot P((z_1(k), z_2(k)), v^1(k)).
\]

The generalisation of the example is given as

\[
P(z(k + 1), w(k)) = \sum_{z(k)} \left( \sum_{v(k)} \left( \sum_s \gamma L_1^s \right) \cdot P(v(k)) \right) \cdot P(z(k)), \quad (9.4)
\]
where $z, v, w,$ and $s$ denote the network’s state, input, output and coupling signals as introduced in Section 3.3.2. Without proof it is clear from the derivation that the simulation approaches with on-line- and off-line-composition return the identical results. The algorithm for the approach with on-line-composition can be given as follows:

\textbf{Algorithm 9.3} (Simulation with on-line-composition)
\begin{itemize}
  \item Initialise: $k = 0, p(z(0))$ is given
  \item Get $p(v(k))$ from measurement
  \item For all combinations of $z$ and $w$ do
    \begin{itemize}
      \item a) Apply Equation (9.4) to gain $P(z(k+1), w(k))$
      \item b) Insert result into distribution vector $p(z(k+1), w(k))$
    \end{itemize}
  \item Apply \( p(z(k+1)) = \sum_{w(k)} p(z(k+1), w(k)) \)
  \item Apply \( p(w(k)) = \sum_{z(k+1)} p(z(k+1), w(k)) \)
  \item Stop on user demand, else $k := k + 1$
  \item Repeat from Step 1
\end{itemize}
\textbf{Result:} $p(w(k))$

The algorithm was implemented in Matlab/Simulink and tested on a number of examples. The applicability of the algorithm is demonstrated on an application example in Section 12.3.

\section*{9.5. Conclusion}

A method has been developed for the simulation of the behaviour of stochastic automata networks. The resulting algorithm has been implemented and tested. The method is based on gaining the simulation result using only the needed information from the component relations of the network. The calculations in the algorithm are scalar to further reduce the memory requirements. The advantage of the simulation approach with on-line-composition is the maintenance of the memory advantage of the network representation. The drawback are the high number of calculations which need to be performed during runtime. The simulation result takes longer to compute, but it is possible to compute systems of a size which could not be simulated so far, because of memory restrictions.
9. Simulation of Stochastic Automata Networks
Abstraction of Networks of Quantised Systems

So far, it has been assumed that the automaton models for the presented methods and algorithms are given. In this section it is discussed how these models can be gained using abstraction methods for quantised systems, i.e. how physical systems can be described as automaton models. The abstraction of quantised systems is motivated in Section 10.1. In Section 10.2 the quantised system is defined and its abstraction to a single automaton is formally given. The extension to networks is presented in Sections 10.3 and 10.4. Section 10.5 introduces an abstraction method which assures the completeness of the resulting network. The main results are summarised in Section 10.6.

10.1. Motivation

Throughout this work the size of the automaton models has been the motivation for the development of new process supervision methods. Clearly, models of this size cannot be generated by hand, i.e. it is not possible to look at a given system and write down the automaton table except for small academic examples. For a complex physical system a systematic method to automatically generate its automaton model is needed. The process of generating the model is called abstraction. A monolithic abstraction method generating a single discrete-time automaton has already been developed in [65,68,93]. The aim of this section is to extend this approach to composite systems.

Analogously to the diagnostic result sets used in the previous sections it is required that the generated automaton model be complete with respect to the definition given below.
10. Abstraction of Networks of Quantised Systems

Definition 10.1 (Completeness of a model): A model with the behaviour $B_a$ is complete if and only if it includes the full behaviour $B_q$ of the original system, i.e.

$$B_a \supseteq B_q$$

holds.

The completeness of the model closes the line of argumentation concerning the completeness of the diagnostic result. Only if the model is complete and the diagnostic result is complete given the model can the diagnostic result be complete with respect to the original system.

The abstraction process has been investigated by a number of groups, e.g. [5, 6, 76, 86, 100, 103]. However, all of these approaches are designed to yield asynchronous discrete-event systems as opposed to the synchronous systems used in this book. Instead, the results presented in this section are based upon [93] which introduces a method for gaining a stochastic discrete-time automaton. In this chapter it is shown that the extension to composite systems is not straight-forward and a method is presented which assures the completeness of the abstracted composite system.

10.2. Abstraction of Quantised Systems

10.2.1. Quantised Systems

A quantised system as depicted in Figure 10.1 consists of a continuous-variable system at its core, an Injector $I$, and Quantiser $Q$. The continuous-variable system can be nonlinear and given as a discrete-time system

$$x(k+1) = g(x(k), u(k)), \quad y(k) = h(x(k), u(k)),$$

or continuous-time system

$$\dot{x} = g(x(t), u(t)), \quad x(0) = x_0 \quad y(t) = h(x(t), u(t)).$$

The variables $x \in \mathcal{R}^{nx} \subset \mathbb{R}^{nx}$, $u \in \mathcal{R}^{nu} \subset \mathbb{R}^{nu}$, $y \in \mathcal{R}^{ny} \subset \mathbb{R}^{ny}$ denote the state, input and output of the system, respectively. The sets $\mathcal{R}$ are finite, i.e. all signals are bounded.
10.2. Abstraction of Quantised Systems

The **quantiser** transforms the numerical signal $\mathbf{y}$ into the quantised signal $[\mathbf{y}] \in \mathcal{N}_y = \{1, 2, \ldots, R\}$ where $\mathcal{N}_y$ denotes the finite discrete output alphabet. In principle $\mathcal{N}_y$ can contain any type of symbols including verbal descriptions such as “on” and “off”, but integer numbers are used here because of the notational convenience. Given a partitioning $\mathcal{Q}_y$ which divides the output space $\mathbb{R}^{n_y}$ into $R$ non-overlapping bounded sets the quantiser is then defined by the relation

$$[\mathbf{y}] = i \Leftrightarrow \mathbf{y} \in \mathcal{Q}_y(i),$$

(10.5)

i.e. the quantised output is assigned the value $i$ whenever the continuous output lies in the $i$-th partition $\mathcal{Q}_y(i)$. In the simplest case the partition is rectangular and each partition can be described by a lower and upper bound: $\mathcal{Q}_y(i) = [y_i,\text{low},y_i,\text{up}]$.

The **injector** performs the opposite operation and transforms the discrete-valued signal $[\mathbf{u}] \in \mathcal{N}_u = \{1, 2, \ldots, M\}$ into a numerical signal $\mathbf{u}$:

$$[\mathbf{u}] = i \Leftrightarrow \mathbf{u} \in \mathcal{Q}_u(i).$$

(10.6)

The mechanism which chooses an appropriate numerical value out of the set $\mathcal{Q}_u(i)$ is not restricted. Anything from random functions to mean values can be applied. To emphasise that a signal has been injected it is marked by angle brackets: $>[\mathbf{u}]< \in \mathbb{R}^{n_u}$.

The combination of discrete and continuous-variable signals makes the overall system a *hybrid system* [7, 9]. However, seen from outside, the system marked by the dashed box in Figure 10.1 has discrete dynamics only. In process supervision the causal relation from the input to the output is not of interest. Instead, the behaviour $\mathcal{B}_q$ of the quantised system with

$$\mathcal{B}_q \subseteq (\mathcal{N}_u \times \mathcal{N}_y) \times (\mathcal{N}_u \times \mathcal{N}_y)^2 \times \cdots$$

(10.7)

is the basis for the investigation (cf. Section 2.1). The behaviour consists of all possible trajectories

$$B(k_h) = (([\mathbf{u}(0)] \ldots [\mathbf{u}(k_h)]), ([\mathbf{y}(0)] \ldots [\mathbf{y}(k_h)]))$$

(10.8)
of arbitrary length $k_h$ that the system can execute [105]. The information flow of the quantised system can then be interpreted as depicted in Figure 10.2.

![Figure 10.2.: Quantised system with non-causal information flow](image)

### 10.2.2. Abstraction

Abstraction means to approximate the behaviour of the quantised system by means of an automaton model. This abstraction leads to a significant reduction of the model complexity, because an automaton model is far easier to handle than a hybrid nonlinear quantised system. This section summarises the results concerning the abstraction of discrete and continuous-time systems as the basis for the extension to automata networks in Section 10.4.

#### Abstraction of discrete-time systems

After the introduction of a state partitioning $Q_x$ with $[x] \in N_x = \{1, 2, \ldots, N\}$ it is reasonable to define the sets $N_z, N_v, N_w$ of the automaton according to the sets of the quantised system. This means that the following holds:

$$
N_z := N_x \\
z = [x] \\
N_v := N_u \\
v = [u] \\
N_w := N_y \\
w = [y].
$$

In the second step every cell in the partitioned input/state-space of the quantised system is mapped to the output/state-space by means of Equations (10.1)–(10.2) (cf. Figure 10.3). The relation of the resulting automaton is then defined by

$$
L = \{(z', w, z, v) | \exists (x,u) \text{ with } x \in Q_x(z), u \in Q_u(v), \\
\quad g(x,u) \in Q_x(z'), h(x,u) \in Q_y(w)\}.
$$

From Figure 10.3 it is clear that, in general, the abstraction process results in a nondeterministic automaton even if the underlying system is deterministic. The cell $([u] = 2, [x] = 2)$ is mapped into the five shaded cells in the output/state-space. As a
10.2. Abstraction of Quantised Systems

Figure 10.3.: Abstraction of a quantised system

consequence the behavioural relation $L$ of the resulting automaton contains five possible transitions for the pair $(v = 2, z = 2)$. In [68] an algorithm has been given which guarantees the completeness of the resulting automaton, i.e. the automata’s behaviour $B_a$ is a superset of the behaviour of the quantised system:

$$B_a \supseteq B_q.$$ (10.10)

Abstraction of continuous-time systems

For the abstraction of continuous-time systems (10.3)–(10.4) the time-line has to be quantised in addition to the quantising of the signal spaces, i.e. the system has to be sampled with the sample time $T$ (cf. Figure 10.4) [102]. The resulting system description

$$x(k+1) = \tilde{g}(x(k), u(k), T),$$ (10.11)

$$y(k) = h(x(k), u(k))$$ (10.12)

is then applied to map the input/state-space to the output/state-space as described in the previous part of this section. Although numerous discretisation methods are available this chapter focuses on a direct sampling using zero order hold (ZOH) elements [49].

Figure 10.4.: Sampling of a continuous system assuming zero order hold (ZOH)
Example 16:
Consider the simple linear continuous system
\[
\begin{align*}
\dot{x} &= -2x(t) + 5u(t) \\
y(t) &= 0.5x(t). 
\end{align*}
\] (10.13)
(10.14)

Assuming a zero order hold and a sampling time of \( T = 1 \) s the sampled system is given by
\[
\begin{align*}
x(k + 1) &= e^{-2T}x(k) + \int_0^T e^{-\alpha T} \, d\alpha 5u(k) \\
&= 0.135x(k) + 2.16u(k) \\
y(k) &= 0.5x(k).
\end{align*}
\] (10.15)
(10.16)

A \( Z \)-transformation of the sampled system results in the \( Z \)-transfer function
\[
G(z) = \frac{1.08}{z - 0.135}
\]
which will be used later in Section 10.4.3.

10.3. Networks of Quantised Systems

As can be seen in Figure 10.5 the compositional modelling of a quantised system introduces additional quantisers and injectors. In the example the internal signal \( r \) is quantised to the qualitative signal \([r]\) and injected to the continuous-valued signal \( >[r]\leq \tilde{r} \). This requires that the injector’s partitioning \( Q_{\tilde{r}} \) contains the quantiser’s partitioning \( Q_r \). In other words, the connection of subsystems demands a fitting interface. In the remainder of this chapter it is assumed that injector and quantiser at the interface between two modular subsystems use the identical partitionings.

It is clear that the information lost through the quantisation cannot be regained through an injection, i.e. in general \( r \neq \tilde{r} \) holds. The second effect, also known as the wrapping-effect in interval arithmetic, is demonstrated in the next example and motivates the following lemma.

Example 17:
Consider the system depicted in Figure 10.6 consisting of two quantised systems in a series connection. Figure 10.7a shows the mapping of one cell in the input/state-space of System 1 to its output/state-space. (The dashed horizontal and vertical lines mark the borders of the partitions.) Because of the quantisation of the output of System 1 this exact area (darker coloured) may not be precisely described but only the larger marked area (light coloured). Therefore, in System 2 the injected signal \( \tilde{r} \) can assume any value within the latter area. Clearly, applying System 2 on this larger set leads to an also larger set for the output \( y \) of the overall system (Figure 10.7b).
10.4. Abstraction of Networks

10.4.1. Approach to Abstraction

The difference between the monolithic and modular abstraction process of composite quantised systems is clarified in Figure 10.8. In the monolithic case the relation between the qualitative input and output of the network, i.e. the signals $[u]$ and $[y]$, is of interest. The output is thereby gained by using the continuous-variable model of the overall system. In the modular case each component is abstracted separately. The abstraction of the left system yields an automaton model which describes the relation $x_{i}(k+1) = 0.135x(k) + 2.16u(k)$ and $r(k) = 0.5u(k)$, and the abstraction of the right system yields an automaton model which describes the relation $y(k) = 0.5x_{i}(k) + 0.5r(k)$. The behaviour of a monolithic quantised system $B_{mono}$ is a subset of the behaviour $B_{mod}$ of the same system using a modular quantisation, i.e.

$$B_{mod} \supseteq B_{mono}$$

holds.

PROOF:
See Appendix B.4.2.

10.4. Abstraction of Networks
10. Abstraction of Networks of Quantised Systems

Figure 10.7.: Exemplary comparison of the behaviour of modular and monolithic quantised systems. The lighter shaded area marks the behaviour of the modular system.

between the signals $[u]$ and $[r]$, whereas the abstraction of the right system results in an automaton model which describes the relation between the signals $[r]$ and $[y]$. For both abstraction approaches it is demanded that the global behaviour, i.e. the relation between the signals $[u]$ and $[y]$ is complete. The modular abstraction is investigated in the following sections and its result is compared to the result of the monolithic abstraction described in Section 10.2.2.

10.4.2. Abstraction of discrete-time systems

It has been proven above that the introduction of additional quantisers changes the output of the overall network of quantised systems with the consequence that its behaviour increases, i.e. more input/output-sequences are possible. This consequence can directly be transferred to the behaviour of the automata networks which have been gained from the modular abstraction of the quantised system.

Let $A_{mono}$ with the relation $L_{mono}$ denote the automaton gained through monolithic
10.4. Abstraction of Networks

a) Monolithic

b) Modular

Figure 10.8.: Non-causal information flow in composite quantised systems

abstraction of the quantised system. In case of the exemplary quantised system depicted in Figure 10.5a the relation is given by

$$L_{\text{mono}} = \{(\hat{z}', w, \hat{z}, v) | \exists (x_1, x_2, u) \in Q_{x_1x_2}(\hat{z}), u \in Q_u(v),
\tilde{g}(x_1, x_2, u) \in Q_{x_1x_2}(\hat{z}'), \tilde{h}(x_1, x_2, u) \in Q_y(w)\}, \quad (10.17)$$

where $\tilde{g}, \tilde{h}$ denote the state and output equations of the overall continuous-variable system and $Q_{x_1x_2}$ denotes the partitioning of its state-space. The relations of the two automata gained through compositional abstraction (Figure 10.5b) are given by

$$L_1 = \{(z'_1, s, z_1, v) | \exists (x_1, u) \text{ with } x_1 \in Q_{x_1}(z'_1), u \in Q_u(v),
\tilde{g}_1(x_1, u) \in Q_{x_1}(z'_1), \tilde{h}_1(x_1, u) \in Q_r(s)\},$$

$$L_2 = \{(z'_2, w, z_2, s) | \exists (x_2, \tilde{r}) \text{ with } x_2 \in Q_{x_2}(z_2), \tilde{r} \in Q_r(s),
\tilde{g}_2(x_2, \tilde{r}) \in Q_{x_2}(z'_2), \tilde{h}_2(x_2, \tilde{r}) \in Q_y(w)\}.$$
an abstraction of the same system using modular quantisation, i.e.

\[ L_{\text{mono}} \subseteq L_1 \Join L_2 \Join \cdots \Join L_\gamma \]

holds.

PROOF:

See Appendix B.2.6

Theorem 10.2 (Completeness):
The system resulting from a modular abstraction of a network of discrete-time quantised systems is complete.

PROOF:
It has been proven in [93] that the automaton resulting from an abstraction of a quantised system is complete, i.e. the automaton’s behaviour includes the system’s behaviour. Thus, \( L_{\text{mono}} \) is complete. Because Theorem 10.1 holds, the system gained by modular abstraction is also complete.

10.4.3. Abstraction of continuous-time systems

The compositional abstraction of networks of continuous-time quantised systems introduces additional complications caused by the sampling of the subsystems. It is known in continuous system theory that the sampling of the overall system (Figure 10.9a) leads to different results than the combination of the sampled subsystems (Figure 10.9b). This is demonstrated in the following example.

![Figure 10.9](image)

**Figure 10.9:** Different approaches for sampling compositional systems

**Example 18:**
Consider a network consisting of two identical systems given by Equations (10.13)–(10.14) in a serial connection as depicted in Figure 10.9. If both systems are sampled separately the resulting system \( G_{\text{mod}} \) is given by

\[ G_{\text{mod}}(z) = G(z) \cdot G(z) = \frac{1.17}{(z - 0.135)^2}. \]
However, the sampling of the overall system
\[\ddot{x} = \begin{pmatrix} -2 & 0 \\ -2.5 & -2 \end{pmatrix} \dot{x}(t) + \begin{pmatrix} 5 \\ 0 \end{pmatrix} u(t)\]  
\[\ddot{y}(t) = (0 \ 0.5) \dot{x}(t)\]  
results in the Z-transfer function
\[\tilde{G}_{\text{mono}}(z) = \frac{0.93z + 0.24}{(z-0.135)^2},\]
which clearly differs from the above function except for \(z = 1\). □

The reason for this difference lies in the introduction of additional hold elements. It is therefore not to be expected that the systems resulting from modular and compositional abstraction are identical. Indeed, the completeness of the compositional abstracted system is not guaranteed. This is demonstrated by the following example.

**Example 19:** Consider the system depicted in Figure 10.10 illustrating a reservoir with an overflow. Assume the level of water \(h_1\) to be above the connecting pipe at the time \(T\) causing the water level \(h_2\) to rise. If shortly after this time \(T + \delta T\), \(h_1\) falls below the pipe level, the water level \(h_2\) will remain constant. However, the discrete-time model with ZOH assumes \(h_1\) to be above the pipe level until the next sampling time \(2T\). Now assume the partitioning of the signals as depicted in Figure 10.10 by the dashed horizontal lines. Then the water level \(h_2\) lies in different partitions depending on the assumed trend of \(q\). Thus, the abstraction returns an automaton which does not necessarily include the behaviour of the original system. The resulting model is not complete. □

This consequence is stated in the following lemma.

**Lemma 10.2:** The completeness of the system resulting from the modular abstraction of a network of continuous-time quantised systems is not guaranteed. □

### 10.5. Assuring Completeness

Input signals whose values change distinctly during the sampling period \(T\) result in large deviations between the original and the sampled systems. Whereas system inputs can be set accordingly by an operator, the internal coupling signals cannot be influenced directly, but are given through the system dynamics. The obvious way to assure that the assumption of constant input signals is not violated is to reduce the sampling period \(T\) until the input signals stay within certain bounds.
However, with a small $T$ the abstracted system usually does not fulfill the demands for applications such as observation and diagnosis. Here, the discrete-event model should include the major dynamics. For a small $T$ the signals usually stay in the same partition and the resulting automaton consists almost entirely of self-loops, i.e. transitions which do not leave the current state $z$. As a consequence the major movement is not visible any more. To put it in a nutshell, this approach to assure completeness is not applicable whenever the sampling period cannot be freely chosen, because of application requirements.

The approach suggested in the remainder of this section takes the movement of the input signal into account by giving an upper and lower border for its movement and abstracting the system for all values within these bounds. As a result it is guaranteed that the original system behaviour is included in the behaviour of the automaton model (completeness) at the cost of an increased behavioural relation. This approach is explicated in the following.

Again, this approach is presented for the serial connection depicted in Figure 10.5b without loss of generality. Because all signals $x_1$, $u$ and $r$ are bounded there exists a Lipschitz-constant $\Lambda > 0$ for the output function $\dot{h}_1$ such that
10.5. Assuring Completeness

\[ \| h_1(x_1(t_1), u(t_1)) - h_1(x_1(t_2), u(t_2)) \| \leq \Lambda \left\| \begin{pmatrix} x_1(t_1) - x_1(t_2) \\ u(t_1) - u(t_2) \end{pmatrix} \right\| \] (10.20)

\[ \| r(t_1) - r(t_2) \| \leq \Lambda \left\| \begin{pmatrix} x_1(t_1) - x_1(t_2) \\ u(t_1) - u(t_2) \end{pmatrix} \right\| \] (10.21)

holds for any two time points \( t_1 \) and \( t_2 \). Analogously, a boundary for the derivative exists:

\[ \| g_1(x_1(t_1), u(t_1)) - g_1(x_1(t_2), u(t_2)) \| \leq \Lambda_x \left\| x_1(t_1) - x_1(t_2) \right\|. \] (10.22)

The maximum alteration of the input \( \| u(t_1) - u(t_2) \| \) is also given. Then for every initial value of \( r \) the constant \( \Lambda \) can be used to give an upper and lower bound for the signal trajectory:

\[ \| r(t_1) - r(t_2) \| \leq \Lambda d_{\text{max}}. \] (10.23)

This can be extended to trajectory bundles by adding the boundaries to the minimum and maximum value of the trajectories’ starting points. Figure 10.11 shows exemplary the boundary of a trajectory bundle where the starting set is one partition in \( Q_r \).

![Bounded trajectory bundle](image)

Figure 10.11.: Bounded trajectory bundle

To assure the completeness the alteration of \( r \) within these boundaries has to be considered in the abstraction process of the subsystems which use \( r \) as an input signal (cf. Figure 10.12). With the extended partition

\[ \tilde{Q}_r(s) = [s_{\text{low}} - \Lambda d_{\text{max}}, s_{\text{up}} + \Lambda d_{\text{max}}] \] (10.24)

the behavioural relation of the automaton gained by the abstraction of the second
10. Abstraction of Networks of Quantised Systems

The system of Figure 10.5b is given by

\[ L_2 = \{ (z'_2, w, z_2, s) | \exists (x_2, \tilde{r}) \text{ with } x_2 \in Q_{x_2}(z_2), \tilde{r} \in \hat{Q}_r(s), \]
\[ g_2(x_2, \tilde{r}) \in Q_{x_2}(z'_2), h_2(x_2, \tilde{r}) \in Q_y(w) \} \]

(10.25)

Figure 10.12.: Abstraction using an overestimation of the original partition

The theorem stated below follows directly from this derivation.

**Theorem 10.3 (Complete abstraction of modular continuous-time systems):**

An automaton \( A \) gained by abstraction of a quantised system with an input signal, which is non-constant during a sampling-period \( T \), is complete if every partition \( Q_r(i) \) is extended according to Equation (10.24).

\( \square \)

10.6. Conclusion

The major result of this chapter is that the modular abstraction of composite quantised systems is possible and the demand for completeness can be fulfilled. Whereas for discrete-time systems the modular abstraction is straightforward, additional precautions have to be taken for continuous-time systems or else the completeness might be lost. Although no analysis of the computational costs of the abstraction process is given here, it stands to reason that the modularisation reduces the costs distinctly. The advantages of the modular representation is retained throughout the modelling process.

The drawback of a modular abstraction is a over-estimation of the automaton model. Especially in the abstraction of continuous-valued systems the overestimation of the signal’s movements may lead to a model with a behaviour which is far larger than the original system behaviour. Although the resulting model is complete, its explanatory power and therefore its applicability in process supervision might then be limited if no precautions are taken.
Direct feedback in synchronised automata networks introduces a direct dependency of a component input upon itself through a signal path within the network. In continuous system theory such a feedback yields an algebraic loop, which may render the overall system ill-posed. In this chapter the feedback problem is formally introduced and criteria for testing if the feedback renders the network ill-defined are given. After a short introduction the feedback problem is described in Section 11.2 for both continuous and discrete systems. In Section 11.3 criteria for testing whether a feedback leads to a well-defined system are presented.

11.1. Introduction

As described in the previous chapters a network consists of several interconnected automata whereby all state transitions are synchronised by the same clock. This synchronisation may become ill-defined if the state transitions of one or more automata depend upon each other through direct feedback paths within the network. This problem, called the feedback problem occurs whenever a signal depends directly and instantaneously on itself. This may lead to a conflict resulting in blocking or even a not well-defined system.

This chapter investigates the problem of direct feedback in automata networks with synchronised signals and develops criteria to test whether a given automata network is well-defined. So far, in literature the problem has been avoided by simply excluding conflicts per definition or by dealing only with Moore automata for which this problem cannot occur [62, 77].
11.2. Feedback in Interconnected Systems

Direct feedback in interconnected systems means that a signal depends directly on itself:

\[ y = f(y). \]  \hspace{1cm} (11.1)

This is not to be confused with closed loops as used in control theory which contain integrators where the signal depends on its temporal derivative and not directly on itself. A solution of Equation (11.1) is called fixed point. A direct feedback may have no, one or more than one fixed points. A feedback with a unique fixed point \( \bar{y} = f(\bar{y}) \) is called well-formed [62].

Definition 11.1 (well-formed): An implicit equation (11.1) is called well-formed if and only if it has exactly one solution.

11.2.1. Direct Feedback in Continuous Systems

In continuous systems theory, direct feedback is well known. The relation (11.1) is the result of an “algebraic loop” within interconnected systems. An example for a direct feedback is the equation \( y = 1 - y \). It has the unique solution \( \bar{y} = 0.5 \) and is therefore well-formed. If Equation (11.1) has no or more than one solution, the overall system is said to be ill-defined. Methods to detect algebraic loops and to avoid them has been studied in literature about simulation techniques and large-scale systems [64,92].

The solution of such an algebraic loop is often found numerically by transforming the problem into finding the root in \( 0 = \bar{y} - f(\bar{y}) \) [64, 92]. For a well-formed loop this problem can be solved using standard methods of numerical theory. However, in discrete systems the signal values are often symbols and therefore algebraic operations are not defined. This prohibits to transfer the methods of continuous system theory to discrete systems.

11.2.2. Direct Feedback in Discrete Systems

In analogy to continuous systems, a direct feedback in discrete systems results when the output \( w \) depends directly on itself as shown in Fig. 11.1. Closing the loop forces the input to be equal to the output: \( v(k) = w(k). \)\(^1\) If the output \( w(k) \) depends directly on the input \( v(k) \) by means of the output relation \( H \), a relation of the form (11.1) occurs. Whenever the output relation forces the output to assume a value different from

\(^1\)Therefore \( \mathcal{N}_v \supseteq \mathcal{N}_w \) must hold. This will be assumed for the remainder of the paper.
the input, a conflict occurs. The main aim of this chapter is to investigate this situation.

Note, that this problem is restricted to synchronous systems. Indeed, because the inputs and outputs in asynchronous systems are not evaluated at the same time, it is impossible for conflicts as described above to occur [60].

Example 20:
Given is a deterministic automaton \( \mathcal{A}^d \) with \( \mathcal{N}_v = \mathcal{N}_w = \{1, 2\} \) and a feedback connection as depicted in Figure 11.1. If the automata’s output relation \( H^d \) generates the output \( w = 1 \) whenever the input is \( v = 2 \) and \( w = 2 \) whenever \( v = 1 \), the feedback condition \( w = v \) cannot be fulfilled. The loop is ill-formed.

11.2.3. The Origin of Loops

Algebraic loops do not exist in reality. Their origin is a direct consequence of the modelling assumption which creates a simplified description of the physically existing system. Neglecting details, which seem not to contribute to the main properties of the system, leads to the mathematical deficiency. In continuous systems such loops usually occur when neglecting parasitical resistors or when approximating quick dynamical effects by static elements.

In discrete systems such loops may occur, because the technological system is simplified by sampling and quantising the signals as described in Chapter 10. Figure 11.2 shows two examples for loops in discrete systems. In the left example the exit of the XOR-gate is not stable and the loop is not well-formed. The circuit can be designed, but what happens depends on the neglected propagation and delay times. In the right example a discrete system is controlled by a discrete controller. If the system has a direct feed-through and the controller is a simple static controller (\( v(k) = h(w(k)) \)), the loop may not be well-formed. The cause of this lies in the assumption that the signal values can change infinitely fast. In large coupled systems it may not be intuitively clear if a loop is well-formed or if the modelling process has to account for additional details.
11.2.4. Direct Feedback Condition

The direct feedback problem occurs only in components which are strongly connected and have direct feed-through. Two components $i$ and $j$ are strongly connected if there exists a signal path $p$ from component $i$ to $j$ as well as from component $j$ to $i$. Both paths together form a loop. A component has a direct feed-through if its output $w(k)$ depends on its input $v(k)$. A Moore automaton has no direct feed-through. Instead, its output depends only on the automaton state $z(k)$ as defined in Section 3.2.10. The lemma stated below follows directly from that.

**Lemma 11.1 (Moore automata in loops [62]):** In an automata network a loop is well-formed if it contains one or more Moore automata.

The question to be answered in the next section is under what conditions ill-formed loops occur in coupled automata for which the output function depends explicitly on the input $v$.

11.3. Solution to the Feedback Problem

The basic problem can be studied for a simple network consisting of a single automaton with a feedback connection as depicted in Figure 11.1. The equivalent automaton of this autonomous system can be obtained by applying the composition rules given in Section 3.3.4. However, only if the feedback loop is well-formed the composition results in an autonomous automaton according to the definitions given in Section 3.2.5. In the following section it is investigated under what conditions the composition of a deterministic, nondeterministic or stochastic automaton with feedback is well-formed. The results are then extended to automata networks and illustrated by extensive examples (cf. [125]).

11.3.1. Single Automaton with Feedback
11.3. Solution to the Feedback Problem

Deterministic Automaton

The formal composition of a deterministic automaton (3.2) with a feedback connection (Fig. 11.1) results in an autonomous system \( \hat{A}^d = (\mathcal{N}_z, \hat{L}^d, z(0)) \). The relation \( \hat{L}^d \) is given by

\[
\hat{L}^d(z', z) = \sum_{w=1}^{R} L^d(z', w, z, w), \tag{11.2}
\]

assuring \( v = w \) for all \( w \in \mathcal{N}_w \).

**Definition 11.2:** The composition of a deterministic automaton with a feedback connection is said to be *well-formed* if for its output function the relation

\[
\sum_{w=1}^{R} H^d(w, z, w) = 1, \quad \forall z \in \mathcal{N}_z \tag{11.3}
\]

holds.

**Theorem 11.1:**
The equivalent automaton of a system consisting of a deterministic automaton with a feedback connection is again a deterministic automaton iff the feedback composition is well-formed.

**Proof:**
Transforming Relation (11.3) results in

\[
\sum_{w=1}^{R} H^d(w, z, w) = 1 \iff \sum_{w=1}^{R} \sum_{z'=1}^{N} L^d(z', w, z, w) = 1 \\
\iff \sum_{z'=1}^{N} \hat{L}^d(z', z) = 1,
\]

whereby the last relation satisfies Equation (3.16).

**Corollary 11.1:** If the relation \( \sum_{w=1}^{R} H^d(w, z, w) > 1 \) is satisfied, the feedback connection leads to a nondeterministic automaton. On the other hand, \( \sum_{w=1}^{R} H^d(w, z, w) = 0 \) results in an automaton which is not live.

**Example 21:**
Consider the deterministic automaton \( A^d \) with \( \mathcal{N}_z = \{1, 2\} \) and \( \mathcal{N}_v = \mathcal{N}_w = \{1, 2\} \). Its dynamics is given in Fig. 11.3a as an automaton graph. The arcs are labelled with the respective
input/output pair \( v/w \). A feedback loop as shown in Fig. 11.1 is introduced, resulting in the autonomous system shown in Fig. 11.3b. As Theorem 11.1 is satisfied the resulting system is a deterministic automaton. Obviously, the system is obtained by eliminating all transitions with \( v \neq w \).

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig11_3.png}
\caption{Automata graphs for Example 21}
\end{figure}

\textbf{Example 22:}
Consider the deterministic automaton \( \mathcal{A}^d \) with \( \mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w \) defined as above, whose dynamics is given in Fig. 11.4a. The composition of \( \mathcal{A}^d \) with a feedback loop results in the system shown in Fig. 11.4b. Theorem 11.1 is not fulfilled, therefore the resulting system is not a deterministic automaton.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig11_4.png}
\caption{Automata graphs for Example 22}
\end{figure}

\textbf{Nondeterministic Automaton}

Analogously to the deterministic case the nondeterministic automaton (3.7) with feedback is an autonomous system \( \hat{\mathcal{A}}^n = (\mathcal{N}_z, \hat{L}^n, z(0)) \) with the relation

\[ \hat{L}^n(z', z) = R \bigvee_{w=1} L^n(z', w, z, w), \quad (11.4) \]

assuring \( v = w \) for all \( w \).

\textbf{Definition 11.3:} The composition of a nondeterministic automaton with a feedback connection is said to be \textit{well-formed} if for its output function the relation

\[ \bigvee_{w=1}^R H^n(w, z, w) = 1, \quad \forall z \in \mathcal{N}_z \quad (11.5) \]
Theorem 11.2:
The equivalent automaton of a system consisting of a nondeterministic automaton with a feedback connection is a nondeterministic automaton iff the feedback composition is well-formed.

Proof:
Transforming Relation (11.5) results in
\[ R \bigvee_{w=1}^{H}(w,z,w) = 1 \iff R \bigvee_{w=1}^{N} L^n(z',w,z,w) = 1 \]
\[ \iff \bigvee_{z'=1}^{N} \hat{L}^n(z',z) = 1, \]
whereby the last relation satisfies Equation (3.17).

Corollary 11.2: If \( \sum_{w=1}^{R} H^n(w,z,w) = 0 \) holds, the feedback connection leads to a nondeterministic automaton which is not live.

Stochastic Automaton
The feedback of a stochastic automaton (3.11) results in an autonomous automaton \( \hat{A}^s = (\mathcal{N}_z, \hat{L}^s, p(z(0))) \) [81,93] with the relation
\[ \hat{L}^s(z'|z) = \sum_{w=1}^{R} L^s(z',w|z,w). \] (11.6)

Definition 11.4: The composition of a stochastic automaton with a feedback connection is said to be well-formed if for its output function
\[ \sum_{w=1}^{R} H^s(w|z,w) = 1, \forall z \in \mathcal{N}_z \] (11.7)
11. Direct Feedback in Automata Networks

Theorem 11.3:
The equivalent automaton of a system consisting of a stochastic automaton with a feedback connection is a stochastic automaton iff the feedback composition is well-formed.

Proof:
Transforming Relation (11.7) results in
\[ \sum_{w=1}^{R} H^x(w|z, w) = 1 \iff \sum_{w=1}^{R} \sum_{z'=1}^{N} L^x(z', w|z, w) = 1 \]
\[ \iff \sum_{z'=1}^{N} \hat{L}^x(z'|z) = 1, \]
whereby the last relation satisfies Equation (3.18).

Interestingly, if equation (3.18) is not satisfied the resulting equivalent automaton (11.6) is not a representation of a stochastic process any more. As opposed to the deterministic or nondeterministic case, the class of automata which fulfills the condition of the above theorem is expected to be very small compared to the class of stochastic automata in general. Also, a well-formed loop with a nondeterministic automaton embedded in a given stochastic automaton is a necessary but not a sufficient condition for the stochastic automaton to be well-formed.

Example 23:
Consider a stochastic automaton \( \mathcal{A}^x \) with \( N_z = \{1, 2\} \) and \( N_v = N_w = \{1, 2\} \). Its dynamics is given in Fig. 11.5a as an automaton graph. The arcs are labelled with the respective input/output pair and the probability for the transition \( v/w/P \). The introduction of a feedback connection results in the system shown in Fig. 11.5b. As
\[ \sum_{w=1}^{2} H^x(w|z, w) = 0.85 \neq 1 \quad \forall z \in \{1, 2\} \]
holds, the system is not well-formed. The resulting system (Fig. 11.5b) is not a stochastic process as the sum of the probabilities of the transitions leaving a state is not 1.

11.3.2. Feedback in Automata Networks
The above definitions can easily be extended to cover automata with multiple inputs and outputs and automata networks. This will be done in this section for stochastic
11.3. Solution to the Feedback Problem

![Automata graphs](image)

Figure 11.5.: Automata graphs for Example 23

automata only, bearing in mind that the definitions for the deterministic and nondeter-
ministic case can be derived easily therefrom.

**Definition 11.5:** The composition of a feedback loop containing \( \gamma \) stochastic au-
tomata is said to be well-formed if

\[
\sum_{w^1=1}^{R^1} \cdots \sum_{w^\rho=1}^{R^\rho} \sum_{s^1=1}^{Q^1} \cdots \sum_{s^\kappa=1}^{Q^\kappa} \prod_{i=1}^{\gamma} H^i_{s^i} = 1
\]  

(11.8)

holds for all state and input values of the network.  

This means that the output relations \( H^i_{s^i} \) of all automata \( i \) in the feedback loop are evaluated for all outputs \( w \) and coupling signals \( s \) that occur in the loop. The relations are then multiplied (element by element) and the result is summed up for all possible signal values.

**Theorem 11.4:**

The equivalent automaton of a system consisting of a feedback loop containing stochastic automata is a stochastic process iff the loop is well-formed.  

**Proof:**

See Appendix B.4.3

As in general the stochastic automata of a network are not stochastically indepen-
dent the above definition cannot be reformulated as a criterion of the form

\[ \sum H_1 \cdot \sum H_2 \cdot \ldots \cdot \sum H_\gamma, \]

which would make it possible to test the automata separately. Hence, for the network to be well-formed it is neither necessary nor sufficient that the single automata of the network are well-formed.
Example 24 (cf. also [93]):
Consider a stochastic automata network with two automata $A^s_1$ and $A^s_2$ as depicted in Figure 11.6. To test the conditions of Theorem 11.4 the sum (11.8) has to be determined. For the state $z_1 = 1$ of automaton $A^s_1$ and state $z_2 = 1$ of automaton $A^s_2$ this results to
\[
H^s_1(s = 1|z_1 = 1, w = 1)H^s_2(w = 1|z_2 = 1, s = 1) + \\
H^s_1(s = 1|z_1 = 1, w = 2)H^s_2(w = 2|z_2 = 1, s = 1) + \\
H^s_1(s = 2|z_1 = 1, w = 1)H^s_2(w = 1|z_2 = 1, s = 2) + \\
H^s_1(s = 2|z_1 = 1, w = 2)H^s_2(w = 2|z_2 = 1, s = 2) = \\
1 \cdot 0.5 + 0.5 \cdot 0.5 + 0 \cdot 0.5 + 0.5 \cdot 0.5 = 1.
\]

Figure 11.6.: Stochastic automata network for Example 24

As the sum for the remaining state combinations $(z_1 = 1, z_2 = 2)$, $(z_1 = 2, z_2 = 1)$, and $(z_1 = 2, z_2 = 2)$ is also 1, the theorem is fulfilled and the composition is well-formed. Hence the automaton network describes a stochastic process. The same result could have been obtained by applying Lemma 11.1, because automaton $A^s_2$ has no direct feed-through meaning its output $w(k)$ does not depend on its input $s(k)$.

\[\square\]

11.4. Bypassing the Feedback Problem

The feedback problem can be bypassed according to Lemma 11.1 by inserting a Moore automaton into the feedback loop. The simplest Moore automaton for this task is a shift register. This, of course, does not solve the original feedback problem, but simply creates a new network with a similar behaviour as the original network with the advantage of being well-formed. The drawback of this procedure is that the completeness of the new model is not guaranteed. However, practical experience has shown that the deviations between the behaviours of the original network and the network with shift register are usually very small. The applicability of such an altered network model is demonstrated in the following example.
11.5. Conclusion

Example 25:
Consider a two-tank system as depicted in Figure 11.7. Both tanks are connected via a pipe. The modular abstraction of these two components results in a network model as shown in Figure 11.8a. The feedback arises, because the liquid level of each tank depends directly on the other tank. In general, this network is not well-formed. The insertion of a shift register results in the altered network given in Figure 11.8b. Figure 11.9 shows the result of a simulation of the behaviour of the altered network together with a simulation of the original continuous-valued system. It can be seen that the signals overlap completely. At least for this input sequence the insertion of the shift register has not led to a noticeable alteration of the network output.

![Figure 11.7: Two-tank system](image)

![Figure 11.8: Block diagram of the two-tank system](image)

11.5. Conclusion

The feedback problem has been posed for automata in analogy to the algebraic loop in continuous-variable systems. The feedback in networks with synchronised signals can render the overall system ill-defined whenever no unique solution for the loop exists. Necessary and sufficient conditions have been found for determining if the composition of a direct feedback loop is well-formed. As these conditions can be implemented easily they are an enhancement to the “brute-force” testing methods that have been used so far.
However, the criteria are rather restrictive. Especially for stochastic automata networks it is to be expected that, in general, they are not well-formed. On the other hand, the example has shown that in practical cases the insertion of a shift register might only lead to small deviations in the behaviour in which case it is possible to use the model in applications like simulation or observation. It has then to be kept in mind that the completeness is not guaranteed any more.
This chapter demonstrates the applicability of the given approaches through application examples. Section 12.1 gives an example of the diagnosis of the air-path of a diesel engine with a turbo charger. Sections 12.2 and 12.3 describe the diagnosis and simulation of a three-tank system.

12.1. Centralised Diagnosis of the Air-Path of a Diesel Engine

Nowadays, diagnosis of automobiles is mainly restricted to signal-based methods, where the measured sensor values are tested for compliance with threshold values [26, 88]. Although signal-based methods can only identify a limited class of faults, they have become widely accepted because of their simple and cheap implementation. However, faults that do not alter the static system behaviour but only influence the signal dynamics cannot be found by using these methods. Then methods have to be applied that use a dynamic model of the plant such as the diagnostic methods presented in this book. The remainder of this section describes the successful application of the approach given in Chapter 7 on the air-path of a diesel engine [112].

The air-path that is to be diagnosed is used in combination with turbo-charged diesel engines and consists of the following components (Figure 12.1): the hot-wire air flow meter (HFM), the air-compressor, the intercooler, the container for the compressed gas, the cylinder for the combustion of the fuel-air-mix, the exhaust gas recirculation valve (EGR valve), the turbine with a variable turbine geometry (VTG), and the exhaust pipe. The air-path is subject to the input signals: engine speed $n_E$, fuel flow per stroke $m_F$, the effective area of the EGR valve $A_{EGR}$, and of the turbine $A_{VTG}$. The following output signals can be measured: the incoming air flow $q_1$, the pressure $p_2$ in
12. Application Example

Figure 12.1.: Air-path of a diesel engine

the container, and the temperature $T_2$ in the container. The continuous-variable system model contains seven state variables, four input and three output signals (Fig. 12.2). The special structure of the air-path results in a strong coupling among the different system states. The recirculated exhaust gas affects the air-mix that streams into the engine by changing its temperature, pressure, and oxygen content. In turn, the air-mix influences the exhaust gas leaving the engine. A second feedback couples the input air flow with the exhaust gas flow through the turbo charger.

Figure 12.2.: Input and output signals of the air-path

Faults that occur in the components of this system influence the emission, which is subject to legal restrictions. Therefore the sensor and actuator faults listed in Table 12.1 are considered for diagnosis. The diagnostic task is to detect and identify the fault cases $f = 1, \ldots, 10$ using only the available sensor and actor information described in the previous paragraph.

In the first step the signals are quantised and the system is abstracted to a stochastic automaton. Fig. 12.3 shows the quantisation of the signal $q_1$. The rectangles indicate
12.1. Centralised Diagnosis of the Air-Path of a Diesel Engine

<table>
<thead>
<tr>
<th>$f$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Faultless operation</td>
</tr>
<tr>
<td>2</td>
<td>HFM: offset +0.02 kg/s</td>
</tr>
<tr>
<td>3</td>
<td>HFM: breakdown 0.0001 kg/s</td>
</tr>
<tr>
<td>4</td>
<td>HFM: drift -20%</td>
</tr>
<tr>
<td>5</td>
<td>EGR: blocked closed</td>
</tr>
<tr>
<td>6</td>
<td>EGR: blocked open</td>
</tr>
<tr>
<td>7</td>
<td>$T_2$-sensor: drift +30%</td>
</tr>
<tr>
<td>8</td>
<td>$T_2$-sensor: drift -30%</td>
</tr>
<tr>
<td>9</td>
<td>$p_2$-sensor: drift +30%</td>
</tr>
<tr>
<td>10</td>
<td>$p_2$-sensor: drift -30%</td>
</tr>
</tbody>
</table>

Table 12.1: Fault cases

in which interval the signal lies at the respective time instant. As can be seen in the figure, the trajectories of the signal $q_1$ for the two fault cases $f = 1$ and $f = 4$ can be distinguished easily in spite of the rough discretisation.

Figure 12.3: Qualitative and quantitative signal $q_1$ for the fault cases $f = 1$ (black) and $f = 4$ (blue/grey)

The results obtained by diagnosing the different faults are partially presented in the following. For the full results confer [112]. As described in Chapter 7, the diagnostic algorithm returns a probability for the occurrence of each fault. This probability is
12. Application Example

marked by the intensity of the bars in the following diagrams. A dark grey bar means that the occurrence of this fault is probable whereas a light grey bar an unlikely fault. The absence of a bar means that this fault can be excluded.

It has turned out that the Faults $f = 3$, $f = 7$, $f = 8$, $f = 9$ and $f = 10$ are very easy to be diagnosed. All of them are diagnosed correctly as soon as the diagnosis starts. This is exemplarily shown for Fault $f = 9$ in Fig. 12.4. Is the air path affected by the fault $f = 2$, the fault cases $f = 1$, $f = 2$ and $f = 4$ are stated possible when the diagnosis starts (Figure 12.5). During the diagnosis the probability of $f = 4$ decreases continuously and the probability of $f = 2$ increases accordingly. At 1.9 seconds a measurement has occurred that is impossible in the faultless case. Therefore the faultless case is excluded and the diagnostic system returns the correct result: the fault case $f = 2$ is identified. The identification of the other fault cases is just as good except for $f = 6$, which can only be detected, but not identified.

12.2. Diagnosis of a Three-Tank System

12.2.1. Application Setup

The application example used to test the the different diagnostic approaches is a three-tank system as depicted in Figures 12.6 and 12.7 which has been available at
12.2. Diagnosis of a Three-Tank System

the Institute of Automation and Computer Control at the Ruhr-Universität Bochum. The system consists of three liquid tanks which are connected through pipes with electronic valves. Additionally, each tank has an outflow which can be opened using manually operated valves to simulate leakages. Tank 3 has two further outflows with electronic valves. Tank 1 and Tank 3 can be filled with water by pumps whose power can be set continuously. The maximum liquid level of each tank is 60cm. Each tank is equipped with a continuous level sensor and five discrete level sensors which can only detect if the water level is below or above the sensor.

As a first step the system has been modelled as a continuous state-space model where the liquid levels of the three tanks are the system states and also the system output. The valve positions and the power of the pumps form the input signals [30]. If the tanks are treated as separate components, as it is the case in this section, each tank needs to know the liquid level of the neighbouring tanks. Only then each component has the full information to determine the value of its system state, because the out- and inflow of a tank depends on the liquid level of the neighbouring tanks. The block diagram of the three-tank system is given in Fig. 12.8.

For the abstraction process (cf. Chapter 10) the signals have to be quantised. For the qualitative diagnosis the discrete level sensors are used which naturally divide the state space of each tank into six equally spaced intervals. All valves are considered as discrete with the two discrete states “open” and “closed”. The power of the two pumps is divided into ten intervals each from “shut off” to “full power”.

The three tanks are abstracted separately with a sampling time of 15s resulting in three relations. The relation of Tank 1 contains 5,264 transitions, which is only a
fraction of the number of possible transitions: $|\mathcal{N}_{v_1}| \cdot |\mathcal{N}_{w_1}| \cdot |\mathcal{N}_{z_1}|^2 = 103680$. The relation of Tank 2 contains 12,169 transitions and the relation of Tank 3 even 20,279 transitions. The abstraction of each tank took less than 10 minutes. To compare the numbers to the monolithic representation the system has also been abstracted to a single automaton. This equivalent automaton has $1.44 \cdot 10^8$ transitions (using about 5 GByte of memory) and its abstraction took more than a week. Because the NAN contains loops it had to be tested for direct feedback. The test showed that the overall network is well-formed. However, because the system has not been abstracted using the method\textsuperscript{1} described in Section 10, the completeness of the model is not guaranteed. But because simulation tests showed no inconsistent IO-sequences, the automaton model can be seen as “quite complete”.

12.2.2. Diagnosis

For the diagnosis the following setup has been chosen. All three tanks may have a leakage simulated by opening the valves V1X, V2X, or V3X, i.e. the respective signals are treated as unmeasurable fault signals. To make the diagnostic task more

\textsuperscript{1}This method has not been implemented yet.
12.2. Diagnosis of a Three-Tank System

difficult it is said that the liquid level of Tank 2 is not measurable. The valves $V_{30H}$ and $V_{30L}$ are closed, all other valves are opened. Both pumps run with full power.

In decentralised diagnosis each tank is diagnosed independently. Because Tank 2 does not possess a measurable output signal, it is not diagnosable and it is not possible to exclude any of the local faults. The diagnosis of Tank 1 and Tank 3 returns the correct result (leakage in both tanks) after three steps as shown in Figure 12.9. In Figure 12.10 the liquid levels of all three tanks are given for the faultless case and in case of a leakage in all tanks. It can be seen that the trajectories differ greatly for the faultless and faulty case, which, in principle, allows for a quick and safe diagnosis. However, the lack of information about the surrounding tanks and therefore about the flow through the connecting pipes delays the exclusion of the faultless case.

In the coordinated diagnosis a coordinator refines the diagnostic result of the decentralised diagnosers as described in Section 6 by reconstructing the unmeasurable coupling signals. Thus, the diagnosis of Tank 2 is possible in spite of the lack of measurable signals. The diagnostic result depicted in Figure 12.11 shows that the coordination greatly enhances the result of the diagnosis.
12. Application Example

Figure 12.9.: Decentralised diagnosis of a three-tank system

Figure 12.10.: Liquid levels of the tanks. The dashed lines mark the faultless case.

12.3. Simulation of a Three-Tank System

The application of the simulation approach with on-line composition as introduced in Chapter 9 (cf. also [122]) is demonstrated using a variation of the three-tank system where all valves are constantly set to get the setup as depicted in Figure 12.12a. Each tank is modelled separately by a SA, whereby the states $z_1$, $z_2$, and $z_3$ of the automata encode the liquid levels $h_1$, $h_2$, and $h_3$ of the respective tanks. These liquid levels are also the automata’s output. The first tank can be filled by means of a pump. Its current delivery is encoded as the network’s input $v^1$. The automata of the network exchange the information about their liquid levels as depicted in Figure 12.12b. It is assumed that the liquid level of Tank 1 is not measurable, therefore the output of automaton $A_1$ is modelled as the unmeasurable coupling system $s^1$.

The system signals are quantised as in the previous example. The cardinalities of
12.3. Simulation of a Three-Tank System

Figure 12.11.: Coordinated diagnosis of a three-tank system

Figure 12.12.: Simulation setup

the domains of the network are then given by

\[ \text{card}(\mathcal{N}_v) = 10, \quad \text{card}(\mathcal{N}_w) = 36, \quad \text{card}(\mathcal{N}_z) = 216. \]

The behavioural relation \( L_1 \) of the automaton \( A_1 \) contains 667 transitions, \( L_2 \) contains 391 transitions, and \( L_3 \) contains 65 transitions. The behavioural relation of the equivalent automaton of this simple system contains 13,000 transitions which is far larger than the sum of the transitions stored in the network description.

For the experiment the plant has been set up with the pump under full power the whole time and starting liquid levels given as \( h_1(0) = 0.5 \text{m}, h_2(0) = 0.1 \text{m}, h_3(0) = 0.5 \text{m} \), which corresponds to the discrete states

\[ z_1(0) = 5, \quad z_2(0) = 1, \quad z_3(0) = 5. \]

The simulation result is depicted in Figure 12.13. The discrete probability distributions over the automaton states are depicted in grey scales as introduced in Section 12.1 and are superimposed by the measured continuous liquid levels of the three-
12. Application Example

tank system. It is apparent that the measurement fits well into the simulation result. The simulation has also been performed using the equivalent automaton of the network with identical results which are therefore not depicted separately. The simulation approach using on-line-composition takes about 290 seconds with a 1.5GHz P3 PC. The simulation using the equivalent automaton takes about 190 seconds not including the time needed for the composition of the network (16s).

Figure 12.13.: Simulation result
Summary

This book has presented two approaches (characteristic functions and behavioural relations) for modelling synchronous nondeterministic and stochastic automata networks. Both representations have been put to use in different approaches to diagnosis. The novelty in this is the solving of the diagnostic tasks for completely different information structures with the aim of reducing the computational complexity.

For nondeterministic automata networks the diagnostic task has successfully been decentralised leading to a significant reduction in computational costs. Moreover, is has been proven that the diagnostic result of the developed method is complete, i.e. it is guaranteed that no fault is overseen. The drawback of the decentralised approach is that this result is not sound, but might contain more fault cases in comparison to the diagnostic methods using a centralised information structure.

By introducing a superordinate coordinator structural knowledge about the system is used to refine the result of the decentralised diagnosis. It has been proven that this novel approach returns the identical result as a centralised approach, in other words the best possible result. The coordination is based on the natural join operation of relational algebra. The distinct computational advantage has been retained.

However, it has also been proven that, in general, decentralised diagnosis of stochastic automata networks is not possible. Several criteria for testing if two automata are stochastically independent and therefore separable have been given. Although an algorithm has been developed which allows to diagnose a network without the need of calculating the overall equivalent automata beforehand, the computational advantage of the approaches developed for the nondeterministic case has not been reached. The same limitations hold for the simulation of the behaviour of stochastic automata networks, i.e. the simulation cannot be performed on a component bases.

Based on the concept developed for the diagnosis of stochastic automata networks a
13. Summary

A simulation method has been developed which retains the full memory advantage of the network representation.

For the compositional modelling a method has been presented which allows to abstract each component of a plant separately as an automaton while assuring the completeness of the resulting network. Furthermore, criteria for testing a network with direct feedthrough for being well-formed has been given.

With these results this book provides a framework for the application of synchronised automata networks in the diagnosis of technological systems. However, in some areas this work is only the starting point for further investigations. An important task is to bring the presented methods into a form which allows their application in the practical field. Furthermore, several problems remain open. The analysis of the optimal decentralisation of an existing system is an important task. Under the same aspect it is advisable to consider the fault diagnostic task right from the beginning in the planning phase of a technological system, instead of adding the diagnosis afterwards. The coordination approach using the natural join operator suggests itself for reconstructing information which has been lost on a communication channel.

It is believed that the presented results have shown that the compositional treatment of systems brings on many advantages and is interesting for future research.
[1] **General Literature**


[3] **DIN 25448 - Ausfalleffektanalyse / Failure mode and effects analyses FMEA.** Mai 1990


[61] Larsson, M.: Behavioural and Structural Model Based Approaches to Discrete Diagnosis. Linköping, Sweden, Linköpings Universitet, Diss., 1999

185


[95] SU, R.: Distributed Diagnosis for Discrete-Event Systems. Toronto, University of Toronto, Diss., 2004


[II] Contributions by the Author


[III] Supervised Diploma and Research Studies


[120] DRÜPPEL, S.: Diagnose eines Luftsystems mit qualitativen Modellen, Ruhr-Universität Bochum, Diplomarbeit, 2005


[125] PACHE, Jens: *Analyse von stochastischen Automatennetzen*, Ruhr-Universität Bochum, Diplomarbeit, 2004

[126] RENSCH, K.: *Qualitative Diagnose eines Druckluftgenerators*, Ruhr-Universität Bochum, Studienarbeit, 2004

Table A.1.: Notation

<table>
<thead>
<tr>
<th>Type</th>
<th>Convention</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>scalars, scalar functions</td>
<td>lower-case, italic</td>
<td>$v, g$</td>
</tr>
<tr>
<td>vectors, vector functions</td>
<td>lower-case, italic, bold</td>
<td>$x, g$</td>
</tr>
<tr>
<td>relations, behavioural function</td>
<td>upper-case, italic, bold</td>
<td>$L$</td>
</tr>
<tr>
<td>sequences of scalars</td>
<td>upper-case, italic</td>
<td>$V, W$</td>
</tr>
<tr>
<td>sequences of sets</td>
<td>upper-case, italic, bold</td>
<td>$V, W$</td>
</tr>
<tr>
<td>sets of scalars</td>
<td>upper-case, calligraphic</td>
<td>$\mathcal{N}_z, \mathcal{F}$</td>
</tr>
<tr>
<td>sets of sets</td>
<td>upper-case, calligraphic, bold</td>
<td>$\mathcal{N}_z, \mathcal{F}$</td>
</tr>
</tbody>
</table>

Programming code, e.g. MySQL commands, is given using a monospace font.

Table A.2.: Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>Diagnostic algorithm</td>
<td>14</td>
</tr>
<tr>
<td>$A$</td>
<td>Automaton</td>
<td>26</td>
</tr>
<tr>
<td>$\mathcal{A}_z$</td>
<td>Automata network</td>
<td>38</td>
</tr>
<tr>
<td>$B$</td>
<td>Measurement sequence</td>
<td>12</td>
</tr>
<tr>
<td>$C$</td>
<td>Constraint for the selection operation $\sigma_C$</td>
<td>22</td>
</tr>
<tr>
<td>$B$</td>
<td>Behaviour</td>
<td>11</td>
</tr>
<tr>
<td>$D$</td>
<td>Diagnoser</td>
<td>14</td>
</tr>
<tr>
<td>$f$</td>
<td>Fault signal</td>
<td>26</td>
</tr>
<tr>
<td>$\mathcal{F}$</td>
<td>Fault transition function</td>
<td>31</td>
</tr>
<tr>
<td>$\mathcal{F}$</td>
<td>Set of faults (result of diagnostic task)</td>
<td>12</td>
</tr>
<tr>
<td>$g$</td>
<td>State function</td>
<td>144</td>
</tr>
</tbody>
</table>
### A. Notation, Symbols, Operators

#### Table A.2.: Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G )</td>
<td>State transition function</td>
<td>30</td>
</tr>
<tr>
<td>( h )</td>
<td>Output function</td>
<td>144</td>
</tr>
<tr>
<td>( H )</td>
<td>Output function</td>
<td>30</td>
</tr>
<tr>
<td>( I )</td>
<td>Initial condition</td>
<td>14</td>
</tr>
<tr>
<td>( k )</td>
<td>Discrete time counter</td>
<td>25</td>
</tr>
<tr>
<td>( K(\bullet) )</td>
<td>Cartesian product of the domains of ( \bullet )</td>
<td>20</td>
</tr>
<tr>
<td>( L )</td>
<td>Characteristic function, behavioural relation</td>
<td>26</td>
</tr>
<tr>
<td>( M )</td>
<td>Number of symbols of an input signal</td>
<td>26</td>
</tr>
<tr>
<td>( N )</td>
<td>Size of an automata’s state domain</td>
<td>26</td>
</tr>
<tr>
<td>( N )</td>
<td>Set of symbols (domain of a signal)</td>
<td>19</td>
</tr>
<tr>
<td>( \mathcal{N} )</td>
<td>Set containing the domains of several signals</td>
<td>36</td>
</tr>
<tr>
<td>( Q )</td>
<td>Number of symbols of an coupling signal</td>
<td>38</td>
</tr>
<tr>
<td>( Q )</td>
<td>Partitioning of a quantised system</td>
<td>144</td>
</tr>
<tr>
<td>( R )</td>
<td>Number of symbols of an output signal</td>
<td>26</td>
</tr>
<tr>
<td>( \mathcal{R} )</td>
<td>Diagnostic result</td>
<td>14</td>
</tr>
<tr>
<td>( S )</td>
<td>Number of fault symbols of a fault signal</td>
<td>26</td>
</tr>
<tr>
<td>( t )</td>
<td>Time</td>
<td>23</td>
</tr>
<tr>
<td>( u )</td>
<td>Input signal (continuous-valued)</td>
<td>144</td>
</tr>
<tr>
<td>( v )</td>
<td>Input signal (discrete-valued)</td>
<td>26</td>
</tr>
<tr>
<td>( \nu )</td>
<td>Set containing an automata’s input signals</td>
<td>36</td>
</tr>
<tr>
<td>( w )</td>
<td>Output signal (discrete-valued)</td>
<td>26</td>
</tr>
<tr>
<td>( \omega )</td>
<td>Set containing an automata’s output signals</td>
<td>36</td>
</tr>
<tr>
<td>( x )</td>
<td>State signal (continuous-valued)</td>
<td>144</td>
</tr>
<tr>
<td>( y )</td>
<td>Output signal (continuous-valued)</td>
<td>144</td>
</tr>
<tr>
<td>( z )</td>
<td>State signal (discrete-valued)</td>
<td>26</td>
</tr>
<tr>
<td>( \mathcal{z} )</td>
<td>Network state consisting out of several automata states</td>
<td>144</td>
</tr>
<tr>
<td>( \mathcal{Z} )</td>
<td>Set of states (result of observation task)</td>
<td>68</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>Number of automata in a network</td>
<td>38</td>
</tr>
<tr>
<td>( \kappa )</td>
<td>Number of network coupling signals</td>
<td>38</td>
</tr>
<tr>
<td>( \mu )</td>
<td>Number of network input signals</td>
<td>38</td>
</tr>
<tr>
<td>( \rho )</td>
<td>Number of network output signals</td>
<td>38</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>Number of network fault signals</td>
<td>38</td>
</tr>
</tbody>
</table>
### Table A.3.: Indices and accents

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d, n, s$</td>
<td>superscript</td>
<td>$d =$ deterministic, $n =$ nondeterministic, $s =$ stochastic used for automata and relations</td>
</tr>
<tr>
<td>$v, w, s, f, z$</td>
<td>index</td>
<td>$v =$ input, $w =$ output, $s =$ coupling, $f =$ fault, $z =$ state used for domains $\mathcal{N}$ and sets of domains $\mathcal{N}$</td>
</tr>
<tr>
<td>$p$</td>
<td>index</td>
<td>marks a stochastic variable</td>
</tr>
<tr>
<td>$\text{centr}, \text{dec}$</td>
<td>index</td>
<td>result obtained by a centralised or decentralised approach</td>
</tr>
<tr>
<td>integer $i$</td>
<td>superscript</td>
<td>marks the unique name of a signal or domain of a network</td>
</tr>
<tr>
<td>$\hat{\text{—}}$ (hat)</td>
<td>accent</td>
<td>derived by composition used for automata, relations and signals</td>
</tr>
<tr>
<td>$\bar{\text{—}}$ (bar)</td>
<td>accent</td>
<td>measurable signals used for sets of signals and sequences</td>
</tr>
</tbody>
</table>

### Table A.4.: Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{sch}()$</td>
<td>schema</td>
</tr>
<tr>
<td>$\text{dom}()$</td>
<td>domain</td>
</tr>
<tr>
<td>$\pi$</td>
<td>projection</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>selection</td>
</tr>
<tr>
<td>$\bowtie$</td>
<td>natural join</td>
</tr>
<tr>
<td>$\lor$</td>
<td>logical OR (disjunction)</td>
</tr>
<tr>
<td>$\bigvee_{s=1}^{n}$</td>
<td>evaluate expression for $s = 1 \ldots n$ and connect all with OR-operation</td>
</tr>
<tr>
<td>$\land$</td>
<td>logical AND (conjunction)</td>
</tr>
<tr>
<td>$\bigwedge_{s=1}^{n}$</td>
<td>evaluate expression for $s = 1 \ldots n$ and connect all with AND-operation</td>
</tr>
</tbody>
</table>
Table A.5.: Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L(k)$</td>
<td>$L(z(k+1), w(k), z(k), v(k), f)$</td>
</tr>
<tr>
<td>$\Sigma_z$</td>
<td>$\sum_{i=1}^{\kappa} \sum_{z=1}^{N_i}$ $(v(0), v(1), \ldots, v(k))$</td>
</tr>
<tr>
<td>$V(0\ldots k)$</td>
<td>deterministic automaton</td>
</tr>
<tr>
<td>DA</td>
<td>deterministic automata network</td>
</tr>
<tr>
<td>DAN</td>
<td>discrete event system</td>
</tr>
<tr>
<td>NA</td>
<td>nondeterministic automaton</td>
</tr>
<tr>
<td>NAN</td>
<td>nondeterministic automata network</td>
</tr>
<tr>
<td>QS</td>
<td>quantised system</td>
</tr>
<tr>
<td>SA</td>
<td>stochastic automaton</td>
</tr>
<tr>
<td>SAN</td>
<td>stochastic automata network</td>
</tr>
<tr>
<td>ZOH</td>
<td>zero order hold</td>
</tr>
</tbody>
</table>

**Possibility:** The possibility function $\text{Poss}(e_1, \ldots, e_n)$ assigns the simultaneous occurrence of the events $e_i$ a possibility represented by the numbers 0 and 1:

$$\text{Poss}(e_1, \ldots, e_n) \in \{0, 1\}.$$ 

Formally, $\text{Poss}(e_{p1} = e_1, \ldots, e_{pn} = e_n) = 1$ holds iff it is possible that all variables $e_{pi}$ assume the respective values $e_i$ simultaneously, else $\text{Poss}(\bullet) = 0$. The result (“true” or “1”) and (“false” or “0”) is also called the logical value. The values are usually associated with events. The association of the values with the respective variables is then called the occurrence of events. For notational convenience the variables $e_{pi}$ will be omitted in the remainder of the book. Hence, instead of $\text{Poss}(e_{p1} = e_1)$ the term $\text{Poss}(e_1)$ is written.

**Probability:** The probability function $P(e_1, \ldots, e_n)$ assigns simultaneous occurrence of the events $e_i$ a probability represented by the numbers between 0 and 1:

$$P(e_1, \ldots, e_n) \in [0, 1].$$ 

Formally, $P(e_{p1} = e_1, \ldots, e_{pn} = e_n)$ returns the joined probability of the random variables $e_{pi}$ assuming the respective values $e_i$ simultaneously. Analogously to the possibility function the values are usually associated with events. The association of the values with the respective variables is then called the occurrence of events. The function $P(e_{p1} = e_1, \ldots, e_{pn} = e_n | b_{p1} = b_1, \ldots, b_{pn} = b_m)$ is called conditional probability. It returns the joined probability of the random variables $e_{pi}$ assuming the respective values $e_i$ simultaneously under the condition that $b_{p1} = b_1, \ldots, b_{pn} = b_m$ hold. The random variables will usually be omitted in the remainder of the book for notational convenience.
B.1. Proof of the Composition Rules

It is common knowledge that the parallel composition operation is identical to the synchronous product for discrete-event systems and the serial composition operation is identical to the biased synchronised product \([47, 57, 58, 106]\). The correctness of the composition operations has been proven in \([93]\) for the stochastic automata network. Therefore, in this work the proofs will only be given for the nondeterministic automata network. The proofs for the deterministic case can be constructed analogously.

B.1.1. Parallel Composition

The correctness of the parallel composition operation for the nondeterministic case is proven by applying the associative and commutative law of boolean algebra \([18]\):

\[
Poss(z'_1, w^1, z_1, v^1, f^1) \land Poss(z'_2, w^2, z_2, v^3, f^2) = Poss(z'_1, w^1, z_1, v^1, f^1, z'_2, w^2, z_2, v^3, f^2) = Poss(z'_1, z'_2, w^1, w^2, z_1, z_2, v^1, v^2, v^3, f^1, f^2).
\]

B.1.2. Serial Composition

The correctness of the serial composition for a nondeterministic automata network is proven analogously to the parallel composition:

\[
\bigvee_{s'} Poss(z'_1, w^1, s', z_1, v^1, f^1) \land Poss(z'_2, w^2, z_2, v^3, s'^1, f^2) = \bigvee_{s'} Poss(z'_1, z'_2, w^1, w^2, z_1, z_2, v^1, v^2, s'^1, f^1, f^2) = Poss(z'_1, z'_2, w^1, w^2, z_1, z_2, v^1, v^2, f^1, f^2).
\]
B. Proofs

B.1.3. Feedback Composition

For a better understanding the proof of the feedback composition for the NAN is given using the extended notation with random variables. These variables are marked with the index \( p \). Furthermore, the coupling signal \( s^1 \) is divided into two parts, namely the output of the automaton \( s^{1*} \) and the input \( s^1 \). The identity of both parts is used in the second line.

\[
\begin{align*}
\forall s^1 : \text{Poss}(z'_{p1} = z'_1, w^1_p = w^1, s^{1*} = s^1, z_{p1} = z_1, v^1_p = v^1, s_p^{1*} = s^1, f^1_p = f^1) \\
= \forall s^1 : \text{Poss}(z'_{p1} = z'_1, w^1_p = w^1, s^{1*} = s^1, z_{p1} = z_1, v^1_p = v^1, f^1_p = f^1) \quad \text{because } s^{1*} = s_p^{1*} \\
= \text{Poss}(z'_{p1} = z'_1, w^1 = w^1, z_{p1} = z_1, v^1_p = v^1, f^1_p = f^1) \\
= \text{Poss}(z'_1, w^1, z_1, v^1, f^1)
\end{align*}
\]

B.2. Proof of Completeness and Soundness

B.2.1. Proof of Theorem 7.1

For completeness and soundness according to Equation (2.3) it has to be proven that

\[ P(f|V, W) = 0 \iff P(W|V, f) = 0 \]

holds for all \( f \in \mathcal{N}_f \). From

\[
\begin{align*}
P(W(0...k)|V(0...k), f) \cdot P(f) &> 0 \\
\Leftrightarrow \sum_{z(0)} P(W(0...k)|V(0...k), z(0), f) \cdot P(z(0)) \cdot P(f) &> 0 \quad \text{(cf. Eqn. (2.30) in [93])}
\end{align*}
\]

\[
\begin{align*}
\Leftrightarrow \sum_{Z(0, ..., k+1)} L^i(0) L^i(1) \cdots L^i(k) \cdot P(z(0)) \cdot P(f) &> 0 \\
\Leftrightarrow \sum_{Z(0, ..., k+1)} \sum_{f} L^i(0) L^i(1) \cdots L^i(k) \cdot P(z(0)) \cdot P(f) &> 0 \\
\Leftrightarrow P(f|V, W) &> 0
\end{align*}
\]

it is clear that for non vanishing initial conditions \( P(z(0)) \) and \( P(f) \) the fault \( f \) will be excluded from the set of fault candidates iff the measurements are inconsistent with the model. The denominator vanishes only if the measurement is inconsistent for all fault cases which has been excluded per definition.

B.2.2. Proof of Theorem 4.1

The nondeterministic automaton can be seen as a simplification of a stochastic automaton where

\[ P > 0 \iff \text{Poss} = 1 \quad \text{and} \quad P = 0 \iff \text{Poss} = 0 \]
B.2. Proof of Completeness and Soundness

holds. Therefore, the proof of completeness and soundness can be applied analogously to the proof in Appendix B.2.1.

B.2.3. Proof of Theorem 4.8

The observation result is given as

\[ Z_{cent}(k) = \{ z(k) | Poss(z(k), \mathbf{V}(0\ldots k), \mathbf{W}(0\ldots k)) = 1 \}. \]

Transforming the right hand side into a non-recursive form results in

\[
\begin{aligned}
Poss(z(k), \mathbf{V}(0\ldots k), \mathbf{W}(0\ldots k)) &= \bigvee_f \bigvee_{z(k+1)} \mathbf{L}^n(k) \land Poss(z(k), f, \mathbf{V}(0\ldots k-1), \mathbf{W}(0\ldots k-1)) \\
&= \bigvee_f \bigvee_{z(k+1)} \mathbf{L}^n(k) \land Poss(z(k-1), f, \mathbf{V}(0\ldots k-2), \mathbf{W}(0\ldots k-2)) \\
&= \bigvee_f \bigvee_{z(k+1)} \bigvee_{z(k-2)} \cdots \bigvee \mathbf{L}^n(k) \land \cdots \land \mathbf{L}^n(0) \land Poss(z(0), f).
\end{aligned}
\]

This proves that the result includes exactly those states \( z(k) \) for which the measured sequence is included in the system behaviour, i.e. \( Poss(\mathbf{V}(0\ldots k), \mathbf{W}(0\ldots k)) = 1 \) holds, and \( z(0), f \) are included in the initial condition: \( Poss(z(0), f) = 1 \).

B.2.4. Proof of Theorem 5.1

Analogously to Proof in Appendix B.2.1 the relations

\[
\begin{aligned}
Poss(\mathbf{W}_i(0\ldots k), \mathbf{V}_i(0\ldots k), f) \land Poss(f_i) &= 1 \\
\iff \bigvee_{z(0)} Poss(\mathbf{W}_i(0\ldots k), \mathbf{V}_i(0\ldots k), z_i(0), f_i) \land Poss(z_i(0)) \land Poss(f_i) = 1 \\
\iff \bigvee_{Z(0\ldots k+1)} \bigvee_{S(0\ldots k)} \mathbf{L}_i^n(0) \mathbf{L}_i^n(z_i(0)) \land \mathbf{L}_i^n(k) \land Poss(z_i(0)) \land Poss(f_i) = 1 \\
&\quad \overset{(5.4)}{\iff} Poss(f_i, \mathbf{V}_i(0\ldots k), \mathbf{W}_i(0\ldots k)) = 1
\end{aligned}
\]

proof the completeness and soundness. \( S_i(0\ldots k) \) denotes all sequences of the coupling signals up to time step \( k \). A fault \( f \) is included in the set of fault candidates whenever the measurable signals are consistent with the NA.

B.2.5. Proof of Theorem 5.2

The proof will only be given for the projection step of the diagnostic algorithm. The proof for the prediction step can be done analogously. Because of the automata’s Markov property the proof can be restricted to the time step \( k = 0 \).
B. Proofs

\[ \text{Poss}(f, \tilde{v}(0), \tilde{w}(0)) \]
\[ = \bigvee_{z(0)z(1)s(0)} \bigwedge_{i=1}^{\gamma} L^i_t(0) \land \text{Poss}(z_i(0), f_i) \quad \text{(application of composition rule)} \]
\[ = \bigvee_{s(0)} \left( \bigvee_{z(0)z(1)} L^0_t(0) \land \text{Poss}(z_0(0), f_0) \right) \land \left( \bigvee_{z(0)z(1)} L^2_t(0) \land \text{Poss}(z_2(0), f_2) \right) \]
\[ \land \cdots \land \left( \bigvee_{z(0)z(1)} L^\gamma_t(0) \land \text{Poss}(z_\gamma(0), f_\gamma) \right) \]
\[ < \left( \bigvee_{z(0)z(1)x(0)} L^0_t(0) \land \text{Poss}(z_0(0), f_0) \right) \land \left( \bigvee_{z(0)z(1)x(0)} L^2_t(0) \land \text{Poss}(z_2(0), f_2) \right) \]
\[ \land \cdots \land \left( \bigvee_{z(0)z(1)x(0)} L^\gamma_t(0) \land \text{Poss}(z_\gamma(0), f_\gamma) \right) \]
\[ = \text{Poss}(f_1, \tilde{v}_1(0), \tilde{w}_1(0)) \land \text{Poss}(f_2, \tilde{v}_2(0), \tilde{w}_2(0)) \land \cdots \land \text{Poss}(f_\gamma, \tilde{v}_\gamma(0), \tilde{w}_\gamma(0)) \]

The “greater than”-sign in the above relation proves the theorem.

B.2.6. Proof of Theorem 10.1

Without loss of generality the proof is limited to the serial connection of quantised systems as depicted in Figure 10.5(b). The proof can be generalised easily, because any network can be composed to a single system using the appropriate composition rules as proven in Section 3.3.4.

The composition \( \tilde{L} = L_1 \Rightarrow L_2 \) with \( L_1 \) and \( L_2 \) given as above results in

\[ \tilde{L} = \{(z_1', z_2', w, z_1, z_2, v) \mid \exists (x_1, u) \text{ with } x_1 \in Q_{x_1}(z_1), u \in Q_u(v), \]
\[ g_1(x_1, u) \in Q_{x_1}(z_1'), h_1(x_1, u) \in Q_r(s), \]
\[ \text{and } \exists (x_2, \tilde{r}) \text{ with } x_2 \in Q_{x_2}(z_2), \tilde{r} \in Q_r(s), \]
\[ g_2(x_2, \tilde{r}) \in Q_{x_2}(z_2'), h_2(x_2, \tilde{r}) \in Q_y(w) \} \]

which can be simplified to

\[ \tilde{L} = \{(z_1', z_2', w, z_1, z_2, v) \mid \exists (x_1, x_2, u) \text{ with } x_1 \in Q_{x_1}(z_1), x_2 \in Q_{x_2}(z_2), u \in Q_u(v), \]
\[ g_1(x_1, u) \in Q_{x_1}(z_1'), g_2(x_2, \tilde{r}) = g_2(x_2, >[h_1(x_1, u)] <) \in Q_{x_2}(z_2'), \]
\[ h_2(x_2, \tilde{r}) = h_2(x_2, >[h_1(x_1, u)] <) \in Q_y(w) \}. \]
The relation $L_{\text{mono}}$ given in Equation (10.17) can be transformed into

$$L_{\text{mono}} = \{(\tilde{z}', w, \tilde{v}, v) \mid \exists (x_1, x_2, u) \in Q_{x_1, x_2}(\tilde{z}), u \in Q_u(v),$$
$$g_1(x_1, u) \in Q_{x_1}(\tilde{z}'), g_2(x_2, r) = g_2(x_2, h_1(x_1, u)) \in Q_{x_2}(\tilde{z}')$$
$$h_2(x_2, r) = h_2(x_2, h_1(x_1, u)) \in Q_y(w)\}.$$

This proves that the relations differ only in the usage of the signals $r$ or $\tilde{r}$, respectively. As the mapping of any cell in $Q_{x_1, u}$ unto $r$ is a subset of the same area after the quantisation and injection $> [r] \leq \tilde{r}$. Therefore, the mapping of $g_2(x_2, r)$ is a subset of the mapping of $g_2(x_2, \tilde{r})$. This may lead to additional tuples in $\hat{L}$ in comparison to $L_{\text{mono}}$, i.e. $L_{\text{mono}} \subseteq \hat{L}$ holds.

### B.3. Proof of Diagnosability

#### B.3.1. Proof of Corollary 4.5 for Nondeterministic Processes With Unmeasurable Signals

It has to be proven that the following relation holds

$$\text{Poss}(f, V(0 \ldots k), W(0 \ldots k)) = \text{Poss}(f, V(0 \ldots k)).$$

With Equations (4.3), (4.12) and $\text{Poss}(w, v) = 1$, because of the consistency, the relation

$$\text{Poss}(f, V, W) = \bigvee_{s(k) \in \text{Poss}(z(k), v(k), f)} \bigvee_{s(k+1) \in \text{Poss}(z(k+1), v(k), f)} \bigvee_{s(k-1) \in \text{Poss}(z(k-1), v(k), f)} \bigvee_{s(0) \in \text{Poss}(z(0), v(0), f)} \bigvee_{s(0) \in \text{Poss}(z(0), v(0), f)} \bigvee_{s(0) \in \text{Poss}(z(0), v(0), f)}$$

can be simplified to

$$\text{Poss}(f, V, W) = \bigvee_{s(k) \in \text{Poss}(z(k), v(k), f)} \bigvee_{s(k+1) \in \text{Poss}(z(k), v(k), f)} \bigvee_{s(k-1) \in \text{Poss}(z(k), v(k), f)} \bigvee_{s(0) \in \text{Poss}(z(0), v(0), f)} \bigvee_{s(0) \in \text{Poss}(z(0), v(0), f)} \bigvee_{s(0) \in \text{Poss}(z(0), v(0), f)}$$

which proves the above relation.

#### B.3.2. Proof of the Diagnosability of Stochastic Automata Networks

Applying the composition rule (3.40) to a stochastic process represented by an automata network $A^S$, results in

$$L'(z', w|z, v, f) = P(z', w|z, v, f) = \sum_{s} \prod_{i=1}^{y} P(z'_i, w_i|z_i, v_i, f_i).$$
B. Proofs

By introducing bijective mappings
\[ M_{z} = N_{z_{1}} \times \cdots \times N_{z_{\gamma}} \rightarrow N_{\tilde{z}}, \]
\[ M_{v} = N_{v_{1}} \times \cdots \times N_{v_{\mu}} \rightarrow N_{\tilde{v}}, \]
and so on the nondeterministic process can be transferred to the scalar case
\[ P(\tilde{z}', \tilde{w}|\tilde{z}, \tilde{v}, \tilde{f}). \]

According to Corollary 7.2 this process is not partially diagnosable if
\[ P(\tilde{z}', \tilde{w}|\tilde{z}, \tilde{v}, \tilde{f}) = P(\tilde{z}', \tilde{w}|\tilde{z}, \tilde{v}) \]
holds for all \( \tilde{z}', \tilde{z} \in N_{\tilde{z}}, \tilde{v} \in N_{\tilde{v}}, \tilde{w} \in N_{\tilde{w}} \) and a subset of faults \( \tilde{f} \in \tilde{N}_{\tilde{f}} \). This proves that a stochastic automata network is not partially diagnosable if the equivalent automaton is not partially diagnosable. The diagnosability of the SAN follows directly from that.

B.3.3. Proof of Theorem 7.3

With the relation
\[ P(z', \tilde{w}|z, \tilde{v}, f) = P(z', \tilde{w}|z, \tilde{v}) \iff \sum_{s} P(z', w|z, v, f) = \sum_{s} Poss(z', w|z, v) \]
and the application of the composition rules the following derivation puts the network into relation with the equivalent automaton:
\[ L^{s}(z', \tilde{w}|z, \tilde{v}, f) = P(z', \tilde{w}|z, \tilde{v}, f) \]
\[ = \sum_{s} \prod_{i=1}^{\gamma} P(z'_{i}, w_{i}|z_{i}, v_{i}, f_{i}) \]
\[ = \sum_{s} \prod_{i=1}^{\gamma} P(z'_{i}, w_{i}|z_{i}, v_{i}) \]
\[ = P(z', \tilde{w}|z, \tilde{v}). \]

This proves that the overall stochastic process is not diagnosable if all automata of the network are not diagnosable.

B.3.4. Proof of Theorem 7.4

The theorem is proven by contradiction (reductio ad absurdum). Assume that a network is never partially diagnosable concerning a fault \( f_{i} \) when the associated automaton \( A_{s}^{f} \) is not diagnosable. Consider now the example depicted in Fig. B.1 consisting out of two automata in a serial connection with their characteristic functions given in Table B.1. Both automata are static systems, i.e. their state does not change and only the left automaton \( A_{s}^{f} \) is influenced by a fault \( f^{1} \). Clearly, this automaton is not diagnosable, because no output signal is measurable. However, the fault can be identified directly using the right automaton’s output signal, because \( w^{1} = f^{1} \) holds. This contradicts the proposition, which proves the theorem.
B.3. Proof of Diagnosability

Figure B.1.: Network with not diagnosable subautomata

Table B.1.: Automaton tables of the automata depicted in Figure B.1

<table>
<thead>
<tr>
<th>$z^\prime_1$</th>
<th>$s^1_1$</th>
<th>$z_1$</th>
<th>$v^1_1$</th>
<th>$f^1_1$</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$z^\prime_2$</th>
<th>$w^1_1$</th>
<th>$z_2$</th>
<th>$s^1_2$</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1.0</td>
</tr>
</tbody>
</table>

B.3.5. Proof of Corollary 7.6

With Equations (7.6), (7.11) and $\text{Poss}(w|v) = 1$, because of the required consistency, the relation

$$P(f|V,W) = \sum_{z(k+1)} L^s(z(k+1)) \cdot \sum_{z(k-1)} L^s(z(k-1)) \cdot \cdots \sum_{z(0)} L^s(z(0)) \cdot \text{Poss}(z(0), f)$$

holds. The denominator in (7.6) is equal to one, because of the consistency. The above relation can be further simplified to

$$P(f|V,W) = \sum_{z(k+1)} \sum_{z(k)} P(z(k+1)|z(k), v(k), f) \cdot \cdots \sum_{z(0)} P(z(1)|z(0), v(0), f) \cdot P(z(0), f)$$

$$= \sum_{z(k+1)} \sum_{z(k)} P(z(k+1)|z(k), v(k), f) \cdot \cdots \sum_{z(1)} P(z(2)|z(1), v(1), f) \cdot P(z(1), f|v(0))$$

$$= \sum_{z(k+1)} \sum_{z(k)} P(z(k+1)|z(k), v(k), f) \cdot \cdots \sum_{z(2)} P(z(3)|z(2), v(2), f) \cdot P(z(2), f|v(1))$$

$$= P(f|v(k)) = P(f),$$

because the fault is constant and independent of the input signal.

B.3.6. Proof of Lemma 7.5

According to Definition 3.1 the relation

$$P(z', w|z, v, f) = P(z'|z, v, f) \cdot P(w|z, v, f).$$
holds for a Mealy-Automaton. Applying \( \tilde{H}(w|v) = H(w|z, v, f) \) returns

\[
P(z', w|z, v, f) = P(z'|z, v, f) \cdot P(w|v)
\]

which is equal to Equation (7.11) in Definition 2.5.

B.4. Other Proofs

B.4.1. Proof of Lemma 8.2

The proof is performed using the network depicted in Fig. 8.1 without loss of generality. The proof is held in two steps. At first an independence of \( s^1 \) from the remaining signals of \( A_1^x \) is investigated, afterwards the independence of the signals of \( A_2^x \).

**Independence of \( A_1^x \):** If \( s^1 \) is independent of the remaining signals of \( A_1^x \) then

\[
L^x_i(z'_1, s^1, w^1|z_1, v^1, f^1) = P(z'_1, w^1|z_1, v^1, f^1) \cdot P(s^1|z_1, v^1, f^1) = P(z'_1, w^1|z_1, v^1, f^1) \cdot P(s^1)
\]

holds. Therefore

\[
P(f^1|v(0), w(0)) = \frac{\sum_{s^1(0)} \left[ \sum_{z_1(1), z_2(0)} \sum_{z_1(1), z_2(0)} \sum_{f^1} \sum_{f^2} L^x_1(0)P(z_1(0), f^1) \cdot \sum_{z_2(1), z_2(0)} \sum_{f^2} L^x_2(0)P(z_2(0), f^2) \right]}{\sum_{s^1(0)} \left[ \sum_{z_1(1), z_1(0)} \sum_{z_1(1), z_2(0)} \sum_{f^1} \sum_{f^2} L^x_1(0)P(z_1(0), f^1) \cdot \sum_{z_2(1), z_2(0)} \sum_{f^2} L^x_2(0)P(z_2(0), f^2) \right]}
\]

\[
= \frac{\sum_{z'_1, z_1} \sum_{z'_1, z_1} \sum_{z_1, v^1} \sum_{z_2, f^2} L^x_1(0)P(z_1(0), f^1) \cdot \sum_{z_2, f^2} L^x_2(0)P(z_2(0), f^2)}{\sum_{z'_1, z_1} \sum_{z'_1, z_1} \sum_{z_1, v^1} \sum_{z_2, f^2} L^x_1(0)P(z_1(0), f^1) \cdot \sum_{z_2, f^2} L^x_2(0)P(z_2(0), f^2)}
\]

\[
= \frac{P(f^1|v(1), w^1(0)) \cdot P(f^2|v^2(0), w^2(0))}{P(f^1|v(0), w(0))}
\]

**Independence of \( A_2^x \):** If \( s^1 \) is independent of the remaining signals of \( A_2^x \) then

\[
L^x_2(z'_2, s^1, w^2|z_2, v^2, f^1) = P(z'_2, w^2|z_2, v^2, f^1)
\]
holds for all \( s^1 \in \mathcal{N}_{s^1} \). Therefore

\[
P(f|v(0), w(0)) = \sum_{s^1(0)} \left( \sum_{z_1(0), z_1(0)} L_1^0(z_1(0), f^1) \sum_{z_2(0), z_2(0)} L_2^0(z_2(0), f^2) \right)
\]

\[
= \sum_{s^1(0)} \left( \sum_{z_1(0), z_1(0)} \sum_{f^1} L_1^0(z_1(0), f^1) \sum_{z_2(0), z_2(0)} \sum_{f^2} L_2^0(z_2(0), f^2) \right)
\]

\[
= \sum_{s^1(0)} \sum_{z_1, z_1} \sum_{f^1} L_1^0(z_1, f^1) \sum_{z_2, z_2} \sum_{f^2} L_2^0(z_2, f^2) = P(f^1|v^1(0), w^1(0)) \cdot P(f^2|v^2(0), w^2(0))
\]

holds. This proves the lemma.

### B.4.2. Proof of Lemma 10.1

The difference between the behaviours \( B_{\text{mod}} \) and \( B_{\text{mono}} \) is caused by the additional quantisation of internal signals \( r \). Let \( \mathcal{N}_{r,i,j} = h([x] = j, [u] = i) \) be the mapping of any cell of the quantised input/state-space \( Q_{ux} \). The quantisation of every point \( r \in \mathcal{N}_{r,i,j} \) results in the set \( \mathcal{N}_{[r],i,j} \subseteq Q_{rx} \) with \( \mathcal{N}_{[r],i,j} \supseteq \mathcal{N}_{r,i,j} \). Thus, the quantisation enlarges the set, but all values of the original set are still included. Clearly, any further mapping of the set \( \mathcal{N}_{[r],i,j} \) results in an superset of the original mapping of the smaller set \( \mathcal{N}_{r,i,j} \). The relation in the Lemma 10.1 follows directly from that.

### B.4.3. Proof of Theorem 11.4

This proof is done analogously to the proofs of the previous theorems in the chapter.

\[
\sum_{w^1=1}^{R^1} \cdots \sum_{w^\rho=1}^{R^\rho} \sum_{s^1=1}^{Q^1} \cdots \sum_{s^\gamma=1}^{Q^\gamma} \prod_{i=1}^{\gamma} H_i^s = 1 \iff \sum_{w_{\mathcal{N}_s}} \sum_{s_{\mathcal{N}_s}} \sum_{z_{\mathcal{N}_z}} \prod_{i=1}^{\gamma} L_i^s = 1
\]

\[
\iff \sum_{w_{\mathcal{N}_s}} \sum_{s_{\mathcal{N}_s}} \sum_{z_{\mathcal{N}_z}} L^s = 1
\]

The last relation satisfies Equation (3.18) which proves the theorem.
B. Proofs
C.1. Algorithm 4.1 in Pseudo-Code

The algorithm solving the centralised diagnostic problem for nondeterministic automata described using behavioural functions is given below.

Given: Automaton $A^n$, a-priori initial condition $Poss(z^{-1}, f)$
Loop: For $k = 0$ to $k_h$ (diagnostic horizon)
  
  Readin $v(k), w(k)$
  
  For all $f$ in $N_f$
    
    Calculate prediction $Poss(z', f, V, W)$ with (4.1) (store for next loop)
    
    Calculate projection $Poss(f|V, W)$ with (4.3)
  
  End For
  
  If $\bigvee Poss(z', f, V, W) = 0$ then restart with initial condition End If
  
  $F = (Poss(f = 1|V,W), Poss(f = 2|V,W), \ldots)$
  
  Output $F$

End For

Result: Set of faults $F(k_h)$

When using mathematical tools like MATLAB® it suggests itself to implement the algorithm using vector calculations.

C.2. Algorithm 4.1R in MySQL-Code

The code for solving the centralised diagnostic problem for nondeterministic automata using relational algebra is given below.

```sql
> /* Initialisation */
> CREATE TEMPORARY TABLE pre (zz int, f int) ENGINE=HEAP;
```
### C. Algorithms and Implementation

```sql
> CREATE TEMPORARY TABLE tmp (zz int, f int) ENGINE=HEAP;
> INSERT INTO pre SELECT * FROM initial;
> /* The loop */
> INSERT INTO tmp SELECT (zz,f) FROM L,pre WHERE v=... AND w=... AND L.zz=pre.zz AND L.f=pre.f GROUP BY zz,f;
> TRUNCATE pre;
> INSERT INTO pre SELECT * FROM tmp;
> TRUNCATE tmp;
> /* Output */
> SELECT f FROM pre;
```

The loop either has to be called repeatedly by an external programme or the flow control constructs supplied by the database have to be used. Modern databases as e.g. MySQL as of version 5.0 allow to store procedures on the server for fast execution.

### C.3. Algorithm 4.2 in Pseudo-Code

The code for solving the centralised diagnostic problem for nondeterministic automata networks described using behavioural functions is given below.

**Given:** Automaton network $A^n$, a-priori initial conditions $\text{Poss}(z_i(-1),f_i) \forall i$

**Loop:** For $k = 0$ to $k_h$ (diagnostic horizon)

- Readin $v(k), w(k)$
- Compute $\hat{L}$ with (3.39) for $v(k)$ and $w(k)$
- For all $f$ in $N_f$
  - Calculate prediction $\text{Poss}(z',f,V,W)$ with (4.1) (store for next loop)
  - Calculate projection $\text{Poss}(f|V,W)$ with (4.3)

**End For**

If $\bigvee z' \text{Poss}(z',f,V,W) = 0$ then restart with initial condition

**End If**

$\mathcal{F} = (\text{Poss}(f = 1|V,W), \text{Poss}(f = 2|V,W),\ldots)$

**Output $\mathcal{F}$**

**End For**

**Result:** Set of faults $\mathcal{F}(k_h)$

### C.4. Algorithm 4.2 in MySQL-Code

The code for solving the centralised diagnostic problem for nondeterministic automata networks using relational algebra is given below.

```sql
> /* Initialisation */
> CREATE TEMPORARY TABLE pre
> (zz1 int,zz2 int,...,f1 int,f2 int,...) ENGINE=HEAP;
```
The definition of the tables and the calling of the natural join operations have to be altered depending on the actual network structure. The loop either has to be called repeatedly by an external program or the flow control constructs supplied by the database. Modern databases as e.g. MySQL as of version 5.0 allow to store procedures on the server for fast execution.

C.5. Algorithm 7.1 in Pseudo-Code

The algorithm solving the centralised diagnostic problem for stochastic automata is given below in recursive form. Confer [93] for a more detailed description.

Given: Automaton \( \mathcal{A} \), a-priori initial condition \( p(z(0)), p(f(0)) \)

Loop: For \( k = 0 \) to \( k_h \) (diagnostic horizon)

- Measure \( v(k), w(k) \)
- For all \( z' \in \mathcal{N}_z, f \in \mathcal{N}_f \)
  - Calculate prediction \( P(z(k+1), f|V(0...k),W(0...k)) \) with (7.4)-(7.5)
  - Calculate projection \( P(f|V(0...k),W(0...k)) \) with (7.6)

End For

Output \( p(f|V(0...k), W(0...k)) \)

End For

Result: Fault probability distribution \( p(f|V(0...k_h), W(0...k_h)) \)

This algorithm has been implemented for MATLAB/Simulink\textsuperscript{®} as an m-coded s-function and is an integral part of the toolbox DIAMOND\textsuperscript{Q}. As opposed to the algorithm the implementation calculates the result vectorially.
C. Algorithms and Implementation

C.6. Algorithm 7.2 in Pseudo-Code

The algorithm solving the centralised diagnostic problem for SAN is an extension of the Algorithm 7.1. In every loop the part of the network applying to $v^v(k)$ and $w^w(k)$ is composed to a single SA with the characteristic function $\hat{L}^s$.

Given: Automata network $A^s\s = \{A^s_1,\ldots,A^s_\gamma\}$, a-priori initial conditions $p(z_i(0)), p(f_i(0))$ of all $i$ automata

Loop: For $k = 0$ to $k_h$ (diagnostic horizon)
  Measure $v(k), w(k)$
  Calculate $\hat{L}^s(k)$ for measurements using composition rule (3.40)
  For all $z' \in N_z, f \in N_f$
    Calculate prediction $P(z(k+1), f|V(0\ldots k), W(0\ldots k))$ with (7.4)-(7.5)
    Calculate projection $P(f|V(0\ldots k), W(0\ldots k))$ with (7.6)
  End For
Output $p(f|V, W)$
End For

Result: Fault probability distribution $p(f|V(0\ldots k_h), W(0\ldots k_h))$

This algorithm has never been implemented in a toolbox, because of its computational disadvantages. However, the realisation for a specific example is described in [69].

C.7. Enhanced Centralised Diagnostic Algorithm for SAN

This algorithm includes an additional loop due to the on-line composition. The measurements $v(k)$ and $w(k)$ are appended to included the coupling signals $s$. The remaining parts of the algorithm work as stated previously.

Given: Automata network $A^s\s = \{A^s_1,\ldots,A^s_\gamma\}$, a-priori initial conditions $p(z_i(0)), p(f_i(0))$

Loop: For $k = 0$ to $k_h$ (diagnostic horizon)
  Measure $v(k), w(k)$
  For all $z' \in N_z, f \in N_f$
    For all $s \in N_s$
      Calculate prediction $P(z(k+1), f|V(0\ldots k), W(0\ldots k))$ with (7.9)
      Calculate projection $P(f|V(0\ldots k), W(0\ldots k))$ with (7.10)
    End For
  End For
Output $p(f|V, W)$
End For

Result: Fault probability distribution $p(f|V(0\ldots k_h), W(0\ldots k_h))$

This algorithm has been implemented for MATLAB/Simulink® and will be part of the next version of the toolbox DIAMONDQ. To assure that the code is efficient for the specific problem a tool has been programmed to analyse a network and to generate a diagnoser for the problem.
In this section a short introduction to SQL (Structured Query Language) is given. This language is used in a number of database programs like Microsoft SQL Server, Oracle, Borland Interbase, and MySQL, which is used in this book. The language has been standardised in the ISO-norms SQL1 (ISO/IEC 9075:1987 extended in ISO/IEC 9075:1989), SQL92 (ISO/IEC 9075:1992), SQL99 (ISO/IEC 9075:1999) und SQL2003 (ISO/IEC JTC1/SC32/WG3). However, these standards have not been fully implemented in most database programs and many software producers introduce additional extensions. For more information about SQL see [32, 75].

MySQL-commands can be divided into the four groups

- *data query* for retrieving information,
- *data manipulation* for inserting, altering, and deleting information,
- *data definition* for defining and altering the database structure such as the table schemas,
- *user rights options* for granting user privileges (not further discussed in this book).

In the following the basic SQL-commands are explained, whereby the commands are given in upper-case and the arguments in lower-case letters. Optional arguments are encased in square brackets and options which exclude each other are separated by a vertical bar. Commands are separated by a semi-colon, text encased in the symbols /* and */ are ignored by the parser.

**Data query**

The syntax of a data query command is given by

```
SELECT
    [ALL | DISTINCTROW ]
    select_expr, ...
FROM table_references [WHERE where_condition]
    [GROUP BY col_name | expr | position
    [ASC | DESC], ... [WITH ROLLUP]]
```
D. Introduction to SQL

- **select_expr** lists the attributes of the resulting relation (table). The symbol * implies a list of all attributes. Each attribute can be renamed using the AS clause.

- **table_references** indicates the relations from which to retrieve the tuples. It can be a single relation or a join of relations.

- The **WHERE** clause states a condition that tuples must satisfy to be included in the resulting relation. The where_condition is an expression that evaluates to true for each tuple to be selected. Examples for a where_condition are \( a=5 \), \( b=’Hallo’ \), or \( T1.b>T2.b \) AND \( a<2 \). All tuples are returned if there is no WHERE clause.

- **GROUP BY** groups and sorts the retrieved information according to the supplied list of columns. If this command is supplied, aggregate functions can be used in the select_expr such as \( \text{SUM} \) to aggregate the grouped information. This way e.g. the number of orders of each month can be easily output.

- The **HAVING** clause works like the WHERE clause, but is used with aggregate functions.

- **LIMIT** gives the maximum number of tuples in the output relation.

- **PROCEDURE** states a stored function which process the data in the result set.

- **INTO OUTFILE** allows to store the resulting relation in a text-file.

**Examples using Table 3.4 on Page 24:**

```sql
SELECT * FROM T1;
SELECT f1 FROM T1;
SELECT f1 AS fault FROM T1;
```

The first expression returns the full relation \( T1 \) whereas the second expression returns only the first column. The third expression returns the identical tuples as the previous expression, but the column has been renamed to \( \text{fault} \).

```sql
SELECT f1 FROM T1 WHERE s=2;
```

Returns the column \( f1 \), but only the tuples for which the condition \( s = 2 \) holds.

```sql
SELECT f1,f2 FROM T1,T2;
SELECT * FROM T1 NATURAL JOIN T2;
```
The first expression returns the columns $f_1$ and $f_2$ containing all tuples of the Cartesian product of $T_1$ and $T_2$. The second expression returns the natural join of these two tables. If no relation contains "null"-values the natural join is identical to the expression:

$$\text{SELECT } * \text{ FROM } T_1, T_2 \text{ WHERE } T_1.s = T_2.s;$$

The following expression returns the number of tuples in the cartesian product of $T_1$ and $T_2$ where the value of the attribute is 1 or 2 respectively (cf. Table D.1).

$$\text{SELECT } f_1, \text{COUNT}(f_1) \text{ FROM } T_1, T_2$$

Table D.1.: Result of the expression $\text{SELECT } f_1, \text{COUNT}(f_1) \text{ FROM } T_1, T_2$

<table>
<thead>
<tr>
<th>$f_1$</th>
<th>$\text{COUNT}(f_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Data manipulation

The syntax of the three major data manipulation commands is given below. Some options such as row limit and priority have been omitted for complexity reasons.

- The $\text{INSERT}$ command inserts new tuples into an existing relation $\text{tbl}_\text{name}$. In the first variant the tuples are given explicitly in $\text{expr}$, whereas in the second variant the result of a $\text{SELECT}$ expression is inserted into the table.

- The $\text{DELETE}$ command removes all tuples from the relation $\text{tbl}_\text{name}$ for which the $\text{where}_\text{condition}$ holds true.

- The $\text{UPDATE}$ statement updates the columns of existing tuples in the relation $\text{tbl}_\text{name}$ with new values. The $\text{SET}$ clause indicates which columns to modify and the values they should be given. The $\text{WHERE}$ clause, specifies the conditions that identify which tuples to update. With no $\text{WHERE}$ clause, all tuples are updated.

Examples using Table 3.4 on Page 24:

$$\text{INSERT INTO } T_1 \ (f_1, s) \ \text{VALUES (1,3), (2,3);}$$

$$\text{INSERT INTO } T_1 \ (f_1, s) \ \text{SELECT } f_2 \ \text{AS } f_1, s \ \text{FROM } T_2;$$

The first command inserts the two new tuples $(f_1 = 1, s = 3)$ and $(f_1 = 2, s = 3)$ into the relation $T_1$. The second command inserts all tuples from relation $T_2$ into relation $T_1$. 
**D. Introduction to SQL**

```sql
UPDATE T1 SET f1=3 WHERE s=1;
DELETE FROM T1 WHERE s=2 AND f1=1
```

The first expression changes the value of \( f1 \) to 3 in all tuples in T1 where \( s = 1 \) holds. The second expression deletes the tuple \( (f1=1,s=1) \) from relation \( T1 \).

**Data definition**

The syntax of the most prominent data definition commands are stated below. The CREATE TABLE and ALTER TABLE commands are very complex and therefore cannot be discussed in full detail. Especially the topics of keys and indices are far beyond the scope of this work.

```
CREATE DATABASE db_name
CREATE [TEMPORARY] TABLE tbl_name (create_definition,...)
CREATE [TEMPORARY] TABLE tbl_name LIKE old_tbl_name
ALTER TABLE tbl_name alter_spec [, alter_spec] ...
DROP TABLE|DATABASE name
```

- **CREATE DATABASE** creates an empty database.
- **CREATE TABLE** creates a relation with the specified schema. In the first variant the schema is given explicitly whereas in the second variant the schema of another table is copied. The TEMPORARY keyword forces the database program to delete the relation after the connection to the database server is quit.
- **ALTER TABLE** changes the schema of a relation or the relation’s name.
- **The DROP command** deletes a relation or a full database irrevocably.

The **create_definition** for the CREATE TABLE command is defined below.

```
col_name data_type
[ NOT NULL | NULL ] [DEFAULT default_value]
[ AUTO_INCREMENT ] [ UNIQUE [ KEY ] | [ PRIMARY ] KEY ]
```

Here the **data_type** can be any data type supported by the used database program. Additional statements as allowing empty entries or demanding unique entries can be specified. The **alter_spec** mainly includes commands to alter the schema of a relation.

```
ADD [COLUMN] (column_definition,...)
| ALTER [COLUMN] col_name
  {SET DEFAULT literal | DROP DEFAULT}
| CHANGE [COLUMN] old_col_name column_definition
  [FIRST|AFTER col_name]
| MODIFY [COLUMN] column_definition
  [FIRST | AFTER col_name]
| DROP [COLUMN] col_name  | RENAME [TO] new_tbl_name
```
Table E.1.: Equivalent automaton for the proof on Page 128

<table>
<thead>
<tr>
<th>$z'_1$</th>
<th>$z'_2$</th>
<th>$z_1$</th>
<th>$z_2$</th>
<th>$f_1$</th>
<th>$f_2$</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0,45</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0,45</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0,4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0,4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0,45</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0,45</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0,4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0,4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0,05</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0,05</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0,05</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0,05</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0,05</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0,05</td>
</tr>
</tbody>
</table>
**E. Example Relations**

Table E.2.: Behavioural relations for Example 9

<table>
<thead>
<tr>
<th>$z'_1$</th>
<th>$w^1$</th>
<th>$s^1$</th>
<th>$z_1$</th>
<th>$v^1$</th>
<th>$f^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$z'_2$</th>
<th>$w^2$</th>
<th>$z_2$</th>
<th>$v^2$</th>
<th>$s^1$</th>
<th>$f^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>