STUDIES IN *FUZZINESS* AND *SOFT COMPUTING*

Alexander Gegov

Complexity Management in Fuzzy Systems

Springer

Alexander Gegov

Complexity Management in Fuzzy Systems

Alexander Gegov

Complexity Management in Fuzzy Systems

A Rule Base Compression Approach

Springer

Alexander Gegov
University of Portsmouth
School of Computing
Buckingham Building
Po1 3HE Portsmouth
United Kingdom
E-mail: alexander.gegov@port.ac.uk

Library of Congress Control Number: 2006933536

ISSN print edition: 1434-9922
ISSN electronic edition: 1860-0808
ISBN-10 3-540-38883-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-38883-8 Springer Berlin Heidelberg New York

Typesetting: by the author and techbooks using a Springer LaTeX macro package
Cover design: Erich Kirchner, Heidelberg

Printed on acid-free paper SPIN: 11733621 89/techbooks 5 4 3 2 1 0

To the pioneers of European integration

Preface

Doing research is a great adventure
As any adventure sometimes it is hard
You may feel alone and with no idea where to go
But if you have courage and press onwards
You will eventually stand where no one has stood
And see the world as no one has seen it
There can be no better feeling than this!
Adaptation from 'Introduction to Research',
Tom Addis (2004)

The idea about this book has been on the author's mind for almost a decade but it was only about a couple of years ago when the underlying research process was actually started. The reason for this delay has been the insufficient spare time for research being a lecturer in a 'new' UK university where the emphasis is mainly on teaching. And maybe this book would have never been written if the author had not been presented with the chance of developing new teaching modules in fuzzy logic that have given him food for thought in a research related context and have helped him combine efficiently his teaching and research activities.

The title of this book may sound too specialised but it has a much wider meaning. Fuzzy systems are any systems for modelling, simulation, control, prediction, diagnosis, decision making, pattern recognition, image processing, etc. which use fuzzy logic. Although fuzzy logic is an advanced extension of binary logic, the latter is still used predominantly today. The main reason for this anachronism is the high level of complexity that is usually associated with the use of fuzzy logic. So, if we were able to solve this problem by means of a suitable complexity management approach, fuzzy logic would gradually replace binary logic. In this context, the book is aimed at anyone who wants to explore new possibilities for using fuzzy logic more effectively and more efficiently.

Complexity has always been an important attribute in research methods. Even nowadays, in the age of supercomputers capable of performing computations at a very high speed, there are still open problems in many research areas which are due to computational complexity, and in particular, to the lack of adequate methods for managing this complexity. For example, some large scale problems in cosmology and genetics are still unsolvable within a reasonable time with the computational power available at present. In the case of threat from terrorism or natural disasters, the amount of information that is relevant to the decision making process can be so large that it may not be possible to process this information reliably within a reasonable time even with the help of the fastest computers.

We could possibly wait for computer technology to become more powerful to hopefully cope with the current challenges of complexity. Unfortunately, this process may take too long, and if and when it has been finally completed, the expected result may still be quite unreliable. The reason for this is that complexity as an attribute of the world that we live in has not only a quantitative dimension in terms of number and scale but also a qualitative dimension in terms of uncertainty and ambiguity. And until we learn how to deal with the qualitative aspects of complexity, we may never be in the position to solve some of the current problems however advanced the computer technology is. Also, we must bear in mind that efficiency of computer software usually has a much greater impact on computational times associated with large scale problems than the speed of computer hardware on which the software is running.

So, if we want to tackle successfully the current challenges of complexity, we need to be able to develop and implement efficient and intelligent computational algorithms. These algorithms must be capable of not only dealing with the quantitative aspects of complexity on the basis of their efficiency but also with its qualitative aspects by means of their intelligence. In this context, fuzzy systems in the form of rule bases are possibly the best tool available for accounting qualitative aspects of complexity such as the uncertainty of the environment. However, the processes of fuzzification, inference and defuzzification usually make these systems suffer from some quantitative aspects of complexity such as the large number of fuzzy rules and associated operations on the fuzzy membership functions of the inputs and the outputs.

The focus of this book is on the management of complexity in fuzzy rule based systems. This problem has been pushed from a marginal location into the mainstream of fuzzy logic research in recent years. The reason for this move is due to the ever more increasing demand for using fuzzy logic not only for small scale domestic purposes but also in large scale industrial and other applications. As a result, many papers and books on fuzzy logic have started to discuss the complexity aspects of the proposed methods in separate paragraphs or even whole sections. Moreover, a number of specific methods for complexity reduction in fuzzy systems have already been developed and used.

This work is possibly a first attempt to deal with the problems of complexity in fuzzy rule based systems at a monographic level. The underlying philosophy is based on the idea of managing complexity rather than only reducing it. In this context, management is viewed as a group of activities such as perception, understanding, and analysis of complexity with the intention of simplifying it formally in a universal and systematic way. This type of approach is quite different from most of the existing complexity reduction methods, which are usually characterised by a limited application scope and empirical nature. In fact, many of these methods simplify the complexity in fuzzy systems by actually ignoring it without adequate justification and with the hope that the resultant simplified system would behave similarly to the original complex one.

As opposed to the complexity reduction methods mentioned above, this book does not rely on semi-mechanistic simplifications and hopes for good luck in a gamble. It is based on the assumption that the inherent redundancy in fuzzy systems should be exploited not by empirical trials but through a sound and formal systematic process. This redundancy and the resulting complexity are attributes of the virtual fuzzy world created by us for the purpose of dealing with the uncertainty in the real crisp world. So, it is only up to us to identify this redundancy accurately, remove it safely and then arrive quickly at the desired destination in the real world once the job has been done. It may look a bit like the plot in the famous movie 'The Matrix' but whether you believe it or not – that is exactly what this book is all about.

The author would like to thank some people and institutions without whom this book possibly would not have been what it is. He is very grateful to Prof Robert Babuska from the Systems and Control Centre at the Delft University of Technology, The Netherlands, for the enlightening discussions on the topic, and to the Editor-in-chief for this book series Prof Janusz Kacprzyk for the kind invitation to submit this work which had a stimulating effect on the overall research and writing process. He is also quite indebted to his senior colleague Prof Tom Addis for being an inspirational example with his passion for research, for advising him how to reduce the impact of academic paperwork on his research activities and for encouraging him to challenge the widely spread perception in UK universities that the most important research indicators are the amount of external funding secured and the score achieved in the Research Assessment Exercise. The author would like to thank his junior colleague Dr Bart-Floris Visscher for the constructive feedback on some of the initial drafts of this book, his BSc project student Neelamugilan Gobalakrishnan for the software validation of some of the theoretical results and the copyright holder for some of the material presented in Chapter 9, the University of Portsmouth, for giving him the permission to use this material in the book. The cooperation of Jane Chandler, former Head of the Department of Computer Science and Software Engineering, and Prof Ajit Narayanan, Head of the School of Computing, University of Portsmouth, UK, for keeping his teaching duties within reasonable bounds is also gratefully acknowledged. Special thanks must go to the Alexander von Humboldt Foundation of Germany and the European Union Commission in Brussels for the research fellowships granted to the author in the past,

which have significantly helped him enrich himself as a scientist through the international dimensions of research.

In addition, the author would be unfair not to mention Bon Jovi, Nickelback and the likes for providing the musical entertainment during the long and boring typing process as well as his colleagues Sion Reynolds, Patrick Beullens and Luke Stutters from the University of Portsmouth rock band ‘Infra Rouge’ for the relaxing musical practices and the inspirational live performances over the last couple of years. And finally, the spiritual support and encouragement of his family members and closest relatives has been second to none – a big thanks to all of them for helping a dream become a reality.

Portsmouth, UK *Alexander Gegov*
July 2006

Contents

Abbreviations

SRB – single rule base
IRB – identity rule base
TRB – transpose rule base
PRB – permutation rule base
MRB – multiple rule base
FF – feedforward
FB – feedback
CADR – conjunctive antecedents in disjunctive rules
DADR – disjunctive antecedents in disjunctive rules
CACR – conjunctive antecedents in conjunctive rules
DACR – disjunctive antecedents in conjunctive rules
CON – conjunctive
DIS – disjunctive
MIMO – multiple input multiple output
MISO – multiple input single output
MO – multiple output
SO – single output
IFS – initial fuzzy system
RFS – reduced fuzzy system
ERB – equivalent rule base
I/T/P RB – identity/transpose/permutation rule base
P-T RB – permutation-transpose rule base
FRB – feedback rule base
ARBN – arbitrary rule base network
CRBN – canonical rule base network
MRBO – multiple rule base output
SRBO – single rule base output
CS – conventional system
AS – aggregated system
SS – sorted system
FS – filtered system
HS – hierarchical system
EO – elementary operations

1 Introduction

1.1 Quantitative and Qualitative Complexity

Complexity is a common attribute of the world that we live in. Although the recent advances in different technologies have made our life easier in many aspects, these advances also bring new challenges that add to the overall complexity in this world. A typical example in this respect is global warming which is a direct consequence of our habits to consume products that require the use of large amounts of energy. Without a doubt, global warming has already become a very complex environmental problem that requires urgent and coordinated actions by international institutions if we want to save our world for future generations.

In general, complexity can be characterised by two main aspects – quantitative and qualitative. The quantitative aspect usually has to do with concepts such as number and scale. For example, if we look at the Internet, it has been growing at an enormous rate as a result of which the number of web pages has increased dramatically over the recent years. Apart from that, more and more geographic areas are acquiring access to the Internet almost every day and this has contributed significantly to the overall enlargement of its scale. The qualitative aspect is usually related to concepts such as uncertainty and ambiguity. If we take again the example with the Internet, it is obvious that as the level of its quantitative complexity increases in terms of the number of web pages and scale, it becomes more difficult to understand and interpret its behaviour, i.e. its level of qualitative complexity also increases.

Usually, as the level of quantitative complexity in a man-made system such as the Internet increases, it leads to a corresponding increase in the level of its qualitative complexity. This is due to the fact that the growth in number and scale is accompanied by the formation of new relations among the building blocks in the system as a result of which it becomes more difficult to describe and predict these relations, i.e. the system becomes more difficult for perception and more uncertain in its behaviour. Therefore, we may assume that the quantitative aspect of complexity implies the qualitative aspect and this assumption is completely in line with the dialectical philosophical idea that any quantitative changes in a system gradually lead to corresponding qualitative changes.

1.2 Time and Safety Critical Implications

In order to be able to cope with the challenges of complexity, we need to identify its possible implications. In this respect, two of the most important implications are time critical and safety critical problems.

In the case of a time critical problem, we need to find a solution to the problem within a limited period of time – otherwise the solution may become irrelevant. For example, an industrial robot that classifies and assembles different particles according to their size and shape has to do these tasks in a way that matches the speed at which the conveyor carrying these particles is moving. Otherwise, the robot would be late for the corresponding assembly operations and the whole manufacturing process may have to be stopped temporarily so that the particles affected by the slow operations of the robot can be rearranged accordingly. Obviously, an increased level of the quantitative and the qualitative complexity in terms of the number of particles carried by the conveyor and the uncertainty in their size or shape, respectively, would make this time critical problem more difficult.

In the case of a safety critical problem, if we do not find a solution to this problem then this could lead to loss of human life. For example, an on-board aircraft collision avoidance system is supposed to detect other aircrafts within a certain radius, determine their location and speed, and if necessary, instruct the pilot to change course. Otherwise, the aircraft would continue to follow the preset course, which may lead to a collision with another aircraft. Here again, an increased level of the quantitative and the qualitative complexity in terms of the number of other closely flying aircrafts and some uncertainty in their location or speed would obviously make this safety critical problem quite difficult.

Usually, a safety critical problem such as the one with the aircraft is also a time critical problem. In this case, the location and the speed of all closely flying aircrafts have to be determined within a limited time period as otherwise the aircraft may collide with another one while the associated computations are still running. Therefore, we may assume that a safety critical problem usually implies a time critical problem and that is not surprising bearing in mind that safety related issues often have explicit time limits.

1.3 Speed and Intelligence of Computers

Having looked at the main aspects of complexity and its implications, it would be interesting to see how we could possibly minimise the undesirable impact of this complexity. As the world today is becoming more dominated by computers that assist people in their everyday activities, it would be reasonable to assume that computer hardware and software are the main factors determining the potential impact of complexity on these activities. What really matters in this case is the relevant attributes of computer hardware and software such as speed and intelligence.

The speed of computers is usually associated with the clock frequency at which the basic arithmetic operations are performed at the hardware level and the algorithmic efficiency at the software level which represents the amount of computations as a function of the size of the problem to be solved. Obviously, the speed of computers can affect directly the quantitative aspects of complexity by reducing the time that it takes to carry out a certain amount of computations.

As opposed to speed, intelligence in computers is still something quite unclear and hard to quantify but by this term we usually mean the ability of computer hardware and software to behave in a way that resembles intelligent human beings, e.g. to be able to learn and reason. In this case, there is a clear link with the qualitative aspects of complexity because learning and reasoning are especially helpful in an environment that is characterised by uncertainty, i.e. where the ability to see 'invisible' objects and to predict 'unexpected' events could make a big difference.

It would be interesting to see to what extent the speed and intelligence of computer hardware and software could affect the quantitative and qualitative aspects of complexity. There has been an ongoing argument and rivalry on this issue between hardware and software professionals but the actual facts so far appear to be more in favour of the opinion held by the second group. For example, it has been shown that the algorithmic efficiency of software is usually more critical for the reduction of the computational times associated with large scale problems than the clock frequency of hardware. Also, as far as the computer market is concerned, it has been much easier to incorporate 'intelligent' attributes in software than in hardware.

1.4 Past and Current Research in Fuzzy Logic

Research in the field of fuzzy logic has gone a long way since the idea about fuzzy sets was first introduced in the mid 60's of the 20th century by Lotfi Zadeh. Since then, we have witnessed a number of trends most of which have lasted for about a decade.

The late 60's and the 70's were characterised mainly by theoretical works that helped fuzzy logic establish itself as a separate discipline alongside deterministic mathematics and statistics. The main drawback of this period was the highly abstract nature of research, which made many applied scientists ignore fuzzy logic before even making themselves familiar with it to some extent. It was not unusual at that time to hear statements describing fuzzy logic as a totally useless theory that would never work in practice.

However, in the 80's, scientists from Japan started to implement fuzzy logic in a number of domestic appliances such as vacuum cleaners, refrigerators and cookers. Later on, in the 90's, we were able to witness the first successful industrial applications of fuzzy logic. But despite the big number of successful applications of fuzzy logic in this period, the criticism against fuzzy logic did not stop – it has even got bigger in the recent years. This time, the main object of attack coming from both outside and inside of the fuzzy academic community has been the empirical nature of fuzzy logic.

As a whole, the main focus of the criticism on fuzzy logic has been on its inability to behave as a systematic science. A scientific method is usually expected to be applicable for solving a particular problem with a guaranteed success. Unfortunately, that is not the case for most of the known fuzzy logic methods, which are empirical and therefore not quite reliable in terms of the expected results.

So, the way forward for fuzzy logic research in the 21st century looks much clearer now. We have to make fuzzy logic science, which is both fundamental and applied, i.e. a powerful and universal theory that can be also validated and justified in practice. Hopefully, the lessons learned from the mistakes made in the past can help us achieve this goal in the foreseeable future. And the sooner we stop swinging into one or another extreme – the better.

1.5 Complexity Issues in Fuzzy Systems

Fuzzy systems are usually good at capturing the qualitative aspect of complexity by means of their linguistic modeling and approximate reasoning capabilities. However, this comes with a price because the associated fuzzy operations in the fuzzification, the inference and the defuzzification stages increase the level of quantitative complexity of the problem. This increase becomes even more embarrassing as the number of inputs in the fuzzy system gets bigger because the amount of these operations is a function of the number of rules which, on its turn, depends on the number of inputs.

Apart from the increased level of their quantitative complexity, the transparency and interpretability of fuzzy systems tends to deteriorate as the number of fuzzy rules increases. In this case, it is harder to observe and explain what is happening in the system. In other words, although the qualitative complexity of the environment is usually well accounted for by the fuzzy system, the system itself appears to generate 'new' qualitative complexity by its existence. So, it turns out that the problem has been actually moved from the environment that the fuzzy system is supposed to model, to the system itself.

There has been a steadily growing interest in complexity issues of fuzzy systems in recent years [1, 3, 4, 13, 14, 15, 18, 19, 24, 36, 40, 42, 44, 62, 69, 75, 77]. This is due to the fact that fuzzy systems have become more widely used in applications of a larger scale as a result of which the associated complexity becomes more apparent. For example, many recent papers and books on fuzzy logic discuss the complexity aspects of the proposed fuzzy methods in separate sections or even chapters. However, the focus of these discussions is usually on quantitative complexity whereas qualitative complexity is often ignored.

1.6 From Complexity Reduction to Complexity Management

It is not surprising that research in fuzzy systems has been focused mainly on quantitative complexity issues. After all, it is normal to expect current research methodologies to be strongly affected by the dominant profit orientated values in our society and the associated material targets such as improved efficiency and increased productivity. That is why most of the known methods dealing with complexity in fuzzy systems are aimed primarily at reducing the time for the completion of the required computations.

This book preaches a different philosophy. It argues that we have to change our narrow minded and conservative way of thinking only within ‘the box’. Our duty as academics is to open new horizons and suggest viable alternatives to the existing ‘status quo’ in our subjects as well as in the world in general. Trying to break this ‘status quo’ would be the best way of not only discovering new knowledge but also of demonstrating a broad, progressive and independent attitude.

As far as fuzzy systems are concerned, we have to start addressing the complexity issues in these systems from a different angle. The idea of reducing complexity by actually ignoring it in a semi-mechanistic way may suit the market needs but it does not speak well about academia. Before reducing complexity, we need to be able to identify, analyse and understand it properly. This is what planned complexity management is all about and this book argues that such an approach must replace the current approach of chaotic complexity reduction.

The use of this new approach would guarantee that if and when we decide to reduce the quantitative complexity, this process will lead to a systematically correct and scientifically justified result even if we can not achieve big financial gains. Last but not least important, by following this approach we would be in the position to also reduce the qualitative complexity in fuzzy systems which will make them more transparent to us as well as more enjoyable to work with. Because after all, apart from the material things in life, there are spiritual things that matter too, and with that in mind we can hopefully make another ‘fuzzy’ world possible.

1.7 Description of Book Chapters

This book is organised in 10 chapters. The current Chapter 1 is an introduction to the topic of complexity in general and in the context of fuzzy logic. Chapter 2 discusses the most common types of fuzzy rule based systems and analyses their impact on complexity. Chapter 3 reviews most of the existing rule base reduction methods for fuzzy systems and summarises their attributes. Chapter 4 introduces advanced techniques for formal presentation of fuzzy systems based on Boolean matrices and binary relations, which facilitate the overall management of complexity. Chapters 5 and 6 present techniques for formal manipulation of *single rule base* (SRB) fuzzy systems with general and special rule bases, which reduce the qualitative complexity. Among the manipulation techniques presented are vertical, horizontal and output merging as well as splitting of rule bases whereas among the special rule bases considered are the *identity rule base* (IRB), the *transpose rule base* (TRB) and the *permutation rule base* (PRB). Chapters 7 and 8 describe techniques for formal transformation of *multiple rule base* (MRB) fuzzy systems with *feedforward* (FF) and *feedback* (FB) interconnections, which also reduce the qualitative complexity. Among the transformation techniques described are repetitive and combined merging manipulations in the context of self standing inputs and outputs as well as total and partial identity lines whereas among the FB interconnections considered are single, local, global, nested, overlapping, crossed and multiple interconnections. Chapter 9 proposes techniques for formal simplification of fuzzy rule based systems, which reduce the quantitative complexity by aggregation of inconsistent rules and filtration of non-monotonic rules. It also shows case studies with complex fuzzy systems and gives a comparative evaluation of the complexity of these techniques. The last Chapter 10 is a conclusion highlighting the theoretical significance of the formal methodology for complexity management in fuzzy systems, an application framework for this methodology and some possible related future research directions.

2 Basic Types of Fuzzy Rule Based Systems

2.1 Mamdami, Sugeno and Tsukamoto Systems

A common type of fuzzy system is the Mamdami system, which is represented by the if-then rules

$$\begin{array}{c} \text{If } i_1 \text{ is } v_{i1,1} \text{ and/or} \ldots \text{and/or } i_m \text{ is } v_{im,1} \text{ then } o_1 \text{ is } v_{o1,1} \text{ also} \ldots \text{also } o_n \text{ is } v_{on,1} \\ \text{and/or} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \text{and/or} \\ \text{If } i_1 \text{ is } v_{i1,r} \text{ and/or} \ldots \text{and/or } i_m \text{ is } v_{im,r} \text{ then } o_1 \text{ is } v_{o1,r} \text{ also} \ldots \text{also } o_n \text{ is } v_{on,r} \end{array} \tag{2.1}$$

where m is the number of inputs, n is the number of outputs and r is the number of fuzzy rules in the system [43, 66, 81]. In this case, i_p, $p = 1,..,m$ represents the p-th input, $v_{ip,s}$ $p = 1,..,m,$ $s = 1,..,r$ is the linguistic value of the p-th input in the s-th rule, o_q, $q = 1,..,n$ represents the q-th output and $v_{oq,s}$ $q = 1,..,n,$ $s = 1,..,r$ is the linguistic value of the q-th output in the s-th rule.

Sometimes, the term ‘fuzzy set’ is used as a synonym of the term ‘linguistic value’. Both terms are suitable for modelling the uncertainty available in many systems by means of the concept of a fuzzy membership degree. This concept is based on the assumption that an object can belong to a fuzzy set with a membership degree that can take any value in the interval [0,1].

The terms with the inputs and their linguistic values in the ‘if’ part of the rule base are also called antecedents whereas the terms with the outputs and their linguistic values in the ‘then’ part of the rule base are called consequents. Usually, the ‘if’ part of the rule base contains all possible permutations of linguistic values of the inputs. As far as the ‘then’ part of the rule base is concerned, it is unlikely to contain all possible permutations of linguistic values of the outputs.

Generally, the number and the meaning of the linguistic values that each input can take vary as inputs usually have different crisp variation range and specific crisp physical meaning. Although this peculiarity is not reflected explicitly in the rule base represented by Eq. (2.1), it can be easily accounted for.

Also, each rule in the rule base represented by Eq. (2.1) may be allocated a specific weight that reflects the importance of this particular rule.

However, in most cases the individual rules are assumed to have equal weight and by default this fact is not reflected explicitly in the rule base.

Apart from its weight, each rule has a firing strength that reflects the extent to which the antecedent terms in the rule are satisfied. In other words, the firing strength of a rule is a measure of the relative impact of this rule on the outputs of the fuzzy system.

The Mamdami system is the most widely used fuzzy system. The main advantages of this type of system are its widespread acceptance, intuitive character and capability to formalise inputs in the form of expert knowledge. In some cases, when there is no sufficient or sensible data available, using expert knowledge is the only way to collect information about the system that we want to model. Therefore, a Mamdami system is a 100% fuzzy system in which the fuzziness spreads from the inputs to the outputs during the stages of fuzzification, inference and defuzzification.

Another type of fuzzy system is the Sugeno system, which is represented by if-then rules in the form:

$$\begin{array}{c} \text{If } i_1 \text{ is } v_{i1,1} \text{ and/or} \ldots \text{ and/or } i_m \text{ is } v_{im,1} \text{ then } o_1 = f_{1,1}(i_{1,\ldots,}i_m) \text{ also} \ldots \text{ also } o_n = f_{n,1}(i_{1,\ldots,}i_m) \\ \text{and/or} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \text{and/or} \\ \text{If } i_1 \text{ is } v_{i1,r} \text{ and/or} \ldots \text{ and/or } i_m \text{ is } v_{im,r} \text{ then } o_1 = f_{1,r}(i_{1,\ldots,}i_m) \text{ also} \ldots \text{ also } o_n = f_{n,r}(i_{1,\ldots,}i_m) \end{array} \qquad (2.2)$$

The 'if' part of the rule base in a Sugeno system is the same as the one in a Mamdami system. However, the outputs o_q, $q = 1,n$ in the 'then' part of the rule base are not represented by linguistic values but as polynomial functions $f_{q,s}$ (..), $q = 1,..,n,$ $s = 1,..,r$ of the inputs i_p, $p = 1,..,m$ [43, 66]. In most cases, these are first-order polynomials in the form of linear functions. All other considerations made for Mamdami systems are also valid for Sugeno systems.

The Sugeno system is less popular than the Mamdami system but it is also widely used. The main advantages of this type of fuzzy system are its computational efficiency and capability to work well with linear techniques such as proportional-integral-derivative control. In a Sugeno system, the inference and the defuzzification stage are simplified due to the presentation of the outputs as crisp functions of the inputs as a result of which many of the fuzzy operations are replaced with more efficient crisp operations. In other words, a Sugeno system uses a crisp approximation of the outputs and therefore can be viewed as a hybrid between a fuzzy and a crisp system.

A third type of fuzzy system is the Tsukamoto system which is represented by the same if-then rules as the ones used in a Mamdami system. The main difference here is that the fuzzy membership functions for the outputs are monotonic functions with a single shoulder which map the firing strength of each rule to a unique crisp value of the output [43, 66].

In a sense, the Tsukamoto system is a hybrid between the Mamdami and the Sugeno system because the outputs are represented by linguistic values similar to the ones in a Mamdami system whereas the crisp values of the outputs are obtained as functional mappings similar to the ones in a Sugeno system. All other considerations made for Mamdami systems are also valid for Tsukamoto systems.

The Tsukamoto system is the least popular fuzzy system. Although its purpose is to combine the advantages of the Mamdami and the Sugeno system, it has found only limited application due to the constraint imposed by the special type of monotonic fuzzy membership functions for the outputs. Obviously, these functions are not capable of representing adequately the variables in most of the available real systems.

2.2 Conjunctive and Disjunctive Systems

There are four possible variations on each of the fuzzy rule bases represented by Eqs. (2.1)–(2.2)

- *conjunctive antecedents in disjunctive rules* (CADR),
- *disjunctive antecedents in disjunctive rules* (DADR),
- *conjunctive antecedents in conjunctive rules* (CACR),
- *disjunctive antecedents in conjunctive rules* (DACR),

where the terms ‘conjunction’ and ‘disjunction’ in these variations are denoted by the logical ‘and’ and ‘or’ operator, respectively [61, 81].

Sometimes, a fuzzy rule base is called *conjunctive* (CON) or *disjunctive* (DIS), depending on whether the reference is made to the antecedents or the rules. For example, a CADR fuzzy system is usually referred to as a CON system in an antecedent related context and a DIS system in a rule related context.

All types of fuzzy systems, i.e. Mamdami, Sugeno and Tsukamoto, can have any of the above four possible variations of their rule bases. Depending on the type of variation for the rule base, Eqs. (2.1)–(2.2) must be amended appropriately by replacing each ‘and/or’ operator either with an ‘and’ or an ‘or’ operator.

In the case of CON antecedents, all antecedent terms in each rule must be satisfied together. For DIS antecedents, at least one antecedent term in each rule is satisfied. In the case of DIS rules, at least one rule in the rule base is satisfied. For CON rules, all rules in the rule base must be satisfied together.

DIS rules appear to be more common than CON rules. This is due to the fact that it is usually harder to find real systems in which all fuzzy rules can be satisfied together. On the contrary, CON antecedents are more common than DIS ones because the antecedent terms are usually expected to be satisfied together.

The above four variations reflect the different ways in which the antecedent terms in each rule and the individual rules in the rule base are connected. As far as the consequent terms in the rule base are concerned, the operator ‘also’ is identical with the logical ‘and’ operator, i.e. all consequent terms in each rule must be satisfied.

2.3 Multiple Output and Single Output Systems

Most fuzzy systems are *multiple input multiple output* (MIMO) systems, i.e. they have more than one input and more than one output. The fuzzy systems represented by Eqs. (2.1)–(2.2) are precisely of this type. However, it is usually necessary to decompose a MIMO fuzzy system into a number of *multiple input single output* (MISO) systems in order to facilitate the processes of inference and defuzzification [61, 81].

This decomposition can be done in a series of simple manipulations based on the laws of Boolean logic. By definition, each ‘if-then’ rule in a fuzzy rule base represents a logical implication that can be formally described with the logical ‘imp’ operator. In this case, if the antecedent and the consequent part in the rule are true, then the whole rule must be true.

Therefore, each rule in Eqs. (2.1)–(2.2) may be represented in the form

$$\text{If } (A_1 \text{ and/or} \ldots \text{and/or } A_m) \text{ then } (C_1 \text{ and} \ldots \text{and } C_n) \tag{2.3}$$

where A_p, $p=1,..,m$ and C_q, $q=1,..,n$ are the logical propositions describing the antecendent and the consequent terms, respectively.

Equation (2.3) may be rewritten in the following equivalent forms:

$$(A_1 \text{ and/or} \ldots \text{and/or } A_m) \text{ imp } (C_1 \text{ and} \ldots \text{and } C_n) \tag{2.4}$$

$$[\text{not } (A_1 \text{ and/or} \ldots \text{and/or } A_m)] \text{ or } (C_1 \text{ and} \ldots \text{and } C_n) \tag{2.5}$$

$$[\text{not } (A_1 \text{ and/or} \ldots \text{and/or } A_m) \text{ or } C_1] \text{ and} \ldots \text{and } [\text{not } (A_1 \text{ and/or} \ldots \text{and/or } A_m) \text{ or } C_n] \tag{2.6}$$

$$[(A_1 \text{ and/or} \ldots \text{and/or } A_m) \text{ imp } C_1] \text{ and} \ldots \text{and } [(A_1 \text{ and/or} \ldots \text{and/or } A_m) \text{ imp } C_n] \tag{2.7}$$

$$[\text{If } (A_1 \text{ and/or} \ldots \text{and/or } A_m) \text{ then } C_1] \text{ and}\ldots\text{and } [\text{If } (A_1 \text{ and/or} \ldots \text{and/or } A_m) \text{ then } C_n] \tag{2.8}$$

Therefore, the MIMO fuzzy system in Eq. (2.3) can be represented by the *n* logically equivalent MISO fuzzy systems in Eq. (2.8). This result is obviously of great importance as it allows us to decompose any *multiple output* (MO) fuzzy system into a number of *single output* (SO) fuzzy systems where the output for each SO system can be processed separately.

While the above decomposition reduces the qualitative complexity in a fuzzy system by presenting it in a simpler form, it also tends to make us forget the fact that there is usually more than one output in the system. In other words, we usually do not take into account the quantitative complexity in the fuzzy system that is added by the application of all fuzzy operations associated with the other outputs.

All types of MO fuzzy systems, i.e. Mamdani, Sugeno and Tsukamoto, can be decomposed into a number of logically equivalent SO systems. In this case, the fuzzification process is carried out only once but the processes of inference and defuzzification must be repeated for each output.

So, the MO fuzzy system in Eq. (2.1) can be represented by the following n logically equivalent SO fuzzy systems:

$$\begin{gathered}\text{If } i_1 \text{ is } v_{i1,1} \text{ and/or} \ldots \text{and/or } i_m \text{ is } v_{im,1} \text{ then } o_q \text{ is } v_{q1,1}\\ \text{and/or}\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ \text{and/or}\\ \text{If } i_1 \text{ is } v_{i1,r} \text{ and/or} \ldots \text{and/or } i_m \text{ is } v_{im,r} \text{ then } o_q \text{ is } v_{q1,r}\\ q = 1,..,n\end{gathered} \tag{2.9}$$

Similarly, the MO fuzzy system in Eq. (2.2) can be represented by the following n logically equivalent SO fuzzy systems:

$$\begin{gathered}\text{If } i_1 \text{ is } v_{i1,1} \text{ and/or} \ldots \text{and/or } i_m \text{ is } v_{im,1} \text{ then } o_q = f_{q,1}(i_{1,..,} i_m)\\ \text{and/or}\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ \text{and/or}\\ \text{If } i_1 \text{ is } v_{i1,r} \text{ and/or} \ldots \text{and/or } i_m \text{ is } v_{im,r} \text{ then } o_q = f_{q,r}(i_{1,..,} i_m)\\ q = 1,..,n\end{gathered} \tag{2.10}$$

Equations. (2.9)–(2.10) show the impact on the outputs in Eqs. (2.1)–(2.2). In this case, the q-th output o_q, $q = 1,..,n$ in the MO fuzzy system has become the only output in the q-th SO fuzzy system.

2.4 Feedforward and Feedback Systems

Most fuzzy systems are FF systems, i.e. the flow of information in these systems is only from the inputs to the outputs. The fuzzy systems represented by Eqs. (2.1)–(2.2) are precisely of this type. However, some fuzzy systems may have partial flow of information in the opposite direction, i.e. from some outputs to some inputs. Such systems are called FB systems and they have at least one output, which is fed back into a corresponding input [2, 38, 59]. In this case, the input and the output represent the same variable but their linguistic values usually describe this variable at different moments in time due to the delay in the flow of information in both directions.

All types of fuzzy systems, i.e. Mamdami, Sugeno and Tsukamoto, can have either one or the other pattern of information flow. In the case of FB pattern, Eqs. (2.1)–(2.2) must be amended appropriately by specifying explicitly which output-input pairs represent the same variable. For example, if the first output is fed back into the second input, this can be represented by $o_1 = i_2$.

2.5 Single Rule Base and Multiple Rule Base Systems

Most fuzzy systems are SRB systems. They have either one rule base, e.g. a MO fuzzy system, or a number of independent rule bases, e.g. SO fuzzy systems. For example, the MO systems and their SO counterparts represented by Eqs. (2.1)–(2.2) and Eqs. (2.9)–(2.10), respectively, are all SRB systems. In this sense, the most distinctive feature of a SRB system is the isolated nature of its rule bases.

However, some processes can be better modelled by a MRB system, i.e. a system with some interconnections between its rule bases [7, 11, 53, 55]. This is usually the case of multi-stage processes where the outputs from a particular stage are also inputs to one or more subsequent stages. The MRB systems used for describing such processes are usually referred to as ‘chained fuzzy systems’ but we will be using the newly introduced and more general term ‘MRB systems’ instead throughout this book for completeness and consistency.

A MRB system can be described by a network whereby all rule bases in a horizontal row represent a level and all rule bases in a vertical column represent a layer. The numbering of levels is from top to bottom whereas the numbering of layers is from left to right. Interconnections may exist between rule bases residing in the same layer as well as between rule bases, which are in different layers. Some of these interconnections can be in a forward direction, i.e. from a particular layer to one or more subsequent layers. Other interconnections can be in a backward direction, i.e. from a particular layer to the same layer or to preceding layers. The interconnections reflect the nature of the multi-stage process being modelled, i.e. the outputs from each rule base which are also inputs to other rule bases in the same layer, preceding layers or subsequent layers.

The layers in a MRB system represent a temporal hierarchy, i.e. processes that take place sequentially in time. As opposed to this, the levels in a MRB system represent a spatial hierarchy, i.e. processes that are subordinated to each other. Although this spatial subordination is relevant mainly within a particular layer, it is often propagated across the whole network structure in the context of the interconnected rule bases.

The above two types of network hierarchy are often used for modelling systems with the purpose of reducing their quantitative and qualitative complexity. In this sense, the network structure of the fuzzy rule base is either a straightforward reflection of the system being modeled or a design decision aimed at achieving better effectiveness or higher efficiency.

A MRB system with s levels and q layers can be represented by the matrix

$$\begin{array}{llllll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \ldots & \text{layer q-1} & \text{layer q} \\ \text{level 1} & RB_{1,1} & RB_{1,2} & \ldots & RB_{1,q-1} & RB_{1,q} \\ \text{level 2} & RB_{2,1} & RB_{2,2} & \ldots & RB_{2,q-1} & RB_{2,q} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \text{level s-1} & RB_{s-1,1} & RB_{s-1,2} & \ldots & RB_{s-1,q-1} & RB_{s-1,q} \\ \text{level s} & RB_{s,1} & RB_{s,2} & \ldots & RB_{s,q-1} & RB_{s,q} \end{array} \tag{2.11}$$

whose elements $RB_{i,j}$ $I = 1,..,s,\ j = 1,..,q$ are rule bases. Quite often some of the blocks in Eq. (2.11) may not occupied by rule bases and in this case the matrix may have a sparse structure.

In order to define a MRB system fully, each of the existing rule bases in Eq. (2.11) must be given in a form similar to Eq. (2.1), Eq. (2.2), Eq. (2.9) or Eq. (2.10). Also, the interconnections between the rule bases must be given by specifying which outputs from which rule bases are which inputs to which rule bases. This can also be done by the block matrix represented in Eq. (2.12) where the output-input interconnections are given arbitrarily for illustration purposes. In this case, all output-input interconnections are of FF type apart from the ones originating from outputs in the last layer, which are of FB type. If a rule base is missing from the network structure of a MRB system, then the corresponding block in the matrix in Eq. 2.12 will be empty but some of the other blocks must reflect any existing interconnections between rule bases in layers preceding or subsequent to the layer with the missing rule base.

Usually, the outputs and inputs which are not part of any interconnections are not reflected explicitly in the corresponding matrix for the MRB system. Such inputs and outputs stand on their own, i.e. are self standing, and as it can be seen from Eq. (2.12), each rule base from the first to the last but one layer inclusive has only one output which is not self standing. Similarly, each rule base from the second to the last but one layer inclusive has only one input which is not self standing whereas each rule base in the last layer has two inputs which are not self standing.

$$\begin{array}{lllllll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \dots & \text{layer q-1} & \text{layer q} & (2.12) \\
\text{level 1} & o_1 = i_1^{2,2} & o_1 = i_1^{2,3} & \dots & o_1 = i_1^{2,q} & o_1 = i_2^{1,q} & \\
\text{level 2} & o_1 = i_1^{1,2} & o_1 = i_1^{1,3} & \dots & o_1 = i_1^{1,q} & o_2 = i_2^{2,q} & \\
\dots & \dots & \dots & \dots & \dots & \dots & \\
\text{level s-1} & o_1 = i_1^{s,2} & o_1 = i_1^{s,3} & \dots & o_1 = i_1^{s,q} & o_1 = i_2^{s-1,q} & \\
\text{level s} & o_1 = i_1^{s-1,2} & o_1 = i_1^{s-1,3} & \dots & o_1 = i_1^{s-1,q} & o_1 = i_2^{s,q} &
\end{array}$$

Interconnections can be either local or global. If an output from a rule base is fed back into an input to the same rule base, the interconnection is local. However, if an output from a rule base is fed back into an input to another rule base residing in the same or in a different layer, the interconnection is global. In this context, in Eq. (2.12) the interconnections for all layers from the first to the last but one inclusive are global whereas the interconnections for the last layer are local.

In general, the individual rule bases in a MRB system may be of any type such as Mamdami, Sugeno or Tsukamoto systems, CON or DIS systems, FF or FB systems, as well as MO or SO systems. In this case, each output-input interconnection assumes defuzzification and fuzzification of the corresponding output and input, respectively.

2.6 Complexity Analysis in Fuzzy Systems

The type of fuzzy system used may have some impact on the level of complexity of the system being modeled. For example, Sugeno and Tsukamoto systems are usually more efficient than Mamdami systems but they are often less transparent, i.e. what is gained in terms of quantitative complexity is lost in terms of qualitative complexity. CON and DIS systems are not in anyway different from each other in relation to both aspects of complexity. SO systems seem to have slightly better transparency than MO systems but in terms of efficiency they are the same. FB systems are more complex than FF systems in terms of both quantitative and qualitative complexity. And finally, SRB systems are usually less complex than MRB systems in both quantitative and qualitative terms.

Usually, the maximum number of rules in a fuzzy system r is an exponential function of the number of the inputs m and the number of linguistic values w that each of these inputs can take [37, 48, 51, 60, 66]. In most cases, this exponential function is in the form:

$$r = w^{m} \tag{2.13}$$

However, if the number of linguistic values per input is not a constant, then the maximum number of rules in a fuzzy system is given by the arithmetic product

$$r = w_{1} \dots w_{m} \tag{2.14}$$

where $w_{p,}$ $p = 1,.., m$, is the number of linguistic values that the p-th input can take. In this case, although Eq. (2.14) is not in an explicit exponential form, the number of rules r is still an implicit exponential function.

It is obvious from Eqs. (2.1)–(2.2) and Eqs. (2.9)–(2.10) that for a fuzzy system with 2 inputs which can take 5 linguistic values each, the number of rules is only 25. However, for 3 inputs this number is 125, whereas in the case of 4 inputs it becomes 625. This phenomenon is illustrated in a wider context in Fig. 2.1 where each of the 2, 3 or 4 inputs is supposed to take 3, 5, 7, 9 or 11 linguistic values. So, it is not hard to imagine what the impact of more inputs or more linguistic values per input would be on the number of rules.

The considerations presented above show clearly the level of quantitative complexity associated with multivariable fuzzy systems, even for the case of a fairly small number of inputs. Bearing in mind that many real life systems are usually characterised by a much larger number of inputs and often have to be operated in real-time, it is obvious that the resulting quantitative complexity has to be taken very seriously.

Besides this, as the number of rules gets bigger, it usually becomes harder to understand what is happening in the fuzzy system because the transparency of the rules reduces and our ability to interpret them suffers from that. Therefore, the level of qualitative complexity in the fuzzy system also increases with the increase of the number of rules.

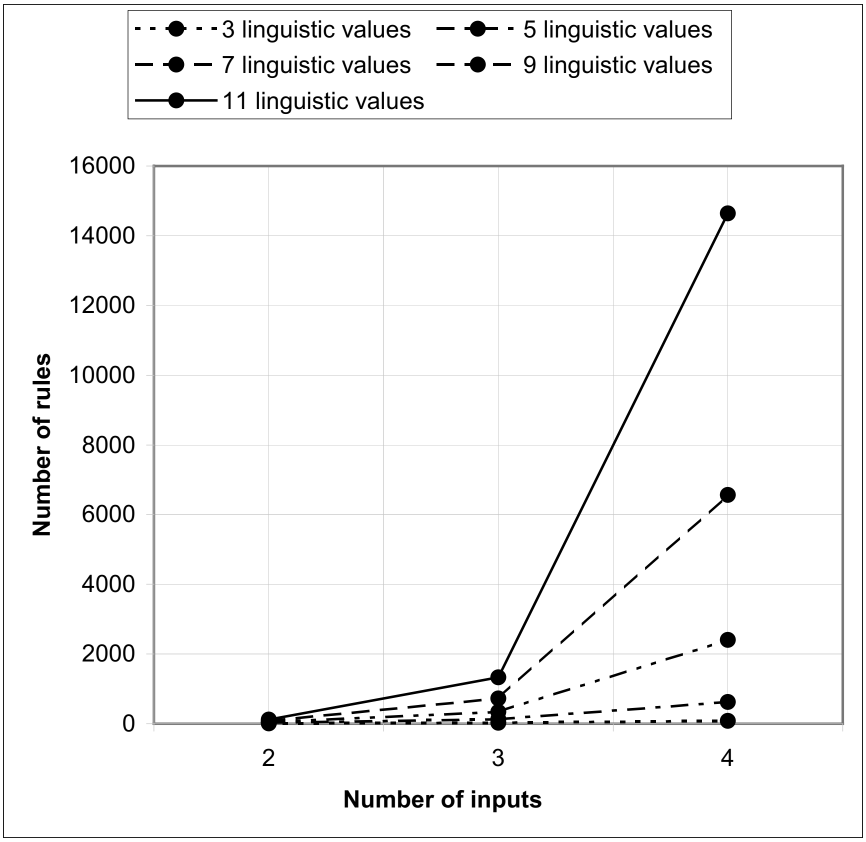

Fig. 2.1. Number of rules as an exponential function of number of inputs

It must be noted that the number of rules in a fuzzy system is only a rough indicator of the quantitative complexity in the system. The exact level of this complexity is a function of the overall amount of fuzzy operations during fuzzification, inference and defuzzification, and this amount itself depends on the number of rules. However, for the purpose of reducing the quantitative complexity in a fuzzy system, it may be sufficient to reduce the number of fuzzy rules without analysing the precise impact of this reduction on the amount of the associated fuzzy operations. In this case, the fuzzy system with the initial number of rules is usually referred to as *initial fuzzy system* (IFS) whereas the fuzzy system with the reduced number of rules is called *reduced fuzzy system* (RFS).

Most of the known methods for complexity reduction in fuzzy systems reduce the number of fuzzy rules by either reducing the number of inputs or the number of linguistic values that these inputs can take. These methods are classified into several groups and are discussed in the next chapter.

3 Rule Base Reduction Methods for Fuzzy Systems

3.1 Removal and Merging of Linguistic Values

The first group of methods for rule base reduction are aimed at removing less significant or merging similar linguistic values [67, 68]. For simplicity, this group of methods and all other methods in the current chapter will be illustrated by integer tables such that the linguistic values of the inputs are coded by integers whereas the outputs are either skipped as irrelevant or presented in a general form.

For example, if the IFS is described by the two inputs i_1 and i_2, and if each of these inputs can take the three linguistic values *small* (S), *medium* (M) and *big* (B), we may decide to remove the value M in which case the values S and B will cover all crisp input values that would otherwise be classified as M. So, if S = 1, M = 2 and B = 3 in the IFS, and if S = 1 and B=3 in the RFS, then these two systems will be illustrated in Figs. 3.1–3.2 and the antecedent parts of their rule bases will be represented by Tables 3.1–3.2.

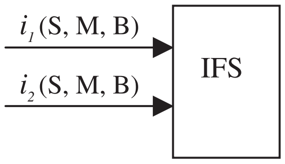

Fig. 3.1. Initial fuzzy system

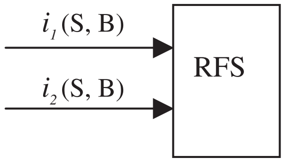

Fig. 3.2. Reduced fuzzy system after removal of linguistic value M

Table 3.1. Initial fuzzy system

Rule number	Linguistic value of i_1	Linguistic value of i_2
1	1	1
2	1	2
3	1	3
4	2	1
5	2	2
6	2	3
7	3	1
8	3	2
9	3	3

Table 3.2. Reduced fuzzy system after removal of linguistic value M = 2

Rule number	Linguistic value of i_1	Linguistic value of i_2
1	1	1
2	1	3
3	3	1
4	3	3

Alternatively, we may decide to merge some of the linguistic values from the IFS in Table 3.1 into new linguistic values, e.g. the values M and B can be merged into a new value called *between medium and big* (MB). So, if S = 1 and MB = 4 in the RFS, then this system will be illustrated in Fig. 3.3 and the antecedent part of its rule base will be represented by Table 3.3.

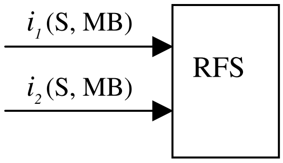

Fig. 3.3. Reduced fuzzy system after merging of linguistic values M and B into MB

Table 3.3. Reduced fuzzy system after merging of linguistic values M and B into MB = 4

Rule number	Linguistic value of i_1	Linguistic value of i_2
1	1	1
2	1	4
3	4	1
4	4	4

From the two methods presented above, the one based on removal of linguistic values is more straightforward but it involves a higher risk as a result of the removal of the corresponding fuzzy set for each of the removed linguistic values. On the other hand, the method based on merging of linguistic values is more difficult to apply due to the necessity to define a new fuzzy set for each of the merged linguistic values but it is less risky.

As a whole, the process of removing and merging of linguistic values is usually associated with loss or aggregation of information. In other words, although the number of rules in the IFS can be substantially reduced, this reduction usually comes with a price because the RFS may not represent adequately the IFS.

3.2 Removal and Fusion of Inputs

The second group of methods for complexity reduction are aimed at removing less significant or fusing similar inputs [42, 50].

For example, if the IFS is described by the three inputs i_1 (position), i_2 (velocity) and i_3 (acceleration), and if each of these inputs can take the two linguistic values *small* (S) and *big* (B), we may decide to remove input i_3. So, if S = 1and B = 2 in both the IFS and the RFS, then these two systems will be illustrated in Figs. 3.4–3.5 and the antecedent parts of their rule bases will be represented by Tables 3.4–3.5.

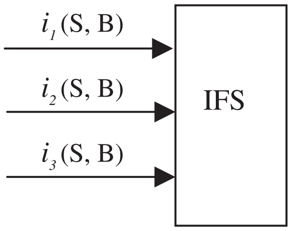

Fig. 3.4. Initial fuzzy system

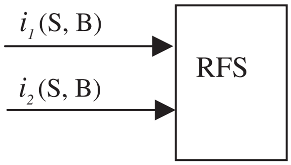

Fig. 3.5. Reduced fuzzy system after removal of input i_3

Table 3.4. Initial fuzzy system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of i_3
1	1	1	1
2	1	1	2
3	1	2	1
4	1	2	2
5	2	1	1
6	2	1	2
7	2	2	1
8	2	2	2

Table 3.5. Reduced fuzzy system after removal of input i_3

Rule number	Linguistic value of i_1	Linguistic value of i_2
1	1	1
2	1	2
3	2	1
4	2	2

Alternatively, we may decide to fuse some of the inputs from the IFS in Table 3.4 into a new input, e.g. the inputs i_2 and i_3 can be fused into a new hybrid input i_4 (velocity/acceleration). In this case, the RFS will be illustrated in Fig. 3.6 and the antecedent part of its rule base will be represented by Table 3.6.

From the two methods presented above, the one based on removal of inputs is more straightforward but it involves a higher risk as a result of the removal of the corresponding variable. On the other hand, the method based

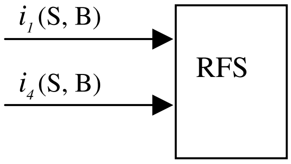

Fig. 3.6. Reduced fuzzy system after fusion of inputs i_2 and i_3 into i_4

Table 3.6. Reduced fuzzy system after fusion of inputs i_2 and i_3 into i_4

Rule number	Linguistic value of i_1	Linguistic value of i_4
1	1	1
2	1	2
3	2	1
4	2	2

on fusion of inputs is more difficult to apply due to the necessity to justify the fusion of particular variables but it is less risky.

As in the case of removal and merging of linguistic values, removal and fusion of inputs is usually associated with loss or aggregation of information. In other words, although the number of rules in the IFS can be substantially reduced, this reduction usually comes with a price because the RFS may not represent adequately the IFS.

3.3 Singular Value Decomposition of Output Matrix

The third group of methods are based on singular value decomposition of the matrix representing the crisp values of the output in a fuzzy system [70, 72, 80]. As a result of this, the number of linguistic values for the inputs in the system is reduced.

For example, if the IFS is described by the two inputs i_1 and i_2, and if each of these inputs can take the five linguistic values *negative big* (NB), *negative small* (NS), *zero* (Z), *positive small* (PS) and *positive big* (PB), then the RFS will have the same inputs but each of them will possibly take only two linguistic values such as *negative* (N) and *positive* (P). So, if NB = 1, NS = 2, Z = 3, PS = 4 and PB = 5 in the IFS, and if N = 6 and P = 7 in the RFS, then these two systems will be illustrated in Figs. 3.7–3.9 and the antecedent parts of their rule bases will be represented by Tables 3.7–3.8.

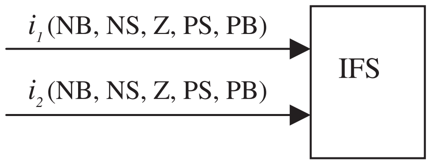

Fig. 3.7. Initial fuzzy system

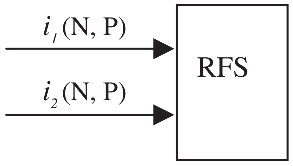

Fig. 3.8. Reduced fuzzy system by singular value decomposition with linguistic values N and P

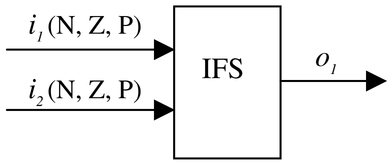

Fig. 3.9. Initial fuzzy system

Table 3.7. Initial fuzzy system

Rule number	Linguistic value of i_1	Linguistic value of i_2
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	2	1
7	2	2
8	2	3
9	2	4
10	2	5
11	3	1
12	3	2
13	3	3
14	3	4
15	3	5
16	4	1
17	4	2
18	4	3
19	4	4
20	4	5
21	5	1
22	5	2
23	5	3
24	5	4
25	5	5

Table 3.8. Reduced fuzzy system by singular value decomposition with linguistic values N = 6 and P = 7

Rule number	Linguistic value of i_1	Linguistic value of i_2
1	6	6
2	6	7
3	7	6
4	7	7

Although this group of methods can be very effective in reducing the number of rules in a fuzzy system, they are applicable mainly for systems with two inputs. In the case of more than two inputs, the singular value decomposition procedure becomes quite complex as the dimension of the space in which the associated matrix is defined increases accordingly. For example, for a system with three inputs, the associated matrix will be defined in a three-dimensional space.

Even in the case of two inputs, the RFS only approximates the behaviour of the IFS and the quality of this approximation can not be guaranteed in advance. Therefore, additional analysis must be carried out to ensure that the RFS is a good approximation of the IFS.

3.4 Conversion into Union Rule Configuration

The fourth group of methods are based on converting the intersection rule configuration of a fuzzy system into a union rule configuration with a smaller number of rules [12, 56, 73]. In this case, the behaviour of the RFS is similar to the one of the IFS.

For example, if the IFS is described by the two inputs i_1, i_2 and the output o_1, and if its rule base is in the intersection rule configuration (i_1 *and* i_2) *then* o_1, then it may be possible to convert this configuration into the union rule configuration *if* (i_1 *then* o_1) *or if* (i_2 *then* o_1). So, if each of the inputs can take the three linguistic values *negative* (N), *zero* (Z) and *positive* (P), and if N = 1, Z = 2 and P = 3 for both the IFS and the RFS, then these two systems will be illustrated in Figs. 3.9–3.10 and their rule bases will be represented by Tables 3.9–3.11.

As in the case of singular value decomposition, the conversion into union rule configuration can be very effective in reducing the number of rules in the IFS. However, this type of conversion can only be applied to a special class of systems called ‘additively separable’. But even in this case, the RFS only approximates the behaviour of the IFS and the quality of this approximation can not be guaranteed in advance. For this reason, additional analysis must be carried out to ensure that the RFS is a good approximation of the IFS.

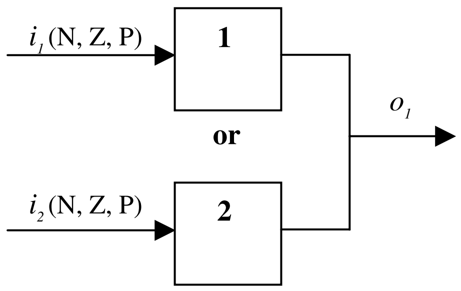

Fig. 3.10. First and second term in the reduced fuzzy system with union rule configuration

Table 3.9. Initial fuzzy system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguisticvalue of o_1
1	1	1	$v_{o1,1}$
2	1	2	$v_{o1,2}$
3	1	3	$v_{o1,3}$
4	2	1	$v_{o1,4}$
5	2	2	$v_{o1,5}$
6	2	3	$v_{o1,6}$
7	3	1	$v_{o1,7}$
8	3	2	$v_{o1,8}$
9	3	3	$v_{o1,9}$

Table 3.10. First term in the reduced fuzzy system with union rule configuration

Rule number	Linguistic value of i_1	Linguistic value of o_1
1	1	$v_{o1,1}$
2	2	$v_{o1,2}$
3	3	$v_{o1,3}$

Table 3.11. Second term in the reduced fuzzy system with union rule configuration

Rule number	Linguistic value of i_2	Linguistic value of o_1
1	1	$v_{o1,1}$
2	2	$v_{o1,2}$
3	3	$v_{o1,3}$

3.5 Spatial Decomposition into Subsystems

The fifth group of methods convert a fuzzy system into spatially decomposed subsystems as a result of which the overall number of rules is reduced [6, 9, 10, 17, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 39, 49, 54, 57, 63, 65, 71, 74, 78, 79, 83, 84]. In this case, the stronger interactions among the subsystems are usually partially compensated whereas the weaker ones are ignored. Depending on the decomposition approach used, the resulting decomposed system has a distributed, decentralised, decoupled or multilevel structure.

For example, if the IFS is described by the two inputs i_1, i_2 and the two output o_1, o_2, then this system can be decomposed into two subsystems such that i_1 and o_1 are the input and the output for the first subsystem whereas i_2 and o_2 are the input and the output for the second subsystem. So, if each of the inputs can take the three linguistic values *negative* (N), *zero* (Z) and *positive* (P), and if N = 1, Z = 2 and P = 3 for both the IFS and the RFS, then these two systems will be illustrated in Figs. 3.11–3.12 and their rule bases will be represented by Tables 3.12–3.14.

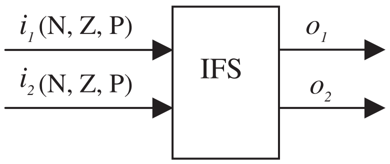

Fig. 3.11. Initial fuzzy system

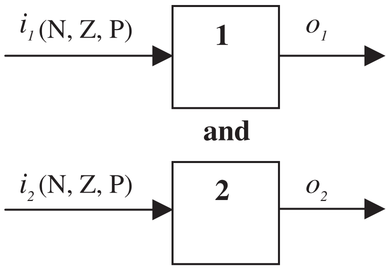

Fig. 3.12. First and second subsystem in the reduced fuzzy system with spatial decomposition

Table 3.12. Initial fuzzy system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of o_1	Linguistic value of o_2
1	1	1	$v_{o1,1}$	$v_{o2,1}$
2	1	2	$v_{o1,2}$	$v_{o2,2}$
3	1	3	$v_{o1,3}$	$v_{o2,3}$
4	2	1	$v_{o1,4}$	$v_{o2,4}$
5	2	2	$v_{o1,5}$	$v_{o2,5}$
6	2	3	$v_{o1,6}$	$v_{o2,6}$
7	3	1	$v_{o1,7}$	$v_{o2,7}$
8	3	2	$v_{o1,8}$	$v_{o2,8}$
9	3	3	$v_{o1,9}$	$v_{o2,9}$

Table 3.13. First subsystem in the reduced fuzzy system with spatial decomposition

Rule number	Linguistic value of i_1	Linguistic value of o_1
1	1	$v_{o1,1}$
2	2	$v_{o1,2}$
3	3	$v_{o1,3}$

Table 3.14. Second subsystem in the reduced fuzzy system with spatial decomposition

Rule number	Linguistic value of i_2	Linguistic value of o_2
1	1	$v_{o2,1}$
2	2	$v_{o2,2}$
3	3	$v_{o2,3}$

Although these methods have been widely used for quite a long time, they have some serious drawbacks. The ability of the RFS to approximate the behaviour of the IFS depends on the strength of the interactions among subsystems and the effectiveness of their compensation, which can not be guaranteed in advance. Therefore, additional analysis must be carried out to ensure that the RFS is a good approximation of the IFS.

3.6 Decomposition into Multilayer Hierarchical Structure

The last group of methods rearrange the inputs in a fuzzy system in a way that leads to the reduction of the number of rules [5, 8, 41, 45, 46, 52, 64, 76, 82]. Actually, the fuzzy system is decomposed into a multilayer hierarchical structure such that each layer has only two inputs and one output.

For example, if the IFS is described by the three inputs i_1, i_2, i_3 and the output o_1, then this system can be decomposed into two layers. In this case, i_1 and i_2 are the inputs to the first layer, a new variable z_1 is an intermediate output from the first layer and an intermediate input to the second layer, i_3 is the other input to the second layer and o_1 is its output. So, if each of the inputs i_1, i_2, i_3 and the intermediate input z_1 can take the three linguistic values *negative* (N), *zero* (Z) and *positive* (P), and if N = 1, Z = 2 and P = 3 for both the IFS and the RFS, then these two systems will be illustrated in Figs. 3.13–3.14 and their rule bases will be represented by Table 3.15–3.17.

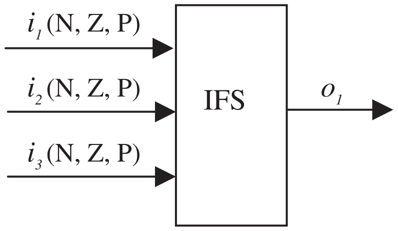

Fig. 3.13. Initial fuzzy system

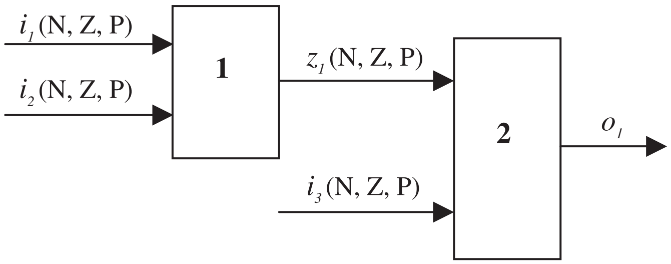

Fig. 3.14. First and second layer in the reduced fuzzy system with multilayer hierarchical structure

Although these methods have become quite popular recently, they have some serious drawbacks such as the unclear interpretation of the intermediate variables and the poor transparency of the rules in the multilayer hierarchical structure. Apart from that, the overall behaviour of the layers in the RFS is only an approximation of the behaviour of the IFS. This is due to the fact that each layer has only two inputs while the other inputs from the IFS are actually ignored. Also, the intermediate variables are usually defuzzified as outputs from one layer and then fuzzified as inputs to the next layer, which leads to further deviation from the behaviour of the IFS. Therefore, additional analysis must be carried out to ensure that the RFS is a good approximation of the IFS.

Table 3.15. Initial fuzzy system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of i_3	Linguistic value of o_1
1	1	1	1	$v_{o1,1}$
2	1	1	2	$v_{o1,2}$
3	1	1	3	$v_{o1,3}$
4	1	2	1	$v_{o1,4}$
5	1	2	2	$v_{o1,5}$
6	1	2	3	$v_{o1,6}$
7	1	3	1	$v_{o1,7}$
8	1	3	2	$v_{o1,8}$
9	1	3	3	$v_{o1,9}$
10	2	1	1	$v_{o1,10}$
11	2	1	2	$v_{o1,11}$
12	2	1	3	$v_{o1,12}$
13	2	2	1	$v_{o1,13}$
14	2	2	2	$v_{o1,14}$
15	2	2	3	$v_{o1,15}$
16	2	3	1	$v_{o1,16}$
17	2	3	2	$v_{o1,17}$
18	2	3	3	$v_{o1,18}$
19	3	1	1	$v_{o1,19}$
20	3	1	2	$v_{o1,20}$
21	3	1	3	$v_{o1,21}$
22	3	2	1	$v_{o1,22}$
23	3	2	2	$v_{o1,23}$
24	3	2	3	$v_{o1,24}$
25	3	3	1	$v_{o1,25}$
26	3	3	2	$v_{o1,26}$
27	3	3	3	$v_{o1,27}$

Table 3.16. First layer in the reduced fuzzy system with multilayer hierarchical structure

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of z_1
1	1	1	$v_{z1,1}$
2	1	2	$v_{z1,2}$
3	1	3	$v_{z1,3}$
4	2	1	$v_{z1,4}$
5	2	2	$v_{z1,5}$
6	2	3	$v_{z1,6}$
7	3	1	$v_{z1,7}$
8	3	2	$v_{z1,8}$
9	3	3	$v_{z1,9}$

Table 3.17. Second layer in the reduced fuzzy system with multilayer hierarchical structure

Rule number	Linguistic value of z_1	Linguistic value of i_3	Linguistic value of o_1
1	1	1	$v_{o1,1}$
2	1	2	$v_{o1,2}$
3	1	3	$v_{o1,3}$
4	2	1	$v_{o1,4}$
5	2	2	$v_{o1,5}$
6	2	3	$v_{o1,6}$
7	3	1	$v_{o1,7}$
8	3	2	$v_{o1,8}$
9	3	3	$v_{o1,9}$

3.7 Comparative Analysis of Reduction Methods

Most of the known methods for rule base reduction in fuzzy systems have serious drawbacks such as empirical nature, limited scope and approximated behaviour of the IFS. A brief comparison of these methods in terms of their attributes is given in Table 3.18.

The empirical nature of most of the methods assumes the use of a ‘trial and error’ approach. The latter is generally unreliable in that the quality of the RFS depends mainly on the good or the bad luck during the trials. Besides this, the limited scope of some of the methods makes them inapplicable to a large group of fuzzy systems. Also, the approximation of the behaviour of the IFS usually compromises the overall quality of the RFS. And finally, although all these methods reduce the quantitative complexity of the IFS, they do not reduce its qualitative complexity.

Table 3.18. Comparison of rule base reduction methods

Method/Attribute	Nature	Scope	Behaviour	Quantitative complexity	Qualitative complexity
First Group	empirical	universal	approximate	reduced	unaffected
Second Group	empirical	universal	approximate	reduced	unaffected
Third Group	systematic	limited	approximate	reduced	unaffected
Fourth Group	systematic	limited	approximate	reduced	unaffected
Fifth Group	empirical	universal	approximate	reduced	unaffected
Sixth Group	empirical	universal	approximate	reduced	unaffected

Obviously, it would be ideal to find a universal and systematic approach to simplifying the IFS, which also guarantees that the behaviour of the simplified system is equivalent to the one of the IFS. The main advantages of this novel approach in comparison to most of the known rule based reduction methods would be its wide applicability irrespective of the properties of the IFS, its capability to lend itself easily to mathematical formalisation and its ability to guarantee that the simplification made to the IFS does not come at a price. On top of that, the quantitative and the qualitative complexity of the IFS would be reduced. A brief description of the attributes of such an approach is shown in Table 3.19.

Table 3.19. Attributes of the novel approach

Approach/ Attribute	Nature	Scope	Behaviour	Quantitative complexity	Qualitative complexity
Novel	systematic	universal	equivalent	reduced	reduced

The underlying philosophy of this novel approach deals with complexity related issues in fuzzy systems not only by reducing the number of fuzzy rules which has an impact mainly on the quantitative complexity of the fuzzy system but from a much wider perspective. This perspective takes into account factors that contribute to the qualitative complexity of the fuzzy system, e.g. the way in which the rules are handled. For this reason, the more general term ‘complexity management’ is used here instead of the relatively specific term ‘complexity reduction’.

In summary, the potential advantages of such an approach in comparison to the known rule base reduction methods are that:

- it would not require any underlying knowledge about the real system that is modelled by the fuzzy system,
- it would guarantee that the behaviour of the RFS is not approximate but equivalent to the behaviour of the IFS,

- it would be applicable to a fuzzy system with any number of inputs/outputs and any type of linguistic values that these inputs/outputs can take,
- it would be based on systematic rather than empirical considerations that justify the approach and guarantee a successful result,
- it would use a simple formal model that is easy to use by people with different mathematical background,
- it would be applicable to a fuzzy system with any type of rule base in terms of the CON and DIS properties of the antecedent terms and the rules,
- it would be applicable for a variable number and meaning of the linguistic values of the inputs and the outputs.

In order to develop such an approach, we need to be able to formally present fuzzy systems and their rule bases. Moreover, such an approach would be in line with the recent trend of using formal methods for fuzzy modelling and control [16, 24]. However, the integer tables used in this chapter are quite simplistic and not suitable for formal handling. For this reason, the next chapter introduces more advanced techniques for formal presentation of fuzzy rule based systems.

4 Formal Presentation of Fuzzy Rule Based Systems

4.1 Basic Properties of Fuzzy Rule Bases

The management of complexity in fuzzy systems relies on the basic properties of fuzzy rule bases. These properties reflect the extent to which of the permutations of linguistic values of inputs and outputs are available as well as the type of mapping between the permutations of linguistic values of the inputs in the antecedent part of the rule base and the corresponding permutations of linguistic values of the outputs in the consequent part.

Four basic properties of fuzzy rule bases are introduced here. Definitions for these properties and examples illustrating them are given below. For consistency with the notation used in the previous chapter, the formal presentation of the properties and the corresponding examples are made initially by integer tables. However, these same properties and examples are represented by more advanced formal techniques in Sects. 4.3–4.4.

Definition 4.1
A fuzzy rule base is complete if and only if all possible permutations of linguistic values of the inputs are present in the antecedent part of the rule base.

Example 4.1
A fuzzy system is described by the two inputs i_1, i_2 and the two outputs o_1, o_2. Each of the inputs can take the two linguistic values *small* (S) and *big* (B), whereas each of the outputs can take the three linguistic values *negative* (N), *zero* (Z) and *positive* (P). So, if S = 1, B = 2, N = 1, Z = 2 and P = 3, then a complete rule base for this system will be represented by Table 4.1.

Table 4.1. Complete rule base for a fuzzy system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of o_1	Linguistic value of o_2
1	1	1	1	1
2	1	2	1	2
3	2	1	1	3
4	2	2	2	1

Definition 4.2

A fuzzy rule base is exhaustive if and only if all possible permutations of linguistic values of the outputs are present in the consequent part of the rule base.

Example 4.2

A fuzzy system is described by the two inputs i_1, i_2 and the two outputs o_1, o_2. Each of the inputs can take the three linguistic values *small* (S), *medium* (M) and *big* (B), whereas each of the outputs can take the two linguistic values *negative* (N) and *positive* (P). So, if S = 1, M = 2, B = 3, N = 1 and P = 2, then an exhaustive rule base for this system will be represented by Table 4.2.

Table 4.2. Exhaustive rule base for a fuzzy system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of o_1	Linguistic value of o_2
1	1	1	1	1
2	1	2	1	2
3	1	3	2	1
4	2	1	2	2

Definition 4.3

A fuzzy rule base is consistent if and only if every available permutation of linguistic values of the inputs is mapped onto only one permutation of linguistic values of the outputs.

Example 4.3

A fuzzy system is described by the two inputs i_1, i_2 and the two outputs o_1, o_2. Each of the inputs can take the three linguistic values *small* (S), *medium* (M) and *big* (B), whereas each of the outputs can take the two linguistic values *negative* (N) and *positive* (P). So, if S = 1, M = 2, B = 3, N = 1 and P = 2, then a consistent rule base for this system will be represented by Table 4.3.

Definition 4.4

A fuzzy rule base is monotonic if and only if every available permutation of linguistic values of the outputs is mapped from only one permutation of linguistic values of the inputs.

Table 4.3. Consistent rule base for a fuzzy system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of o_1	Linguistic value of o_2
1	1	1	1	1
2	1	2	1	1
3	1	3	1	1
4	2	1	1	2
5	2	2	1	2
6	2	3	2	1
7	3	1	2	1
8	3	2	2	2
9	3	3	2	2

Example 4.4

A fuzzy system is described by the two inputs i_1, i_2 and the two outputs o_1, o_2. Each of the inputs can take the two linguistic values *small* (S) and *big* (B) whereas each of the outputs can take the three linguistic values *negative* (N), *zero* (Z) and *positive* (P). So, if S = 1, B = 2, N = 1, Z = 2 and P = 3, then a monotonic rule base for this system will be represented by Table 4.4.

Table 4.4. Monotonic rule base for a fuzzy system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of o_1	Linguistic value of o_2
1	1	1	1	1
2	1	1	1	2
3	1	1	1	3
4	1	2	2	1
5	1	2	2	2
6	2	1	2	3
7	2	1	3	1
8	2	2	3	2
9	2	2	3	3

If a fuzzy rule base does not have some of the four basic properties, then some or more of Definitions 4.1–4.4 will not hold. In this case, the fuzzy rule base will have some of the corresponding dual properties, which are described by the definitions below.

Definition 4.5

A fuzzy rule base is incomplete if and only if at least one of all possible permutations of linguistic values of the inputs is missing from the antecedent part of the rule base.

Definition 4.6
A fuzzy rule base is non-exhaustive if and only if at least one of all possible permutations of linguistic values of the outputs is missing from the consequent part of the rule base.

Definition 4.7
A fuzzy rule base is inconsistent if and only if at least one of all available permutations of linguistic values of the inputs is mapped onto more than one permutation of linguistic values of the outputs.

Definition 4.8
A fuzzy rule base is non-monotonic if and only if at least one of all available permutations of linguistic values of the outputs is mapped from more than one permutation of linguistic values of the inputs.

4.2 Analysis of Rule Base Properties

There is an obvious duality between Definitions 4.1–4.4 and Definitions 4.5–4.8. While each definition from the first four ones describes a fuzzy rule base that has a specific property, each definition from the last four ones describes a fuzzy rule base that does not have this property.

For completeness, a fuzzy rule base must be described by its full property status which shows which of the four basic properties are available using the Boolean values *true* (T) and *false* (F). Such a description for the four fuzzy rule bases from Sect. 4.1 is given in Table 4.5.

Table 4.5. Full property status for fuzzy rule bases represented by integer tables

Fuzzy rule base/Property	Complete	Exhaustive	Consistent	Monotonic
Table 4.1	T	F	T	T
Table 4.2	F	T	T	T
Table 4.3	T	T	T	F
Table 4.4	T	T	F	T

The full property status in Table 4.5 reveals the impact of some changes in the input-output mappings of the fuzzy rule bases on their properties. For example, the integer table in Table 4.2 is an inverse image of the integer table in Table 4.1 whereby the Boolean values for completeness and exhaustiveness in the rule base from the first table are actually inverted with respect to the values in the rule base from the second table. Similarly, the integer table in Table 4.4 is an inverse image of the integer table in Table 4.3 whereby the Boolean values for consistency and monotonousness in the rule base from the first table are inverted with respect to the ones in the rule base from the second table.

It is useful to know to what extent a fuzzy rule base is likely to have each of the four basic properties. Usually, a fuzzy rule base is expected to be complete, i.e. with all possible permutations of linguistic values of inputs available, although that may not always be the case. As far as the permutations of linguistic values of the outputs are concerned, it is fairly common for some of them to be missing and therefore a fuzzy rule base is likely to be non-exhaustive. Ideally, a fuzzy rule base must be consistent, i.e. with each available permutation of linguistic values of inputs yielding only one permutation of linguistic values of outputs. And finally, it is quite common for some permutations of linguistic values of outputs to be yielded by more than one permutation of linguistic values of inputs and therefore a fuzzy rule base is likely to be non-monotonic.

Theoretically speaking, there are 16 possible permutations of Boolean values of properties for a fuzzy rule bases but not all of these permutations are equally desirable. If the level of desirability for the permutations is described by the linguistic values *low* (L), *medium* (M) and *high* (H), then these permutations can be represented by Table 4.6.

Table 4.6. Permutations of properties for a fuzzy rule base

Permutation/ Property	Complete	Exhaustive	Consistent	Monotonic	Desirability
1	T	T	T	T	H
2	T	T	T	F	H
3	T	T	F	T	L
4	T	T	F	F	L
5	T	F	T	T	H
6	T	F	T	F	H
7	T	F	F	T	L
8	T	F	F	F	L
9	F	T	T	T	M
10	F	T	T	F	M
11	F	T	F	T	L
12	F	T	F	F	L
13	F	F	T	T	M
14	F	F	T	F	M
15	F	F	F	T	L
16	F	F	F	F	L

It is worth noting that some of the permutations of Boolean values of properties in Table 4.6 may not be possible for some fuzzy rule bases. For example, if a fuzzy system has two inputs and one output whereby each of them can take three linguistic values, then permutation 1 will be impossible, i.e. the rule base can not be complete, exhaustive, consistent and monotonic at the same time. In particular, if the rule base is complete, it will have 9 rules and therefore some of the linguistic values for the output will appear more than once, i.e. the rule base will be non-monotonic. On the other hand, if the rule base is monotonic with 3 rules such that each linguistic value for the output appears only once, then 6 permutations of linguistic values for the inputs will be missing and therefore the rule base will be incomplete.

It is obvious from Table 4.6 that a complete and consistent fuzzy rule base is very desirable not only because it provides detailed information about the fuzzy system by means of all possible permutations of linguistic values of inputs but also because its inference mechanism is non-contradictory (see permutations 1-2, 5-6). An incomplete but consistent rule base is still desirable due to its non-contradictory inference mechanism although it does not provide detailed information about the fuzzy system (see permutations 9-10, 13-14). However, an inconsistent fuzzy rule base is not desirable irrespective of whether it is complete or not because its inference mechanism is contradictory (see permutations 3-4, 7-8, 11-12, 15-16).

A deeper analysis of Table 4.6 shows that the permutations with high level of desirability are the ones that correspond to a complete and consistent fuzzy rule base. These properties relate to the antecedent part of the rule base over which we usually have some control. In this respect, we can make additional observations on the inputs in order to add the rules with the missing permutations of linguistic values and to make the rule base complete. Also, we can make additional observations on the outputs in order to remove the redundant rules with the same antecedent part and to make the rule base consistent.

Therefore, if a fuzzy rule base is initially in a 'low' property status, we should be able to achieve a 'high' status by means of additional observations on the inputs and the outputs. However, if a 'high' property status is not achievable from a 'low' status for some reason, e.g. due to time constraints on the additional observations, then we should be able to achieve at least a 'medium' status by means of additional observations on the outputs. Obviously, if a fuzzy rule base is initially in a 'medium' property status, then we should be able to achieve a 'high' status by means of additional observations on the inputs.

The required types of additional observations for achieving transitions from one property status to another for fuzzy rule bases are shown in Table 4.7. In this case, a transition is desirable only if it is from a lower to a higher property status although such a transition may not always be possible due to inability to make sufficient additional observations. On the other hand, a transition from a higher to a lower property status is always possible, e.g. it is easy to remove rules or add contradictory rules, but such a transition is undesirable as it actually deteriorates the inference mechanism of the fuzzy system.

Table 4.7. Transitions from one property status to another for fuzzy rule bases

Property status	To low	To medium	To high
From low	–	output observations	input and output observations
From medium	undesirable	–	input observations
From high	undesirable	undesirable	–

The considerations made above show that if a fuzzy rule base is not in a 'high' or at least in a 'medium' property status, we are very likely to be able to achieve a higher status using additional observations on the inputs or the outputs. Such a transition will improve the inference mechanism of the fuzzy system.

4.3 Presentation of Rule Bases by Boolean Matrices

Although integer tables can be quite useful for formal presentation of fuzzy rule bases in relation to their properties, they also have some serious drawbacks. For example, they may contain repetitive information such as rules with the same antecedent parts in inconsistent rule bases or rules with the same consequent parts in non-monotonic rule bases. Also, as integer tables are a bit rigid, they do not appear to be suitable for formal manipulation of fuzzy rule bases, e.g. during a transition to a higher property status. And finally, it may be quite difficult and time consuming to define the property status by means of integer tables, especially in the case of large fuzzy rule bases.

Obviously, more advanced techniques for formal presentation of fuzzy rule bases would be quite helpful. For this purpose, one such technique is introduced here. This technique is based on Boolean matrices that have been thoroughly studied by mathematicians and applied by engineers in a number of areas such as transport networks and communication networks [47]. Some basic definitions for Boolean matrices are given below.

Definition 4.9
An $m \times n$ Boolean matrix is a matrix with m rows and n columns whose elements can take only the values 0 and 1.

Definition 4.10
An $m \times n$ null Boolean matrix is a matrix with m rows and n columns all of whose elements are equal to 0.

Definition 4.11
An $m \times n$ universal Boolean matrix is a matrix with m rows and n columns all of whose elements are equal to 1.

Definition 4.12
A Boolean matrix is square if and only if the number of its rows is equal to the number of its columns.

Definition 4.13
A Boolean matrix is homogenous if and only if its row and column labels are of the same type.

Definition 4.14
An element in a Boolean square matrix is on-diagonal if and only if its row and column index are the same.

Definition 4.15
An element in a Boolean square matrix is off-diagonal if and only if its row and column index are different.

Definition 4.16
An identity Boolean matrix is a square homogenous Boolean matrix all of whose on-diagonal elements are equal to 1 and all of whose off-diagonal elements are equal 0.

The basic operations that can be applied to elements of Boolean matrices are 'addition' and 'multiplication'. They are both binary operations as they can only be applied to two operands. In the case of more than two elements, each of the two operations can be applied in a sequential manner, i.e. step by step, whereby only two elements are considered at each step and the result from the step becomes an operand in the next step.

The 'addition' operation has the effect of taking the 'maximum' of the elements whereas the 'multiplication' operation has the effect of taking the 'minimum' of the elements. Both operations are commutative, i.e. the result is not affected if the positions of the two elements are swapped.

In terms of the values of the first and the second element, there are four different permutations for the 'addition' operation which are described by the following equations:

$$1 + 1 = max\,(1, 1) = 1 \tag{4.1}$$

$$0 + 1 = max\,(0, 1) = 1 \tag{4.2}$$

$$1 + 0 = max\,(1, 0) = 1 \tag{4.3}$$

$$0 + 0 = max\,(0, 0) = 0 \tag{4.4}$$

Similarly, there are four different permutations for the 'multiplication' operation which are described by the following equations:

$$1\,.\,1 = min\,(1, 1) = 1 \tag{4.5}$$

$$0\,.\,1 = min\,(0, 1) = 0 \tag{4.6}$$

$$1\,.\,0 = min\,(1, 0) = 0 \tag{4.7}$$

$$0\,.\,0 = min\,(0, 0) = 0 \tag{4.8}$$

Boolean matrices are multiplied in almost the same way as conventional matrices, i.e. matrices whose elements can take any values. Each element in a Boolean matrix product *A***B* can be obtained by multiplying each row from the first matrix *A* with its counterpart column from the second matrix *B*. In this case, the row index of an element in *A***B* is the same as the index of the corresponding row from the matrix *A*, whereas the column index of an element in *A***B* is the same as the index of the corresponding column from the matrix *B*.

The multiplication compatibility rule for Boolean matrices is the same as the rule for conventional matrices, i.e. the number of columns in the first matrix must be equal to the number of rows in the second matrix. The only difference is that instead of applying the arithmetic 'addition' and 'multiplication' operations on elements of the matrices, we apply the 'maximum' and 'minimum' operations, respectively.

The 'addition' and 'multiplication' operations on elements of Boolean matrices are valid in the context of matrix multiplication only if the two matrices are compatible. Obviously, Boolean matrix multiplication is non-commutative because the swapping of the two matrices may violate the multiplication compatibility rule.

By replacing 'max' with 'or', '0' with 'false' and '1' with 'true' in Eqs. (4.1)–(4.4), the latter can be represented by the following equivalent form:

$$\mathrm{T}\quad or\quad \mathrm{T} = \mathrm{T} \tag{4.9}$$

$$\mathrm{F}\quad or\quad \mathrm{T} = \mathrm{T} \tag{4.10}$$

$$\mathrm{T}\ \mathit{or}\ \mathrm{F}\ = \mathrm{T} \tag{4.11}$$

$$\mathrm{F}\ \mathit{or}\ \mathrm{F} = \mathrm{F} \tag{4.12}$$

Similarly, by replacing ‘min’ with ‘and’, ‘0’ with ‘false’ and ‘1’ with ‘true’ in Eqs. (4.5)–(4.8), the latter can be represented by the following equivalent form:

$$\mathrm{T}\ \mathit{and}\ \mathrm{T}\ = \mathrm{T} \tag{4.13}$$

$$\mathrm{F}\ \mathit{and}\ \mathrm{T}\ = \mathrm{F} \tag{4.14}$$

$$\mathrm{T}\ \mathit{and}\ \mathrm{F}\ = \mathrm{F} \tag{4.15}$$

$$\mathrm{F}\ \mathit{and}\ \mathrm{F} = \mathrm{F} \tag{4.16}$$

Obviously, there exists a clear duality between Eqs. (4.1)–(4.8) and Eqs. (4.9)–(4.16). The first group of equations is based on a set theoretic approach to operations on elements in Boolean matrices while the second group of equations is based on a Boolean logic approach. This specific type of duality is a reflection of the general type of duality that is known to exist between set theory and Boolean logic.

The above duality facilitates the manipulation and the interpretation of fuzzy rule bases, which are presented formally. For example, a set theoretic based presentation of a fuzzy rule base can be easily converted into an equivalent Boolean logic based presentation if this conversion is expected to improve the transparency of the fuzzy rules. In other cases, the opposite type of conversion may be required, i.e. from a Boolean logic based presentation into a set theoretic based presentation.

The process of presenting a fuzzy rule base by a Boolean matrix is almost straightforward when the rule base is already presented by an integer table. The following algorithm gives a step by step description of the process of converting an integer table into a Boolean matrix:

Algorithm 4.1

1. Sort all possible permutations of linguistic values of inputs from the integer table in an ascending order.
2. Sort all possible permutations of linguistic values of outputs from the integer table in an ascending order.
3. Label the rows of the Boolean matrix with the sorted permutations of linguistic values of inputs.

4. Label the columns of the Boolean matrix with the sorted permutations of linguistic values of outputs.
5. Go through all the elements of the Boolean matrix and set each element equal to 1 or 0 using steps 6 and 7.
6. If an element of the Boolean matrix reflects an existing mapping from an input onto an output permutation, set it equal to 1.
7. If an element of the Boolean matrix reflects a non-existing mapping from an input onto an output permutation, set it equal to 0.

By means of Algorithm 4.1, the four examples of fuzzy rule bases presented by the integer tables in Tables 4.1–4.4 can be presented equivalently by the Boolean matrices in Eqs. (4.17)–(4.20), respectively.

$$\begin{array}{cccccccccc}
\text{Inputs/Outputs} & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\
11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
12 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
21 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
22 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{array} \tag{4.17}$$

$$\begin{array}{ccccc}
\text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\
11 & 1 & 0 & 0 & 0 \\
12 & 0 & 1 & 0 & 0 \\
13 & 0 & 0 & 1 & 0 \\
21 & 0 & 0 & 0 & 1 \\
22 & 0 & 0 & 0 & 0 \\
23 & 0 & 0 & 0 & 0 \\
31 & 0 & 0 & 0 & 0 \\
32 & 0 & 0 & 0 & 0 \\
33 & 0 & 0 & 0 & 0
\end{array} \tag{4.18}$$

$$\begin{array}{ccccc}
\text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\
11 & 1 & 0 & 0 & 0 \\
12 & 1 & 0 & 0 & 0 \\
13 & 1 & 0 & 0 & 0 \\
21 & 0 & 1 & 0 & 0 \\
22 & 0 & 1 & 0 & 0 \\
23 & 0 & 0 & 1 & 0 \\
31 & 0 & 0 & 1 & 0 \\
32 & 0 & 0 & 0 & 1 \\
33 & 0 & 0 & 0 & 1
\end{array} \tag{4.19}$$

$$\begin{array}{cccccccccc}
\text{Inputs/Outputs} & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\
11 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
12 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
21 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
22 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{array} \tag{4.20}$$

The full property status of the four fuzzy rule bases described by the integer tables in Tables 4.1– 4.4 was given earlier in Table 4.5. As the conversion of an integer table into a Boolean matrix does not change the basic properties and the associated property status of the fuzzy rule base, we can use Table 4.5 to represent the same rule base by only changing the row labels appropriately. In this case, the table numbers in Table 4.5 will be replaced by the corresponding equation numbers, as shown in Table 4.8.

Table 4.8. Full property status for fuzzy rule bases represented by Boolean matrices

Fuzzy rule base / Property	Complete	Exhaustive	Consistent	Monotonic
Equation 4.17	T	F	T	T
Equation 4.18	F	T	T	T
Equation 4.19	T	T	T	F
Equation 4.20	T	T	F	T

As in the case of integer tables, the full property status in Table 4.8 reveals the impact of some changes in the input-output mappings of the fuzzy rule bases on their properties. For example, the Boolean matrix in Eq. (4.18) is the transpose of the Boolean matrix in Eq. (4.17) whereby the Boolean values for completeness and exhaustiveness for the rule base from the second equation are actually inverted with respect to the values for the rule base from the first equation. Similarly, the Boolean matrix in Eq. (4.20) is the transpose of the Boolean matrix in Eq. (4.19) whereby the Boolean values for consistency and monotonousness in the rule base from the second equation are inverted with respect to the corresponding values for the rule base from the first equation.

The conversion of an integer table into a Boolean matrix does not affect the permutations of properties and the transitions from one property status to another for the associated fuzzy rule base. Therefore, Tables 4.6–4.7 are valid for any fuzzy rule base irrespective of whether it is presented by an integer table or a Boolean matrix.

One of the advantages of Boolean matrices with respect to integer tables is that it is very easy to define the properties of the associated fuzzy rule base on the basis of the properties of the corresponding matrix. This is so because the properties of a fuzzy rule base can be implied directly from some properties of the corresponding matrix such as the number of non-zero elements in its rows and columns. The following definitions show how this works.

Definition 4.17
A fuzzy rule base is complete if and only if each row in its Boolean matrix contains at least one non-zero element.

Definition 4.18
A fuzzy rule base is incomplete if and only if at least one row in its Boolean matrix contains only zero elements.

Definition 4.19
A fuzzy rule base is exhaustive if and only if each column in its Boolean matrix contains at least one non-zero element.

Definition 4.20
A fuzzy rule base is non-exhaustive if and only if at least one column in its Boolean matrix contains only zero elements.

Definition 4.21
A fuzzy rule base is consistent if and only if each row in its Boolean matrix contains not more than one non-zero element.

Definition 4.22
A fuzzy rule base is inconsistent if and only if at least one row in its Boolean matrix contains more than one non-zero element.

Definition 4.23
A fuzzy rule base is monotonic if and only if each column in its Boolean matrix contains not more than one non-zero element.

Definition 4.24
A fuzzy rule base is non-monotonic if and only if at least one column in its Boolean matrix contains more than one non-zero element.

The validity of the definitions above can be checked by examining Table 4.8 and Eqs. (4.17)–(4.20). Obviously, the contents of Table 4.8 can be inferred much more easily from the properties of the Boolean matrices in Eqs. (4.17)–(4.20) than from the structure of the corresponding integer tables in Tables 4.1–4.4. This is so because Boolean matrices compress the information from integer tables, e.g. the matrix labels do not duplicate permutations of linguistic values of inputs in inconsistent rule bases and permutations of linguistic values of outputs in non-monotonic rule bases.

Also, scanning elements in a Boolean matrix works much faster and easier than scanning permutations in integer tables.

4.4 Presentation of Rule Bases by Binary Relations

Although Boolean matrices are superior to integer tables for formal presentation of fuzzy rule bases, they also have some drawbacks. For example, the row and column labels in these matrices contain all possible permutations of linguistic values of inputs and outputs although some of these labels may be redundant, especially in the case of large fuzzy rule bases. This redundancy becomes even more obvious in large sparse Boolean matrices describing incomplete and non-exhaustive fuzzy rule bases.

Obviously, a more advanced technique for formal presentation of fuzzy rule bases would be useful. Ideally, this technique should combine the advantages of integer tables and Boolean matrices by:

- not containing redundant row and column labels for the permutations of linguistic values of inputs in inconsistent rule bases and the permutations of linguistic values of outputs in non-monotonic rule bases,
- not containing redundant rows and columns for the permutations of linguistic values of inputs and outputs in incomplete and non-exhaustive fuzzy rule bases,
- allowing fast and easy scanning of elements for the purpose of defining the properties of the associated fuzzy rule base.

One such technique is introduced here. The technique is based on binary relations that have been thoroughly studied by mathematicians and used by engineers in a number of application areas [47]. Some basic definitions for binary relations and other related terms are given below.

Definition 4.25
An m set is a collection of m distinct objects of the same type with arbitrary ordering.

By 'objects' we mean items, numbers, permutations of linguistic values, etc. By 'distinct' we mean that each object can appear not more than once in a set. By the 'same type' we mean that all objects in a set must be of the same nature, i.e. only items, only numbers, only permutations of linguistic values, etc. By 'arbitrary ordering' we mean that the order of the objects in a set does not matter, i.e. the changing of the ordering of the objects in a set does not change the set itself. By 'collection' we mean that a set is expected to contain at least one object although in some cases the collection could be empty.

Definition 4.26
A null set is an empty collection of objects.

Very often, the term 'element' is used when we make a reference to an object in a set related context. In this case, an element is either a member or not a member of the set, i.e. it either belongs to or does not belong to the set.

Definition 4.27
An $m \times n$ binary relation is a set all of whose elements are pairs whereby the first element in each pair belongs to an m set and the second element in the pair belongs to an n set.

Definition 4.28
A null binary relation is an empty collection of pairs of elements.

Definition 4.29
An $m \times n$ universal binary relation is a set that contains all possible pairs of elements from an m set and an n set.

Definition 4.30
A binary relation is square if and only if the number of elements in the two participating sets is equal.

Definition 4.31
A binary relation is homogenous if and only if the two participating sets are of the same type.

Definition 4.32
A pair in a square binary relation is on-diagonal if and only if its individual elements are in the same position in the two participating sets.

Definition 4.33
A pair in a square binary relation is off-diagonal if and only if its individual elements are in different positions in the two participating sets.

Definition 4.34
An identity binary relation is a square homogenous binary relation that contains all the on-diagonal pairs and none of the off-diagonal pairs from the two participating sets.

Usually, the term maplet is used when we make a reference to a pair in a binary relation related context. In this sense, a pair is either a member or not a member of the relation, i.e. it either belongs to or does not belong to the relation.

A subset of a set may contain only elements, which are members of the set. Similarly, a subrelation of a binary relation may contain only maplets, which are members of the binary relation. Therefore, any set is a subset of itself and any binary relation is a subrelation of itself. Also, by definition a null set is a subset of any set and a null binary relation is a subrelation of any binary relation.

Definition 4.35
The domain of a binary relation is the subset of all individual elements from the first participating set in the relation, which belong to maplets in this relation.

Definition 4.36
The range of a binary relation is the subset of all individual elements from the second participating set in the relation, which belong to maplets in this relation.

The basic operations that can be applied to maplets of binary relations are ‘aggregation’ and ‘composition’. They are both binary operations as they can only be applied to two operands. In the case of more than two maplets, each of the two operations can be applied in a sequential manner, i.e. step by step, whereby only two maplets are considered at each step and the result from the step becomes an operand in the next step.

The ‘aggregation’ operation has the effect of replacing two identical maplets standing in parallel with only one of them, i.e. one of the two maplets is removed. The following equation describes the ‘aggregation’ operation:

$$(e_1, e_2) + (e_1, e_2) = (e_1, e_2) \tag{4.21}$$

The ‘composition’ operation has the effect of replacing two maplets standing in a sequence with a new maplet. The ‘composition’ operation has the effect of putting the first element from the first maplet and the second element from the second maplet in a new maplet provided that the second element from the first maplet and the first element from the second maplet are identical.

As opposed to the ‘aggregation’ operation, the ‘composition’ operation is not commutative, i.e. it may not be possible to apply the operation if the positions of the two maplets are swapped. The following equation describes the ‘composition’ operation:

$$(e_1, e_2) \,.\, (e_2, e_3) = (e_1, e_3) \tag{4.22}$$

If we swap the positions of the two maplets in Eq. (4.22), it will not be possible to apply the 'composition' operation and merge these maplets into a new one. In other words, the operation will not have any effect and the two initial maplets will remain unchanged.

The 'aggregation' and 'composition' operations can also be applied when one of the maplets or even both of them are empty. In particular, the result of aggregating or composing two empty maplets is the empty maplet whereas the result of aggregating or composing a non-empty maplet with an empty maplet is either the non-empty maplet or the empty maplet.

The following equations describe all possible cases for the 'aggregation' and the 'composition' operations when at least one of the maplets is empty:

$$(\ ) + (e_1, e_2) = (e_1, e_2) \tag{4.23}$$

$$(e_1, e_2) + (\ ) = (e_1, e_2) \tag{4.24}$$

$$(\ ) + (\ ) = (\ ) \tag{4.25}$$

$$(\ ) \,.\, (e_1, e_2) = (\ ) \tag{4.26}$$

$$(e_1, e_2) \,.\, (\ ) = (\ ) \tag{4.27}$$

$$(\ ) \,.\, (\ ) = (\ ) \tag{4.28}$$

The 'aggregation' and 'composition' operations are the building blocks for the composition of binary relations just as the 'addition' and 'multiplication' operations are the building blocks for the multiplication of Boolean matrices. Each maplet in a binary relation product $A*B$ can be obtained by composing a set of maplets with an identical first element from the first relation A with its counterpart set of maplets with an identical second element from the second relation B. In this case, the first element of a maplet in $A*B$ is the same as the first element of the corresponding maplets from the relation A, whereas the second element of a maplet in $A*B$ is the same as the second element of the corresponding maplets from the relation B.

The composition compatibility rule for binary relations requires the set of second elements in the maplets from the first relation to be equal to the set of first elements in the maplets from the second relation. The 'aggregation' and 'composition' operations on maplets of binary relations are valid in the context of relational composition only if the two relations are compatible. Obviously, binary relational composition is non-commutative because the swapping of the relations may violate the composition compatibility rule.

As an empty maplet reflects a non-existing maplet in a binary relation, it can be represented formally by the Boolean value 'false'. Similarly, a non-empty maplet that reflects an existing maplet in a binary relation can be represented formally by the Boolean value 'true'. As the 'aggregation' operation describes maplets standing in parallel, it can be represented by the Boolean logical disjunction ‘or’. As opposed to this, the ‘composition’ operation which describes maplets standing in sequence can be represented by the Boolean logical conjunction ‘and’.

Therefore, by replacing ‘+’ with ‘or’, ‘.’ with ‘and’, all empty maplets ( ) with ‘false’ and all non-empty maplets (e_1, e_2) with ‘true’ in Eqs. (4.21)–(4.28), the latter can be represented equivalently by Eqs. (4.1)–(4.8) as follows:

- (4.21) by (4.1),
- (4.22) by (4.5),
- (4.23) by (4.2),
- (4.24) by (4.3),
- (4.25) by (4.4),
- (4.26) by (4.6),
- (4.27) by (4.7),
- (4.28) by (4.8).

As in the case of Boolean matrices, the above equivalences show the obvious duality between Eqs. (4.21)–(4.28) and Eqs. (4.9)–(4.16). The first group of equations is based on a set theoretic approach to operations on maplets in binary relations, whereas the second group of equations is based on a Boolean logic approach. This specific type of duality is a reflection of the general type of duality between set theory and Boolean logic, and as already mentioned in Sect. 4.3, it facilitates the manipulation and the interpretation of fuzzy rule bases which are presented formally.

The process of presenting a fuzzy rule base by a binary relation is almost straightforward when the rule base is already represented by an integer table. The following algorithm gives a step by step description of the process of converting an integer table into a binary relation:

Algorithm 4.2

1. Go through all the rows of the integer table from top to bottom by mapping the pair of permutations of linguistic values of inputs and outputs from each row onto a maplet (as described by steps 2 and 3).
2. Make the permutation of linguistic values of inputs in each table row the first element of the maplets corresponding to this row.
3. Make the permutation of linguistic values of outputs in each table row the second element of the maplets corresponding to this row.

4. Generate a binary relation containing all the maplets created in the previous steps of this algorithm.

By means of Algorithm 4.2, the four examples of fuzzy rule bases presented by the integer tables in Tables 4.1-4.4 can be presented equivalently by the binary relations in Eqs. (4.29)–(4.32), respectively.

$$\{(11, 11), (12, 12), (21, 13), (22, 21)\} \tag{4.29}$$

$$\{(11, 11), (12, 12), (13, 21), (21, 22)\} \tag{4.30}$$

$$\{(11, 11), (12, 11), (13, 11), (21, 12), (22, 12), (23, 21), (31, 21), (32, 22), (33, 22)\} \tag{4.31}$$

$$\{(11, 11), (11, 12), (11, 13), (12, 21), (12, 22), (21, 23), (21, 31), (22, 32), (22, 33)\} \tag{4.32}$$

In the binary relations presented by Eqs. (4.28)–(4.32), the maplets are separated by commas and surrounded by a pair of square brackets. In this case, the opening and the closing square bracket mark the beginning and the end of the binary relation, respectively.

As the conversion of an integer table into a binary relation does not change the basic properties and the associated property status of the fuzzy rule base, we can use Table 4.5 to present the same rule base by only changing the row labels appropriately. In this case, the table numbers in Table 4.5 will be replaced by the corresponding equation numbers, as shown in Table 4.9.

Table 4.9. Full property status for fuzzy rule bases represented by binary relations

Fuzzy rule base / Property	Complete	Exhaustive	Consistent	Monotonic
Equation 4.29	T	F	T	T
Equation 4.30	F	T	T	T
Equation 4.31	T	T	T	F
Equation 4.32	T	T	F	T

As in the case of integer tables and Boolean matrices, the full property status in Table 4.9 reveals the impact of some changes in the input-output mappings of the fuzzy rule bases on their properties. For example, the binary relation in Eq. (4.30) is the inverse of the binary relation in Eq. (4.29) whereby the Boolean values for completeness and exhaustiveness of the rule base from the second equation are actually inverted with respect to the corresponding values for the rule base from the first equation. Similarly, the binary relation in Eq. (4.32) is the inverse of the binary relation in Eq. (4.31) whereby the Boolean values for consistency and monotonousness for the rule base from the second equation are inverted with respect to the corresponding values for the rule base from the first equation.

Here again, the conversion of an integer table into a binary relation does not affect the permutations of properties and the transitions from one property status to another for the associated fuzzy rule base. Therefore, Tables 4.6–4.7 are valid for any fuzzy rule base irrespective of whether it is presented by an integer table or a binary relation.

One of the advantages of binary relations with respect to integer tables is that it is very easy to define the properties of the associated fuzzy rule base on the basis of the properties of the corresponding relation. This is so because the properties of a fuzzy rule base can be implied directly from some properties of the corresponding relation such as total, partial, onto, into, one-to-many and many-to-one mappings. The following definitions show how this works.

Definition 4.37

A fuzzy rule base is complete if and only if its binary relation is a total mapping.

Definition 4.38

A fuzzy rule base is incomplete if and only if its binary relation is a partial mapping.

Definition 4.39

A fuzzy rule base is exhaustive if and only if its binary relation is an onto mapping.

Definition 4.40

A fuzzy rule base is non-exhaustive if and only if its binary relation is an into mapping.

Definition 4.41

A fuzzy rule base is consistent if and only if its binary relation does not contain any one-to-many mappings.

Definition 4.42

A fuzzy rule base is inconsistent if and only if its binary relation contains at least one one-to-many mapping.

Definition 4.43

A fuzzy rule base is monotonic if and only if its binary relation does not contain any many-to-one mappings.

Definition 4.44
A fuzzy rule base is non-monotonic if and only if its binary relation contains at least one many-to-one mapping.

The validity of the definitions above can be checked by examining Table 4.9 and Eqs. (4.29)–(4.32). Obviously, the contents of Table 4.9 can be derived more easily from the properties of the binary relations in Eqs. (4.29)–(4.32) than from the structure of the corresponding integer tables in Tables 4.1-4.4. This is so because binary relations compress the information from integer tables, as already explained in the beginning of the current section. Also, scanning maplets in a binary relation works much faster and easier than scanning permutations in integer tables.

4.5 Comparative Analysis of Formal Presentation Techniques

Both Boolean matrices and binary relations are advanced techniques for formal presentation of fuzzy systems, which facilitate the analysis of fuzzy rule bases and improve their transparency. As these techniques are actually two different ways of describing the same rule base, there exists an obvious duality between them. For example, with regard to the analysis of properties of fuzzy rule bases by Boolean matrices and binary relations, Definitions 4.17–4.24 are the dual counterparts of Definitions 4.37–4.44. Also, with regard to the basic operations on elements of Boolean matrices and maplets in binary relations, Eqs. (4.1)–(4.8) are the dual counterparts of Eqs. (4.21)–(4.28).

Boolean matrices and binary relations are superior to integer tables in terms of representing fuzzy rule bases not only because they compress the information held by integer tables but also because they provide more reliable and efficient ways of analysing this information. Binary relations are possibly better than Boolean matrices as they do not contain empty maplets for non-existent input-output mappings of permutations of linguistic values. As opposed to this, zero elements in Boolean matrices reflect such mappings and this represents some redundancy.

Algorithms 4.1–4.2 show how an integer table presenting a fuzzy rule base can be converted into an equivalent formal presentation as a Boolean matrix or a binary relation. In some cases, however, it may be more sensible to obtain a binary relation from a Boolean matrix rather than from an integer table. The following algorithm gives a step by step description of the process of converting a Boolean matrix into a binary relation:

Algorithm 4.3

1. Go through all the elements of the Boolean matrix and map them onto the maplets of the binary relation using steps 2-3.

2. If an element is equal to 1, map it onto a maplet such that the row label of the element is the first element in the maplet and the column label of the element is the second element in the maplet.
3. If an element is equal to 0, do nothing.

By means of Algorithm 4.3, the four examples of fuzzy rule bases represented by the Boolean matrices in Eqs. (4.17)–(4.20) can be represented equivalently by the binary relations in Eqs. (4.29)–(4.32), respectively.

Algorithms 4.1– 4.3 allow us to convert a formal presentation of a fuzzy rule base into a more advanced one, i.e. from an integer table into a Boolean matrix, from an integer table into a binary relation, and from a Boolean matrix into a binary relation. These conversions are summarised in Table 4.10 and illustrated in Fig. 4.1. The conversions facilitate significantly the analysis of rule bases.

Table 4.10. Conversions between formal presentation techniques for fuzzy rule bases

From / To	Integer table	Boolean matrix	Binary relation
Integer table	–	Algorithm 4.1	Algorithm 4.2
Boolean matrix	unnecessary	–	Algorithm 4.3
Binary relation	unnecessary	unnecessary	–

The opposite conversions of the ones described in Table 4.10 are possible but they do not make much sense. This is so because the aim of conversion is to represent a fuzzy rule base with a more advanced formal technique, which would allow us to better analyse the rule base. For this reason, the opposite conversions are indicated in Table 4.10 as unnecessary.

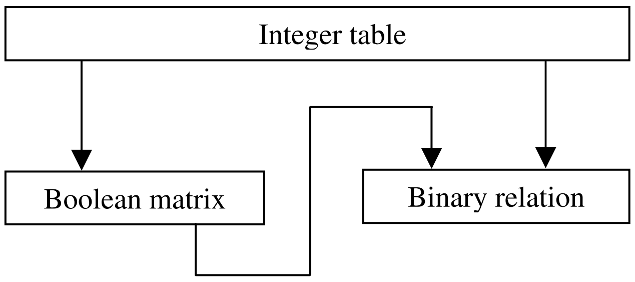

Fig. 4.1. Conversions between formal presentation techniques for fuzzy rule bases

Finally, although binary relations are usually a better formal presentation technique of fuzzy rule bases than Boolean matrices, the latter are underpinned by a matrix theory that appears to be more powerful than the corresponding relational theory. Therefore, Boolean matrices may sometimes be preferred to binary relations as more suitable for a specific task.

4.6 Application Range of Formal Presentation Techniques

The two advanced formal presentation techniques introduced in this chapter are applicable to a wide range of fuzzy rule based systems. These techniques can be applied to Mamdami, Sugeno and Tsukamoto systems, CON and DIS systems, MO and SO systems, FF and FB systems, as well as SRB and MRB systems.

The following paragraphs demonstrate the wide application range of Boolean matrices and binary relations. Integer tables are also considered as part of the formal presentation process of fuzzy rule bases by Boolean matrices or binary relations because the latter can be obtained under the assumption that the corresponding integer table is already available (see Algorithms 4.1–4.2). The demonstration is based on one or more of the four fuzzy rule bases described by Tables 4.1-4.4 and some variations of them, which take into account explicitly different cases.

Example 4.1 describes explicitly a fuzzy system of Mamdami or Tsukamoto type. If this system is of CADR type, i.e. with CON antecedents and DIS rules, then it will be represented by the following 'if-then' rules:

The presentation of this type of fuzzy system by an integer table, Boolean matrix and binary relation was given earlier in Table 4.1, Eq. (4.17) and Eq. (4.29), respectively. These three formal presentations did not show explicitly the way in which the antecedents and the rules are connected. This was done on purpose for simplicity and on the assumption that we can always find these connections from the if-then rules in Eq. (4.33). However, we may want to show the connections for this fuzzy system explicitly by including them in the corresponding integer table, Boolean matrix and binary relation as follows:

$$\begin{gathered}\text{If } i_1 \text{ is S and } i_2 \text{ is S then } o_1 \text{ is N also } o_2 \text{ is N}\\ \text{or}\\ \text{If } i_1 \text{ is S and } i_2 \text{ is B then } o_1 \text{ is N also } o_2 \text{ is Z}\\ \text{or}\\ \text{If } i_1 \text{ is B and } i_2 \text{ is S then } o_1 \text{ is N also } o_2 \text{ is P}\\ \text{or}\\ \text{If } i_1 \text{ is B and } i_2 \text{ is B then } o_1 \text{ is Z also } o_2 \text{ is N}\end{gathered} \tag{4.33}$$

Table 4.11. Integer table with connections for a Mamdami/Tsukamoto fuzzy system

Rule number	Linguistic value of i_1		Linguistic value of i_2	Linguistic value of o_1	Linguistic value of o_2
1	1	and	1	1	1
or					
2	1	and	2	1	2
or					
3	2	and	1	1	3
or					
4	2	and	2	2	1

$$\begin{array}{rccccccccc}
\text{Inputs/Outputs} & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\
\text{and } 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{or} & & & & & & & & & \\
\text{and } 12 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{or} & & & & & & & & & \\
\text{and } 21 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{or} & & & & & & & & & \\
\text{and } 22 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{array} \tag{4.34}$$

$$\{[\text{and } (11, 11)] \text{ or } [\text{and } (12, 12)] \text{ or } [\text{and } (21, 13)] \text{ or } [\text{and } (22, 21)]\} \tag{4.35}$$

We can show explicitly the connections between the antecedents and the rules in any CON/DIS fuzzy system, i.e. CADR, DADR, CACR and DACR, which is of Mamdami or Tsukamoto type. This can be done by including the corresponding ‘and’ or ‘or’ operator in the designated places in the integer table, the Boolean matrix and the binary relation, as shown by Table 4.11, Eq. (4.34) and Eq. (4.35), respectively.

As a Sugeno fuzzy system differs from a Mamdami/Tsukamoto system in the consequent part of the rule base, some variations have to be introduced to account for this difference. For this purpose, we may consider a variation of Example 4.1 describing a Sugeno fuzzy system of CADR type, i.e. with CON antecedents and DIS rules, and with linear functions for the outputs. This system will be represented by the following ‘if-then’ rules:

$$\begin{array}{c}
\text{If } i_1 \text{ is S and } i_2 \text{ is S then } o_1 = 3.\,i_1 + 4.\,i_2 + 5 \text{ also } o_2 = 6.\,i_1 + 7.\,i_2 + 8 \\
\text{or} \\
\text{If } i_1 \text{ is S and } i_2 \text{ is B then } o_1 = 5.\,i_1 + 4.\,i_2 + 3 \text{ also } o_2 = 8.\,i_1 + 7.\,i_2 + 6 \\
\text{or} \\
\text{If } i_1 \text{ is B and } i_2 \text{ is S then } o_1 = 4.\,i_1 + 5.\,i_2 + 6 \text{ also } o_2 = 7.\,i_1 + 8.\,i_2 + 9 \\
\text{or} \\
\text{If } i_1 \text{ is B and } i_2 \text{ is B then } o_1 = 6.\,i_1 + 5.\,i_2 + 4 \text{ also } o_2 = 9.\,i_1 + 8.\,i_2 + 7
\end{array} \tag{4.36}$$

This type of fuzzy system can be represented by an integer table, a Boolean matrix and a binary relation similar to the ones given by Table 4.1, Eq. (4.17) and Eq. (4.29), respectively. We can show explicitly the way in which the antecedents and the rules are connected by including the corresponding ‘and’ or ‘or’ operator in the designated places of the above three formal presentations, as already shown for a Mamdami/Tsukamoto system by Table 4.11, Eq. (4.34) and Eq. (4.35). In this case, the corresponding integer table, Boolean matrix and binary relation will be as follows:

Table 4.12. Integer table with connections for a Sugeno fuzzy system

Rule number	Linguistic value of i_1		Linguistic value of i_2	Linear function coefficients for o_1	Linear function coefficients for o_2
1	1	and	1	3, 4, 5	6, 7, 8
or					
2	1	and	2	5, 3, 4	8, 7, 6
or					
3	2	and	1	4, 5, 6	7, 8, 9
or					
4	2	and	2	6, 5, 4	9, 8, 7

$$\begin{array}{lcccc}
\text{Inputs/Outputs} & 3{:}4{:}5\text{-}6{:}7{:}8 & 5{:}3{:}4\text{-}8{:}7{:}6 & 4{:}5{:}6\text{-}7{:}8{:}9 & 6{:}5{:}4\text{-}9{:}8{:}7 \\
\text{and } 11 & 1 & 0 & 0 & 0 \\
\quad\text{or} & & & & \\
\text{and } 12 & 0 & 1 & 0 & 0 \\
\quad\text{or} & & & & \\
\text{and } 21 & 0 & 0 & 1 & 0 \\
\quad\text{or} & & & & \\
\text{and } 22 & 0 & 0 & 0 & 1
\end{array} \tag{4.37}$$

$$\{[\text{and } (11, 3{:}4{:}5\text{-}6{:}7{:}8)] \text{ or } [\text{and } (12, 5{:}3{:}4\text{-}8{:}7{:}6)] \text{ or } [\text{and } (21, 4{:}5{:}6\text{-}7{:}8{:}9)] \text{ or } [\text{and } (22, 6{:}5{:}4\text{-}9{:}8{:}7)]\} \tag{4.38}$$

As far as some of the notations for the above Sugeno fuzzy system are concerned, the dots (.) in Eq. (4.36) stand for arithmetic multiplication in the linear functions. The commas (,) in Table 4.12, the semi-colons (:) in Eq. (4.37) and the dashes (-) in Eq. (4.38) are used as separators for the coefficients of the linear functions in the integer table, the Boolean matrix, and the binary relation, respectively.

It can be seen from Eq. (4.37) that the Boolean matrix for a Sugeno fuzzy system contains only the column labels that reflect existing permutations of coefficient values for the linear functions. This is so because it would be theoretically unjustifiable and practically impossible to include all possible permutations of such values. This variation can be reflected in the conversion process of an integer table into a Boolean matrix by introducing appropriate changes in Algorithm 4.1.

We can show explicitly the connections between the antecedents and the rules in any CON/DIS fuzzy system, i.e. CADR, DADR, CACR and DACR, which is of Sugeno type. This can be done by including the corresponding 'and' or 'or' operator in the designated places in the integer table, the Boolean matrix and the binary relation, as shown by Table 4.12, Eq. (4.37) and Eq. (4.38), respectively.

The Mamdami/Tsukamoto and the Sugeno fuzzy system described by the 'if-then' rules in Eq. (4.33) and Eq. (4.36) are both with two outputs, i.e. they are MO systems. Therefore, each of these MO systems can be represented by two logically equivalent SO systems as follows

$$\begin{gathered}\text{If } i_1 \text{ is S and } i_2 \text{ is S then } o_1 \text{ is N}\\ \text{or}\\ \text{If } i_1 \text{ is S and } i_2 \text{ is B then } o_1 \text{ is N}\\ \text{or}\\ \text{If } i_1 \text{ is B and } i_2 \text{ is S then } o_1 \text{ is N}\\ \text{or}\\ \text{If } i_1 \text{ is B and } i_2 \text{ is B then } o_1 \text{ is Z}\end{gathered} \tag{4.39}$$

$$\begin{gathered}\text{If } i_1 \text{ is S and } i_2 \text{ is S then } o_2 \text{ is N}\\ \text{or}\\ \text{If } i_1 \text{ is S and } i_2 \text{ is B then } o_2 \text{ is Z}\\ \text{or}\\ \text{If } i_1 \text{ is B and } i_2 \text{ is S then } o_2 \text{ is P}\\ \text{or}\\ \text{If } i_1 \text{ is B and } i_2 \text{ is B then } o_2 \text{ is N}\end{gathered} \tag{4.40}$$

$$\begin{gathered}\text{If } i_1 \text{ is S and } i_2 \text{ is S then } o_1 = 3.\,i_1 + 4.\,i_2 + 5\\ \text{or}\\ \text{If } i_1 \text{ is S and } i_2 \text{ is B then } o_1 = 5.\,i_1 + 4.\,i_2 + 3\\ \text{or}\\ \text{If } i_1 \text{ is B and } i_2 \text{ is S then } o_1 = 4.\,i_1 + 5.\,i_2 + 6\\ \text{or}\\ \text{If } i_1 \text{ is B and } i_2 \text{ is B then } o_1 = 6.\,i_1 + 5.\,i_2 + 4\end{gathered} \tag{4.41}$$

$$\begin{gathered}\text{If } i_1 \text{ is S and } i_2 \text{ is S then } o_2 = 6.\,i_1 + 7.\,i_2 + 8\\ \text{or}\\ \text{If } i_1 \text{ is S and } i_2 \text{ is B then } o_2 = 8.\,i_1 + 7.\,i_2 + 6\\ \text{or}\\ \text{If } i_1 \text{ is B and } i_2 \text{ is S then } o_2 = 7.\,i_1 + 8.\,i_2 + 9\\ \text{or}\\ \text{If } i_1 \text{ is B and } i_2 \text{ is B then } o_2 = 9.\,i_1 + 8.\,i_2 + 7\end{gathered} \tag{4.42}$$

where Eqs. (4.39)–(4.40) and Eqs. (4.41)–(4.42) are the SO counterparts of Eq. (4.33) and Eq. (4.36), respectively.

The formal presentation of the above two fuzzy systems by integer tables, Boolean matrices and binary relations is fairly straightforward, as shown below. In this case, Tables 4.13–4.14 and Eqs. (4.43)–(4.46) represent the Mamdami/Tsukamoto fuzzy system whereas Tables 4.15-4.16 and Eqs. (4.47)–(4.50) represent the Sugeno system.

Table 4.13. Integer table with connections for the first SO Mamdami/Tsukamoto fuzzy system

Rule number	Linguistic value of i_1		Linguistic value of i_2	Linguistic value of o_1
1	1	and	1	1
or				
2	1	and	2	1
or				
3	2	and	1	1
or				
4	2	and	2	2

Table 4.14. Integer table with connections for the second SO Mamdami/ Tsukamoto fuzzy system

Rule number	Linguistic value of i_1		Linguistic value of i_2	Linguistic value of o_2
1	1	and	1	1
or				
2	1	and	2	2
or				
3	2	and	1	3
or				
4	2	and	2	1

Table 4.15. Integer table with connections for the first SO Sugeno fuzzy system

Rule number	Linguistic value of i_1		Linguistic value of i_2	Linear function coefficients for o_1
1	1	and	1	3, 4, 5
or				
2	1	and	2	5, 3, 4
or				
3	2	and	1	4, 5, 6
or				
4	2	and	2	6, 5, 4

Table 4.16. Integer table with connections for the second SO Sugeno fuzzy system

Rule number	Linguistic value of i_1		Linguistic value of i_2	Linear function coefficients for o_2
1	1	and	1	6, 7, 8
or				
2	1	and	2	8, 7, 6
or				
3	2	and	1	7, 8, 9
or				
4	2	and	2	9, 8, 7

$$\begin{array}{rccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ \text{and } 11 & 1 & 0 & 0 \\ \text{or} & & & \\ \text{and } 12 & 1 & 0 & 0 \\ \text{or} & & & \\ \text{and } 21 & 1 & 0 & 0 \\ \text{or} & & & \\ \text{and } 22 & 0 & 1 & 0 \end{array} \tag{4.43}$$

$$\begin{array}{rccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ \text{and } 11 & 1 & 0 & 0 \\ \text{or} & & & \\ \text{and } 12 & 0 & 1 & 0 \\ \text{or} & & & \\ \text{and } 21 & 0 & 0 & 1 \\ \text{or} & & & \\ \text{and } 22 & 1 & 0 & 0 \end{array} \tag{4.44}$$

$$\{[\text{and } (11, 1)] \text{ or } [\text{and } (12, 1)] \text{ or } [\text{and } (21, 1)] \text{ or } [\text{and } (22, 2)]\} \tag{4.45}$$

$$\{[\text{and } (11, 1)] \text{ or } [\text{and } (12, 2)] \text{ or } [\text{and } (21, 3)] \text{ or } [\text{and } (22, 1)]\} \tag{4.46}$$

$$\begin{array}{lcccc} \text{Inputs/Outputs} & 3{:}4{:}5 & 5{:}3{:}4 & 4{:}5{:}6 & 6{:}5{:}4 \\ \text{and } 11 & 1 & 0 & 0 & 0 \\ \quad\text{or} & & & & \\ \text{and } 12 & 0 & 1 & 0 & 0 \\ \quad\text{or} & & & & \\ \text{and } 21 & 0 & 0 & 1 & 0 \\ \quad\text{or} & & & & \\ \text{and } 22 & 0 & 0 & 0 & 1 \end{array} \tag{4.47}$$

$$\begin{array}{lcccc} \text{Inputs/Outputs} & 6{:}7{:}8 & 8{:}7{:}6 & 7{:}8{:}9 & 9{:}8{:}7 \\ \text{and } 11 & 1 & 0 & 0 & 0 \\ \quad\text{or} & & & & \\ \text{and } 12 & 0 & 1 & 0 & 0 \\ \quad\text{or} & & & & \\ \text{and } 21 & 0 & 0 & 1 & 0 \\ \quad\text{or} & & & & \\ \text{and } 22 & 0 & 0 & 0 & 1 \end{array} \tag{4.48}$$

{[and (11, 3:4:5)] or [and (12, 5:3:4)] or [and (21, 4:5:6)] or [and (22, 6:5:4)]} (4.49)

{[and (11, 6:7:8)] or [and (12, 8:7:6)] or [and (21, 7:8:9)] or [and (22, 9:8:7)]} (4.50)

It has been demonstrated so far that Boolean matrices and binary relations can be used for representing formally all basic types of SRB systems of FF type. They can also be used for representing MRB systems of either FF or FB type. For this purpose, let us assume that the four rule bases from Examples 4.1-4.4 are located and interconnected in a MRB system as follows:

- the rule base in Example 4.1 is in level 1 of layer 1, i.e. it is denoted by RB_{11},
- the rule base in Example 4.2 is in level 1 of layer 2, i.e. it is denoted by RB_{12},
- the rule base in Example 4.3 is in level 2 of layer 1, i.e. it is denoted by RB_{21},
- the rule base in Example 4.4 is in level 2 of layer 2, i.e. it is denoted by RB_{22},
- the outputs of RB_{11} are fed forward into the inputs to RB_{12},
- the outputs of RB_{21} are fed forward into the inputs to RB_{22},

- the outputs of RB_{12} are fed back into the inputs to RB_{21},
- the outputs of RB_{22} are fed back into the inputs to RB_{11}.

Therefore, the MRB system can be represented by the following block matrices:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_{11} & RB_{12} \\ \text{level 2} & RB_{21} & RB_{22} \end{array} \qquad (4.51)$$

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_i = i_i^{1,2},\ i=1,2 & o_i = i_i^{2,1},\ i=1,2 \\ \text{level 2} & o_i = i_i^{2,2},\ i=1,2 & o_i = i_i^{1,1},\ i=1,2 \end{array} \qquad (4.52)$$

The block matrices in Eqs. (4.51)–(4.52) are private cases of the block matrices in Eqs. (2.11)–(2.12) for the four rule bases in Examples 4.1-4.4. In this context, the block matrix in Eq. (4.51) specifies the location of individual rule bases in the network structure of the MRB system and the block matrix in Eq. (4.52) specifies the output-input interconnections between individual rule bases.

Equations (4.51)–(4.52) represent a structural description of a MRB system which is a general type of description. Therefore, it must be accompanied by a detailed description of each individual rule base in the form of an integer table, Boolean matrix or binary relation. As far as the individual rule bases are concerned, they can be of any type, i.e. Mamdami, Sugeno or Tsukamoto systems, CON or DIS systems, FF or FB systems, as well as MO or SO systems.

The block matrix in Eq. (4.52) is for a MRB system whereas an integer table is for a SRB system. In particular, the integer table represents the input-output mappings in a rule base while the block matrix represents the output-input mappings between rule bases. Therefore, the block matrix in Eq. (4.52) can be converted into an equivalent Boolean matrix and binary relation by means of algorithms similar to Algorithms 4.1–4.2. In this case, the Boolean matrix and the binary relation will be represented by Eq. (4.53) and Eq. (4.54), respectively.

Undoubtedly, both Boolean matrices and binary relations are advanced techniques for formal presentation of fuzzy rule bases, which facilitate the complexity management in fuzzy systems. These techniques compress the information about the rule base contained by the corresponding integer table and this compression reduces the quantitative complexity in the fuzzy system. At the same time, the qualitative complexity of the system is also reduced because the rule base is represented in a more transparent way that makes the rules easier for interpretation.

However, both Boolean matrices and binary relations have been demonstrated so far only in the context of passive analysis of the associated fuzzy system. In order to make an active impact on the fuzzy system, e.g. when we want to simplify it or change some of its properties, we need to be able to manipulate the formal presentation of the system appropriately. This issue is discussed in detail in the next chapter.

$$\begin{array}{ccccc} \text{Rule base / Rule base} & RB_{11} & RB_{21} & RB_{12} & RB_{22} \\ RB_{11} & 0 & 0 & 1 & 0 \\ RB_{21} & 0 & 0 & 0 & 1 \\ RB_{12} & 0 & 1 & 0 & 0 \\ RB_{22} & 1 & 0 & 0 & 0 \end{array} \qquad (4.53)$$

$$\{(RB_{11}, RB_{12}), (RB_{21}, RB_{22}), (RB_{12}, RB_{21}), (RB_{22}, RB_{11})\} \qquad (4.54)$$

5 Formal Manipulation of Fuzzy Rule Based Systems

5.1 Preliminaries on Rule Base Manipulation

The advanced techniques for formal presentation of fuzzy rule bases introduced in the previous chapter are essential to the complexity management in fuzzy systems. These techniques not only compress the information contained by the fuzzy system but they also facilitate the identification of the properties of the fuzzy rule base. However, sometimes we may want to change the properties of a fuzzy rule base by manipulating its formal presentation appropriately. For example, we may want to make an inconsistent rule base consistent or an incomplete rule base complete.

Six techniques for formal manipulation of fuzzy rule bases are introduced in this chapter. These techniques can be applied mainly in the context of MRB systems as the corresponding manipulations usually affect the interconnections between individual rule bases at some stage of the manipulation process. However, we may have to deal only with SRB systems at other stages of the manipulation process.

Integer tables are not used in this chapter because they do not lend themselves easily to formal manipulation. On the contrary, Boolean matrices and binary relations are very convenient for formal manipulation and therefore they are used for demonstrating the basic operations for formal manipulation of fuzzy rule bases, as shown further in Sects. 5.2–5.7.

Numerous examples are presented further in this chapter. Their purpose is to demonstrate the rule base manipulation techniques and to show the impact of the manipulation on the properties of the rule bases involved, i.e. which properties of the participating rule bases are preserved or lost in the resultant rule bases.

5.2 Vertical Merging Manipulation of Rule Bases

The process of merging two fuzzy rule bases in parallel into a single fuzzy rule base is called 'vertical merging' and it is shown in Fig. 5.1. This type of manipulation can be applied to rule bases residing at different levels within the same layer of a MRB system. Obviously, the number of levels in this layer will be reduced as a result of this manipulation.

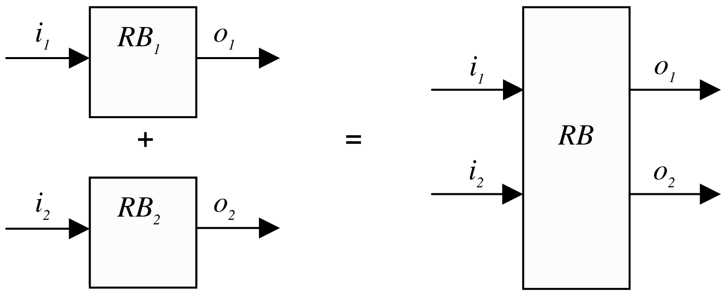

Fig. 5.1. Vertical merging of rule bases RB_1 and RB_2 into rule base RB

In order to illustrate the vertical merging manipulation, we introduce an operation called 'vertical composition'. This operation is binary and it can be applied to only two operands at a time. The operands in this case are the Boolean matrices or the binary relations representing the operand rule bases. The result from the application of this operation is a single Boolean matrix or binary relation representing the product rule base.

Algorithms 5.1 and 5.2 demonstrate the application of the vertical composition operation to Boolean matrices and binary relations, respectively.

Algorithm 5.1

1. Construct all possible permutations of row labels from the operand matrices and sort them.
2. Construct all possible permutations of column labels from the operand matrices and sort them.
3. Label the rows of the product matrix with the sorted permutations of row labels from the operand matrices.
4. Label the columns of the product matrix with the sorted permutations of column labels from the operand matrices.
5. Go through all the elements of the operand matrices and set each element of the product matrix equal to 1 or 0, as described in steps 6 and 7.
6. If an element of the product matrix is mapped from a pair of non-zero elements in the product matrices, set this element equal to 1.
7. If an element of the product matrix is mapped from a pair of elements in the product matrices at least one of which is zero, set this element equal to 0.

Algorithm 5.2

1. Construct all possible pairs of maplets from the operand relations such that the first maplet in each pair comes from the first relation and the second maplet comes from the second relation.
2. Merge each of these pairs of maplets into a new maplet, as described in steps 3 and 4.
3. Make the first element in each new maplet equal to the corresponding permutation of first elements from the associated pair of maplets.
4. Make the second element in each new maplet equal to the corresponding permutation of second elements from the associated pair of maplets.
5. Generate the product binary relation containing all new maplets created in the previous two steps of this algorithm.

Example 5.1

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

$$RB_1: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{5.1}$$

$$RB_1: \{(1, 2), (2, 1)\} \tag{5.2}$$

$$RB_2: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{5.3}$$

$$RB_2: \{(1, 2), (2, 2)\} \tag{5.4}$$

The vertical merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1 + RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

$$RB: \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 0 & 1 \\ 12 & 0 & 0 & 0 & 1 \\ 21 & 0 & 1 & 0 & 0 \\ 22 & 0 & 1 & 0 & 0 \end{array} \tag{5.5}$$

$$RB\text{: } \{(11, 22), (12, 22), (21, 12), (22, 12)\} \tag{5.6}$$

The rule bases above have the following properties:

- RB_1 – complete, consistent, exhaustive, monotonic,
- RB_2 – complete, consistent, non-exhaustive, non-monotonic,
- RB – complete, consistent, non-exhaustive, non-monotonic.

Example 5.2

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

$$RB_1\text{:} \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{5.7}$$

$$RB_1\text{: } \{(1, 2), (2, 2)\} \tag{5.8}$$

$$RB_2\text{:} \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{5.9}$$

$$RB_2\text{: } \{(1, 1), (2, 1)\} \tag{5.10}$$

The vertical merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1 + RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

$$RB\text{:} \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 1 & 0 \\ 12 & 0 & 0 & 1 & 0 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 1 & 0 \end{array} \tag{5.11}$$

$$RB\text{: } \{(11, 21), (12, 21), (21, 21), (22, 21)\} \tag{5.12}$$

The rule bases above have the following properties:

- RB_1 – complete, consistent, non-exhaustive, non-monotonic,
- RB_2 – complete, consistent, non-exhaustive, non-monotonic,
- RB – complete, consistent, non-exhaustive, non-monotonic.

Example 5.3

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 1 & 1 \end{array} \tag{5.13}$$

$$RB_1: \{(2, 1), (2, 2)\} \tag{5.14}$$

$$RB_2: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{5.15}$$

$$RB_2: \{(1, 2), (2, 1)\} \tag{5.16}$$

The vertical merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1 + RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

$$RB: \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 \\ 21 & 0 & 1 & 0 & 1 \\ 22 & 1 & 0 & 1 & 0 \end{array} \tag{5.17}$$

$$RB: \{(21, 12), (22, 11), (21, 22), (22, 21)\} \tag{5.18}$$

The rule bases above have the following properties:

- RB_1 – incomplete, inconsistent, exhaustive, monotonic,
- RB_2 – complete, consistent, exhaustive, monotonic,
- RB – incomplete, inconsistent, exhaustive, monotonic.

Example 5.4

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

$$RB_1: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 1 & 1 & 1 \\ 2 & 0 & 0 \end{array} \tag{5.19}$$

$$RB_1: \{(1, 1), (1, 2)\} \tag{5.20}$$

$$RB_2: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 1 & 0 & 0 \\ 2 & 1 & 1 \end{array} \tag{5.21}$$

$$RB_2: \{(2, 1), (2, 2)\} \tag{5.22}$$

The vertical merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1 + RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

$$RB: \quad \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 0 & 0 & 0 & 0 \\ 12 & 1 & 1 & 1 & 1 \\ 21 & 0 & 0 & 0 & 0 \\ 22 & 0 & 0 & 0 & 0 \end{array} \tag{5.17}$$

$$RB: \{(12, 11), (12, 12), (12, 21), (12, 22)\} \tag{5.24}$$

The rule bases above have the following properties:

- RB_1 – incomplete, inconsistent, exhaustive, monotonic,
- RB_2 – incomplete, inconsistent, exhaustive, monotonic,
- RB – incomplete, inconsistent, exhaustive, monotonic.

Example 5.5

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 3 & 0 & 0 & 0 \end{array} \tag{5.25}$$

$$RB_1: \{(1, 1), (2, 1), (2, 2)\} \tag{5.26}$$

$$RB_2: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 1 & 1 \end{array} \tag{5.27}$$

$$RB_2: \{(2, 3), (3, 2), (3, 3)\} \tag{5.28}$$

The vertical merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1 + RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

$$RB: \begin{array}{cccccccccc} \text{Inputs/Outputs} & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\ 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 22 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 23 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 31 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \tag{5.29}$$

$$\begin{aligned} RB: \{&(12, 13), (13, 12), (13, 13), \\ &(22, 13), (23, 12), (23, 13), \\ &(22, 23), (23, 22), (23, 23)\} \end{aligned} \tag{5.30}$$

The rule bases above have the following properties:

- RB_1 – incomplete, inconsistent, non-exhaustive, non-monotonic,
- RB_2 – incomplete, inconsistent, non-exhaustive, non-monotonic,
- RB – incomplete, inconsistent, non-exhaustive, non-monotonic.

Example 5.6
The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{c|ccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ \hline 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 \end{array} \tag{5.31}$$

$$RB_1: \{(1, 2), (2, 1), (3, 3)\} \tag{5.32}$$

$$RB_2: \begin{array}{c|ccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ \hline 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 \end{array} \tag{5.33}$$

$$RB_2: \{(1, 1), (2, 3), (3, 2)\} \tag{5.34}$$

The vertical merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1 + RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

$$RB: \begin{array}{c|ccccccccc} \text{Inputs/Outputs} & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\ \hline 11 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 13 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 21 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 22 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 23 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 31 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \tag{5.35}$$

$$\begin{aligned} RB: \{&(11, 21), (12, 23), (13, 22), \\ &(21, 11), (22, 13), (23, 12), \\ &(31, 31), (32, 33), (33, 32)\} \end{aligned} \tag{5.36}$$

The rule bases above have the following properties:

- RB_1 – complete, consistent, exhaustive, monotonic,
- RB_2 – complete, consistent, exhaustive, monotonic,
- RB – complete, consistent, exhaustive, monotonic.

The examples above show that if at least one of the two operand matrices does not have a specific property then the product matrix does not have this property either. In particular, the second operand matrix in Example 5.1 and the two operand matrices in Example 5.2 are non-exhaustive and non-monotonic as are the associated product matrices. Also, the first operand matrix in Example 5.3 and the two operand matrices in Example 5.4 are incomplete and inconsistent as are the associated product matrices. And finally, the two operand matrices in Example 5.5 are incomplete, inconsistent, non-exhaustive and non-monotonic as is the associated product matrix.

In addition, the examples above show that if each of the two operand matrices has a specific property then the product matrix has this property too. In particular, the two operand matrices in Examples 5.1–5.2 are complete and consistent as are the associated product matrices. Also, the two operand matrices in Examples 5.3–5.4 are exhaustive and monotonic as are the associated product matrices. And finally, the two operand matrices in Example 5.6 are complete, consistent, exhaustive and monotonic as is the associated product matrix.

5.3 Vertical Splitting Manipulation of Rule Bases

The process of splitting a single fuzzy rule base into two fuzzy rule bases in parallel is called 'vertical splitting' and it is shown in Fig. 5.2. This type of manipulation can be applied to a rule base of a SRB system or a rule base that is part of a MRB system. Obviously, as a result of this manipulation a SRB system will be represented as a number of smaller SRB systems, whereas for a MRB system the number of levels in the corresponding layer will be increased.

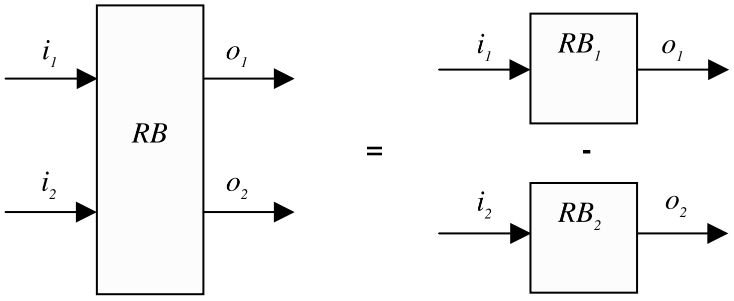

Fig. 5.2. Vertical splitting of rule base RB into rule bases RB_1 and RB_2

In order to illustrate the vertical splitting manipulation, we introduce an operation called ‘vertical decomposition’. This operation is unary and it can be applied to only one operand at a time. The operand in this case is the Boolean matrix or the binary relation representing the operand rule base. The result from the application of this operation is a couple of Boolean matrices or binary relations representing the product rule bases.

Algorithms 5.3 and 5.4 demonstrate the application of the vertical decomposition operation to Boolean matrices and binary relations, respectively.

Algorithm 5.3

1. Construct a pair of all possible sub-permutations of row labels from the operand matrix and sort them.
2. Construct a pair of all possible sub-permutations of column labels from the operand matrix and sort them.
3. Label the rows of the first product matrix with the first element from the pair of all possible sorted sub-permutations of row labels from the operand matrix.
4. Label the columns of the first product matrix with the first element from the pair of all possible sorted sub-permutations of column labels from the operand matrix.
5. Label the rows of the second product matrix with the second element from the pair of all possible sorted sub-permutations of row labels from the operand matrix.
6. Label the columns of the second product matrix with the second element from the pair of all possible sorted sub-permutations of column labels from the operand matrix.
7. Go through all the elements of the operand matrix and set each element of the two product matrices equal to 1 or 0, as described in steps 8 and 9.
8. If an element of a product matrix is mapped from a non-zero element in the operand matrix, set this element equal to 1.
9. If an element of the product matrix is mapped from a zero element in the operand matrix, set this element equal to 0.

Algorithm 5.4

1. Construct all pairs of maplets from the operand relation such that the conditions in steps 2-5 are satisfied.
2. The first sub-element of the first element in each pair of maplets is the same.
3. The first sub-element of the second element in each pair of maplets is the same.
4. The second sub-element of the first element in each pair of maplets is the same.

5. The second sub-element of the second element in each pair of maplets is the same.
6. If any of the conditions from the previous four steps of this algorithm can not be satisfied, then vertical splitting is not possible.
7. Rearrange the maplets from the operand relation following the procedure in steps 8-11.
8. Make each first sub-element of a first element in each pair of maplets the first element in a pair of the first product relation.
9. Make each first sub-element of a second element in each pair of maplets the second element in a pair of the first product relation.
10. Make each second sub-element of a first element in each pair of maplets the first element in a pair of the second product relation.
11. Make each second sub-element of a second element in each pair of maplets the second element in a pair of the second product relation.
12. Generate the two product binary relations containing all maplets created in the previous four steps of this algorithm.

Example 5.7

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

RB: (5.37)

Inputs/Outputs	11	12	21	22
11	0	0	0	1
12	0	0	0	1
21	0	1	0	0
22	0	1	0	0

$$RB: \{(11, 22), (12, 22), (21, 12), (22, 12)\} \quad (5.38)$$

The vertical splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1 - RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

RB_1: (5.39)

Inputs/Outputs	1	2
1	0	1
2	1	0

$$RB_1: \{(1, 2), (2, 1)\} \quad (5.40)$$

$$RB_2\text{:} \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{5.41}$$

$$RB_2\text{: } \{(1, 2), (2, 2)\} \tag{5.42}$$

The rule bases above have the following properties:

- *RB* – complete, consistent, non-exhaustive, non-monotonic,
- RB_1 – complete, consistent, exhaustive, monotonic,
- RB_2 – complete, consistent, non-exhaustive, non-monotonic.

Example 5.8
The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

$$RB\text{:} \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 1 & 0 \\ 12 & 0 & 0 & 1 & 0 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 1 & 0 \end{array} \tag{5.43}$$

$$RB\text{: } \{(11, 21), (12, 21), (21, 21), (22, 21)\} \tag{5.44}$$

The vertical splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1 - RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

$$RB_1\text{:} \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{5.45}$$

$$RB_1\text{: } \{(1, 2), (2, 2)\} \tag{5.46}$$

$$RB_2\text{:} \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{5.47}$$

$$RB_2\text{: } \{(1, 1), (2, 1)\} \tag{5.48}$$

The rule bases above have the following properties:

- *RB* – complete, consistent, non-exhaustive, non-monotonic,
- *RB_1* – complete, consistent, non-exhaustive, non-monotonic,
- *RB_2* – complete, consistent, non-exhaustive, non-monotonic.

Example 5.9
The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

$$RB\text{:}\quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 \\ 21 & 0 & 1 & 0 & 1 \\ 22 & 1 & 0 & 1 & 0 \end{array} \tag{5.49}$$

$$RB\text{: } \{(21, 12), (22, 11), (21, 22), (22, 21)\} \tag{5.50}$$

The vertical splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1 – RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

$$RB_1\text{:}\quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 1 & 1 \end{array} \tag{5.51}$$

$$RB_1\text{: } \{(2, 1), (2, 2)\} \tag{5.52}$$

$$RB_2\text{:}\quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{5.53}$$

$$RB_2\text{: } \{(1, 2), (2, 1)\} \tag{5.54}$$

The rule bases above have the following properties:

- *RB* – incomplete, inconsistent, exhaustive, monotonic,

- RB_1 – incomplete, inconsistent, exhaustive, monotonic,
- RB_2 – complete, consistent, exhaustive, monotonic.

Example 5.10

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

$$RB: \quad \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 0 & 0 \\ 12 & 1 & 1 & 1 & 1 \\ 21 & 0 & 0 & 0 & 0 \\ 22 & 0 & 0 & 0 & 0 \end{array} \tag{5.55}$$

$$RB: \{(12, 11), (12, 12), (12, 21), (12, 22)\} \tag{5.56}$$

The vertical splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1 - RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

$$RB_1: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 0 & 0 \end{array} \tag{5.57}$$

$$RB_1: \{(1, 1), (1, 2)\} \tag{5.58}$$

$$RB_2: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 1 & 1 \end{array} \tag{5.59}$$

$$RB_2: \{(2, 1), (2, 2)\} \tag{5.60}$$

The rule bases above have the following properties:

- *RB* – incomplete, inconsistent, exhaustive, monotonic,
- RB_1 – incomplete, inconsistent, exhaustive, monotonic,
- RB_2 – incomplete, inconsistent, exhaustive, monotonic.

Example 5.11
The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

$$RB: \begin{array}{c|ccccccccc}
\text{Inputs/Outputs} & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\
11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
12 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
13 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
22 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
23 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
31 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array} \qquad (5.61)$$

$$\begin{aligned} RB: \{&(12, 13), (13, 12), (13, 13), \\ &(22, 13), (23, 12), (23, 13), \\ &(22, 23), (23, 22), (23, 23)\} \end{aligned} \qquad (5.62)$$

The vertical splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1 – RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{c|ccc}
\text{Inputs/Outputs} & 1 & 2 & 3 \\
1 & 1 & 0 & 0 \\
2 & 1 & 1 & 0 \\
3 & 0 & 0 & 0
\end{array} \qquad (5.63)$$

$$RB_1: \{(1, 1), (2, 1), (2, 2)\} \qquad (5.64)$$

$$RB_2: \begin{array}{c|ccc}
\text{Inputs/Outputs} & 1 & 2 & 3 \\
1 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 \\
3 & 0 & 1 & 1
\end{array} \qquad (5.65)$$

$$RB_2: \{(2, 3), (3, 2), (3, 3)\} \qquad (5.66)$$

The rule bases above have the following properties:

- *RB* – incomplete, inconsistent, non-exhaustive, non-monotonic,
- RB_1 – incomplete, inconsistent, non-exhaustive, non-monotonic,
- RB_2 – incomplete, inconsistent, non-exhaustive, non-monotonic.

Example 5.12

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

$$RB: \begin{array}{c|ccccccccc} \text{Inputs/Outputs} & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\ 11 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 13 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 21 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 22 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 23 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 31 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \quad (5.67)$$

$$\begin{aligned} RB: \{&(11, 21), (12, 23), (13, 22), \\ &(21, 11), (22, 13), (23, 12), \\ &(31, 31), (32, 33), (33, 32)\} \end{aligned} \quad (5.68)$$

The vertical splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1 - RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{c|ccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 \end{array} \quad (5.69)$$

$$RB_1: \{(1, 2), (2, 1), (3, 3)\} \quad (5.70)$$

$$RB_2: \begin{array}{c|ccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 \end{array} \quad (5.71)$$

$$RB_2: \{(1, 1), (2, 3), (3, 2)\} \quad (5.72)$$

The rule bases above have the following properties:

- RB – complete, consistent, exhaustive, monotonic,
- RB_1 – complete, consistent, exhaustive, monotonic,
- RB_2 – complete, consistent, exhaustive, monotonic.

The examples above show that if the operand matrix does not have a specific property then at least one of the two product matrices does not have this property either. In particular, the operand matrices in Examples 5.7–5.8 are non-exhaustive and non-monotonic as are the second product matrix in the first example and the two product matrices in the second example. Also, the operand matrices in Examples 5.9–5.10 are incomplete and inconsistent as are the first product matrix in the first example and the two product matrices in the second example. And finally, the operand matrix in Example 5.11 is incomplete, inconsistent, non-exhaustive and non-monotonic as are the two product matrices.

In addition, the examples above show that if the operand matrix has a specific property then the product matrices have this property too. In particular, the operand matrices in Examples 5.7–5.8 are complete and consistent as are all four product matrices. Also, the operand matrices in Examples 5.9–5.10 are exhaustive and monotonic as are all four product matrices. And finally, the operand matrix in Example 5.12 is complete, consistent, exhaustive and monotonic as are the two product matrices.

5.4 Horizontal Merging Manipulation of Rule Bases

The process of merging two fuzzy rule bases in sequence into a single fuzzy rule base is called ‘horizontal merging’ and it is shown in Fig. 5.3. This type of manipulation can be applied to rule bases residing in different layers within the same level of a MRB system. Obviously, the number of layers in this level will be reduced as a result of this manipulation.

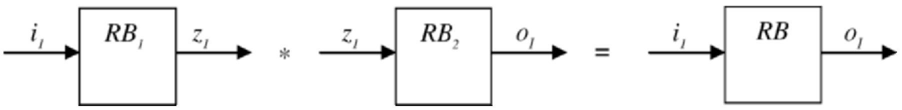

Fig. 5.3. Horizontal merging of rule bases RB_1 and RB_2 into rule base RB

In order to illustrate the horizontal merging manipulation, we introduce an operation called ‘horizontal composition’. This operation is binary and it can be applied to only two operands at a time. The operands in this case are the Boolean matrices or the binary relations representing the operand rule bases. The result from the application of this operation is a single Boolean matrix or binary relation representing the product rule base.

Algorithms 5.5 and 5.6 demonstrate the application of the vertical composition operation to Boolean matrices and binary relations, respectively.

Algorithm 5.5

1. Label the rows of the product matrix with the row labels from the first operand matrix.
2. Label the columns of the product matrix with the column labels from the second operand matrix.
3. Set each element of the product matrix equal to 1 or 0 by mapping it from the corresponding row in the first operand matrix and the corresponding column in the second operand matrix, as described in step 4.
4. Find the product matrix by multiplying the operand matrices using the operations for 'addition' and 'multiplication' of elements, as defined by Eqs. (4.1)–(4.8) in Sect. 4.3.

Algorithm 5.6

1. Construct all maplets in the product relation so that the first element in each maplet is equal to the first element in a maplet in the first relation and the second element in each maplet is equal to the second element in a maplet in the second relation, as described in step 2.
2. Compose the operand relations using the operations for 'aggregation' and 'composition' of elements, as defined by Eqs. (4.21)–(4.28) in Sect. 4.4.

Example 5.13

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 0 & 0 \end{array} \qquad (5.73)$$

$$RB_1: \{(1, 1), (1, 2)\} \qquad (5.74)$$

$$RB_2: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \qquad (5.75)$$

$$RB_2: \{(1, 2), (2, 2)\} \tag{5.76}$$

The horizontal merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1{*}RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

$$RB: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 0 \end{array} \tag{5.77}$$

$$RB: \{(1, 2)\} \tag{5.78}$$

The rule bases above have the following properties:

- RB_1 – incomplete, inconsistent, exhaustive, monotonic,
- RB_2 – complete, consistent, non-exhaustive, non-monotonic,
- RB – incomplete, consistent, non-exhaustive, monotonic.

Example 5.14

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{5.79}$$

$$RB_1: \{(1, 1), (2, 1)\} \tag{5.80}$$

$$RB_2: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 0 & 0 \end{array} \tag{5.81}$$

$$RB_2: \{(1, 1), (1, 2)\} \tag{5.82}$$

The horizontal merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1{*}RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

$$RB: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 1 & 1 & 1 \\ 2 & 1 & 1 \end{array} \tag{5.83}$$

$$RB: \{(1, 1), (1, 2), (2, 1), (2, 2)\} \tag{5.84}$$

The rule bases above have the following properties:

- RB_1 – complete, consistent, non-exhaustive, non-monotonic,
- RB_2 – incomplete, inconsistent, exhaustive, monotonic,
- RB – complete, inconsistent, exhaustive, non-monotonic.

Example 5.15

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 1 & 1 & 0 & 0 \\ 12 & 0 & 0 & 0 & 1 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 0 & 0 \end{array} \tag{5.85}$$

$$RB_1: \{(11, 11), (11, 12), (12, 22), (21, 21)\} \tag{5.86}$$

$$RB_2: \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 0 & 0 & 0 & 1 \\ 12 & 0 & 0 & 0 & 1 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 1 & 0 & 0 \end{array} \tag{5.87}$$

$$RB_2: \{(11, 22), (12, 22), (21, 21), (22, 12)\} \tag{5.88}$$

The horizontal merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1 * RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

$$\begin{array}{c|cccc} RB: \text{ Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 0 & 1 \\ 12 & 0 & 1 & 0 & 0 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 0 & 0 \end{array} \quad (5.89)$$

$$RB: \{(11, 22), (12, 12), (21, 21)\} \quad (5.90)$$

The rule bases above have the following properties:

- RB_1 – incomplete, inconsistent, exhaustive, monotonic,
- RB_2 – complete, consistent, non-exhaustive, non-monotonic,
- *RB* – incomplete, consistent, non-exhaustive, monotonic.

Example 5.16

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

$$\begin{array}{c|cccc} RB_1: \text{ Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 1 & 0 & 0 \\ 12 & 0 & 1 & 0 & 0 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 0 & 1 \end{array} \quad (5.91)$$

$$RB_1: \{(11, 12), (12, 12), (21, 21), (22, 22)\} \quad (5.92)$$

$$\begin{array}{c|cccc} RB_2: \text{ Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 0 & 0 \\ 12 & 1 & 1 & 0 & 0 \\ 21 & 0 & 0 & 0 & 1 \\ 22 & 0 & 0 & 1 & 0 \end{array} \quad (5.93)$$

$$RB_2: \{(12, 11), (12, 12), (21, 22), (22, 21)\} \quad (5.94)$$

The horizontal merging of RB_1 and RB_2 into a product rule base *RB* will be denoted by RB_1*RB_2 = *RB* where RB will be presented by the following Boolean matrix and binary relation:

$$RB: \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 1 & 1 & 0 & 0 \\ 12 & 1 & 1 & 0 & 0 \\ 21 & 0 & 0 & 0 & 1 \\ 22 & 0 & 0 & 1 & 0 \end{array} \qquad (5.95)$$

$$RB: \{(11, 11), (11, 12), (12, 11), (12, 12), (21, 22), (22, 21)\} \qquad (5.96)$$

The rule bases above have the following properties:

- RB_1 – complete, consistent, non-exhaustive, non-monotonic,
- RB_2 – incomplete, inconsistent, exhaustive, monotonic,
- RB – complete, inconsistent, exhaustive, non-monotonic.

The examples above show that if one of the two operand matrices does not have a specific property, then this may be compensated by the absence of another property in the other operand matrix, in which case the product matrix will have one of the two lacking properties. In particular, the first operand matrix in Examples 5.13 and 5.15 is inconsistent but the product matrix is consistent due to the compensating effect of the second operand matrix, which is non-monotonic. Also, the second operand matrix in Examples 5.13 and 5.15 is non-monotonic but the product matrix is monotonic due to the compensating effect of the first operand matrix, which is inconsistent. In addition, the second operand matrix in Examples 5.14 and 5.16 is incomplete but the product matrix is complete due to the compensating effect of the first operand matrix, which is non-monotonic. And finally, the first operand matrix in Examples 5.14 and 5.16 is non-exhaustive but the product matrix is exhaustive due to the compensating effect of the second operand matrix, which is inconsistent.

In addition, the examples above show that if one of the two operand matrices has a specific property, then this property may be preserved in the product matrix even if the other operand matrix does not have the property. In particular, the first operand matrix in Examples 5.13 and 5.15 is monotonic as is the product matrix in these examples although the second product matrix is non-monotonic. Also, the first operand matrix in Examples 5.14 and 5.16 is complete as is the product matrix in these examples although the second operand matrix is incomplete. In addition, the second operand matrix in Examples 5.13 and 5.15 is consistent as is the product matrix in these examples although the first product matrix is inconsistent. And finally, the second operand matrix in Examples 5.14 and 5.16 is exhaustive as is the product matrix in these examples although the first operand matrix is non-exhaustive.

5.5 Horizontal Splitting Manipulation of Rule Bases

The process of splitting a single fuzzy rule base into two fuzzy rule bases in sequence is called ‘horizontal splitting’ and it is shown in Fig. 5.4. This type of manipulation can be applied to a rule base of a SRB system or a rule base that is a part of a MRB system. Obviously, as a result of this manipulation a SRB system will be represented as a number of smaller SRB systems whereas the number of layers in the corresponding level of a MRB system will be increased.

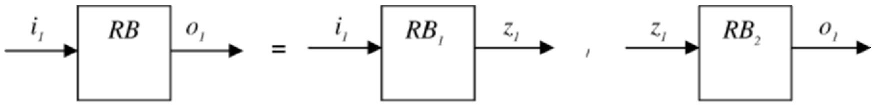

Fig. 5.4. Horizontal splitting of rule base *RB* into rule bases RB_1 and RB_2

In order to illustrate the horizontal splitting manipulation, we introduce an operation called ‘horizontal decomposition’. This operation is unary and it can be applied to only one operand at a time. The operand in this case is the Boolean matrix or the binary relation representing the operand rule base. The result from the application of this operation is a couple of Boolean matrices or binary relations representing the product rule bases.

Algorithms 5.7 and 5.8 demonstrate the application of the horizontal decomposition operation to Boolean matrices and binary relations, respectively.

Algorithm 5.7

1. Label the rows of the first product matrix with the row labels from the operand matrix.
2. Label the columns of the second product matrix with the column labels from the operand matrix.
3. Label the columns of the first product matrix as the rows of the second product matrix.
4. Set each element of the product matrices equal to 1 or 0 by mapping it from the corresponding row and column in the operand matrix, as described in step 5.
5. Find the product matrices such that the result of their multiplication using the operations for ‘addition’ and ‘multiplication’ of elements is the operand matrix, as defined by Eqs. (4.1)–(4.8) in Sect. 4.3.

Algorithm 5.8

1. Construct all maplets in the product relations such that the first element in each maplet in the operand relation is equal to the first element in a maplet in the first product relation and the second element in each maplet in the product relation is equal to the second element in a maplet in the second operand relation, as described by step 3.
2. Construct all maplets in the product relations so that the second element in a maplet in the first relation is equal to the first element in a maplet in the second relation, as described by step 3.
3. Find the product relations so that the result of their composition using the operations for 'aggregation' and 'composition' of elements is the operand relation, as defined by Eqs. (4.21)–(4.28) in Sect. 4.4.

Example 5.17

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

$$RB: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 0 \end{array} \tag{5.97}$$

$$RB: \{(1, 2)\} \tag{5.98}$$

The horizontal splitting of*RB* into product rule bases RB_1 and RB_2 will be denoted by $RB_1 = RB/RB_2$ where RB_1 and RB_2 may be presented by the following Boolean matrices and binary relations:

$$RB_1: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 0 & 0 \end{array} \tag{5.99}$$

$$RB_1: \{(1, 1), (1, 2)\} \tag{5.100}$$

$$RB_2: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{5.101}$$

$$RB_2\text{: } \{(1, 2), (2, 2)\} \tag{5.102}$$

The rule bases above have the following properties:

- *RB* – incomplete, consistent, non-exhaustive, monotonic,
- RB_1 – incomplete, inconsistent, exhaustive, monotonic,
- RB_2 – complete, consistent, non-exhaustive, non-monotonic.

Example 5.18

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

$$RB\text{: } \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 1 & 1 \end{array} \tag{5.103}$$

$$RB\text{: } \{(1, 1), (1, 2), (2, 1), (2, 2)\} \tag{5.104}$$

The horizontal splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1 / RB_2$ where RB_1 and RB_2 may be presented by the following Boolean matrices and binary relations:

$$RB_1\text{: } \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{5.105}$$

$$RB_1\text{: } \{(1, 1), (2, 1)\} \tag{5.106}$$

$$RB_2\text{: } \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 0 & 0 \end{array} \tag{5.107}$$

$$RB_2\text{: } \{(1, 1), (1, 2)\} \tag{5.108}$$

The rule bases above have the following properties:

- *RB* – complete, inconsistent, exhaustive, non-monotonic,
- RB_1 – complete, consistent, non-exhaustive, non-monotonic,
- RB_2 – incomplete, inconsistent, exhaustive, monotonic.

Example 5.19

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

$$RB:\quad \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 0 & 0 & 0 & 1 \\ 12 & 0 & 1 & 0 & 0 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 0 & 0 \end{array} \tag{5.109}$$

$$RB\text{: }\{(11, 22), (12, 12), (21, 21)\} \tag{5.110}$$

The horizontal splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1 / RB_2$ where RB_1 and RB_2 may be presented by the following Boolean matrices and binary relations:

$$RB_1:\quad \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 1 & 1 & 0 & 0 \\ 12 & 0 & 0 & 0 & 1 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 0 & 0 \end{array} \tag{5.111}$$

$$RB_1\text{: }\{(11, 11), (11, 12), (12, 22), (21, 21)\} \tag{5.112}$$

$$RB_2:\quad \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 0 & 0 & 0 & 1 \\ 12 & 0 & 0 & 0 & 1 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 1 & 0 & 0 \end{array} \tag{5.113}$$

$$RB_2\text{: }\{(11, 22), (12, 22), (21, 21), (22, 12)\} \tag{5.114}$$

The rule bases above have the following properties:

- *RB* – incomplete, consistent, non-exhaustive, monotonic,
- RB_1 – incomplete, inconsistent, exhaustive, monotonic,
- RB_2 – complete, consistent, non-exhaustive, non-monotonic.

Example 5.20

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

$$RB: \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 1 & 1 & 0 & 0 \\ 12 & 1 & 1 & 0 & 0 \\ 21 & 0 & 0 & 0 & 1 \\ 22 & 0 & 0 & 1 & 0 \end{array} \tag{5.115}$$

$$RB: \{(11, 11), (11, 12), (12, 11), (12, 12), (21, 22), (22, 21)\} \tag{5.116}$$

The horizontal splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1 / RB_2$ where RB_1 and RB_2 may be presented by the following Boolean matrices and binary relations:

$$RB_1: \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 1 & 0 & 0 \\ 12 & 0 & 1 & 0 & 0 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 0 & 1 \end{array} \tag{5.117}$$

$$RB_1: \{(11, 12), (12, 12), (21, 21), (22, 22)\} \tag{5.118}$$

$$RB_2: \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 0 & 0 \\ 12 & 1 & 1 & 0 & 0 \\ 21 & 0 & 0 & 0 & 1 \\ 22 & 0 & 0 & 1 & 0 \end{array} \tag{5.119}$$

$$RB_2: \{(12, 11), (12, 12), (21, 22), (22, 21)\} \tag{5.120}$$

The rule bases above have the following properties:

- *RB* – complete, inconsistent, exhaustive, non-monotonic,
- RB_1 – complete, consistent, non-exhaustive, non-monotonic,
- RB_2 – incomplete, inconsistent, exhaustive, monotonic.

The examples above show that if the operand matrix does not have a specific property then this property may be generated in one of the two product matrices. In particular, the operand matrices in Examples 5.17 and 5.19 are incomplete and non-exhaustive but the first product matrix in these examples is exhaustive and the second product matrix is complete. Also, the operand matrices in Examples 5.18 and 5.20 are inconsistent and non-monotonic but the first product matrix in these examples is consistent and the second product matrix is monotonic.

In addition, the examples above show that if the operand matrix does not have a specific property then one of the two product matrices does not have this property either. In particular, the operand matrices in Examples 5.17 and 5.19 are incomplete and non-exhaustive as are the first and the second product matrix in these examples, which are incomplete and non-exhaustive, respectively. Also, the operand matrices in Examples 5.18 and 5.20 are inconsistent and non-monotonic as are the first and the second product matrix in these examples, which are non-monotonic and inconsistent, respectively.

5.6 Output Merging Manipulation of Rule Bases

The process of representing two SO systems with common inputs as a MO system with the same inputs as the two SO systems is called 'output merging' and it is shown in Fig. 5.5. This type of manipulation can be applied to rule bases residing at different levels within the same layer of a MRB system. Obviously, the number of levels in this layer will be reduced as a result of this manipulation.

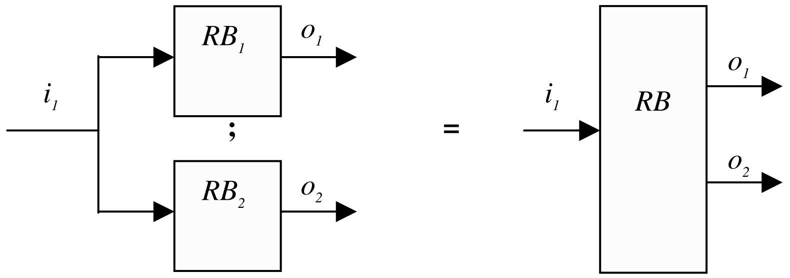

Fig. 5.5. Output merging of rule bases RB_1 and RB_2 into rule base RB

In order to illustrate the output merging manipulation, we introduce an operation called 'output composition'. This operation is binary and it can be applied to only two operands at a time. The operands in this case are the Boolean matrices or the binary relations representing the operand rule bases. The result from the application of this operation is a single Boolean matrix or binary relation representing the product rule base.

Algorithms 5.9 and 5.10 demonstrate the application of the output composition operation to Boolean matrices and binary relations, respectively.

Algorithm 5.9

1. Label the rows of the product matrix with the common row labels of the two operand matrices.
2. Label the columns of the product matrix with the sorted permutations of the column labels of the two operand matrices.
3. Go through all the elements of the operand matrices and set each element of the product matrix equal to 1 or 0, as described in steps 4 and 5.
4. If an element of the product matrix is mapped from two non-zero elements in the operand matrices, set this element equal to 1.
5. If an element of the product matrix is mapped from two elements in the operand matrices such that at least one of them is zero, set this element equal to 0.

Algorithm 5.10

1. Construct pairs of maplets from the two operand relations such that the first maplet comes from the first relation, the second maplet comes from the second relation and the first element of the maplets in each pair is the same.
2. Merge each of these pairs of maplets into a new maplet for the product relation, as described in steps 3 and 4.
3. Make the first element in each new maplet for the product relation equal to the first element from the corresponding pair of maplets for the operand relations.
4. Construct the second element in each new maplet for the product relation by concatenating the second elements from the corresponding pair of maplets for the operand relations such that the second element from the first maplet comes first and the second element from the second maplet comes second in the concatenation.
5. Generate the product binary relation containing all new maplets created in the previous two steps of this algorithm.

Example 5.21

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

RB_1: (5.121)

Inputs/Outputs	1	2
11	1	0
12	0	1
21	1	0
22	0	0

$$RB_1: \{(11, 1), (12, 2), (21, 1)\} \quad (5.122)$$

RB_2: (5.123)

Inputs/Outputs	1	2
11	1	0
12	1	0
21	0	1
22	0	0

$$RB_2: \{(11, 1), (12, 1), (21, 2)\} \quad (5.124)$$

The output merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1;RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

RB: (5.125)

Inputs/Outputs	11	12	21	22
11	1	0	0	0
12	0	0	1	0
21	0	1	0	0
22	0	0	0	0

$$RB: \{(11, 11), (12, 21), (21, 12)\} \quad (5.126)$$

The rule bases above have the following properties:

- RB_1 – incomplete, consistent, exhaustive, non-monotonic,
- RB_2 – incomplete, consistent, exhaustive, non-monotonic,
- RB – incomplete, consistent, non-exhaustive, monotonic.

Example 5.22

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

RB_1: (5.127)

Inputs/Outputs	1	2
11	1	0
12	0	1
21	0	1
22	0	0

RB_1: {(11, 1), (12, 2), (21, 2)} (5.128)

RB_2: (5.129)

Inputs/Outputs	1	2
11	1	1
12	1	0
21	0	1
22	0	0

RB_2: {(11, 1), (11, 2), (12, 1), (21, 2)} (5.130)

The output merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1;RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

RB: (5.131)

Inputs/Outputs	11	12	21	22
11	1	1	0	0
12	0	0	1	0
21	0	0	0	1
22	0	0	0	0

RB: {(11, 11), (11, 12), (12, 21), (21, 22)} (5.132)

The rule bases above have the following properties:

- RB_1 – incomplete, consistent, exhaustive, non-monotonic,
- RB_2 – incomplete, inconsistent, exhaustive, non-monotonic,
- RB – incomplete, inconsistent, exhaustive, monotonic.

Example 5.23
The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

RB_1: (5.133)

Inputs/Outputs	1	2
11	1	1
12	1	0
21	0	1
22	0	0

RB_1: {(11, 1), (11, 2), (12, 1), (21, 2)} (5.134)

RB_2: (5.135)

Inputs/Outputs	1	2
11	0	1
12	1	0
21	1	0
22	0	0

RB_2: {(11, 2), (12, 1), (21, 1)} (5.136)

The output merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1;RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

RB: (5.137)

Inputs/Outputs	11	12	21	22
11	0	1	0	1
12	1	0	0	0
21	0	0	1	0
22	0	0	0	0

RB: {(11, 12), (11, 22), (12, 11), (21, 21)} (5.138)

The rule bases above have the following properties:

- RB_1 – incomplete, inconsistent, exhaustive, non-monotonic,
- RB_2 – incomplete, consistent, exhaustive, non-monotonic,
- RB – incomplete, inconsistent, exhaustive, monotonic.

Example 5.24
The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

RB_1: (5.139)

Inputs/Outputs	1	2
11	1	0
12	0	0
21	0	0
22	0	0

$$RB_1: \{(11, 1)\} \tag{5.140}$$

RB_2: (5.141)

Inputs/Outputs	1	2
11	0	1
12	0	0
21	0	0
22	0	0

$$RB_2: \{(11, 2)\} \tag{5.142}$$

The output merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1;RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

RB: (5.143)

Inputs/Outputs	11	12	21	22
11	0	1	0	0
12	0	0	0	0
21	0	0	0	0
22	0	0	0	0

$$RB: \{(11, 12)\} \tag{5.144}$$

The rule bases above have the following properties:

- RB_1 – incomplete, consistent, non-exhaustive, monotonic,
- RB_2 – incomplete, consistent, non-exhaustive, monotonic,
- RB – incomplete, consistent, non-exhaustive, monotonic.

Example 5.25

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

RB_1: (5.145)

Inputs/Outputs	1	2
11	0	1
12	0	1
21	0	1
22	0	1

$$RB_1: \{(11, 2), (12, 2), (21, 2), (22, 2)\} \tag{5.146}$$

RB_2: (5.147)

Inputs/Outputs	1	2
11	1	0
12	1	0
21	1	0
22	1	0

$$RB_2: \{(11, 1), (12, 1), (21, 1), (22, 1)\} \tag{5.148}$$

The output merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1;RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

RB: (5.149)

Inputs/Outputs	11	12	21	22
11	0	0	1	0
12	0	0	1	0
21	0	0	1	0
22	0	0	1	0

$$RB: \{(11, 21), (12, 21), (21, 21), (22, 21)\} \tag{5.150}$$

The rule bases above have the following properties:

- RB_1 – complete, consistent, non-exhaustive, non-monotonic,
- RB_2 – complete, consistent, non-exhaustive, non-monotonic,
- RB – complete, consistent, non-exhaustive, non-monotonic.

Example 5.26
The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

RB_1: (5.151)

Inputs/Outputs	1	2
11	0	1
12	0	1
21	0	1
22	1	0

$$RB_1: \{(11, 2), (12, 2), (21, 2), (22, 1)\} \tag{5.152}$$

RB_2: (5.153)

Inputs/Outputs	1	2
11	0	1
12	0	1
21	0	1
22	0	1

$$RB_2: \{(11, 2), (12, 2), (21, 2), (22, 2)\} \tag{5.154}$$

The output merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1;RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

RB: (5.155)

Inputs/Outputs	11	12	21	22
11	0	0	0	1
12	0	0	0	1
21	0	0	0	1
22	0	1	0	0

$$RB: \{(11, 22), (12, 22), (21, 22), (22, 12)\} \tag{5.156}$$

The rule bases above have the following properties:

- RB_1 – complete, consistent, exhaustive, non-monotonic,
- RB_2 – complete, consistent, non-exhaustive, non-monotonic,
- RB – complete, consistent, non-exhaustive, non-monotonic.

Example 5.27
The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

RB₁: (5.157)

Inputs/Outputs	1	2
11	1	0
12	1	0
21	1	0
22	1	0

$$RB_1: \{(11, 1), (12, 1), (21, 1), (22, 1)\} \tag{5.158}$$

RB_2: (5.159)

Inputs/Outputs	1	2
11	1	0
12	0	1
21	0	1
22	0	1

$$RB_2: \{(11, 1), (12, 2), (21, 2), (22, 2)\} \tag{5.160}$$

The output merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1;RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

RB: (5.161)

Inputs/Outputs	11	12	21	22
11	1	0	0	0
12	0	1	0	0
21	0	1	0	0
22	0	1	0	0

$$RB: \{(11, 11), (12, 12), (21, 12), (22, 12)\} \tag{5.162}$$

The rule bases above have the following properties:

- RB_1 – complete, consistent, non-exhaustive, non-monotonic,
- RB_2 – complete, consistent, exhaustive, non-monotonic,
- RB – complete, consistent, non-exhaustive, non-monotonic.

Example 5.28
The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices and binary relations:

$$RB_1: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 1 & 0 \\ 12 & 1 & 0 \\ 21 & 0 & 1 \\ 22 & 0 & 1 \end{array} \tag{5.163}$$

$$RB_1: \{(11, 1), (12, 1), (21, 2), (22, 2)\} \tag{5.164}$$

$$RB_2: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 1 & 0 \\ 12 & 1 & 0 \\ 21 & 0 & 1 \\ 22 & 1 & 0 \end{array} \tag{5.165}$$

$$RB_2: \{(11, 1), (12, 1), (21, 2), (22, 1)\} \tag{5.166}$$

The output merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1;RB_2 = RB$ where RB will be presented by the following Boolean matrix and binary relation:

$$RB: \quad \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 1 & 0 & 0 & 0 \\ 12 & 1 & 0 & 0 & 0 \\ 21 & 0 & 0 & 0 & 1 \\ 22 & 0 & 0 & 1 & 0 \end{array} \tag{5.167}$$

$$RB: \{(11, 11), (12, 11), (21, 22), (22, 21)\} \tag{5.168}$$

The rule bases above have the following properties:

- RB_1 – complete, consistent, exhaustive, non-monotonic,
- RB_2 – complete, consistent, exhaustive, non-monotonic,
- RB – complete, consistent, non-exhaustive, non-monotonic.

The examples above show that the product matrix is complete if the operand matrices are both complete (see Examples 5.25–5.28) but it is incomplete otherwise, i.e. if at least one of the operand matrices is incomplete (see Examples 5.21–5.24). As far as consistency is concerned, if the operand matrices are both consistent then the product matrix is also consistent (see Example 5.21 and Examples 5.24–5.28). However, if at least one of the operand matrices is inconsistent then the product matrix is inconsistent too (see Examples 5.22–5.23).

In addition, the examples above show that if the operand matrices are both exhaustive then the product matrix may be exhaustive (see Examples 5.22–5.23) as well as non-exhaustive (see Example 5.21 and Example 5.28). However, if at least one of the operand matrices is non-exhaustive then the product matrix is non-exhaustive (see Examples 5.24–5.27). As far as monotonousness is concerned, the product matrix may be monotonic if the operand matrices are both monotonic (see Example 5.24) or at least one of them is non-monotonic (see Examples 5.21–5.23). However, if the operand matrices are both non-monotonic (see Examples 5.25–5.28) then product matrix is non-monotonic too.

5.7 Output Splitting Manipulation of Rule Bases

The process of representing a MO system with two outputs as two SO systems with the same common inputs as the MO system is called 'output splitting' and it is shown in Fig. 5.6. This type of manipulation can be applied to a rule base of a SRB system or a rule base that is part of a MRB system. Obviously, as a result of this manipulation a SRB system will be represented as a number of smaller SRB systems, whereas for a MRB system the number of levels in the corresponding layer will be increased.

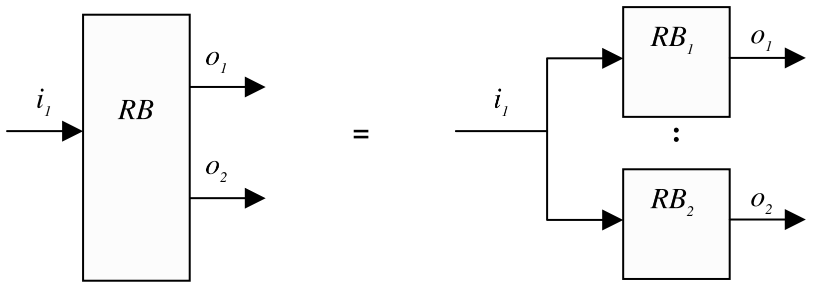

Fig. 5.6. Output splitting of rule base *RB* into rule bases RB_1 and RB_2

In order to illustrate the output splitting manipulation, we introduce an operation called 'output decomposition'. This operation is unary and it can be applied to only one operand at a time. The operand in this case is the Boolean matrix or the binary relation representing the operand rule base. The result from the application of this operation is a couple of Boolean matrices or binary relations representing the product rule bases.

Algorithms 5.11 and 5.12 demonstrate the application of the output decomposition operation to Boolean matrices and binary relations, respectively.

Algorithm 5.11

1. Label the rows of each of the two product matrices with the sorted permutations of row labels from the operand matrix.
2. Label the columns of the first product matrix with the sorted permutations of the corresponding first element from the column labels in the operand matrix.
3. Label the columns of the second product matrix with the sorted permutations of the corresponding second element from the column labels in the operand matrix.
4. Go through all the elements of the operand matrix and set each element of the two product matrices equal to 1 or 0, as described in steps 5 and 6.
5. If an element of a product matrix is mapped from a non-zero element in the operand matrix, set this element equal to 1.
6. If an element of a product matrix is mapped from a zero element in the operand matrix, set this element equal to 0.

Algorithm 5.12

1. Split each maplet from the operand relation into a pair of new maplets for the two product relations, as described in steps 2, 3 and 4.
2. Make the first element in each new maplet for each of the product relations equal to the first element from the corresponding maplet for the operand relation.
3. Make the second element in each new maplet for the first product relation equal to the corresponding first part of the second element from the maplet for the operand relation.
4. Make the second element in each new maplet for the second product relation equal to the corresponding second part of the second element from the maplet for the operand relation.
5. Generate the two product binary relations containing all new maplets created in the previous three steps of this algorithm.

Example 5.29

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

$$RB: \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 1 & 0 & 0 & 0 \\ 12 & 0 & 0 & 1 & 0 \\ 21 & 0 & 1 & 0 & 0 \\ 22 & 0 & 0 & 0 & 0 \end{array} \qquad (5.169)$$

$$RB: \{(11, 11), (12, 21), (21, 12)\} \tag{5.170}$$

The output splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1{:}RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 1 & 0 \\ 12 & 0 & 1 \\ 21 & 1 & 0 \\ 22 & 0 & 0 \end{array} \tag{5.171}$$

$$RB_1: \{(11, 1), (12, 2), (21, 1)\} \tag{5.172}$$

$$RB_2: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 1 & 0 \\ 12 & 1 & 0 \\ 21 & 0 & 1 \\ 22 & 0 & 0 \end{array} \tag{5.173}$$

$$RB_2: \{(11, 1), (12, 1), (21, 2)\} \tag{5.174}$$

The rule bases above have the following properties:

- *RB* – incomplete, consistent, non-exhaustive, monotonic,
- RB_1 – incomplete, consistent, exhaustive, non-monotonic,
- RB_2 – incomplete, consistent, exhaustive, non-monotonic.

Example 5.30

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

$$RB: \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 1 & 1 & 0 & 0 \\ 12 & 0 & 0 & 1 & 0 \\ 21 & 0 & 0 & 0 & 1 \\ 22 & 0 & 0 & 0 & 0 \end{array} \tag{5.175}$$

$$RB: \{(11, 11), (11, 12), (12, 21), (21, 22)\} \tag{5.176}$$

The output splitting of RB into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1{:}RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 1 & 0 \\ 12 & 0 & 1 \\ 21 & 0 & 1 \\ 22 & 0 & 0 \end{array} \tag{5.177}$$

$$RB_1: \{(11, 1), (12, 2), (21, 2)\} \tag{5.178}$$

$$RB_2: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 1 & 1 \\ 12 & 1 & 0 \\ 21 & 0 & 1 \\ 22 & 0 & 0 \end{array} \tag{5.179}$$

$$RB_2: \{(11, 1), (11, 2), (12, 1), (21, 2)\} \tag{5.180}$$

The rule bases above have the following properties:

- RB – incomplete, inconsistent, exhaustive, monotonic,
- RB_1 – incomplete, consistent, exhaustive, non-monotonic,
- RB_2 – incomplete, inconsistent, exhaustive, non-monotonic.

Example 5.31

The operand rule base RB is presented by the following Boolean matrix and binary relation:

$$RB: \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 1 & 0 & 1 \\ 12 & 1 & 0 & 0 & 0 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 0 & 0 \end{array} \tag{5.181}$$

$$RB: \{(11, 12), (11, 22), (12, 11), (21, 21)\} \tag{5.182}$$

The output splitting of RB into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1{:}RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

RB_1: (5.183)

Inputs/Outputs	1	2
11	1	1
12	1	0
21	0	1
22	0	0

$$RB_1: \{(11, 1), (11, 2), (12, 1), (21, 2)\} \tag{5.184}$$

RB_2: (5.185)

Inputs/Outputs	1	2
11	0	1
12	1	0
21	1	0
22	0	0

$$RB_2: \{(11, 2), (12, 1), (21, 1)\} \tag{5.186}$$

The rule bases above have the following properties:

- RB – incomplete, inconsistent, exhaustive, monotonic,
- RB_1 – incomplete, inconsistent, exhaustive, non-monotonic,
- RB_2 – incomplete, consistent, exhaustive, non-monotonic.

Example 5.32

The operand rule base RB is presented by the following Boolean matrix and binary relation:

RB: (5.187)

Inputs/Outputs	11	12	21	22
11	0	1	0	0
12	0	0	0	0
21	0	0	0	0
22	0	0	0	0

$$RB\text{: } \{(11, 12)\} \tag{5.188}$$

The output splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1{:}RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

RB_1: (5.189)

Inputs/Outputs	1	2
11	1	0
12	0	0
21	0	0
22	0	0

$$RB_1\text{: } \{(11, 1)\} \tag{5.190}$$

RB_2: (5.191)

Inputs/Outputs	1	2
11	0	1
12	0	0
21	0	0
22	0	0

$$RB_2\text{: } \{(11, 2)\} \tag{5.192}$$

The rule bases above have the following properties:

- *RB* – incomplete, consistent, non-exhaustive, monotonic,
- RB_1 – incomplete, consistent, non-exhaustive, monotonic,
- RB_2 – incomplete, consistent, non-exhaustive, monotonic.

Example 5.33

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

RB: (5.193)

Inputs/Outputs	11	12	21	22
11	0	0	1	0
12	0	0	1	0
21	0	0	1	0
22	0	0	1	0

$$RB\text{: } \{(11, 21), (12, 21), (21, 21), (22, 21)\} \tag{5.194}$$

The output splitting of RB into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1{:}RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

RB_1: (5.195)

Inputs/Outputs	1	2
11	0	1
12	0	1
21	0	1
22	0	1

$$RB_1\text{: } \{(11, 2), (12, 2), (21, 2), (22, 2)\} \tag{5.196}$$

RB_2: (5.197)

Inputs/Outputs	1	2
11	1	0
12	1	0
21	1	0
22	1	0

$$RB_2\text{: } \{(11, 1), (12, 1), (21, 1), (22, 1)\} \tag{5.198}$$

The rule bases above have the following properties:

- RB – complete, consistent, non-exhaustive, non-monotonic,
- RB_1 – complete, consistent, non-exhaustive, non-monotonic,
- RB_2 – complete, consistent, non-exhaustive, non-monotonic.

Example 5.34

The operand rule base RB is presented by the following Boolean matrix and binary relation:

RB: (5.199)

Inputs/Outputs	11	12	21	22
11	0	0	0	1
12	0	0	0	1
21	0	0	0	1
22	0	1	0	0

$$RB: \{(11, 22), (12, 22), (21, 22), (22, 12)\} \quad (5.200)$$

The output splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1{:}RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

RB_1: (5.201)

Inputs/Outputs	1	2
11	0	1
12	0	1
21	0	1
22	1	0

$$RB_1: \{(11, 2), (12, 2), (21, 2), (22, 1)\} \quad (5.202)$$

RB_2: (5.203)

Inputs/Outputs	1	2
11	0	1
12	0	1
21	0	1
22	0	1

$$RB_2: \{(11, 2), (12, 2), (21, 2), (22, 2)\} \quad (5.204)$$

The rule bases above have the following properties:

- *RB* – complete, consistent, non-exhaustive, non-monotonic,
- RB_1 – complete, consistent, exhaustive, non-monotonic,
- RB_2 – complete, consistent, non-exhaustive, non-monotonic.

Example 5.35

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

RB: (5.205)

Inputs/Outputs	11	12	21	22
11	1	0	0	0
12	0	1	0	0
21	0	1	0	0
22	0	1	0	0

$$RB: \{(11, 11), (12, 12), (21, 12), (22, 12)\} \tag{5.206}$$

The output splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1{:}RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

RB_1: (5.207)

Inputs/Outputs	1	2
11	1	0
12	1	0
21	1	0
22	1	0

$$RB_1: \{(11, 1), (12, 1), (21, 1), (22, 1)\} \tag{5.208}$$

RB_2: (5.209)

Inputs/Outputs	1	2
11	1	0
12	0	1
21	0	1
22	0	1

$$RB_2: \{(11, 1), (12, 2), (21, 2), (22, 2)\} \tag{5.210}$$

The rule bases above have the following properties:

- *RB* – complete, consistent, non-exhaustive, non-monotonic,
- RB_1 – complete, consistent, non-exhaustive, non-monotonic,
- RB_2 – complete, consistent, exhaustive, non-monotonic.

Example 5.36

The operand rule base *RB* is presented by the following Boolean matrix and binary relation:

RB: (5.211)

Inputs/Outputs	11	12	21	22
11	1	0	0	0
12	1	0	0	0
21	0	0	0	1
22	0	0	1	0

$$RB\text{: } \{(11, 11), (12, 11), (21, 22), (22, 21)\} \tag{5.212}$$

The output splitting of *RB* into product rule bases RB_1 and RB_2 will be denoted by $RB = RB_1{:}RB_2$ where RB_1 and RB_2 will be presented by the following Boolean matrices and binary relations:

RB_1: (5.213)

Inputs/Outputs	1	2
11	1	0
12	1	0
21	0	1
22	0	1

$$RB_1\text{: } \{(11, 1), (12, 1), (21, 2), (22, 2)\} \tag{5.214}$$

RB_2: (5.215)

Inputs/Outputs	1	2
11	1	0
12	1	0
21	0	1
22	1	0

$$RB_2\text{: } \{(11, 1), (12, 1), (21, 2), (22, 1)\} \tag{5.216}$$

The rule bases above have the following properties:

- *RB* – complete, consistent, non-exhaustive, non-monotonic,
- RB_1 – complete, consistent, exhaustive, non-monotonic,
- RB_2 – complete, consistent, exhaustive, non-monotonic.

The examples above show that the product matrices are both complete if the operand matrix is complete (see Examples 5.33–5.36) but they are both incomplete otherwise, i.e. if the operand matrix is incomplete (see Examples 5.29–5.32). As far as consistency is concerned, if the operand matrix is consistent then the product matrices are both consistent (see Example 5.29 and Examples 5.32–5.36). However, if the operand matrix is inconsistent then at least one of the product matrices is inconsistent too (see Examples 5.30–5.31).

In addition, the examples above show that if the operand matrix is exhaustive then the product matrices are both exhaustive (see Examples 5.30–5.31). However, if the operand matrix is non-exhaustive then either both product matrices are exhaustive (see Example 5.29 and Example 5.36) or at least one of them is non-exhaustive (see Examples 5.32–5.35). As far as monotonousness is concerned, if the operand matrix is monotonic then the product matrices may be both monotonic (see Example 5.32) or non-monotonic (see Examples 5.29–5.31). However, if the operand matrix in non-monotonic (see Examples 5.33–5.36) then both product matrices are non-monotonic too.

5.8 Comparative Analysis of Formal Manipulation Techniques

The basic operations for formal manipulation of fuzzy rule bases introduced in Sects. 5.2–5.7 are a powerful tool for analysis and synthesis of fuzzy systems. In particular, the three merging operations can be used for building complex fuzzy systems by composing simpler systems whereas the three splitting operations can be used for studying complex fuzzy systems by decomposing them into simpler systems. In this case, the three merging operations have an active impact on the operand fuzzy systems whereas the three splitting operations have a passive impact on the operand fuzzy system.

The vertical and output operations affect the number of levels within a particular layer whereas the horizontal operations affect the number of layers within a particular level. Also, in the case of merging operations the result is always guaranteed with a unique solution, whereas in the case of splitting operations the result is not guaranteed but if it is then the solution may be non-unique.

The considerations presented above on formal manipulation techniques for fuzzy rule bases provide essential information about the main characteristics of these techniques. These characteristics are summarised in Table 5.1.

Table 5.1. Characteristics of formal manipulation techniques for fuzzy rule bases

Technique/ Characteristic	Task	Impact	Component	Result	Solution
Vertical merging	synthesis	active	level	guaranteed	unique
Vertical splitting	analysis	passive	level	not guaranteed	non-unique
Horizontal merging	synthesis	active	layer	guaranteed	unique
Horizontal splitting	analysis	passive	layer	not guaranteed	non-unique
Output merging	synthesis	active	level	guaranteed	unique
Output splitting	analysis	passive	level	not guaranteed	non-unique

The successive application of splitting an operand rule base RB_O into two rule bases RB_1 and RB_2 and merging them together will give a product rule base RB_P that is equal to RB_O. However, the successive application of merging two operand rule bases into a rule base and splitting it will not necessarily give product rule bases that are equal to the operand rule bases. In other words, the sequence of a merging and a splitting operation is an identity mapping only if the first operation in the sequence is the splitting operation. This conclusion applies to vertical, horizontal and output merging and splitting operations, as shown by Eqs. (5.217)–(5.219).

$$\text{If } RB_O = RB_1 - RB_2 \text{ and } RB_1 + RB_2 = RB_P \text{ then } RB_P = RB_O \tag{5.217}$$

$$\text{If } RB_O = RB_1 / RB_2 \text{ and } RB_1 * RB_2 = RB_P \text{ then } RB_P = RB_O \tag{5.218}$$

$$\text{If } RB_O = RB_1 : RB_2 \text{ and } RB_1 ; RB_2 = RB_P \text{ then } RB_P = RB_O \tag{5.219}$$

The three implications above can be easily validated by comparing all examples from Sects. 5.3, 5.5 and 5.7 with their counterpart examples from Sects. 5.2, 5.4 and 5.6.

5.9 Application Range of Formal Manipulation Techniques

The six formal manipulation techniques introduced in this chapter are applicable to a wide range of fuzzy rule based systems. These techniques can be applied to Mamdami, Sugeno and Tsukamoto systems, CON and DIS systems, MO and SO systems, FF and FB systems, as well as SRB and MRB systems.

Examples 5.1–5.36 describe implicitly a fuzzy system of Mamdami or Tsukamoto type. In order to apply the associated rule base manipulation algorithms to Sugeno systems, the crisp outputs in the operand rule bases have to fuzzified, i.e. converted into linguistic values. In this case, the linguistic values of the outputs in the product rule bases can be converted back into crisp values, if necessary.

Examples 5.1–5.36 can be extended easily in accordance with the considerations in Sect. 4.6, if we would like them to describe explicitly Mamdami, Sugeno and Tsukamoto systems. However, this has not been done in this chapter in order to simplify the notations and to put the emphasis on the manipulation rather than the presentation process, which was dealt with in the previous chapter.

As far as CON and DIS systems are concerned, the formal manipulation techniques are directly applicable to them. In this case, the operand and the product rule bases must be of the same type, i.e. CADR, DADR, CACR or DACR.

With respect to MO and SO systems, the formal manipulation techniques are also directly applicable by matching appropriately the number of outputs in the operand and the product rule bases. For example, in the case of vertical and output merging the number of outputs in the product rule base is equal to the sum of the number of outputs in the operand rule bases. Similarly, in the case of vertical and output splitting the number of outputs in the operand rule base is equal to the sum of the number of outputs in the product rule bases. However, in the case of horizontal merging the number of outputs in the product rule base is equal to the number of outputs in the second operand rule base and in the case of horizontal splitting the number of outputs in the second product rule base is equal to the number of outputs in the operand rule base.

In the case of FF systems, the formal manipulation techniques are directly applicable as already demonstrated by Examples 5.1–5.36. However, if any of the operand rule bases are of FB type, then they have to be converted into an *equivalent rule base* (ERB) of FF type before the manipulation techniques can be applied. This conversion requires the specification of the existing output-input interconnections and their type, e.g. local or global, and it is discussed in detail in Chapter 8.

Examples 5.1–5.36 demonstrate the application of formal manipulation techniques in the context of both SRB and MRB systems. In merging manipulations, the operand rule bases are usually part of a MRB system whereas the product rule base may be a SRB system or part of a MRB system. However, in splitting manipulations the operand rule base may be a SRB system or part of a MRB system whereas the product rule bases are usually part of a MRB system.

The vertical, horizontal and output manipulation techniques facilitate the complexity management in fuzzy systems. These techniques allow the compressed information about a fuzzy rule base in the form of a Boolean matrix or binary relation to be manipulated appropriately for the purpose of analysis and synthesis of fuzzy systems.

However, the manipulation techniques demonstrated so far are only with common types of operand rule bases, which do not always make obvious the impact on the corresponding product rule bases. Also, these common types of rule bases may have only limited impact on the properties and the structure of the product rule base. In order to make this impact more obvious and significant, a specific study on formal manipulation with some special operand rule bases is required and this is discussed in detail in the next chapter.

6 Formal Manipulation with Special Rule Bases

6.1 Preliminaries on Special Rule Bases

The techniques for formal manipulation of fuzzy rule bases introduced in the previous chapter facilitate the complexity management in fuzzy systems. These techniques allow the compressed information about the fuzzy system contained in the Boolean matrix and the binary relation to be reorganised for the purpose of analysis or synthesis. This reorganisation is carried out by representing a single operand rule base with a couple of product rule bases, or alternatively, by representing a couple of operand rule bases with a single product rule base. In either case, no special requirements are usually placed on the corresponding operand or product rule bases, i.e. the latter may be any arbitrary rule bases.

However, if we want to reinforce the change of the properties of a SRB system or to change the overall structure of a MRB system by manipulating the corresponding Boolean matrix or binary relation, we may have to use some special rule bases. Such rule bases are the IRB, the TRB and the PRB. These special rule bases are described by the following definitions.

Definition 6.1
A transpose Boolean matrix of a given Boolean matrix is the matrix obtained by representing the rows and the columns of the given matrix as columns and rows, respectively.

Definition 6.2
A transpose binary relation of a given binary relation is the relation obtained by swapping the first and the second element in all maplets of the given relation.

Definition 6.3
A permutation Boolean matrix is a square matrix with exactly one non-zero element in each row and each column.

Definition 6.4
A permutation binary relation is a square relation in which each element of the two participating sets appears only once as a first or a second element in a maplet.

Definition 6.5

An IRB is denoted by RB_I and is formally presented by an identity Boolean matrix or an identity binary relation.

Definition 6.6

A TRB is denoted by RB_T and is formally presented by a transpose Boolean matrix or a transpose binary relation.

Definition 6.7

A PRB is denoted by RB_P and is formally presented by a permutation Boolean matrix or a permutation binary relation.

For completeness and consistency, Definitions 6.1–6.4 on transpose and permutation related concepts must be considered together with Definitions 4.16 and 4.34 on identity related concepts. As far as Definitions 6.5–6.7 are concerned, they follow directly from the above mentioned definitions in that they state how the special rule bases are denoted and presented formally. Integer tables are not used in this chapter because they do not lend themselves easily to formal manipulation.

The formal manipulation techniques with special rule bases introduced here can be applied mainly in the context of MRB systems as the corresponding manipulations usually affect the interconnections between individual rule bases at some stage of the manipulation process. However, we may have to deal only with SRB systems at other stages of this process.

For generality, a special rule base which is an IRB, TRB or PRB will be referred to as an *identity / transpose / permutation rule base* (I/T/P RB). This chapter presents the following manipulation techniques with such rule bases:

- vertical merging from below and above,
- vertical splitting,
- horizontal merging from right and left,
- horizontal splitting,
- output merging from below and above,
- output splitting.

The above manipulation techniques are illustrated in Figs. 6.1–6.9 and by numerous examples further in this chapter. The purpose of the examples is to demonstrate the manipulation techniques with special rule bases and to show the potential impact of the manipulation on the structure of the rule bases involved, i.e. how the patterns from the operand rule bases are replicated or transformed in the product rule bases.

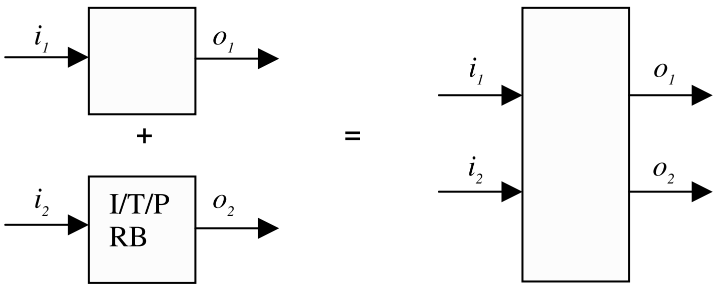

Fig. 6.1. Vertical merging with an identity / transpose / permutation rule base from below

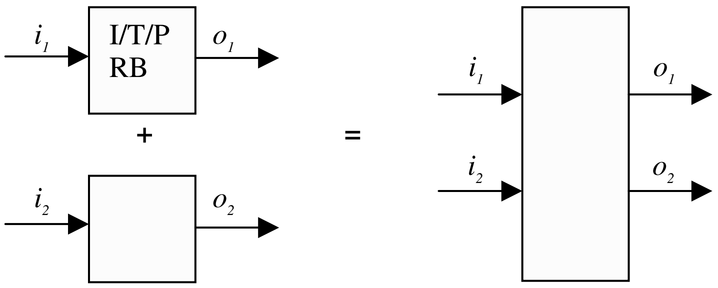

Fig. 6.2. Vertical merging with an identity / transpose / permutation rule base from above

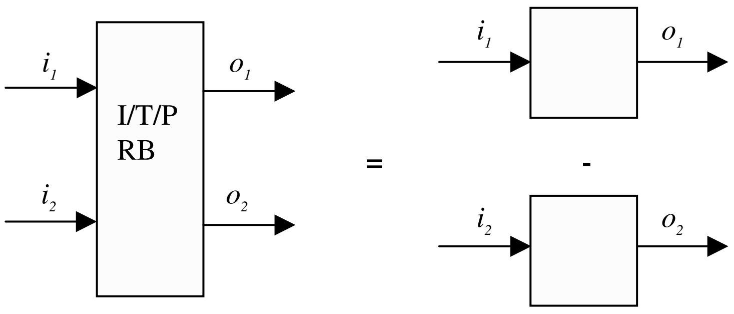

Fig. 6.3. Vertical splitting of an identity/transpose/permutation rule base

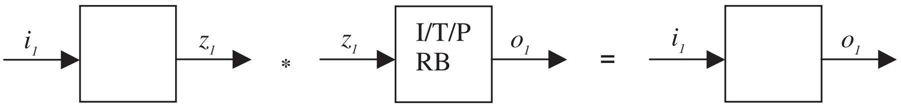

Fig. 6.4. Horizontal merging with an identity/transpose/permutation rule base from right

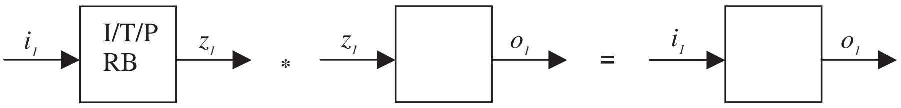

Fig. 6.5. Horizontal merging with an identity/transpose/permutation rule base from left

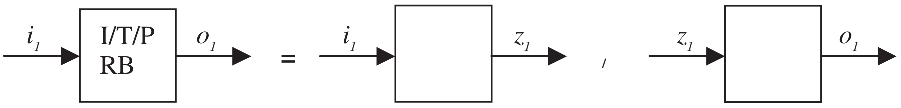

Fig. 6.6. Horizontal splitting of an identity/transpose/permutation rule base

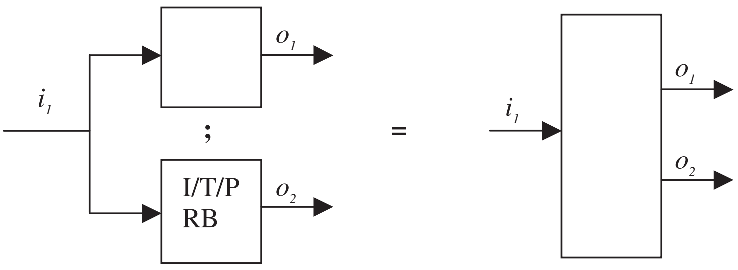

Fig. 6.7. Output merging with an identity/transpose/permutation rule base from below

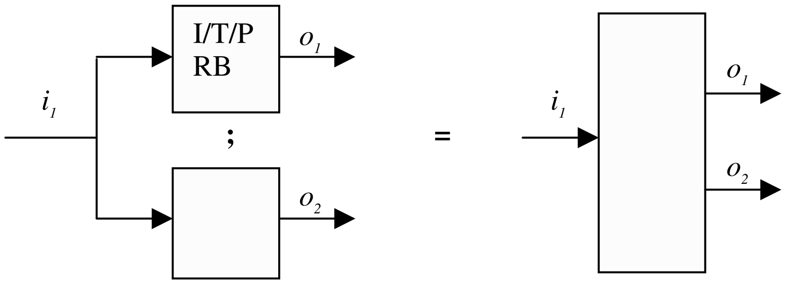

Fig. 6.8. Output merging with an identity/transpose/permutation rule base from above

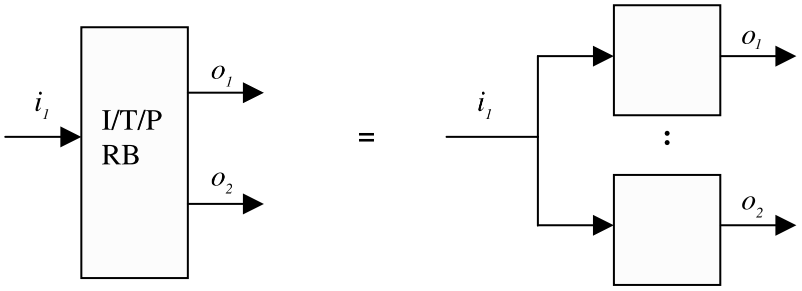

Fig. 6.9. Output splitting of an identity/transpose/permutation rule base

6.2 Manipulation with Identity Rule Bases

Example 6.1

This example demonstrates the technique of vertical merging with an IRB from below. The operand rule bases *RB* and RB_I are presented by the following Boolean matrices and binary relations:

$$RB: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{6.1}$$

$$RB: \{(1, 2), (2, 1)\} \tag{6.2}$$

$$RB_I: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.3}$$

$$RB_I\text{: } \{(1, 1), (2, 2)\} \tag{6.4}$$

The vertical merging of RB and RB_I into a product rule base RB_M will be denoted by $RB+RB_I = RB_M$ where RB_M will be presented by the following block Boolean matrix and binary relation:

$$RB_M\text{: } \begin{array}{c|cc|cc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 0 & 0 & 1 & 0 \\ 12 & 0 & 0 & 0 & 1 \\ \hline 21 & 1 & 0 & 0 & 0 \\ 22 & 0 & 1 & 0 & 0 \end{array} \tag{6.5}$$

$$RB_M\text{: } \{(11, 21), (12, 22), (21, 11), (22, 12)\} \tag{6.6}$$

In this case, the positions of the non-zero blocks in RB_M map the positions of the non-zero elements in the Boolean matrix from above RB whereby each non-zero block in RB_M is equal to the identity Boolean matrix from below RB_I.

Example 6.2

This example demonstrates the technique of vertical merging with an IRB from above. The operand rule bases RB_I and RB are presented by the following Boolean matrices and binary relations:

$$RB_I\text{: } \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.7}$$

$$RB_I\text{: } \{(1, 1), (2, 2)\} \tag{6.8}$$

$$RB\text{: } \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{6.9}$$

$$RB\text{: } \{(1, 2), (2, 1)\} \tag{6.10}$$

The vertical merging of RB_I and RB into a product rule base RB_M will be denoted by $RB_I+RB = RB_M$ where RB_M will be presented by the block following Boolean matrix and binary relation:

$$RB_M: \begin{array}{c|cc|cc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 0 & 1 & 0 & 0 \\ 12 & 1 & 0 & 0 & 0 \\ \hline 21 & 0 & 0 & 0 & 1 \\ 22 & 0 & 0 & 1 & 0 \end{array} \qquad (6.11)$$

$$RB_M: \{(11, 12), (12, 11), (21, 22), (22, 21)\} \qquad (6.12)$$

In this case, the positions of the non-zero blocks in RB_M map the positions of the non-zero elements in the identity Boolean matrix from above RB_I whereby each non-zero block in RB_M is equal to the Boolean matrix from below RB.

Example 6.3

This example demonstrates the technique of vertical splitting of an IRB. The operand rule base RB_I is presented by the following block Boolean matrix and binary relation:

$$RB_I: \begin{array}{c|cc|cc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 1 & 0 & 0 & 0 \\ 12 & 0 & 1 & 0 & 0 \\ \hline 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 0 & 1 \end{array} \qquad (6.13)$$

$$RB_I: \{(11, 11), (12, 12), (21, 21), (22, 22)\} \qquad (6.14)$$

The vertical splitting of RB_I into product rule bases RB_{S1} and RB_{S2} will be denoted by $RB_I = RB_{S1}–RB_{S2}$ where RB_{S1} and RB_{S2} will be presented by the following Boolean matrices and binary relations:

$$RB_{S1}: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \qquad (6.15)$$

$$RB_{S1}: \{(1, 1), (2, 2)\} \qquad (6.16)$$

$$RB_{S2}: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.17}$$

$$RB_{S2}: \{(1, 1), (2, 2)\} \tag{6.18}$$

In this case, each of the two product Boolean matrices RB_{S1} and RB_{S2} is an IRB that is equal to the non-zero on-diagonal blocks in the operand Boolean matrix RB_I.

Example 6.4

This example demonstrates the technique of horizontal merging with an IRB from right. The operand rule bases *RB* and RB_I are presented by the following Boolean matrices and binary relations:

$$RB: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{6.19}$$

$$RB: \{(1, 2), (2, 1)\} \tag{6.20}$$

$$RB_I: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.21}$$

$$RB_I: \{(1, 1), (2, 2)\} \tag{6.22}$$

The horizontal merging of *RB* and RB_I into a product rule base RB_M will be denoted by $RB{*}RB_I = RB_M$ where RB_M will be presented by the following Boolean matrix and binary relation:

$$RB_M: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{6.23}$$

$$RB_M: \{(1, 2), (2, 1)\} \tag{6.24}$$

In this case, the product rule base RB_M is the same as the operand rule base from left RB. This follows from the equality of the corresponding Boolean matrices and binary relations for these rule bases, as shown by Eqs. (6.19)–(6.20) and Eqs. (6.23)–(6.24).

Example 6.5

This example demonstrates the technique of horizontal merging with an IRB from left. The operand rule bases RB_I and RB are presented by the following Boolean matrices and binary relations:

$$RB_I: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.25}$$

$$RB_I: \{(1, 1), (2, 2)\} \tag{6.26}$$

$$RB: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{6.27}$$

$$RB: \{(1, 2), (2, 1)\} \tag{6.28}$$

The horizontal merging of RB_I and RB into a product rule base RB_M will be denoted by $RB_I{*}RB = RB_M$ where RB_M will be presented by the following Boolean matrix and binary relation:

$$RB_M: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{6.29}$$

$$RB_M: \{(1, 2), (2, 1)\} \tag{6.30}$$

In this case, the product rule base RB_M is the same as the operand rule base from right RB. This follows from the equality of the corresponding Boolean matrices and binary relations for these rule bases, as shown by Eqs. (6.27)–(6.28) and Eqs. (6.29)–(6.30).

Example 6.6

This example demonstrates the technique of horizontal splitting of an IRB. The operand rule base RB_I is presented by the following Boolean matrix and binary relation:

$$RB_I: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{array} \quad (6.31)$$

$$RB_I: \{(1, 1), (2, 2), (3, 3)\} \quad (6.32)$$

The horizontal splitting of RB_I into product rule bases RB_{S1} and RB_{S2} will be denoted by $RB_I = RB_{S1}/RB_{S2}$ where RB_{S1} and RB_{S2} may be presented by the following Boolean matrices and binary relations:

$$RB_{S1}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 \end{array} \quad (6.33)$$

$$RB_{S1}: \{(1, 2), (2, 3), (3, 1)\} \quad (6.34)$$

$$RB_{S2}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \end{array} \quad (6.35)$$

$$RB_{S2}: \{(1, 3), (2, 1), (3, 2)\} \quad (6.36)$$

In this case, the product rule bases RB_{S1} and RB_{S2} are both PRBs. Also, each of them is a TRB with respect to the other. This follows from the comparison of the corresponding Boolean matrices and binary relations for these rule bases, as shown by Eqs. (6.33)–(6.34) and Eqs. (6.35)–(6.36).

Example 6.7

This example demonstrates the technique of output merging with an IRB from below. The operand rule bases RB and RB_I are presented by the following Boolean matrices and binary relations:

$$RB: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{6.37}$$

$$RB: \{(1, 2), (2, 1)\} \tag{6.38}$$

$$RB_I: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.39}$$

$$RB_I: \{(1, 1), (2, 2)\} \tag{6.40}$$

The output merging of *RB* and RB_I into a product rule base RB_M will be denoted by $RB;RB_I = RB_M$ where RB_M will be presented by the following block Boolean matrix and binary relation:

$$RB_M: \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 \end{array} \tag{6.41}$$

$$RB_M: \{(1, 21), (2, 12)\} \tag{6.42}$$

In this case, the positions of the non-zero 1×2 sub-blocks in the product Boolean matrix RB_M map the positions of the non-zero elements in the first operand Boolean matrix *RB* whereas the positions of the non-zero elements within these sub-blocks map the positions of the non-zero elements in the second Boolean matrix RB_I.

Example 6.8

This example demonstrates the technique of output merging with an IRB from above. The operand rule bases RB_I and *RB* are presented by the following Boolean matrices and binary relations:

$$RB_I: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.43}$$

$$RB_I\text{: } \{(1, 1), (2, 2)\} \tag{6.44}$$

$$RB\text{:} \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{6.45}$$

$$RB\text{: } \{(1, 2), (2, 1)\} \tag{6.46}$$

The output merging of RB_I and RB into a product rule base RB_M will be denoted by $RB_I;RB = RB_M$ where RB_M will be presented by the following block Boolean matrix and binary relation:

$$RB_M\text{:} \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 1 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 \end{array} \tag{6.47}$$

$$RB_M\text{: } \{(1, 12), (2, 21)\} \tag{6.48}$$

In this case, the positions of the non-zero 1×2 sub-blocks in the product Boolean matrix RB_M map the positions of the non-zero elements in the first operand Boolean matrix RB_I whereas the positions of the non-zero elements within these sub-blocks map the positions of the non-zero elements in the second Boolean matrix *RB*.

Example 6.9

This example demonstrates the technique of output splitting of an IRB. The operand rule base RB_I is presented by the following block Boolean matrix and binary relation:

$$RB_I\text{:} \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 1 & 0 & 0 & 0 \\ 12 & 0 & 1 & 0 & 0 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 0 & 1 \end{array} \tag{6.49}$$

$$RB_I\text{: } \{(11, 11), (12, 12), (21, 21), (22, 22)\} \tag{6.50}$$

The output splitting of RB_I into product rule bases RB_{S1} and RB_{S2} will be denoted by $RB = RB_{S1}:RB_{S2}$ where RB_{S1} and RB_{S2} will be presented by the following block Boolean matrices and binary relations:

$$RB_{S1}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 1 & 0 \\ 12 & 1 & 0 \\ 21 & 0 & 1 \\ 22 & 0 & 1 \end{array} \tag{6.51}$$

$$RB_{S1}: \{(11, 1), (12, 1), (21, 2), (22, 2)\} \tag{6.52}$$

$$RB_{S2}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 1 & 0 \\ 12 & 0 & 1 \\ 21 & 1 & 0 \\ 22 & 0 & 1 \end{array} \tag{6.53}$$

$$RB_{S2}: \{(11, 1), (12, 1), (21, 2), (22, 2)\} \tag{6.54}$$

In this case, the blocks in the second product Boolean matrix RB_{S2} map the non-zero on-diagonal blocks in the operand Boolean matrix RB_I whereas the positions of the non-zero elements in the first product Boolean matrix RB_{S1} map the positions of the non-zero 1×2 sub-blocks in RB_I.

6.3 Manipulation with Transpose Rule Bases

Example 6.10

This example demonstrates the technique of vertical merging with a TRB from below and above. In this case, the two operand rule bases RB_{T1} and RB_{T2} are transpose to each other and they are presented by the following Boolean matrices and binary relations:

$$RB_{T1}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{6.55}$$

$$RB_{T1}: \{(1, 1), (2, 1)\} \tag{6.56}$$

$$RB_{T2}: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 0 & 0 \end{array} \tag{6.57}$$

$$RB_{T2}: \{(1, 1), (1, 2)\} \tag{6.58}$$

The vertical merging of RB_{T1} and RB_{T2} into a product rule base RB_M will be denoted by RB_{T1}+RB_{T2} = RB_M where RB_M will be presented by the following block Boolean matrix and binary relation:

$$RB_M: \quad \begin{array}{c|cc|cc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 1 & 1 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 \\ 21 & 1 & 1 & 0 & 0 \\ 22 & 0 & 0 & 0 & 0 \end{array} \tag{6.59}$$

$$RB_M: \{(11, 11), (11, 12), (21, 11), (21, 12)\} \tag{6.60}$$

The positions of the non-zero blocks in the product Boolean matrix RB_M map the positions of the non-zero elements in the operand Boolean matrix from above RB_{T1} whereby each non-zero block in RB_M is equal to the operand Boolean matrix from below RB_{T2}.

Example 6.11

This example demonstrates the technique of vertical splitting of TRBs, i.e. rule bases which are transpose to each other. The two operand rule bases RB_{T1} and RB_{T2} are handled separately with the purpose to find out how the symmetrical patterns in them are replicated in the corresponding product rule bases.

The operand rule base RB_{T1} is presented by the following block Boolean matrix and binary relation:

$$RB_{T1}: \quad \begin{array}{c|cc|cc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 1 & 0 \\ 12 & 0 & 0 & 1 & 0 \\ 21 & 1 & 0 & 0 & 0 \\ 22 & 1 & 0 & 0 & 0 \end{array} \tag{6.61}$$

$$RB_{T1}: \{(11, 21), (12, 21), (21, 11), (22, 11)\} \tag{6.62}$$

The vertical splitting of RB_{T1} into product rule bases RB_{S1} and RB_{S2} will be denoted by $RB_{T1} = RB_{S1}–RB_{S2}$ where RB_{S1} and RB_{S2} will be presented by the following Boolean matrices and binary relations:

$$RB_{S1}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{6.63}$$

$$RB_{S1}: \{(1, 2), (2, 1)\} \tag{6.64}$$

$$RB_{S2}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{6.65}$$

$$RB_{S2}: \{(1, 1), (2, 1)\} \tag{6.66}$$

The positions of the non-zero elements in the first product Boolean matrix RB_{S1} map the positions of the two identical non-zero blocks in the operand Boolean matrix RB_{T1}. As far as the second product Boolean matrix RB_{S2} is concerned, it is the same as each of the two identical non-zero blocks in the operand Boolean matrix RB_{T1}.

The operand rule base RB_{T2} is presented by the following block Boolean matrix and binary relation:

$$RB_{T2}: \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 1 & 1 \\ 12 & 0 & 0 & 0 & 0 \\ 21 & 1 & 1 & 0 & 0 \\ 22 & 0 & 0 & 0 & 0 \end{array} \tag{6.67}$$

$$RB_{T2}: \{(11, 21), (11, 22), (21, 11), (21, 12)\} \tag{6.68}$$

The vertical splitting of RB_{T2} into product rule bases RB_{S3} and RB_{S4} will be denoted by $RB_{T2} = RB_{S3}–RB_{S4}$ where RB_{S3} and RB_{S4} will be presented by the following Boolean matrices and binary relations:

$$RB_{S3}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{6.69}$$

$$RB_{S3}: \{(1, 2), (2, 1)\} \tag{6.70}$$

$$RB_{S4}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 0 & 0 \end{array} \tag{6.71}$$

$$RB_{S4}: \{(1, 1), (1, 2)\} \tag{6.72}$$

The positions of the non-zero elements in the first product Boolean matrix RB_{S3} map the positions of the two identical non-zero blocks in the operand Boolean matrix RB_{T2}. As far as the second product Boolean matrix RB_{S4} is concerned, it is the same as each of the two identical non-zero blocks in the operand Boolean matrix RB_{T2}.

A more detailed inspection of this example shows that the patterns from the two operand rule bases RB_{T1} and RB_{T2} are replicated in the corresponding product rule bases RB_{S1}, RB_{S2}, RB_{S3} and RB_{S4} in a manner that matches the symmetry in RB_{T1} and RB_{T2}. In particular, RB_{S1} is equal to RB_{S3} whereas RB_{S2} and RB_{S4} are transpose to each other.

Example 6.12

This example demonstrates the technique of horizontal merging of TRBs. The two operand rule bases RB_{T1} and RB_{T2} are handled together in two separate cases whereby in the second case their positions are swapped in relation to the first case.

For the first case, the operand rule bases RB_{T1} and RB_{T2} are presented by the following Boolean matrices and binary relations:

$$RB_{T1}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \end{array} \tag{6.73}$$

$$RB_{T1}: \{(1, 1), (2, 1), (3, 2)\} \tag{6.74}$$

$$RB_{T2}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 \end{array} \tag{6.75}$$

$$RB_{T2}\text{: } \{(1, 1), (1, 2), (2, 3)\} \tag{6.76}$$

The horizontal merging of RB_{T1} and RB_{T2} into a product rule base RB_{M12} will be denoted by $RB_{T1}*RB_{T} = RB_{M12}$ where RB_{M12} will be presented by the following block Boolean matrix and binary relation:

$$RB_{M12}\text{: } \begin{array}{c|ccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 1 & 0 \\ 2 & 1 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{array} \tag{6.77}$$

$$RB_{M12}\text{: } \{(1, 1), (1, 2), (2, 1), (2, 2), (3, 3)\} \tag{6.78}$$

For the second case, the positions of the operand rule bases RB_{T1} and RB_{T2} are swapped, i.e. RB_{T2} is from left and RB_{T1} is from right. Therefore, the horizontal merging of RB_{T2} and RB_{T1} into a product rule base RB_{M21} will be denoted by $RB_{T2}*RB_{T1} = RB_{M21}$ where RB_{M21} will be presented by the following block Boolean matrix and binary relation:

$$RB_{M21}\text{: } \begin{array}{c|ccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 \end{array} \tag{6.79}$$

$$RB_{M21}\text{: } \{(1, 1), (2, 2)\} \tag{6.80}$$

A more detailed inspection of this example shows that the symmetrical patterns from the two operand rule bases RB_{T1} and RB_{T2} are transformed in the corresponding product rule bases RB_{M12} and RB_{M21}. In particular, RB_{M12} and RB_{M21} both have a block-diagonal structure, i.e. all blocks outside the main block-diagonal are zeros. In the first case, the main diagonal blocks of RB_{M12} are of universal type, i.e. with all elements non-zero. In the second case, the main diagonal blocks of RB_{M21} are of either identity type, i.e. with all elements on the main diagonal non-zero and the remaining elements zeros, or of null type, i.e. with all elements zeros.

Example 6.13

This example demonstrates the technique of horizontal splitting of TRBs whereby the two operand rule bases RB_{T1} and RB_{T2} are handled separately. In this case, the operand rule base RB_{T1} is presented by the following Boolean matrix and binary relation:

$$RB_{T1}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 \end{array} \tag{6.81}$$

$$RB_{T1}: \{(1, 2), (2, 3)\} \tag{6.82}$$

The horizontal splitting of RB_{T1} into product rule bases RB_{S1} and RB_{S2} will be denoted by $RB_{T1} = RB_{S1} / RB_{S2}$ where RB_{S1} and RB_{S2} may be presented by the following Boolean matrices and binary relations:

$$RB_{S1}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 1 \\ 3 & 0 & 0 & 0 \end{array} \tag{6.83}$$

$$RB_{S1}: \{(1, 1), (2, 2), (2, 3)\} \tag{6.84}$$

$$RB_{S2}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 0 & 1 \end{array} \tag{6.85}$$

$$RB_{S2}: \{(1, 2), (2, 3), (3, 3)\} \tag{6.86}$$

The zero row in the first product Boolean matrix RB_{S1} maps the zero row in the operand Boolean matrix RB_{T1}. As far as the second product Boolean matrix RB_{S2} is concerned, its zero column maps the zero column in the operand Boolean matrix RB_{T1}.

The operand rule base RB_{T2} is presented by the following Boolean matrix and binary relation:

$$RB_{T2}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \end{array} \tag{6.87}$$

$$RB_{T2}: \{(2, 1), (3, 2)\} \qquad (6.88)$$

The horizontal splitting of RB_{T2} into product rule bases RB_{S3} and RB_{S4} will be denoted by $RB_{T2} = RB_{S3} / RB_{S4}$ where RB_{S3} and RB_{S4} may be presented by the following Boolean matrices and binary relations:

$$RB_{S3}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 1 \end{array} \qquad (6.89)$$

$$RB_{S3}: \{(2, 1), (3, 2), (3, 3)\} \qquad (6.90)$$

$$RB_{S4}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 0 & 1 & 0 \end{array} \qquad (6.91)$$

$$RB_{S4}: \{(1, 1), (2, 2), (3, 2)\} \qquad (6.92)$$

The zero row in the first product Boolean matrix RB_{S3} maps the zero row in the operand Boolean matrix RB_{T2}. As far as the second product Boolean matrix RB_{S4} is concerned, its zero column maps the zero column in the operand Boolean matrix RB_{T2}.

A more detailed inspection of this example shows that the patterns from the two operand rule bases RB_{T1} and RB_{T2} are transformed in the corresponding product rule bases RB_{S1}, RB_{S2}, RB_{S3} and RB_{S4} in a manner that matches the symmetry in RB_{T1} and RB_{T2}. In particular, RB_{S2} and RB_{S3} are transpose to each other as are RB_{S1} and RB_{S4}. In other words, the transposition of the operand Boolean matrix RB_{T1} into RB_{T2} has an inverse transformation effect on the corresponding product Boolean matrices, i.e. RB_{S1} and RB_{S2} must be transposed and have their positions swapped in order to obtain their counterparts RB_{S3} and RB_{S4}.

Example 6.14

This example demonstrates the technique of output merging with a TRB from below and above. In this case, the two operand rule bases RB_{T1} and RB_{T2} are transpose to each other and they are presented by the following Boolean matrices and binary relations:

$$RB_{T1}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{6.93}$$

$$RB_{T1}: \{(1, 2), (2, 2)\} \tag{6.94}$$

$$RB_{T2}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 1 & 1 \end{array} \tag{6.95}$$

$$RB_{T2}: \{(2, 1), (2, 2)\} \tag{6.96}$$

The output merging of RB_{T1} and RB_{T2} into a product rule base RB_M will be denoted by RB_{T1}; $RB_{T2} = RB_M$ where RB_M will be presented by the following block Boolean matrix and binary relation:

$$RB_M: \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 1 \end{array} \tag{6.97}$$

$$RB_M: \{(2, 21), (2, 22)\} \tag{6.98}$$

The position of the non-zero block in the product Boolean matrix RB_M maps the position of the non-zero 2×1 sub-block in the operand Boolean matrix from above RB_{T1} whereby this non-zero block is equal to the operand Boolean matrix from below RB_{T2}.

Example 6.15

This example demonstrates the technique of output splitting of TRBs whereby the two operand rule bases RB_{T1} and RB_{T2} are handled separately. In this case, the operand rule base RB_{T1} is presented by the following block Boolean matrix and binary relation:

$$RB_{T1}: \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 1 & 0 & 0 \\ 12 & 0 & 0 & 1 & 0 \\ 21 & 0 & 0 & 0 & 1 \\ 22 & 0 & 1 & 0 & 0 \end{array} \tag{6.99}$$

$$RB_{T1}: \{(11, 12), (12, 21), (21, 22), (22, 12)\} \tag{6.100}$$

The output splitting of RB_{T1} into product rule bases RB_{S1} and RB_{S2} will be denoted by $RB_{T1} = RB_{S1}:RB_{S2}$ where RB_{S1} and RB_{S2} will be presented by the following Boolean matrices and binary relations:

$$RB_{S1}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 1 & 0 \\ 12 & 0 & 1 \\ 21 & 0 & 1 \\ 22 & 1 & 0 \end{array} \tag{6.101}$$

$$RB_{S1}: \{(11, 1), (12, 2), (21, 2), (22, 1)\} \tag{6.102}$$

$$RB_{S2}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 0 & 1 \\ 12 & 1 & 0 \\ 21 & 0 & 1 \\ 22 & 0 & 1 \end{array} \tag{6.103}$$

$$RB_{S2}: \{(11, 2), (12, 1), (21, 2), (22, 2)\} \tag{6.104}$$

It is obvious that the rows in the first product Boolean matrix RB_{S1} map the non-zero 1×2 sub-blocks in the operand Boolean matrix RB_{T1}. As far as the second product Boolean matrix RB_{S2} is concerned, its non-zero elements map the non-zero 1×2 sub-blocks in a Boolean matrix that is obtained by swapping the second and the third column in RB_{T1} together with their labels. This new matrix stands for a new rule base RB_{T1N} in which the positions of the two outputs have been swapped, i.e. the second output from RB_{T1} has become first whereas its first output has become second. Obviously, the corresponding binary relation does not change as a result of this column swap because the column labels are swapped as well.

The new rule base RB_{T1N} is presented by the following block Boolean matrix and binary relation:

$$RB_{T1N}: \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 21 & 12 & 22 \\ 11 & 0 & 0 & 1 & 0 \\ 12 & 0 & 1 & 0 & 0 \\ 21 & 0 & 0 & 0 & 1 \\ 22 & 0 & 0 & 1 & 0 \end{array} \tag{6.105}$$

$$RB_{T1N}: \{(11, 12), (12, 21), (21, 22), (22, 12)\} \tag{6.106}$$

The operand rule base RB_{T2} is presented by the following block Boolean matrix and binary relation:

$$RB_{T2}: \quad \begin{array}{c|cc|cc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 0 & 0 & 0 & 0 \\ 12 & 1 & 0 & 0 & 1 \\ \hline 21 & 0 & 1 & 0 & 0 \\ 22 & 0 & 0 & 1 & 0 \end{array} \tag{6.107}$$

$$RB_{T2}: \{(12, 11), (12, 22), (21, 12), (22, 21)\} \tag{6.108}$$

The vertical splitting of RB_{T2} into product rule bases RB_{S3} and RB_{S4} will be denoted by $RB_{T2} = RB_{S3}{:}RB_{S4}$ where RB_{S3} and RB_{S4} will be presented by the following block Boolean matrices and binary relations:

$$RB_{S3}: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 11 & 0 & 0 \\ 12 & 1 & 1 \\ \hline 21 & 1 & 0 \\ 22 & 0 & 1 \end{array} \tag{6.109}$$

$$RB_{S3}: \{(12, 1), (12, 2), (21, 1), (22, 2)\} \tag{6.110}$$

$$RB_{S4}: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 11 & 0 & 0 \\ 12 & 1 & 1 \\ \hline 21 & 0 & 1 \\ 22 & 1 & 0 \end{array} \tag{6.111}$$

$$RB_{S4}: \{(12, 1), (12, 2), (21, 2), (22, 1)\} \tag{6.112}$$

It is obvious that the rows in the first product Boolean matrix RB_{S3} map the non-zero 1×2 sub-blocks in the operand Boolean matrix RB_{T2}. As far as the second product Boolean matrix RB_{S4} is concerned, its non-zero elements map the non-zero 1×2 sub-blocks in a Boolean matrix that is obtained by swapping the second and the third column in RB_{T2} together with their labels. This new matrix stands for a new rule base RB_{T2N} in which the positions of the two outputs have been swapped, i.e. the second output from RB_{T2} has become first whereas its first output has become second. Here again, the corresponding binary relation does not change as a result of this column swap because the column labels are swapped as well.

The new rule base RB_{T2N} is presented by the following block Boolean matrix and binary relation:

RB_{T2N}: (6.113)

Inputs/Outputs	11	21	12	22
11	0	0	0	0
12	1	0	0	1
21	0	0	1	0
22	0	1	0	0

$$RB_{T2N}: \{(12, 11), (12, 22), (21, 12), (22, 21)\} \tag{6.114}$$

A more detailed inspection of this example shows that the patterns from the two operand rule bases RB_{T1} and RB_{T2} are replicated in the corresponding product rule bases RB_{S1} and RB_{S3} in a manner that matches the symmetry in RB_{T1} and RB_{T2}. In particular, the rows in RB_{S3} map the 1×2 sub-blocks in RB_{T2} which are obtained by transposing the corresponding 2×1 blocks in RB_{T1} as part of the transposition process from RB_{T1} to RB_{T2}. Similarly, the rows in RB_{S1} map the 1×2 sub-blocks in RB_{T1} which are obtained by transposing the corresponding 2×1 blocks in RB_{T2} as part of the transposition process from RB_{T2} to RB_{T1}.

The above conclusions become more obvious when RB_{T1} and RB_{T2} are presented in a form that makes their 2×1 blocks explicit, i.e.:

RB_{T1}: (6.115)

Inputs/Outputs	11	12	21	22
11	0	1	0	0
12	0	0	1	0
21	0	0	0	1
22	0	1	0	0

RB_{T2}: (6.116)

Inputs/Outputs	11	12	21	22
11	0	0	0	0
12	1	0	0	1
21	0	1	0	0
22	0	0	1	0

6.4 Manipulation with Permutation Rule Bases

Example 6.16

This example demonstrates the technique of vertical merging with a PRB from below and above. The two operand rule bases RB_{P1} and RB_{P2} are presented by the following Boolean matrices and binary relations:

RB_{P1}: (6.117)

Inputs/Outputs	1	2	3
1	0	0	1
2	1	0	0
3	0	1	0

$$RB_{P1}: \{(1, 3), (2, 1), (3, 2)\} \tag{6.118}$$

RB_{P2}: (6.119)

Inputs/Outputs	1	2	3
1	1	0	0
2	0	0	1
3	0	1	0

$$RB_{P2}: \{(1, 1), (2, 3), (3, 2)\} \tag{6.120}$$

The vertical merging of RB_{P1} and RB_{P2} into a product rule base RB_M will be denoted by $RB_{P1} + RB_{P2} = RB_M$ where RB_M will be presented by the following block Boolean matrix and binary relation:

RB_M: (6.121)

Inputs/Outputs	11	12	13	21	22	23	31	32	33
11	0	0	0	0	0	0	1	0	0
12	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	1	0	0
21	1	0	0	0	0	0	0	0	0
22	0	0	1	0	0	0	0	0	0
23	0	1	0	0	0	0	0	0	0
31	0	0	0	1	0	0	0	0	0
32	0	0	0	0	0	1	0	0	0
33	0	0	0	0	1	0	0	0	0

$$\begin{aligned} RB_M\!: \{&(11, 31), (12, 33), (13, 31),\\ &(21, 11), (22, 13), (23, 12),\\ &(31, 21), (32, 23), (33, 22)\} \end{aligned} \tag{6.122}$$

In this case, the permutation pattern from the two operand rule bases RB_{P1} and RB_{P2} is replicated in the product rule base RB_M, i.e. the latter is a PRB.

Example 6.17

This example demonstrates the technique of vertical splitting of a PRB. The operand rule base RB_P is presented by the following block Boolean matrix and binary relation:

$$RB_P\!: \quad \begin{array}{c|cc|cc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 0 & 0 & 0 & 1 \\ 12 & 0 & 0 & 1 & 0 \\ \hline 21 & 0 & 1 & 0 & 0 \\ 22 & 1 & 0 & 0 & 0 \end{array} \tag{6.123}$$

$$RB_P\!: \{(11, 22), (12, 21), (21, 12), (22, 11)\} \tag{6.124}$$

The vertical splitting of RB_P into product rule bases RB_{S1} and RB_{S2} will be denoted by $RB_P = RB_{S1}–RB_{S2}$ where RB_{S1} and RB_{S2} will be presented by the following Boolean matrices and binary relations:

$$RB_{S1}\!: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 0 & 1 \\ 12 & 1 & 0 \end{array} \tag{6.125}$$

$$RB_{S1}\!: \{(11, 2), (12, 1)\} \tag{6.126}$$

$$RB_{S2}\!: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 11 & 0 & 1 \\ 12 & 1 & 0 \end{array} \tag{6.127}$$

$$RB_{S2}\!: \{(11, 2), (12, 1)\} \tag{6.128}$$

In this case, the permutation pattern from the operand rule base RB_P is replicated in the product rule bases RB_{S1} and RB_{S2}, i.e. each of them is a PRB.

Example 6.18

This example demonstrates the technique of horizontal merging with a PRB from right and left. The operand rule bases RB_{P1} and RB_{P2} are presented by the following Boolean matrices and binary relations:

$$RB_{P1}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \end{array} \tag{6.129}$$

$$RB_{P1}: \{(1, 3), (2, 1), (3, 2)\} \tag{6.130}$$

$$RB_{P2}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 \end{array} \tag{6.131}$$

$$RB_{P2}: \{(1, 1), (2, 3), (3, 2)\} \tag{6.132}$$

The horizontal merging of RB_{P1} and RB_{P2} into a product rule base RB_M will be denoted by $RB_{P1}*RB_{P2} = RB_M$ where RB_M will be presented by the following block Boolean matrix and binary relation:

$$RB_M: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 \end{array} \tag{6.133}$$

$$RB_M: \{(1, 2), (2, 1), (3, 3)\} \tag{6.134}$$

In this case, the permutation pattern from the two operand rule bases RB_{P1} and RB_{P2} is replicated in the product rule base RB_M, the latter is a PRB.

Example 6.19

This example demonstrates the technique of horizontal splitting of a PRB. The operand rule base RB_P is presented by the following Boolean matrix and binary relation:

$$RB_P: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 \end{array} \tag{6.133}$$

$$RB_P: \{(1, 2), (2, 3), (3, 1)\} \tag{6.134}$$

The horizontal splitting of RB_P into product rule bases RB_{S1} and RB_{S2} will be denoted by $RB_P = RB_{S1} / RB_{S2}$ where RB_{S1} and RB_{S2} may be presented by the following Boolean matrices and binary relations:

$$RB_{S1}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 \end{array} \tag{6.135}$$

$$RB_{S1}: \{(1, 1), (2, 3), (3, 2)\} \tag{6.136}$$

$$RB_{S2}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 \end{array} \tag{6.137}$$

$$RB_{S2}: \{(1, 2), (2, 1), (3, 3)\} \tag{6.138}$$

In this case, the permutation pattern from the operand rule base RB_P is replicated in the product rule bases RB_{S1} and RB_{S2}, i.e. each of them is a PRB.

Example 6.20

This example demonstrates the technique of output merging with a PRB from below and above. The operand rule bases RB_{P1} and RB_{P2} are presented by the following Boolean matrices and binary relations:

$$RB_{P1}: \begin{array}{c|ccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \end{array} \tag{6.139}$$

$$RB_{P1}: \{(1, 3), (2, 1), (3, 2)\} \tag{6.140}$$

$$RB_{P2}: \begin{array}{c|ccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 \end{array} \tag{6.141}$$

$$RB_{P2}: \{(1, 1), (2, 3), (3, 2)\} \tag{6.142}$$

The output merging of RB_{P1} and RB_{P2} into a product rule base RB_M will be denoted by $RB_{P1};RB_{P2} = RB_M$ where RB_M will be presented by the following block Boolean matrix and binary relation:

$$RB_M: \begin{array}{c|ccc|ccc|ccc} \text{Inputs/Outputs} & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array} \tag{6.143}$$

$$RB_M: \{(1, 31), (2, 13), (3, 22)\} \tag{6.144}$$

In this case, the permutation pattern from the two operand rule bases RB_{P1} and RB_{P2} is replicated in the non-zero 1×3 sub-blocks in the product rule base RB_M, as shown by Eqs. (6.143)–(6.144). In particular, the positions of these blocks map the positions of the non-zero elements from the first operand matrix whereby the positions of the non-zero elements in the sub-blocks map the positions of the non-zero elements from the second operand matrix.

Example 6.21

This example demonstrates the technique of output splitting of a PRB. The operand rule base RB_P is presented by the following block Boolean matrix and binary relation:

$$RB_P: \quad \begin{array}{c|cc|cc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 0 & 0 & 1 & 0 \\ 12 & 1 & 0 & 0 & 0 \\ \hline 21 & 0 & 1 & 0 & 0 \\ 22 & 0 & 0 & 0 & 1 \end{array} \tag{6.145}$$

$$RB_P: \{(11, 21), (12, 11), (21, 12), (22, 22)\} \tag{6.146}$$

The output splitting of RB_P into product rule bases RB_{S1} and RB_{S2} will be denoted by $RB_P = RB_{S1}:RB_{S2}$ where RB_{S1} and RB_{S2} will be presented by the following block Boolean matrices and binary relations:

$$RB_{S1}: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 11 & 0 & 1 \\ 12 & 1 & 0 \\ \hline 21 & 1 & 0 \\ 22 & 0 & 1 \end{array} \tag{6.147}$$

$$RB_{S1}: \{(11, 2), (12, 1), (21, 1), (22, 2)\} \tag{6.148}$$

$$RB_{S2}: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 11 & 1 & 0 \\ 12 & 1 & 0 \\ \hline 21 & 0 & 1 \\ 22 & 0 & 1 \end{array} \tag{6.149}$$

$$RB_{S2}: \{(11, 1), (12, 1), (21, 2), (22, 2)\} \tag{6.150}$$

In this case, the non-zero elements in the first product Boolean matrix RB_{S1} map the non-zero 1×2 sub-blocks in the operand Boolean matrix RB_P. As far as the second product Boolean matrix RB_{S2} is concerned, its non-zero elements map the non-zero 1×2 sub-blocks in a Boolean matrix that is obtained by swapping the second and the third column in RB_P together with their labels. This new matrix stands for a new rule base RB_{PN} in which the positions of the two outputs have been swapped, i.e. the second output from RB_P has become first whereas its first output has become second. Obviously, the corresponding binary relation does not change as a result of this column swap because the column labels are swapped as well.

The new rule base RB_{PN} is presented by the following block Boolean matrix and binary relation:

$$RB_{PN}: \quad \begin{array}{c|cc|cc} \text{Inputs/Outputs} & 11 & 21 & 12 & 22 \\ \hline 11 & 0 & 1 & 0 & 0 \\ 12 & 1 & 0 & 0 & 0 \\ \hline 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 0 & 1 \end{array} \tag{6.151}$$

$$RB_{PN}: \{(11, 21), (12, 11), (21, 12), (22, 22)\} \tag{6.152}$$

A more detailed inspection of this example shows that the patterns from the operand rule base RB_P are replicated and transformed in the corresponding product rule bases RB_{S1} and RB_{S2} in a manner that matches the permutation pattern in RB_P. In particular, this permutation pattern is replicated in the blocks of RB_{S1} as well as in a new Boolean matrix RB_{S2N} that is obtained by swapping the second and the third row of RB_{S2} together with their labels. This new matrix stands for a new rule base RB$_{S2N}$ in which the positions of the two inputs have been swapped, i.e. the second input from RB_{S2} has become first whereas its first output has become second. Obviously, the corresponding binary relation does not change as a result of this row swap because the row labels are swapped as well.

The new rule base RB_{S2N} is presented by the following block Boolean matrix and binary relation:

$$RB_{S2N}: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 11 & 1 & 0 \\ 21 & 0 & 1 \\ \hline 12 & 1 & 0 \\ 22 & 0 & 1 \end{array} \tag{6.153}$$

$$RB_{S2N}: \{(11, 1), (21, 2), (12, 1), (22, 2)\} \tag{6.154}$$

6.5 Specific Cases with Special Rule Bases

The techniques for formal manipulation with special rule bases introduced in the previous sections of this chapter relate to the most common cases. However, sometimes we may have some specific cases such as:

- vertical merging of two IRBs,
- horizontal merging of two IRBs,
- horizontal splitting of an IRB,
- horizontal merging of two rule bases such that each of them is a *permutation-transpose rule base* (P-T RB), i.e. a PRB as well as a TRB.

The above specific cases are illustrated in Figs. 6.10–6.13 and by several examples further in this section. The purpose of the examples is to demonstrate the specific cases with special rule bases and to show the potential impact of the manipulation on the structure of the rule bases involved, i.e. how the patterns from the operand rule bases are replicated or transformed in the product rule bases.

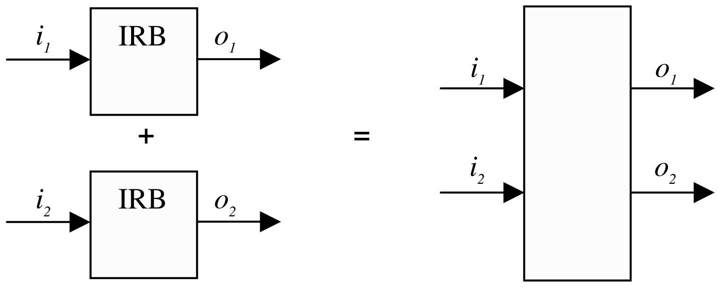

Fig. 6.10. Vertical merging of two identity rule bases

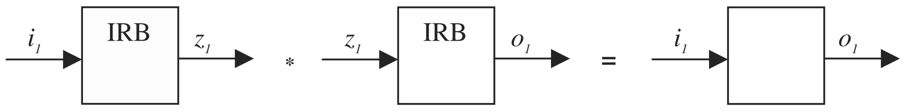

Fig. 6.11. Horizontal merging of two identity rule bases

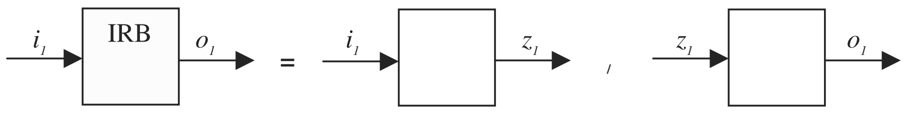

Fig. 6.12. Horizontal splitting of an identity rule base

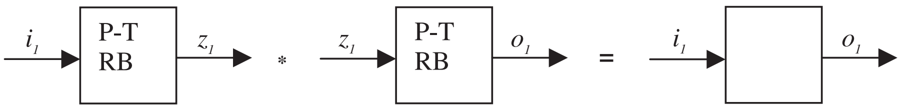

Fig. 6.13. Horizontal merging of two permutation-transpose rule bases

Example 6.22

This example demonstrates the technique of vertical merging of two IRBs. The operand rule bases RB_{I1} and RB_{I2} are presented by the following Boolean matrices and binary relations:

$$RB_{I1}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.155}$$

$$R_{I1}: \{(1, 1), (2, 2)\} \tag{6.156}$$

$$RB_{I2}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.157}$$

$$RB_{I2}: \{(1, 1), (2, 2)\} \tag{6.158}$$

The product rule base RB_M for the vertical merging of RB_{I1} and RB_{I2} will be denoted by $RB_M = RB_{I1} / RB_{I2}$ and will be presented by the following block Boolean matrix and binary relation:

$$RB_M: \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 1 & 0 & 0 & 0 \\ 12 & 0 & 1 & 0 & 0 \\ 21 & 0 & 0 & 1 & 0 \\ 22 & 0 & 0 & 0 & 1 \end{array} \tag{6.159}$$

$$RB_M: \{(11, 11), (12, 12), (21, 21), (22, 22)\} \tag{6.160}$$

In this case, the product rule base RB_M is an IRB, as shown by Eqs. (6.159)–(6.160).

Example 6.23

This example demonstrates the technique of horizontal merging of two IRBs. The operand rule bases RB_{I1} and RB_{I2} are presented by the following Boolean matrices and binary relations:

$$RB_{I1}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.161}$$

$$R_{I1}: \{(1, 1), (2, 2)\} \tag{6.162}$$

$$RB_{I2}: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.163}$$

$$RB_{I2}: \{(1, 1), (2, 2)\} \tag{6.164}$$

The product rule base RB_M for the vertical merging of RB_{I1} and RB_{I2} will be denoted by $RB_M = RB_{I1} * RB_{I2}$ and will be presented by the following Boolean matrix and binary relation:

$$RB_M: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{6.165}$$

$$RB_M: \{(1, 1), (2, 2)\} \tag{6.166}$$

In this case, the product rule base RB_M is an IRB, as shown by Eqs. (6.165)–(6.166).

Example 6.24

This example demonstrates the technique of horizontal splitting of an IRB. The operand rule base RB_I is presented by the following Boolean matrix and binary relation:

$$RB_I: \quad \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{array} \tag{6.167}$$

$$RB_I: \{(1, 1), (2, 2), (3, 3)\} \tag{6.168}$$

The product rule bases RB_{S1} and RB_{S2} for the horizontal splitting of RB_I will be denoted by $RB_{S1};RB_{S2} = RB_I$ and may be presented by the following Boolean matrices and binary relations:

$$RB_{S1}: \quad \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{array} \tag{6.169}$$

$$RB_{S1}: \{(1, 1), (2, 2), (3, 3)\} \tag{6.170}$$

$$RB_{S2}: \quad \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{array} \tag{6.171}$$

$$RB_{S2}: \{(1, 1), (2, 2), (3, 3)\} \tag{6.172}$$

It is obvious that the product rule bases RB_{S1} and RB_{S2} are both IRBs, as shown by Eqs. (6.169)–(6.170) and Eqs. (6.171)–(6.172).

Example 6.25

This example demonstrates the technique of horizontal merging of two P-T RBs. The two operand rule bases RB_{PT1} and RB_{PT2} are handled together in two separate cases whereby in the second case their positions are swapped in relation to the first case.

For the first case, the operand rule bases RB_{PT1} and RB_{PT2} are presented by the following Boolean matrices and binary relations:

$$RB_{PT1}: \quad \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \end{array} \tag{6.173}$$

$$RB_{PT1}: \{(1, 3), (2, 1), (3, 2)\} \tag{6.174}$$

$$RB_{PT2}: \quad \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 \end{array} \tag{6.175}$$

$$RB_{PT2}: \{(1, 2), (2, 3), (3, 1)\} \tag{6.176}$$

.

The product rule base RB_{M12} for the horizontal merging of RB_{PT1} and RB_{PT2} will be denoted by $RB_{M12} = RB_{PT1}*RB_{PT2}$ and will be presented by the following Boolean matrix and binary relation:

$$RB_{M12}: \quad \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{array} \tag{6.177}$$

$$RB_{M12}: \{(1, 1), (2, 2), (3, 3)\} \tag{6.178}$$

For the second case, the positions of the operand rule bases RB_{PT1} and RB_{PT2} are swapped, i.e. RB_{PT2} is from left and RB_{PT1} is from right. Therefore, the product rule base RB_{M21} for the vertical merging of RB_{PT2} and RB_{PT1} will be denoted by $RB_{M21} = RB_{PT2}*RB_{PT1}$ and will be presented by the following block Boolean matrix and binary relation:

$$RB_{M21}: \quad \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{array} \tag{6.179}$$

$$RB_{M21}: \{(1, 1), (2, 2), (3, 3)\} \tag{6.180}$$

As shown by Eqs. (6.177)–(6.178) and Eqs. (6.179)–(6.180), the symmetrical patterns from the two operand rule bases RB_{PT1} and RB_{PT2} are transformed in the corresponding product rule bases RB_{M12} and RB_{M21} in a specific way. In particular, RB_{M12} and RB_{M21} are IRBs and are equal to each other. Therefore, the product rule base is not affected by the positions of the operand rule bases, i.e. the horizontal merging of two PRB-TRBs is commutative.

6.6 Analysis of Manipulation Techniques with Special Rule Bases

The examples from the previous sections on formal manipulation with special rule bases provide a valuable insight into how patterns from operand rule bases are replicated and transformed in the corresponding product rule bases. It is obvious in this case that patterns in a rule base define its properties uniquely whereas the opposite is not true, i.e. a property may be reflected in more than one pattern. For example, an identity pattern in a rule base implies uniquely that it is complete, consistent, exhaustive and monotonic. However, a rule base with all these properties does not necessarily imply that it has the identity pattern as it could have the permutation pattern instead.

Also, patterns are more general and flexible descriptors of fuzzy systems than properties. While properties can be either preserved or lost during formal manipulation, patterns can be replicated or even transformed. In addition, patterns are more structurally orientated descriptors than properties. This structural orientation makes patterns more relevant to MRB systems whereas properties remain more relevant to SRB systems or MRB systems represented equivalently by SRB systems. All this shows that patterns are more suitable for synthesis of rule bases while properties are more suitable for analysis.

As far as the general study of patterns in rule bases is concerned, Boolean matrices are a more suitable tool than binary relations and that is why the latter have been marginalised in this chapter. For example, it is much easier to recognise a pattern in a rule base by inspecting its Boolean matrix rather than the corresponding binary relation.

The three merging manipulations introduced in the current and the previous chapter usually lead to a unique product rule base, i.e. the patterns and the properties of the operand rule bases determine uniquely the patterns and the properties of the product rule base. However, in the three splitting manipulations, the patterns and the properties of the operand rule base often do not determine uniquely the patterns and the properties of the product rule bases. For example, an arbitrary rule base can be horizontally split such that either the first or the second product rule base is an IRB whereas the other product rule base is equal to the operand rule base. Also, a 3×3 IRB can be vertically split such that either the first or the second product rule base is a 2×2 IRB whereas the other product rule base is a 1×1 IRB. And finally, a 3×3 IRB can be output split such that either the first or the second product rule base is a 3×2 rule base whereas the other product rule base is a 3×1 rule base.

As far as the impact of the different manipulation techniques on the structure of the rule base is concerned, it can be summarised as follows:

- vertical merging removes a level within a layer,
- vertical splitting generates a level within a layer,
- horizontal merging removes a layer within a level,
- horizontal splitting generates a layer within a level,
- output merging removes a level within a layer,
- output splitting generates a level within a layer.

The above conclusions are valid for manipulation with arbitrary rule bases as well as special rule bases. They are usually valid for MRB systems although in some cases either the operand or the product rule base may be a SRB system.

Also, bearing in mind that merging operations remove cells such as layers or levels while splitting operations generate cells, the conclusion can be made that merging is more relevant to synthesis of rule bases whereas splitting is more relevant to analysis. This conclusion is also completely in line with the general assumption that synthesis is an active approach leading to a deterministic result whereas analysis is a passive approach that may not yield a deterministic result.

The manipulation techniques with arbitrary and special rule bases have been demonstrated in the previous and the current chapter in a relatively isolated context, i.e. with product rule bases, which are SRB systems. In the case of complex fuzzy systems, i.e. systems containing rule bases interconnected within a fairly complex structure, the product rule bases themselves may turn out to be interconnected with other rule bases and as such will be parts of a MRB system. In order to account for these circumstances, a special study on the resulting transformation of the associated rule bases is needed and this is discussed in detail in the next chapter.

7 Formal Transformation of Fuzzy Rule Based Systems

7.1 Preliminaries on Rule Base Transformation

The techniques for formal manipulation with arbitrary and special rule bases introduced in the previous two chapters are a powerful tool for complexity management in fuzzy systems. However, due to their basic nature, these techniques can be used only in SRB systems or very simple MRB systems. So, if we want to deal with complex MRB systems, we have to use more advanced formal manipulation techniques.

To distinguish these advanced techniques from the basic manipulation techniques, they will be referred to as formal transformation techniques. As opposed to formal manipulation, formal transformation deals with more complex types of reorganisation in a fuzzy system whereby a single operand rule base is represented by more than just a couple of product rule bases, or alternatively, a fairly complex structure of operand rule bases is represented by a single product rule base. In either case, no special requirements are usually put on the corresponding operand or product rule bases, i.e. the latter may be any arbitrary rule bases.

For consistency, the formal transformation techniques for fuzzy systems are introduced in the current chapter by examples whereby the technique highlighted in a particular example uses techniques from some preceding examples as prerequisites. As opposed to the previous three chapters, the notations here are not presented at the same level of detail but they are sufficient for understanding the underlying process for each transformation technique. And finally, although the range of transformation techniques presented in this chapter is not exhaustive, it provides a fairly good basis for the development of new transformation techniques, if necessary.

7.2 Repetitive Merging Manipulations

The most basic type of formal transformation in a fuzzy rule based system is the one in which a merging manipulation operation is applied in a repetitive manner. For example, if we have three or more operand rule bases standing in parallel within a particular layer of a MRB system, we may want to merge them vertically into a single operand rule base. Also, if we have three or more operand rule bases standing in sequence within a particular level of a MRB system, we may want to merge them horizontally into a single operand rule base. Therefore, it is useful to know if the merging manipulations involved are commutative and associative, as discussed in Examples 7.1–7.2.

Example 7.1

Vertical and horizontal merging manipulations are both non-commutative and therefore it is not allowed to swap the positions of operand rule bases in repetitive merging. In particular, the swapping of the positions of the operand rule bases RB_1 and RB_2 will affect the product rule base, as shown briefly in Figs. 7.1–7.2 and in more detail by Eqs. (7.1)–(7.2).

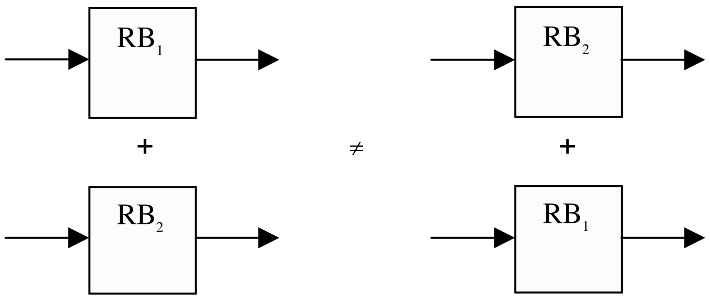

Fig. 7.1. Non-commutativity of vertical merging

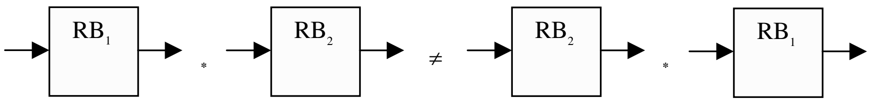

Fig. 7.2. Non-commutativity of horizontal merging

$$\text{If } RB_{1+2} = RB_1 + RB_2 \text{ and } RB_{2+1} = RB_2 + RB_1 \text{ then } RB_{1+2} \neq RB_{2+1} \tag{7.1}$$

$$\text{If } RB_{1*2} = RB_1 * RB_2 \text{ and } RB_{2*1} = RB_2 * RB_1 \text{ then } RB_{1*2} \neq RB_{2*1} \tag{7.2}$$

To illustrate the above implications, we consider the operand rule bases RB_1 and RB_2, which are presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{7.3}$$

$$RB_1: \{(1, 2), (2, 1)\} \tag{7.4}$$

$$RB_2: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{7.5}$$

$$RB_2: \{(1, 1), (2, 1)\} \tag{7.6}$$

In this case, the product rule bases RB_{1+2} and RB_{2+1} for the vertical merging of RB_1 and RB_2 will be presented by the following different Boolean matrices and binary relations:

$$RB_{1+2}: \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 1 & 0 \\ 12 & 0 & 0 & 1 & 0 \\ 21 & 1 & 0 & 0 & 0 \\ 22 & 1 & 0 & 0 & 0 \end{array} \tag{7.7}$$

$$RB_{1+2}: \{(11, 21), (12, 21), (21, 11), (22, 11)\} \tag{7.8}$$

$$RB_{2+1}: \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 1 & 0 & 0 \\ 12 & 1 & 0 & 0 & 0 \\ 21 & 0 & 1 & 0 & 0 \\ 22 & 1 & 0 & 0 & 0 \end{array} \tag{7.9}$$

$$RB_{2+1}: \{(11, 12), (12, 11), (21, 12), (22, 11)\} \tag{7.10}$$

Similarly, the product rule bases RB_{1*2} and RB_{2*1} for the horizontal merging of RB_1 and RB_2 will be presented by the following different Boolean matrices and binary relations:

RB_{1*2}: (7.11)

Inputs/Outputs	1	2
1	1	0
2	1	0

$$RB_{1*2}: \{(1, 1), (2, 1)\} \tag{7.12}$$

RB_{2*1}: (7.13)

Inputs/Outputs	1	2
1	0	1
2	0	1

$$RB_{2*1}: \{(1, 2), (2, 2)\} \tag{7.14}$$

Example 7.2

Vertical and horizontal merging manipulations are both associative and therefore it is possible to change the order of operations on operand rule bases in repetitive merging. In other words, the changing of the order of operations on the operand rule bases RB_1, RB_2 and RB_3 will not affect the product rule base, as shown briefly in Figs. 7.3–7.4 and in more detail by Eqs. (7.15)–(7.16).

$$\text{If } RB_{(1+2)+3} = (RB_1 + RB_2) + RB_3 \text{ and } RB_{1+(2+3)} = RB_1 + (RB_2 + RB_3) \quad \text{then } RB_{(1+2)+3} = RB_{1+(2+3)} \tag{7.15}$$

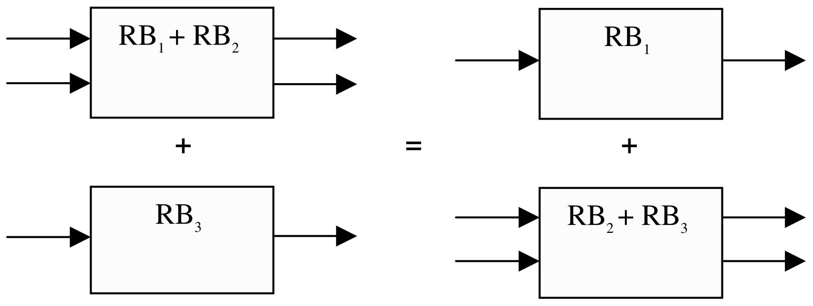

Fig. 7.3. Associativity of vertical merging

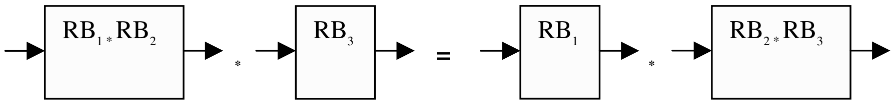

Fig. 7.4. Associativity of horizontal merging

$$\text{If } RB_{(1*2)*3} = (RB_1 * RB_2) * RB_3 \text{ and } RB_{1*(2*3)} = RB_1 * (RB_2 * RB_3) \quad (7.16)$$
$$\text{then } RB_{(1*2)*3} = RB_{1*(2*3)}$$

To illustrate the above implications, we consider the operand rule bases RB_1, RB_2 and RB_3, which are presented by the following Boolean matrices and binary relations:

RB_1: (7.17)

Inputs/Outputs	1	2
1	1	0
2	1	0

RB_1: {(1, 1), (2, 1)} (7.18)

RB_2: (7.19)

Inputs/Outputs	1	2
1	0	1
2	1	0

RB_2: {(1, 2), (2, 1)} (7.20)

RB_3: (7.21)

Inputs/Outputs	1	2
1	0	1
2	0	1

RB_3: {(1, 2), (2, 2)} (7.22)

In this case, both product rule bases $RB_{(1+2)+3}$ and $RB_{1+(2+3)}$ for the vertical merging of RB_1, RB_2 and RB_3 will be presented by the following Boolean matrix and binary relation:

$$RB_{(1+2)+3} = RB_{1+(2+3)}: \begin{array}{c|cccccccc} \text{Inputs/Outputs} & 111 & 112 & 121 & 122 & 211 & 212 & 221 & 222 \\ \hline 111 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 112 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 121 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 122 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 211 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 212 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 221 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 222 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \tag{7.23}$$

$$RB_{(1+2)+3} = RB_{1+(2+3)}: \{(111, 122), (112, 122), (121, 112), (122, 112), (211, 122), (212, 122), (221, 112), (222, 112)\} \tag{7.24}$$

Similarly, both product rule bases $RB_{(1*2)*3}$ and $RB_{1*(2*3)}$ for the horizontal merging of RB_1, RB_2 and RB_3 will be presented by the following Boolean matrix and binary relation:

$$RB_{(1*2)*3} = RB_{1*(2*3)}: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{7.25}$$

$$RB_{1*(2*3)} = RB_{1*(2*3)}: \{(1, 2), (2, 2)\} \tag{7.26}$$

7.3 Combined Merging Manipulations

A more advanced type of formal transformation in a fuzzy rule based system is the one in which a merging manipulation operation is combined with a splitting manipulation operation or vice versa. For example, if we have an operand rule base with one input and two outputs residing in a particular layer as well as two other operand rule bases with one input and one output each residing in the next layer, we could then feed each of the two outputs from the first rule base as an input to each of the other two rule bases. Also, if we have an operand rule base with one input and one output residing in a particular level as well as two other operand rule bases with one input and one output each such that they both reside in the next level, we could then form a rule base whose inputs are the inputs to the first and the second rule base and whose outputs are outputs from the first and the third rule base. Therefore, it is useful to know if the merging and splitting manipulations involved are distributive and interchangeable, as discussed in Examples 7.3–7.4.

Example 7.3

Vertical and horizontal merging manipulations are both non-distributive with respect to each other. Therefore, it is not allowed to expand any brackets specifying the order in which these operations are to be applied. In particular, if the operand rule base RB_1 stands separately from the rule bases RB_2 and RB_3, the combining of RB_1 with RB_2 and RB_3 in a distributive context will affect the product rule base, as shown briefly in Figs. 7.5–7.6 and in more detail by Eqs. (7.27)–(7.28).

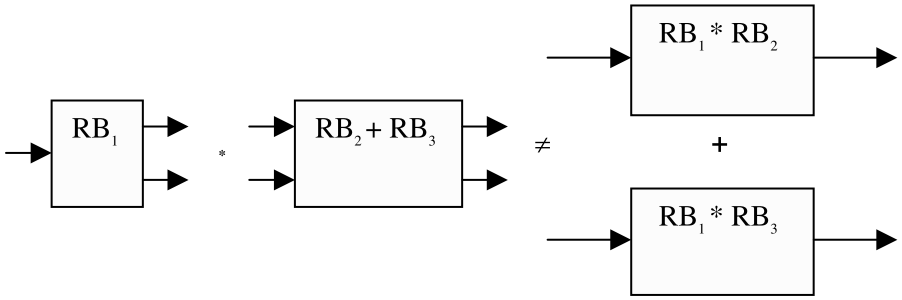

Fig. 7.5. Non-distributivity of vertical merging with respect to horizontal merging

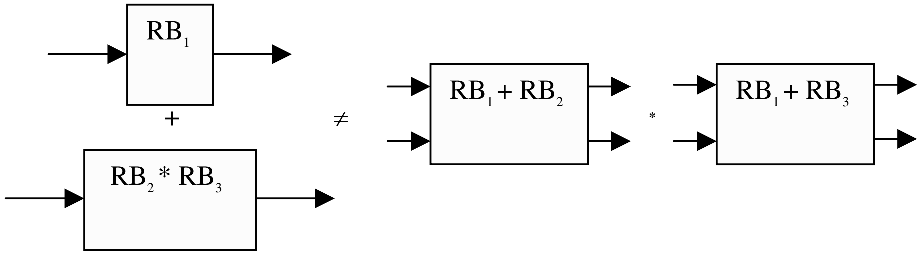

Fig. 7.6. Non-distributivity of horizontal merging with respect to vertical merging

$$\begin{aligned} &\text{If } RB_{1*(2+3)} = RB_1 * (RB_2 + RB_3) \\ &\text{and } RB_{(1*2)+(1*3)} = (RB_1 * RB_2) + (RB_1 * RB_3) \\ &\text{then } RB_{1*(2+3)} \neq RB_{(1*2)+(1*3)} \end{aligned} \qquad (7.27)$$

$$\begin{aligned} &\text{If } RB_{1+(2*3)} = RB_1 + (RB_2 * RB_3) \\ &\text{and } RB_{(1+2)*(1+3)} = (RB_1 + RB_2) * (RB_1 + RB_3) \\ &\text{then } RB_{1+(2*3)} \neq RB_{(1+2)*(1+3)} \end{aligned} \tag{7.28}$$

To illustrate the implications in Eq. (7.27), we consider the operand rule bases RB_1, RB_2 and RB_3, which are presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 \end{array} \tag{7.29}$$

$$RB_1: \{(1, 21), (2, 12)\} \tag{7.30}$$

$$RB_2: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{7.31}$$

$$RB_2: \{(1, 2), (2, 2)\} \tag{7.32}$$

$$RB_3: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{7.33}$$

$$RB_3: \{(1, 1), (2, 1)\} \tag{7.34}$$

The product rule base $RB_{1*(2+3)}$ for the combined merging of RB_1, RB_2 and RB_3 will be presented by the following Boolean matrix and binary relation:

$$RB_{1*(2+3)}: \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 0 \end{array} \tag{7.35}$$

$$RB_{1*(2+3)}: \{(1, 21), (2, 21)\} \tag{7.36}$$

In this case, the product rule base $RB_{(1*2)+(1*3)}$ for the combined merging of RB_1, RB_2 and RB_3 will not exist due to the dimensional mismatch between the rule base RB_1 and the other two rule bases RB_2 and RB_3. In other words, the number of outputs for RB_1 is different from the number of inputs for RB_2 and RB_3.

To illustrate the implications in Eq. (7.28), we consider the operand rule bases RB_1, RB_2 and RB_3, which are presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{7.37}$$

$$RB_1: \{(1, 2), (2, 1)\} \tag{7.38}$$

$$RB_2: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{7.39}$$

$$RB_2: \{(1, 1), (2, 1)\} \tag{7.40}$$

$$RB_3: \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{7.41}$$

$$RB_3: \{(1, 2), (2, 2)\} \tag{7.42}$$

In this case, the product rule bases $RB_{1+(2*3)}$ and $RB_{(1+2)*(1+3)}$ for the combined merging of RB_1, RB_2 and RB_3 will be presented by the following different Boolean matrices and binary relations:

$$RB_{1+(2*3)}: \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 0 & 1 \\ 12 & 0 & 0 & 0 & 1 \\ 21 & 0 & 1 & 0 & 0 \\ 22 & 0 & 1 & 0 & 0 \end{array} \tag{7.43}$$

$$\mathrm{RB}_{1+(2*3)}: \{(11, 22), (12, 22), (21, 12), (22, 12)\} \tag{7.44}$$

$\mathrm{RB}_{(1+2)*(1+3)}$: (7.45)

Inputs/Outputs	11	12	21	22
11	0	1	0	0
12	0	1	0	0
21	0	0	0	1
22	0	0	0	1

$$\mathrm{RB}_{(1+2)*(1+3)}: \{(11, 12), (12, 12), (21, 22), (22, 22)\} \tag{7.46}$$

Example 7.4

Vertical and horizontal merging manipulations are interchangeable and therefore it is possible to change accordingly the order of operations on operand rule bases in combined merging. In other words, the changing of the order of operations on the operand rule bases RB_1, RB_2, RB_3 and RB_4 will not affect the product rule base, as shown briefly in Fig. 7.7 and in more detail by Eq. (7.47).

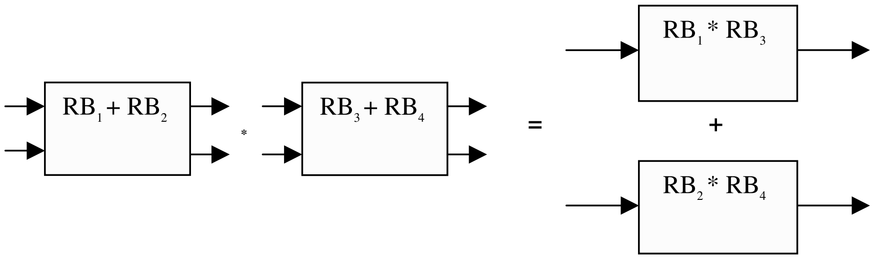

Fig. 7.7. Interchangeability of vertical and horizontal merging

$$\begin{aligned} \text{If } RB_{(1+2)*(3+4)} &= (RB_1 + RB_2)*(RB_3 + RB_4) \\ \text{and } RB_{(1*3)+(2*4)} &= (RB_1 * RB_3)+(RB_2 * RB_4) \\ \text{then } RB_{(1+2)*(3+4)} &= RB_{(1*3)+(2*4)} \end{aligned} \tag{7.47}$$

To illustrate the above implication, we consider the operand rule bases RB_1, RB_2, RB_3 and RB_4, which are presented by the following Boolean matrices and binary relations:

$$RB_1: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{7.48}$$

$$RB_1: \{(1, 2), (2, 1)\} \tag{7.49}$$

$$RB_2: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{7.50}$$

$$RB_2: \{(1, 1), (2, 1)\} \tag{7.51}$$

$$RB_3: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{7.52}$$

$$RB_3: \{(1, 2), (2, 2)\} \tag{7.53}$$

$$RB_4: \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{7.54}$$

$$RB_4: \{(1, 2), (2, 1)\} \tag{7.55}$$

Both product rule bases $RB_{(1+2)*(3+4)}$ and $RB_{(1*3)+(2*4)}$ for the combined merging of RB_1, RB_2, RB_3 and RB_4 will be presented by the following Boolean matrix and binary relation:

$$RB_{(1+2)*(3+4)} = RB_{(1*3)+(2*4)}: \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 0 & 1 \\ 12 & 0 & 0 & 0 & 1 \\ 21 & 0 & 0 & 0 & 1 \\ 22 & 0 & 0 & 0 & 1 \end{array} \tag{7.56}$$

$$RB_{(1+2)*(3+4)} = RB_{(1*3)+(2*4)}: \{(11, 22), (12, 22), (21, 22), (22, 22)\} \tag{7.57}$$

7.4 Self Standing Inputs and Outputs

As already mentioned in Chapter 2, a MRB system may be presented by a sparse matrix with empty cells for the missing rule bases. In this context, inputs or outputs spanning across at least one but not all layers are referred to as 'self standing'. Such inputs and outputs describe identity mappings, which may be presented by IRBs. In this case, the dimension of the corresponding identity Boolean matrix and binary relation RB_I is equal to the number of inputs and outputs in the rule base, i.e. an IRB with two inputs and two outputs will be denoted by RB_{I2}.

Alternatively, identity mappings may also be presented by a couple of P-T RBs. As already mentioned in Chapter 6, the horizontal merging of two P-T RBs yields an IRB. In this case, the dimension of the corresponding permutation Boolean matrix and binary relation RB_P is equal to the number its inputs and outputs, i.e. a PRB with three inputs and three outputs will be denoted by RB_{P3}.

The number of inputs to a MRB system is equal to the sum of the number of inputs to each rule base in the first layer, including any self standing inputs. Similarly, the number of outputs from a MRB system is equal to the sum of the number of outputs from each rule base in the last layer, including any self standing outputs.

Any inputs to a MRB system which are fed directly into a rule base residing in a layer after the first one are self standing. In this case, the number of layers through which a self standing input passes before being fed into a rule base gives the length of this input. This length is always greater than or equal to 1 and less than or equal to the number of layers in the MRB system minus 1. In this context, Example 7.5 shows how a self standing input can be presented by an IRB RB_{I1} and how an ERB RB_E for the entire MRB system can be derived. In addition, Example 7.6 shows how an ERB RB_E can be derived in accordance with the interchangeability property, by either vertical-horizontal or horizontal-vertical merging.

Any outputs from a rule base residing in a layer before the last one in a MRB system which are fed directly as outputs from the MRB system are self standing. In this case, the number of layers through which a self standing output passes after being fed from a rule base gives the length of this output. This length is always greater than or equal to 1 and less than or equal to the number of layers in the MRB system minus 1. In this context, Example 7.7 shows how a self standing output can be presented by an IRB RB_{I1} and how an ERB RB_E for the entire MRB system can be derived. In addition, Example 7.8 shows how an ERB RB_E can be derived in accordance with the interchangeability property, by either vertical-horizontal or horizontal-vertical merging.

Example 7.5

A MRB system with 2 levels and 2 layers is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_{1x1}^{1,1} & RB_{2x1}^{1,2} \\ \text{level 2} & RB_{II}^{2,1} & \end{array} \tag{7.58}$$

The output-input interconnections for the above MRB system are given by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1^{1,1} = i_1^{1,2} & o_1^{1,2} \\ \text{level 2} & o_1^{2,1} = i_2^{1,2} & \end{array} \tag{7.59}$$

Equation (7.58) shows that the MRB system has two inputs and one output. It also shows that the rule base in level 1 of layer 1 has one input and one output, the IRB in level 2 of layer 1 has one input and one output, the rule base in level 1 of layer 2 has two inputs and one output whereas the rule base in level 2 of layer 2 is missing. In addition, Eq. (7.59) shows that the first output from the rule base in level 1 of layer 1 is the same as the first input to the rule base in level 1 of layer 2 whereas the self standing input which is also the first output from the IRB in level 2 of layer 1 is the same as the second input to the rule base in level 1 of layer 2.

Although the only purpose of the IRB is to represent a self standing input in level 2 of layer 1, this rule base must be counted separately. Therefore, the MRB system consists of three rule bases, as shown in Fig. 7.8.

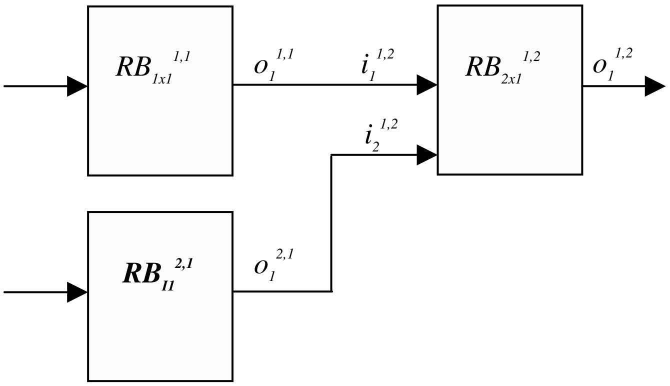

Fig. 7.8. Multiple rule base system with three rule bases and self standing input in level 2 of layer 1

The rule bases $RB_{1x1}^{1,1}$, $RB_{I1}^{2,1}$ and $RB_{2x1}^{1,2}$ are presented by the following Boolean matrices and binary relations:

$RB_{1x1}^{1,1}$: (7.60)

Inputs/Outputs	1	2
1	0	1
2	1	0

$$RB_{1x1}^{1,1}: \{(1, 2), (2, 1)\} \tag{7.61}$$

$RB_{I1}^{2,1}$: (7.62)

Inputs/Outputs	1	2
1	1	0
2	0	1

$$RB_{I1}^{2,1}: \{(1, 1), (2, 2)\} \tag{7.63}$$

$RB_{2x1}^{1,2}$: (7.64)

Inputs/Outputs	1	2
11	0	1
12	0	1
21	1	0
22	1	0

$$RB_{2x1}^{1,2}: \{(11, 2), (12, 2), (21, 1), (22, 1)\} \tag{7.65}$$

The ERB RB_E for the MRB system is derived by the formula in Eq. (7.66) and is presented by the Boolean matrix and the binary relation in Eqs. (7.67)–(7.68).

$$RB_E = (RB_{1x1}^{1,1} + RB_{I1}^{2,1}) * RB_{2x1}^{1,2} \tag{7.66}$$

RB_E: (7.67)

Inputs/Outputs	1	2
11	1	0
12	1	0
21	0	1
22	0	1

$$RB_E: \{(11, 1), (12, 1), (21, 2), (22, 2)\} \tag{7.68}$$

Example 7.6

A MRB system with 2 levels and 2 layers is presented by the following matrix:

$$\begin{array}{lcc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_{11}^{1,1} & RB_{1x1}^{1,2} \\ \text{level 2} & RB_{1x1}^{2,1} & RB_{1x1}^{2,2} \end{array} \qquad (7.69)$$

The output-input interconnections for the above MRB system are given by the following matrix:

$$\begin{array}{lcc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1^{1,1} = i_1^{1,2} & o_1^{1,2} \\ \text{level 2} & o_1^{2,1} = i_1^{2,2} & o_1^{2,2} \end{array} \qquad (7.70)$$

Equation (7.69) shows that the MRB system has two inputs and two outputs. It also shows that each of the rule bases including the IRB in level 1 of layer 1 has one input and one output. In addition, Eq. (7.70) shows that the self standing input which is also the first output from the IRB in level 1 of layer 1 is the same as the first input to the rule base in level 1 of layer 2 whereas the the first output from the rule base in level 2 of layer 1 is the same as the first input to the rule base in level 2 of layer 2.

Although the only purpose of the IRB is to represent a self standing input in level 1 of layer 1, this rule base must be counted separately. Therefore, the MRB system consists of four rule bases, as shown in Fig. 7.9.

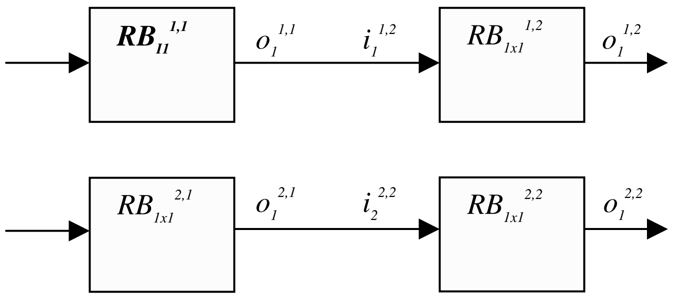

Fig. 7.9. Multiple rule based system with four rule bases and self standing input in level 1 of layer 1

The rule bases $RB_{11}^{1,1}$, $RB_{1x1}^{2,1}$, $RB_{1x1}^{1,2}$ and $RB_{1x1}^{2,2}$ are presented by the following Boolean matrices and binary relations:

$$RB_{11}^{1,1}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{7.71}$$

$$RB_{11}^{1,1}: \{(1, 1), (2, 2)\} \tag{7.72}$$

$$RB_{1x1}^{2,1}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{7.73}$$

$$RB_{1x1}^{2,1}: \{(1, 1), (2, 1)\} \tag{7.74}$$

$$RB_{1x1}^{1,2}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{7.75}$$

$$RB_{1x1}^{1,2}: \{(1, 2), (2, 1)\} \tag{7.76}$$

$$RB_{1x1}^{2,2}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{7.77}$$

$$RB_{1x1}^{2,2}: \{(1, 2), (2, 2)\} \tag{7.78}$$

The ERB RB_E for the MRB system is derived as either $RB_{(1+2)*(3+4)}$ or $RB_{(1*3)+(2*4)}$ from Eq. (7.47). By using a superscript notation that is consistent with the subscript notation for the above four rule bases, RB_E will be obtained from Eq. (7.79).

$$RB_E = RB^{(1,1+2,1)*(1,2+2,2)} = RB^{(1,1*1,2)+(2,1*2,2)} \tag{7.79}$$

In this case, the ERB RB_E for both the vertical-horizontal and the horizontal-vertical merging of $RB_{II}^{1,1}$, $RB_{1x1}^{2,1}$, $RB_{1x1}^{1,2}$ and $RB_{1x1}^{2,2}$ will be presented by the following Boolean matrix and binary relation:

$$RB_E: \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 0 & 0 & 1 \\ 12 & 0 & 0 & 0 & 1 \\ 21 & 0 & 1 & 0 & 0 \\ 22 & 0 & 1 & 0 & 0 \end{array} \tag{7.80}$$

$$RB_E: \{(11, 22), (12, 22), (21, 12), (22, 12)\} \tag{7.81}$$

Example 7.7

A MRB system with 2 levels and 2 layers is presented by the following matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & & RB_{II}^{1,2} \\ \text{level 2} & RB_{1x2}^{2,1} & RB_{1x1}^{2,2} \end{array} \tag{7.82}$$

The output-input interconnections for the above MRB system are given by the following matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & & o_1^{1,2} \\ \text{level 2} & \begin{array}{c} o_1^{2,1} = i_1^{1,2} \\ o_2^{2,1} = i_1^{2,2} \end{array} & o_1^{2,2} \end{array} \tag{7.83}$$

Equation (7.82) shows that the MRB system has one input and two outputs. It also shows that the rule base in level 1 of layer 1 is missing, the rule base in level 2 of layer 1 has one input and two outputs, the rule base in level 1 of layer 2 is an IRB with one input and one output whereas the rule base in level 2 of layer 2 has one input and one output. In addition, Eq. (7.83) shows that the first output from the rule base in level 2 of layer 1 is self standing as it the same as the first input to the IRB in level 1 of layer 2 whereas the second output from the rule base in level 2 of layer 1 is the same as the input to the rule base in level 2 of layer 2.

Although the only purpose of the IRB is to represent a self standing output in level 1 of layer 2, this rule base must be counted separately. Therefore, the MRB system consists of three rule bases, as shown in Fig. 7.10.

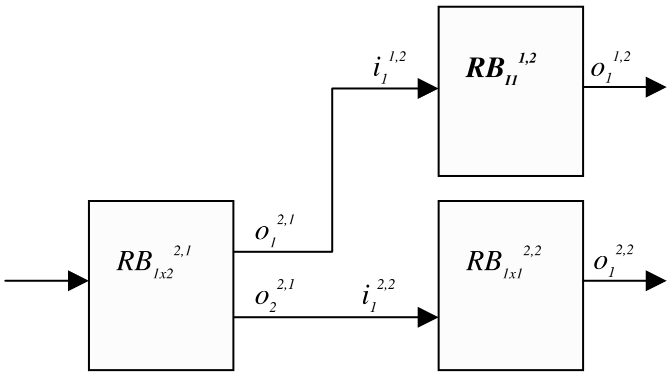

Fig. 7.10. Multiple rule base system with three rule bases and self standing output in level 1 of layer 2

The rule bases $RB_{1x2}^{2,1}$, $RB_{11}^{1,2}$ and $RB_{1x1}^{2,2}$ are presented by the following Boolean matrices and binary relations:

$$RB_{2x1}^{1,2}: \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 1 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 0 \end{array} \tag{7.84}$$

$$RB_{2x1}^{1,2}: \{(1, 22), (2, 11)\} \tag{7.85}$$

$$RB_{11}^{1,2}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{7.86}$$

$$RB_{11}^{1,2}: \{(1, 1), (2, 2)\} \tag{7.87}$$

$$RB_{1x1}^{2,2}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{7.88}$$

$$RB_{1x1}^{2,2}: \{(1, 2), (2, 1)\} \tag{7.89}$$

The ERB RB_E for the MRB system is derived by the formula in Eq. (7.90) and is presented by the Boolean matrix and the binary relation in Eqs. (7.91)–(7.92).

$$RB_E = RB_{1x2}^{2,1} * (RB_{I1}^{1,2} + RB_{1x1}^{2,2}) \tag{7.90}$$

$$RB_E: \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 \end{array} \tag{7.91}$$

$$RB_E: \{(1, 21), (2, 12)\} \tag{7.92}$$

Example 7.8
A MRB system with 2 levels and 2 layers is presented by the following matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_{1x1}^{1,1} & RB_{1x1}^{1,2} \\ \text{level 2} & RB_{1x1}^{2,1} & RB_{I1}^{2,2} \end{array} \tag{7.93}$$

The output-input interconnections for the above MRB system are given by the following matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1^{1,1} = i_1^{1,2} & o_1^{1,2} \\ \text{level 2} & o_1^{2,1} = i_1^{2,2} & o_1^{2,2} \end{array} \tag{7.94}$$

Equation (7.93) shows that the MRB system has two inputs and two outputs. It also shows that each of the rule bases including the IRB in level 2 of layer 2 has one input and one output. In addition, Eq. (7.94) shows that the first output from the rule base in level 2 of layer 1 is self standing as it is the same as the first input to the IRB in level 2 of layer 2 whereas the first output from the rule base in level 1 of layer 1 is the same as the first input to the rule base in level 1 of layer 2.

Although the only purpose of the IRB is to represent a self standing output in level 2 of layer 2, this rule base must be counted separately. Therefore, the MRB system consists of four rule bases, as shown in Fig. 7.11.

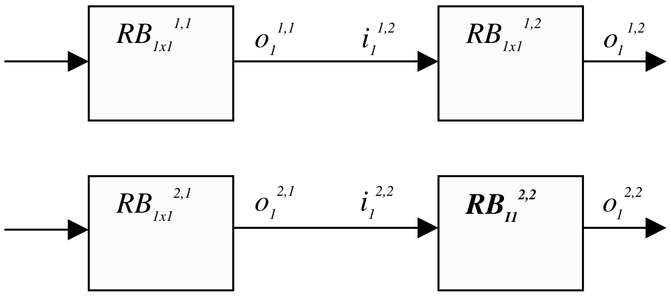

Fig. 7.11. Multiple rule base system with four rule bases and self standing output in level 2 of layer 2

The rule bases $RB_{1x1}^{1,1}$, $RB_{1x1}^{2,1}$, $RB_{1x1}^{1,2}$ and $RB_{II}^{2,2}$ are presented by the following Boolean matrices and binary relations:

$RB_{1x1}^{1,1}$: (7.95)

Inputs/Outputs	1	2
1	0	1
2	0	1

$$RB_{1x1}^{1,1}: \{(1, 2), (2, 2)\} \tag{7.96}$$

$RB_{1x1}^{2,1}$: (7.97)

Inputs/Outputs	1	2
1	0	1
2	1	0

$$RB_{1x1}^{2,1}: \{(1, 2), (2, 1)\} \tag{7.98}$$

$RB_{1x1}^{1,2}$: (7.99)

Inputs/Outputs	1	2
1	1	0
2	1	0

$$RB_{1x1}^{1,2}: \{(1, 1), (2, 1)\} \tag{7.100}$$

$$RB_{II}^{2,2}: \quad \begin{array}{c|cc} \text{Inputs/Outputs} & 1 & 2 \\ \hline 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{7.101}$$

$$RB_{II}^{2,2}: \{(1, 1), (2, 2)\} \tag{7.102}$$

The ERB RB_E for the MRB system is derived again as either $RB_{(1+2)*(3+4)}$ or $RB_{(1*3)+(2*4)}$ from Eq. (7.47). In particular, RB_E will be obtained from Eq. (7.79) using the superscript notation introduced there.

So, the ERB RB_E for both the vertical-horizontal and the horizontal-vertical merging of $RB_{II}^{1,1}$, $RB_{1x1}^{2,1}$, $RB_{1x1}^{1,2}$ and $RB_{II}^{2,2}$ will be presented by the following Boolean matrix and binary relation:

$$RB_E: \quad \begin{array}{c|cccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ \hline 11 & 0 & 1 & 0 & 0 \\ 12 & 1 & 0 & 0 & 0 \\ 21 & 0 & 1 & 0 & 0 \\ 22 & 1 & 0 & 0 & 0 \end{array} \tag{7.103}$$

$$RB_E: \{(11, 12), (12, 11), (21, 12), (22, 11)\} \tag{7.104}$$

7.5 Total and Partial Identity Lines

As shown in the previous section, self standing inputs and outputs in a MRB system can be presented by IRBs. In this case, the corresponding identity mappings are at either side of the MRB system and spanning across one or more successive layers starting with the first layer or ending with the last layer.

However, sometimes we may have identity mappings across the whole of a MRB system, which are referred to as identity lines. As these lines span all the layers of the MRB system, they are called total lines. In this case, any linguistic value of an input at the beginning of a total line propagates unchanged as an output at the end of the line. The number of layers spanned by a total line gives the length of the line and it is always equal to the number of layers in the MRB system.

Similarly, we may have identity mappings somewhere in the middle of a MRB system, which are also referred to as identity mapping lines. As the inputs of these lines are the same as the outputs from a particular rule base and their outputs are the same as the inputs to another rule base, such lines are called partial lines. In this case, any linguistic value of an input in the beginning of a partial line propagates unchanged as an output at the end of the line. The number of layers spanned by a partial line gives the length of this line and it is always greater than or equal to 1 and less than or equal to the overall number of layers in the MRB system minus 2.

Examples 7.9–7.10 show how a total line of length 2 can be presented by a couple of IRBs RB_{II} and a couple of P-T RBs RB_{PTI}, respectively. Example 7.11 shows how a partial line of length 1 can be presented by an IRB RB_{II}. Also, all three examples show how an ERB RB_E can be derived by either vertical-horizontal or horizontal-vertical merging in accordance with the interchangeability property.

Example 7.9

A MRB system with 2 levels and 2 layers is presented by the following matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_{II}^{1,1} & RB_{II}^{1,2} \\ \text{level 2} & RB_{1x1}^{2,1} & RB_{1x1}^{2,2} \end{array} \tag{7.105}$$

The output-input interconnections for the above MRB system are given by the following matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1^{1,1} = i_1^{1,2} & o_1^{1,2} \\ \text{level 2} & o_1^{2,1} = i_1^{2,2} & o_1^{2,2} \end{array} \tag{7.106}$$

Equation (7.105) shows that the MRB system has two inputs and two outputs. It also shows that each of the rule bases including the IRB in level 1 of layers 1 and the IRB in level 1 of layer 2 has one input and one output. In addition, Eq. (7.106) shows that the two IRBs in level 1 of layers 1 and 2 represent a total line whereas the first output from the rule base in level 2 of layer 1 is the same as the first input to the rule base in level 2 of layer 2.

Although the only purpose of the two IRBs is to represent a total line in level 1 of layers 1 and 2, these rule bases must be counted separately. Therefore, the MRB system consists of four rule bases, as shown in Fig. 7.12.

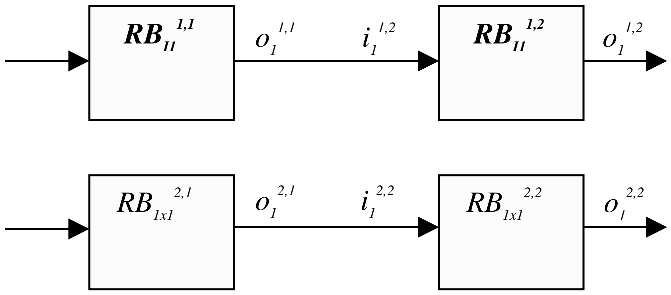

Fig. 7.12. Multiple rule base system with four rule bases and total line in level 1 of layers 1 and 2

The rule bases $RB_{II}^{1,1}$, $RB_{1x1}^{2,1}$, $RB_{II}^{1,2}$ and $RB_{1x1}^{2,2}$ are presented by the following Boolean matrices and binary relations:

$$RB_{II}^{1,1}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{7.107}$$

$$RB_{II}^{1,1}: \{(1, 1), (2, 2)\} \tag{7.108}$$

$$RB_{1x1}^{2,1}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{7.109}$$

$$RB_{1x1}^{2,1}: \{(1, 1), (2, 1)\} \tag{7.110}$$

$$RB_{II}^{1,2}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{7.111}$$

$$RB_{II}^{1,2}: \{(1, 1), (2, 2)\} \tag{7.112}$$

$$RB_{1x1}^{2,2}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{7.113}$$

$$RB_{1x1}^{2,2}: \{(1, 2), (2, 2)\} \tag{7.114}$$

The ERB RB_E for the MRB system is derived as either $RB_{(1+2)*(3+4)}$ or $RB_{(1*3)+(2*4)}$ from Eq. (7.47). In particular, RB_E will be obtained from Eq. (7.79) using the superscript notation introduced there.

So, the ERB RB_E for the vertical-horizontal and the horizontal-vertical merging of $RB_{II}^{1,1}$, $RB_{1x1}^{2,1}$, $RB_{II}^{1,2}$ and $RB_{1x1}^{2,2}$ will be presented by the following Boolean matrix and binary relation:

$$RB_E: \quad \begin{array}{ccccc} \text{Inputs/Outputs} & 11 & 12 & 21 & 22 \\ 11 & 0 & 1 & 0 & 0 \\ 12 & 0 & 1 & 0 & 0 \\ 21 & 0 & 0 & 0 & 1 \\ 22 & 0 & 0 & 0 & 1 \end{array} \tag{7.115}$$

$$RB_E: \{(11, 12), (12, 12), (21, 22), (22, 22)\} \tag{7.116}$$

Example 7.10

A MRB system with 2 levels and 2 layers is presented by the following matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_{PT1}^{1,1} & RB_{PT1}^{1,2} \\ \text{level 2} & RB_{1x1}^{2,1} & RB_{1x1}^{2,2} \end{array} \tag{7.117}$$

The output-input interconnections for the above MRB system are given by the following matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1^{1,1} = i_1^{1,2} & o_1^{1,2} \\ \text{level 2} & o_1^{2,1} = i_1^{2,2} & o_1^{2,2} \end{array} \tag{7.118}$$

Equation (7.117) shows that the MRB system has two inputs and two outputs. It also shows that each of the rule bases including the P-T RB in level 1 of layer 1 and the P-T RB in level 1 of layer 2 has one input and one output. In addition, Eq. (7.118) shows that the two P-T RBs in level 1 of layers 1 and 2 represent a total line whereas the first output from the rule base in level 2 of layer 1 is the same as the first input to the rule base in level 2 of layer 2.

Although the only purpose of the two P-T RBs is to represent a total line in level 1 of layers 1 and 2, these rule bases must be counted separately. Therefore, the MRB system consists of four rule bases, as shown in Fig. 7.13.

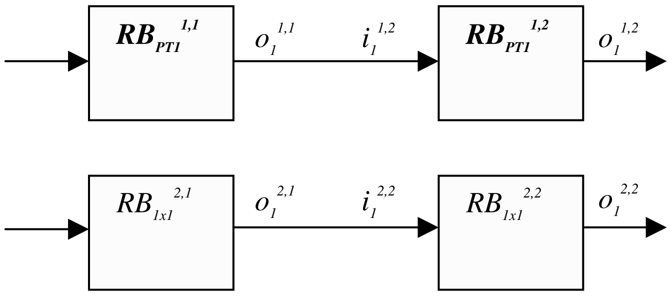

Fig. 7.13. Multiple rule base system with four rule bases and total line in level 1 of layers 1 and 2

The rule bases $RB_{PT1}^{1,1}$, $RB_{1x1}^{2,1}$, $RB_{PT1}^{1,2}$, and $RB_{1x1}^{2,2}$ are presented by the following Boolean matrices and binary relations:

$$RB_{PT1}^{1,1}: \quad \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \end{array} \tag{7.119}$$

$$RB_{PT1}^{1,1}: \{(1, 3), (2, 1), (3, 2)\} \tag{7.120}$$

$$RB_{1x1}^{2,1}: \quad \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \end{array} \tag{7.121}$$

$$RB_{1x1}^{2,1}: \{(1, 1), (2, 1), (3, 2)\} \tag{7.122}$$

$$RB_{PT1}^{1,2}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 \end{array} \quad (7.123)$$

$$RB_{PT1}^{1,2}: \{(1, 2), (2, 3), (3, 1)\} \quad (7.124)$$

$$RB_{1x1}^{2,2}: \begin{array}{cccc} \text{Inputs/Outputs} & 1 & 2 & 3 \\ 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 0 & 1 \end{array} \quad (7.125)$$

$$RB_{1x1}^{2,2}: \{(1, 2), (2, 3), (3, 3)\} \quad (7.126)$$

The ERB RB_E for the MRB system is derived as either $RB_{(1+2)*(3+4)}$ or $RB_{(1*3)+(2*4)}$ from Eq. (7.47). In particular, RB_E will be obtained from Eq. (7.79) using the superscript notation introduced there.

So, the ERB RB_E for both vertical-horizontal and the horizontal-vertical merging of $RB_{PT1}^{1,1}$, $RB_{1x1}^{2,1}$, $RB_{PT1}^{1,2}$ and $RB_{1x1}^{2,2}$ will be presented by the following Boolean matrix and binary relation:

$$RB_E: \begin{array}{cccccccccc} \text{Inputs/Outputs} & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\ 11 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 21 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 22 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 23 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 31 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \quad (7.127)$$

$$\begin{aligned} RB_E: \{&(11, 12), (12, 12), (13, 13), \\ &(21, 22), (22, 22), (23, 23), \\ &(31, 32), (32, 32), (33, 33)\} \end{aligned} \quad (7.128)$$

Example 7.11

A MRB system with 2 levels and 3 layers is presented by the following matrix:

$$\begin{array}{cccc}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & RB_{1x1}^{1,1} & RB_{1x1}^{1,2} & RB_{1x1}^{1,3} \\
\text{level 2} & RB_{1x1}^{2,1} & RB_{II}^{2,2} & RB_{1x1}^{2,3}
\end{array} \tag{7.129}$$

The output-input interconnections for the above MRB system are given by the following matrix:

$$\begin{array}{cccc}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & o_1^{1,1} = i_1^{1,2} & o_1^{1,2} = i_1^{1,3} & o_1^{1,3} \\
\text{level 2} & o_1^{2,1} = i_1^{2,2} & o_1^{2,2} = i_1^{2,3} & o_1^{2,3}
\end{array} \tag{7.130}$$

Equation (7.129) shows that the MRB system has two inputs and two outputs. It also shows that each of the rule bases including the IRB in level 2 of layer 2 has one input and one output. In addition, Eq. (7.130) shows that the IRB in level 1 of layer 2 represents a partial line as the input at its beginning is the same as the output from the rule base in level 2 of layer 1 while the output at its end is the same as the input to the rule base in level 2 of layer 3. Also, the first output from the rule base in level 1 of layer 1 is the same as the first input to the rule base in level 1 of layer 2 and the first output from the rule base in level 1 of layer 2 is the same as the first input to the rule base in level 1 of layer 3.

Although the only purpose of the IRB is to represent a partial line in level 2 of layer 2, this rule base must be counted separately. Therefore, the MRB system consists of six rule bases, as shown in Fig. 7.13.

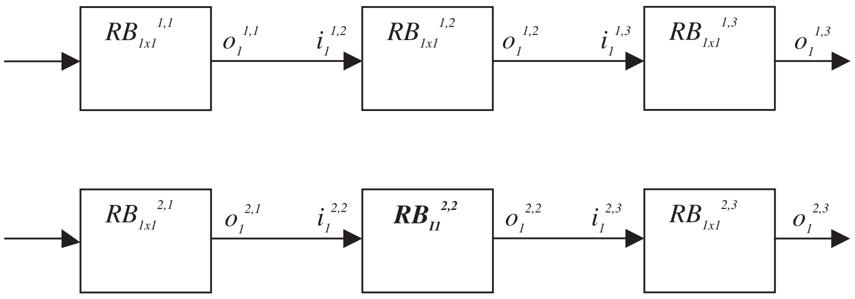

Fig. 7.14. Multiple rule base system with six rule bases and partial line in level 2 of layer 2

The rule bases $RB_{1x1}^{1,1}$, $RB_{1x1}^{2,1}$, $RB_{1x1}^{1,2}$, $RB_{11}^{2,2}$, $RB_{1x1}^{1,3}$ and $RB_{1x1}^{2,3}$ are presented by the following Boolean matrices and binary relations:

$$RB_{1x1}^{1,1}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \tag{7.131}$$

$$RB_{1x1}^{1,1}: \{(1, 1), (2, 1)\} \tag{7.132}$$

$$RB_{1x1}^{2,1}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{7.133}$$

$$RB_{1x1}^{2,1}: \{(1, 2), (2, 1)\} \tag{7.134}$$

$$RB_{1x1}^{1,2}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{7.135}$$

$$RB_{1x1}^{1,2}: \{(1, 2), (2, 1)\} \tag{7.136}$$

$$RB_{11}^{2,2}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{array} \tag{7.137}$$

$$RB_{11}^{2,2}: \{(1, 1), (2, 2)\} \tag{7.138}$$

$$RB_{1x1}^{1,3}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \tag{7.139}$$

$$RB_{1x1}^{1,3}: \{(1, 2), (2, 2)\} \tag{7.140}$$

$$RB_{1x1}^{2,3}: \quad \begin{array}{ccc} \text{Inputs/Outputs} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \tag{7.141}$$

$$RB_{1x1}^{2,3}: \{(1, 2), (2, 1)\} \tag{7.142}$$

The ERB RB_E for the MRB system is derived as either $RB_{(1+2)*(3+4)*(5+6)}$ or $RB_{(1*3*5)+(2*4*6)}$ by extending Eq. (7.47) for a rule base with three layers. In particular, RB_E will be obtained from Eq. (7.79) using the superscript notation introduced there.

So, the ERB RB_E for the vertical-horizontal and the horizontal-vertical merging of $RB_{1x1}^{1,1}, RB_{1x1}^{2,1}, RB_{1x1}^{1,2}, RB_{11}^{2,2}, RB_{1x1}^{1,3}$ and $RB_{1x1}^{2,3}$ will be presented by the following Boolean matrix and binary relation:

RB_E: (7.143)

Inputs/Outputs	11	12	21	22
11	0	0	1	0
12	0	0	0	1
21	0	0	1	0
22	0	0	0	1

$$RB_E: \{(11, 21), (12, 22), (21, 21), (22, 22)\} \tag{7.144}$$

7.6 Comparative Analysis of Formal Transformation Techniques

The basic operations for formal manipulation of fuzzy rule bases introduced in Sects. 7.2–7.5 provide a powerful tool for analysis and synthesis of fuzzy systems. In particular, the repetitive merging manipulations and the associated combined merging manipulations can be used for simplifying complex MRB systems by representing them with equivalent SRB systems. In this context, the associative and interchangeable features of merging manipulations make the underlying formal transformation process quite flexible. In addition, the self standing inputs, self standing outputs, total identity lines and partial identity lines are effective means of converting MRB systems from an arbitrarily complex form into a simpler canonical form, as shown further in Chapter 8. In this case, the ability of rule bases to represent the above topological peculiarities facilitates significantly the associated formal transformation process.

The repetitive and combined merging manipulations affect actively levels and layers within a MRB system as part of the design process, i.e. during synthesis, whereas the self standing inputs, self standing outputs, total identity lines and partial identity lines affect passively only levels in this system as part of the study process, i.e. during analysis. Also, the result in all cases above is guaranteed whereby both merging manipulations as well as inputs, outputs, total and partial lines lead to unique solutions. The only requirement is that repetitive merging is applied in the context of associativity whereas combined merging is applied in the context in interchangeability.

The considerations presented above on formal transformation techniques for fuzzy rule bases provide essential information about the main characteristics of these techniques. These characteristics are summarised in Table 7.1.

Table 7.1. Characteristics of formal transformation techniques for fuzzy rule bases

Technique/ Characteristic	Task	Impact	Component	Result	Solution
Repetitive merging	synthesis	active	level/layer	guaranteed	unique
Combined merging	synthesis	active	level/layer	guaranteed	unique
Self standing inputs	analysis	passive	level	guaranteed	unique
Self standing outputs	analysis	passive	lavel	guaranteed	unique
Total identity lines	analysis	passive	level	guaranteed	unique
Partial identity lines	analysis	passive	level	guaranteed	unique

7.7 Application Range of Formal Transformation Techniques

The formal transformation techniques introduced in this chapter are applicable to a wide range of MRB systems. These techniques can be applied to Mamdami, Sugeno and Tsukamoto systems, most CON and DIS systems, MO and SO systems, as well as FF and FB systems.

Examples 7.1–7.11 describe implicitly a fuzzy system of Mamdami or Tsukamoto type. In order to apply the associated rule base manipulation algorithms to Sugeno systems, the crisp outputs from all rule bases residing in all layers from the first to the last but one have to fuzzified, i.e. converted into linguistic values. Obviously, the outputs from the rule bases in the last layer can be kept with their crisp values.

Examples 7.1–7.11 can be extended easily in accordance with the considerations in Sect.4.6, if we would like them to describe explicitly Mamdami, Sugeno and Tsukamoto systems. However, this has not been done in this chapter in order to simplify the notations and to put the emphasis on the transformation rather than the presentation process, which was dealt with in the Chapter 4.

As far as CON and DIS systems are concerned, the formal manipulation techniques are directly applicable to most of them. In this case, almost all rule bases in the MRB system must be of the same CADR or CACR type, i.e. with CON antecedents and either DIS or CON rules. This is so because when the antecedents in a particular rule base are of DIS type, they can not be merged horizontally with the consequents in another rule base on the left of it due to the CON type of consequents by definition. The only exception in this case can be made for the rule bases in the first layer of a MRB system whose antecedents can be of any type because they do not take part in any horizontal merging manipulations.

The formal transformation techniques presented in this chapter facilitate the complexity management in fuzzy systems. These techniques allow the compressed information about the fuzzy rule bases in a MRB system in the form of Boolean matrices or binary relations to be transformed appropriately for the purpose of analysis and synthesis of fuzzy systems.

However, the formal transformation techniques demonstrated so far deal with MRB systems, which contain only FF rule bases. In order to show how these techniques can be used for MRB systems containing also FB rule bases, a detailed study is presented in the next chapter.

8 Formal Transformation of Feedback Rule Bases

8.1 Preliminaries on Feedback Rule Bases

As already mentioned in Sect. 2.4, the information flow in a SRB system is usually in a forward direction, i.e. from the inputs to the outputs of the rule base, but sometimes there may also be information flow in a backward direction, i.e. from some outputs to some inputs. This FB may be as simple as an identity mapping line whereby an output is fed back unchanged directly into an input. In other cases, the FB may be of a more complex nature whereby the output passes through a *feedback rule base* (FRB) that changes it before it is fed into the corresponding input.

As discussed in Sect. 2.5, some of the information flow in a MRB system may be in a backward direction, i.e. from an output of a rule base residing in a particular layer to an input of the same rule base or another rule base residing in the same or a preceding layer. In analogy with the SRB system case above, this FB could be either simple or complex, i.e. the output is either unchanged or changed before being fed into the corresponding input.

Several types of FB interconnections are considered further in this chapter and illustrated by examples. Most of these examples are about MRB systems in which the FB is more complex than in SRB systems. In these cases, the ERB of the MRB system is derived by transforming the FB interconnections into FF interconnections, wherever possible, and then applying the techniques introduced in Chapter 7, or by transforming all complex feedback interconnections into simple feedback interconnections and then checking the ERB accordingly.

8.2 Transforsmation of Rule Bases with Simple Feedback

In a fuzzy rule based system with simple feedback, the linguistic value of the corresponding output in each rule is identical with the linguistic value of the associated input in the same rule. In this case, the rule base is constrained by this identity type of FB and therefore it is necessary to check if the constraints are met. A general case with simple FB is presented in Fig. 8.1.

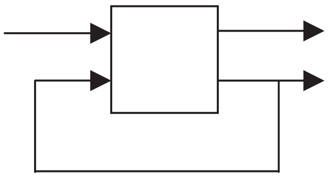

Fig. 8.1. Simple feedback

The notion of simple FB is illustrated further by six basic examples with SRB systems. It is assumed that each input and output in these examples can take three linguistic values, e.g. small, medium and big. As already shown in Chapter 3, these linguistic values can be coded by the positive integers 1, 2 and 3, respectively. In particular, Example 8.1 shows how a 1×1 rule base can be presented by a simple FB pattern, Examples 8.2–8.3 refer to two diagonal patterns of simple FB in a 2×1 and a 1×2 rule base, respectively, whereas Examples 8.4–8.6 deal with three patterns of simple FB in a 2×2 rule base.

In all six examples, the question marks (?) in the Boolean matrices represent elements, which may be but are not necessarily 1's. Similarly, the subscript question marks ($_?$) of the associated pairs in the corresponding binary relations represent only possible but not necessarily existing maplets. In the case when all question marks in a Boolean matrix represent 1's and all associated subscript question marks in the corresponding binary relation represent existing maplets, the underlying rule base is both complete and exhaustive. Otherwise, it is either incomplete or non-exhaustive.

Example 8.1

This example illustrates a simple FB pattern for a 1×1 rule base. The rule base RB_{1x1} has one input (i_1) and one output (o_1) whereby o_1 is fed back unchanged into i_1, i.e. for each rule the linguistic value of o_1 is the same as the linguistic value of i_1. The Boolean matrix and the binary relation for this rule base must reflect this FB constraint, as shown by Eqs. (8.1)–(8.2).

$$RB_{1x1}: \quad \begin{array}{cccc} i_1/o_1 & 1 & 2 & 3 \\ 1 & ? & 0 & 0 \\ 2 & 0 & ? & 0 \\ 3 & 0 & 0 & ? \end{array} \tag{8.1}$$

$$RB_{1x1}: \{(1, 1)_?, (2, 2)_?, (3, 3)_?\} \tag{8.2}$$

Example 8.2

This example illustrates a simple FB pattern for a 2×1 rule base. The rule base RB_{2x1} has two inputs (i_1 and i_2) and one output (o_1) whereby o_1 is fed back unchanged into i_2, i. e. for each rule the linguistic value of o_1 is the same as the linguistic value of i_2. The block Boolean matrix and the binary relation for this rule base must reflect this FB constraint, as shown by Eqs. (8.3)–(8.4).

$$RB_{2x1}: \begin{array}{cccc} i_1\, i_2 \,/\, o_1 & 1 & 2 & 3 \\ 11 & ? & 0 & 0 \\ 12 & 0 & ? & 0 \\ 13 & 0 & 0 & ? \\ & & & \\ 21 & ? & 0 & 0 \\ 22 & 0 & ? & 0 \\ 23 & 0 & 0 & ? \\ & & & \\ 31 & ? & 0 & 0 \\ 32 & 0 & ? & 0 \\ 33 & 0 & 0 & ? \end{array} \tag{8.3}$$

$$\begin{aligned} RB_{2x1}\text{: } \{&(11, 1)_?, (12, 2)_?, (13, 3)_?, \\ &(21, 1)_?, (22, 2)_?, (23, 3)_?, \\ &(31, 1)_?, (32, 2)_?, (33, 3)_?\} \end{aligned} \tag{8.4}$$

Example 8.3

This example illustrates a simple FB pattern for a 1×2 rule base. The rule base RB_{1x2} has one input (i_1) and two outputs (o_1 and o_2) whereby o_2 is fed back unchanged into i_1, i.e. for each rule, the linguistic value of o_2 is the same as the linguistic value of i_1. The block Boolean matrix and the binary relation for this rule base must reflect this FB constraint, as shown by Eqs. (8.5)–(8.6).

$$RB_{1x2}: \begin{array}{cccccccccc} i_1 \,/\, o_1\, o_2 & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\ 1 & ? & 0 & 0 & ? & 0 & 0 & ? & 0 & 0 \\ 2 & 0 & ? & 0 & 0 & ? & 0 & 0 & ? & 0 \\ 3 & 0 & 0 & ? & 0 & 0 & ? & 0 & 0 & ? \end{array} \tag{8.5}$$

$$\begin{aligned} RB_{1x2}\text{: } \{&(1, 11)_?, (2, 12)_?, (3, 13)_?, \\ &(1, 21)_?, (2, 22)_?, (3, 23)_?, \\ &(1, 31)_?, (2, 32)_?, (3, 33)_?\} \end{aligned} \tag{8.6}$$

Example 8.4

This example illustrates a simple FB pattern for a 2×2 rule base. The rule base RB_{2x2} has two inputs (i_1 and i_2) and two outputs (o_1 and o_2) whereby o_1 is fed back unchanged into i_2, i.e. for each rule, the linguistic value of o_1 is the same as the linguistic value of i_2. The block Boolean matrix and the binary relation for this rule base must reflect this FB constraint, as shown by Eqs. (8.7)–(8.8).

$$RB_{2x2}: \begin{array}{c|ccc|ccc|ccc}
i_1\, i_2 \,/\, o_1\, o_2 & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\
\hline
11 & ? & ? & ? & 0 & 0 & 0 & 0 & 0 & 0 \\
12 & 0 & 0 & 0 & ? & ? & ? & 0 & 0 & 0 \\
13 & 0 & 0 & 0 & 0 & 0 & 0 & ? & ? & ? \\
\hline
21 & ? & ? & ? & 0 & 0 & 0 & 0 & 0 & 0 \\
22 & 0 & 0 & 0 & ? & ? & ? & 0 & 0 & 0 \\
23 & 0 & 0 & 0 & 0 & 0 & 0 & ? & ? & ? \\
\hline
31 & ? & ? & ? & 0 & 0 & 0 & 0 & 0 & 0 \\
32 & 0 & 0 & 0 & ? & ? & ? & 0 & 0 & 0 \\
33 & 0 & 0 & 0 & 0 & 0 & 0 & ? & ? & ?
\end{array} \tag{8.7}$$

$$\begin{aligned}
RB_{2x2}: \{&(11, 11)_?, (11, 12)_?, (11, 13)_?, \\
&(12, 21)_?, (12, 22)_?, (12, 23)_?, \\
&(13, 31)_?, (13, 32)_?, (13, 33)_?, \\
&(21, 11)_?, (21, 12)_?, (21, 13)_?, \\
&(22, 21)_?, (22, 22)_?, (22, 23)_?, \\
&(23, 31)_?, (23, 32)_?, (23, 33)_?, \\
&(31, 11)_?, (31, 12)_?, (31, 13)_?, \\
&(32, 21)_?, (32, 22)_?, (32, 23)_?, \\
&(33, 31)_?, (33, 32)_?, (33, 33)_?\}
\end{aligned} \tag{8.8}$$

Example 8.5

This example illustrates a simple FB pattern for a 2×2 rule base. The rule base RB_{2x2} has two inputs (i_1 and i_2) and two outputs (o_1 and o_2) whereby o_2 is fed back unchanged into i_1, i.e. for each rule, the linguistic value of o_2 is the same as the linguistic value of i_1. The block Boolean matrix and the binary relation for this rule base must reflect this FB constraint, as shown by Eqs. (8.9)–(8.10).

$$RB_{2x2}: \begin{array}{c|ccc ccc ccc}
i_1\, i_2 \,/\, o_1\, o_2 & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\
11 & ? & 0 & 0 & ? & 0 & 0 & ? & 0 & 0 \\
12 & ? & 0 & 0 & ? & 0 & 0 & ? & 0 & 0 \\
13 & ? & 0 & 0 & ? & 0 & 0 & ? & 0 & 0 \\
21 & 0 & ? & 0 & 0 & ? & 0 & 0 & ? & 0 \\
22 & 0 & ? & 0 & 0 & ? & 0 & 0 & ? & 0 \\
23 & 0 & ? & 0 & 0 & ? & 0 & 0 & ? & 0 \\
31 & 0 & 0 & ? & 0 & 0 & ? & 0 & 0 & ? \\
32 & 0 & 0 & ? & 0 & 0 & ? & 0 & 0 & ? \\
33 & 0 & 0 & ? & 0 & 0 & ? & 0 & 0 & ?
\end{array} \tag{8.9}$$

$$\begin{aligned}
RB_{2x2}: \{&(11, 11)_?, (12, 11)_?, (13, 11)_?,\\
&(11, 21)_?, (12, 21)_?, (13, 21)_?,\\
&(11, 31)_?, (12, 31)_?, (13, 31)_?,\\
&(21, 12)_?, (22, 12)_?, (23, 12)_?,\\
&(21, 22)_?, (22, 22)_?, (23, 22)_?,\\
&(21, 32)_?, (22, 32)_?, (23, 32)_?,\\
&(31, 13)_?, (32, 13)_?, (33, 13)_?,\\
&(31, 23)_?, (32, 23)_?, (33, 23)_?,\\
&(31, 33)_?, (32, 33)_?, (33, 33)_?\}
\end{aligned} \tag{8.10}$$

Example 8.6

This example illustrates a simple FB pattern for a 2×2 rule base. The rule base RB_{2x2} has two inputs (i_1 and i_2) and two outputs (o_1 and o_2) whereby o_1 is fed back unchanged into i_2 and o_2 is fed back unchanged into i_1, i.e. for each rule, the linguistic value of o_1 is the same as the linguistic value of i_2 and the linguistic value of o_2 is the same as the linguistic value of i_1. The block Boolean matrix and the binary relation for this rule base must reflect this FB constraint, as shown by Eqs. (8.11)–(8.12).

$$RB_{2x2}: \begin{array}{c|ccc ccc ccc}
i_1\, i_2 \,/\, o_1\, o_2 & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\
11 & ? & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
12 & 0 & 0 & 0 & ? & 0 & 0 & 0 & 0 & 0 \\
13 & 0 & 0 & 0 & 0 & 0 & 0 & ? & 0 & 0 \\
21 & 0 & ? & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
22 & 0 & 0 & 0 & 0 & ? & 0 & 0 & 0 & 0 \\
23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ? & 0 \\
31 & 0 & 0 & ? & 0 & 0 & 0 & 0 & 0 & 0 \\
32 & 0 & 0 & 0 & 0 & 0 & ? & 0 & 0 & 0 \\
33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ?
\end{array} \tag{8.11}$$

$$\begin{aligned} RB_{2x2}\colon \{&(11, 11)_?, (12, 21)_?, (13, 31)_?, \\ &(21, 12)_?, (22, 22)_?, (23, 32)_?, \\ &(31, 13)_?, (32, 23)_?, (33, 33)_?\} \end{aligned} \tag{8.12}$$

8.3 Transformation of Rule Bases with Local Feedback

Local feedback is the most basic type of complex feedback. A fuzzy rule based system with local FB is constrained because the linguistic value of the corresponding output in each rule is mapped by a FB function onto a linguistic value of the associated input in the same rule. However, these two linguistic values are not supposed to be the same as in the case of simple FB because the FB function is usually a non-identity function. In the private case of an identity FB function, the local FB is reduced to the simple FB discussed in the previous section. In the general case of local FB, it is necessary to check that the associated rule base satisfies the constraints imposed by this FB. This general case is presented in Fig. 8.2.

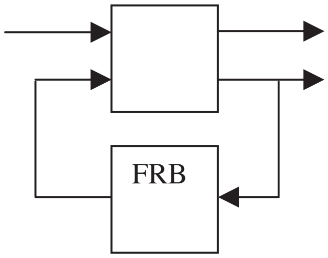

Fig. 8.2. Local feedback

The notion of local FB is illustrated further by six basic examples with SRB systems. These examples are similar to the ones from the previous section whereby the only difference here is the non-identity type of FB. In particular, Example 8.7 shows how local FB can be presented in a 1×1 rule base, Examples 8.8–8.9 refer to two types of local FB in a 2×1 and a 1×2 rule base, respectively, whereas Examples 8.10–8.12 deal with three types of local FB in a 2×2 rule base.

The presence of a non-identity FB function in these examples is not sufficient to classify the corresponding rule bases as MRB systems. Although such a function has an underlying FRB, there must be at least two rule bases interconnected in a FF manner in order to have a MRB system. In this context, the procedure described in the examples below starts with a

SRB system which is transformed into a MRB system by moving the FB function from the FB loop to a newly introduced FF path and representing this function as a rule base. Once the FB loop has been removed, the MRB system is transformed into an equivalent SRB system by vertical and horizontal merging manipulations.

For consistency, the notation used further in this section and in the whole current chapter is the same for both SRB and MRB systems, i.e. with an explicit presentation of each level and layer. In order to trace all FB and FF interconnections during the formal transformation procedure, the matrices used in the examples contain not only references to all individual rule bases but also to all inputs and outputs for each rule base as well as to all output-input FB functions available.

Example 8.7
A 1×1 rule base with local FB is presented by the followingmatrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB,\, i_1, o_1 \end{array} \tag{8.13}$$

The output-input interconnections for this rule base are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & F(o_1) = i_1 \end{array} \tag{8.14}$$

Equation (8.13) shows that the rule base *RB* has one input (i_1) and one output (o_1). In addition, Eq. (8.14) shows that the output from the rule base o_1 is mapped by the FB function *F* onto the input to the rule base i_1.

By introducing a second layer with a rule base RB_F that replaces the FB function *F*, the initial SRB system with complex FB is transformed into an equivalent MRB system with simple FB that embraces the two rule bases in sequence RB and RB_F.

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB,\, i_1, o_1 & RB_F,\, i_F, o_F \end{array} \tag{8.15}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1 = i_F & o_F = i_1 \end{array} \tag{8.16}$$

The transformation of the SRB system into an equivalent MRB system has led to the appearance of the rule base RB_F in the FF part of the MRB system as well as the appearance of two new interconnection variables i_F and o_F. In this context, Eq. (8.15) shows that the MRB system has one input (i_1) and one output (o_F). In addition, Eq. (8.16) shows that the output o_1 from RB is the same as the input i_F to RB_F whereas the output o_F from RB_F is the same as the input i_1 to RB.

The MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.17).

$$RB_E = RB * RB_F \tag{8.17}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_1, o_F \end{array} \tag{8.18}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_F = i_1 \end{array} \tag{8.19}$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby the linguistic values of the output o_F are fed back unchanged into the input i_1 for each of the fuzzy rules. In other words, the rule base RB_E must have the same structure as the rule base RB_{1x1} from Example 8.1, as specified by Eqs. (8.1)–(8.2).

Example 8.8
A 2×1 rule base with localFB is presented by the followingmatrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB, i_1, i_2, o_1 \end{array} \tag{8.20}$$

The output-input interconnections for this rule base are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & F(o_1) = i_2 \end{array} \qquad (8.21)$$

Equation (8.20) shows that the rule base *RB* has two inputs (i_1 and i_2) and one output (o_1). In addition, Eq. (8.21) shows that the output from the rule base o_1 is mapped by the FB function *F* onto the input to the rule base i_2.

By introducing a second layer with a rule base RB_F that replaces the FB function *F*, the initial SRB system with complex FB is transformed into an equivalent MRB system with simple FB that embraces the two rule bases in sequence RB and RB_F.

The MRB system is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB,\ i_1, i_2, o_1 & RB_F,\ i_F, o_F \end{array} \qquad (8.22)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1 = i_F & o_F = i_2 \end{array} \qquad (8.23)$$

The transformation of the SRB system into an equivalent MRB system has led to the appearance of RB_F in the FF part of the MRB system as well as the appearance of the two new interconnection variables i_F and o_F. In this context, Eq. (8.22) shows that the MRB system has two inputs (i_1 and i_2) and one output (o_F). In addition, Eq. (8.23) shows that the output o_1 from *RB* is the same as the input i_F to RB_F whereas the output o_F from RB_F is the same as the input i_2 to *RB*.

The MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.24).

$$RB_E = RB * RB_F \qquad (8.24)$$

The SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_1, i_2, o_F \end{array} \qquad (8.25)$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_F = i_2 \end{array} \tag{8.26}$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby the linguistic values of the output o_F are fed back unchanged into the input i_2 for each of the fuzzy rules. In other words, the rule base RB_E must have the same structure as the rule base RB_{2x1} from Example 8.2, as specified by Eqs. (8.3)–(8.4).

Example 8.9
A 1×2 rule base with local FB is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB,\, i_1,\, o_1,\, o_2 \end{array} \tag{8.27}$$

The output-input interconnections for this rule base are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & F(o_2) = i_1 \end{array} \tag{8.28}$$

Equation (8.27) shows that the rule base RB has one input (i_1) and two outputs (o_1 and o_2). In addition, Eq. (8.28) shows that the output from the rule base o_2 is mapped by the FB function F onto the input to the rule base i_1.

By introducing a second layer with two levels such that the first level is occupied by an IRB RB_I mapping the output o_1 and the second level is occupied by a rule base RB_F replacing the FB function F, the initial SRB system with complex FB is transformed into an equivalent MRB system with simple FB and three rule bases (RB, RB_I and RB_F). In this case, RB_I and RB_F are standing in parallel in layer 2 whereas RB and RB_F are standing in sequence in level 2.

The MRB system is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & & RB_I,\, i_I,\, o_I \\ \text{level 2} & RB,\, i_1,\, o_1,\, o_2 & RB_F,\, i_F,\, o_F \end{array} \tag{8.29}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & & o_I \\ \text{level 2} & o_1 = i_I & o_F = i_1 \\ & o_2 = i_F & \end{array} \qquad (8.30)$$

The transformation of the SRB system into an equivalent MRB system has led to the appearance of RB_I and RB_F in the FF part of the MRB system as well as the appearance the four new interconnection variables i_I, o_I, i_F and o_F. In this context, Eq. (8.29) shows that the MRB system has one input (i_1) and two outputs (o_I and o_F). In addition, Eq. (8.30) shows that the output o_1 from *RB* is the same as the input i_I to RB_I and the output o_2 from *RB* is the same as the input i_F to RB_F, whereas the output o_F from RB_F is the same as the input i_1 to *RB*.

The MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.31).

$$RB_E = \text{RB} * (RB_I + RB_F) \qquad (8.31)$$

The SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_I,\ o_I,\ o_F \end{array} \qquad (8.32)$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_I \\ & o_F = i_1 \end{array} \qquad (8.33)$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby the linguistic values of the output o_F are fed back unchanged into the input i_1 for each of the fuzzy rules. In other words, the rule base RB_E must have the same structure as the rule base RB_{1x2} from Example 8.3, as specified by Eqs. (8.5)–(8.6).

Example 8.10

A 2×2 rule base with local FB is presented by the following matrix:

.

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB,\ i_1, i_2, o_1, o_2 \end{array} \tag{8.34}$$

The output-input interconnections for this rule base are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & F(o_1) = i_2 \end{array} \tag{8.35}$$

Equation (8.34) shows that the rule base *RB* has two inputs (i_1 and i_2) and two outputs (o_1 and o_2). In addition, Eq. (8.35) shows that the output from the rule base o_1 is mapped by the FB function *F* onto the input to the rule base i_2.

By introducing a second layer with two levels such that the first level is occupied by a rule base RB_F replacing the FB function *F* and the second level is occupied by an IRB RB_I mapping the output o_2, the initial SRB system with complex FB is transformed into an equivalent MRB system with simple FB and three rule bases (RB, RB_F and RB_I). In this case, RB_F and RB_I are standing in parallel in layer 2 whereas RB and RB_F are standing in sequence in level 1.

The MRB system is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB,\ i_1, i_2, o_1, o_2 & RB_F,\ i_F, o_F \\ \text{level 2} & & RB_I,\ i_I, o_I \end{array} \tag{8.36}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & \begin{array}{l} o_1 = i_F \\ o_2 = i_I \end{array} & o_F = i_2 \\ \text{level 2} & & o_I \end{array} \tag{8.37}$$

The transformation of the SRB system into an equivalent MRB system has led to the appearance of RB_F and RB_I in the FF part of the MRB system as well as the appearance the four new interconnection variables i_F, o_F, i_I and o_I. In this context, Eq. (8.36) shows that the MRB system has two inputs (i_1 and i_2) and two outputs (o_F and o_I). In addition, Eq. (8.37) shows that the output o_1 from *RB* is the same as the input i_F to RB_F and the output o_2 from

RB is the same as the input i_I to RB_I, whereas the output o_F from RB_F is the same as the input i_2 to *RB*.

The MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.38).

$$RB_E = \mathrm{RB} * (RB_F + RB_I) \tag{8.38}$$

The SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_I, i_2, o_F, o_I \end{array} \tag{8.39}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_F = i_2 \\ & o_I \end{array} \tag{8.40}$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby the linguistic values of the output o_F are fed back unchanged into the input i_I for each of the fuzzy rules. In other words, the rule base RB_E must have the same structure as the rule base RB_{2x2} from Example 8.4, as specified by Eqs. (8.7)–(8.8).

Example 8.11
A 2×2 rule base with local FB is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB, i_1, i_2, o_1, o_2 \end{array} \tag{8.41}$$

The output-input interconnections for this rule base are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & F(o_2) = i_1 \end{array} \tag{8.42}$$

Equation (8.41) shows that the rule base *RB* has two inputs (i_1 and i_2) and two outputs (o_1 and o_2). In addition, Eq. (8.42) shows that the output from the rule base o_2 is mapped by the FB function F onto the input to the rule base i_1.

By introducing a second layer with two levels such that the first level is occupied by an IRB RB_I mapping the output o_I and the second level is occupied by a rule base RB_F replacing the FB function F, the initial SRB system with complex FB is transformed into an equivalent MRB system with simple FB and three rule bases (RB, RB_I and RB_F). In this case, RB_I and RB_F are standing in parallel in layer 2 whereas RB and RB_F are standing in sequence in level 2.

The MRB system is presented by the following matrix:

$$\begin{array}{lll}
\text{level/layer} & \text{layer 1} & \text{layer 2} \\
\text{level 1} & & RB_I,\ i_I,\ o_I \\
\text{level 2} & RB,\ i_1,\ i_2,\ o_1,\ o_2 & RB_F,\ i_F,\ o_F
\end{array} \tag{8.43}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll}
\text{level/layer} & \text{layer 1} & \text{layer 2} \\
\text{level 1} & & o_I \\
\text{level 2} & \begin{array}{l} o_1 = i_I \\ o_2 = i_F \end{array} & o_F = i_1
\end{array} \tag{8.44}$$

The transformation of the SRB system into an equivalent MRB system has led to the appearance of RB_I and RB_F in the FF part of the MRB system as well as the appearance the four new interconnection variables i_I, o_I, i_F and o_F. In this context, Eq. (8.43) shows that the MRB system has two inputs (i_1 and i_2) and two outputs (o_I and o_F). In addition, Eq. (8.44) shows that the output o_1 from RB is the same as the input i_I to RB_I and the output o_2 from RB is the same as the input i_F to RB_F, whereas the output o_F from RB_F is the same as the input i_1 to RB.

The MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.45).

$$RB_E = \text{RB} * (RB_I + RB_F) \tag{8.45}$$

The SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_1, i_2, o_1, o_F \end{array} \tag{8.46}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_1 \\ & o_F = i_1 \end{array} \tag{8.47}$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby the linguistic values of the output o_F are fed back unchanged into the input i_1 for each of the fuzzy rules. In other words, the rule base RB_E must have the same structure as the rule base RB_{2x2} from Example 8.5, as specified by Eqs. (8.9)–(8.10).

Example 8.12
A 2×2 rule base with local FB is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB, i_1, i_2, o_1, o_2 \end{array} \tag{8.48}$$

The output-input interconnections for this rule base are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & F1(o_1) = i_2 \\ & F2(o_2) = i_1 \end{array} \tag{8.49}$$

Equation (8.48) shows that the rule base *RB* has two inputs (i_1 and i_2) and two outputs (o_1 and o_2). In addition, Eq. (8.49) shows that the output from the rule base o_1 is mapped by the FB function *F1* onto the input to the rule base i_2 whereas the output from the rule base o_2 is mapped by the FB function *F2* onto the input to the rule base i_1.

By introducing a second layer with two levels such that the first level is occupied by a rule base RB_{F1} replacing the FB function *F1* and the second level is occupied by a rule base RB_{F2} replacing the FB function *F2*, the initial SRB system with complex FB is transformed into an equivalent MRB system with simple FB and three rule bases (*RB*, RB_{F1} and RB_{F2}). In this case, RB_{F1} and RB_{F2} are standing in parallel in layer 2 whereas RB and RB_{F2} are standing in sequence in level 2.

The MRB system is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & & RB_{F1}, i_{F1}, o_{F1} \\ \text{level 2} & RB, i_1, i_2, o_1, o_2 & RB_{F2}, i_{F2}, o_{F2} \end{array} \tag{8.50}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & & o_{F1} = i_2 \\ \text{level 2} & \begin{array}{l} o_1 = i_{F1} \\ o_2 = i_{F2} \end{array} & o_{F2} = i_1 \end{array} \tag{8.51}$$

The transformation of the SRB system into an equivalent MRB system has led to the appearance of RB_{F1} and RB_{F2} in the FF part of the MRB system as well as the appearance the four new interconnection variables i_{F1}, o_{F1}, i_{F2} and o_{F2}. In this context, Eq. (8.50) shows that the MRB system has two inputs (i_1 and i_2) and two outputs (o_{F1} and o_{F2}). In addition, Eq. (8.51) shows that the output o_1 from RB is the same as the input i_{F1} to RB_{F1} and the output o_2 from RB is the same as the input i_{F2} to RB_{F2}, whereas the output o_{F1} from RB_{F1} is the same as the input i_2 to RB and the output o_{F2} from RB_{F2} is the same as the input i_1 to RB.

The MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.52).

$$RB_E = \text{RB} * (RB_{F1} + RB_{F2}) \tag{8.52}$$

The SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_1, i_2, o_{F1}, o_{F2} \end{array} \tag{8.53}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & \begin{array}{l} o_{F1} = i_2 \\ o_{F2} = i_1 \end{array} \end{array} \tag{8.54}$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby the linguistic values of the outputs o_{F1} and o_{F2} are fed back unchanged into the inputs i_2 and i_1, respectively, for each of the fuzzy rules. In other words, the rule base RB_E must have the same structure as the rule base RB_{2x2} from Example 8.6, as specified by Eqs. (8.11)–(8.12).

8.4 Transformation of Rule Bases with Global Feedback

Global feedback is a fairly basic type of complex FB that can be viewed as an extension of simple or local FB. While simple and local FB embrace only one rule base in a particular level and layer of the network structure for a MRB system, global FB embraces at least two rule bases residing in adjacent levels or layers. Three general cases with global FB are presented in Figs. 8.3–8.5.

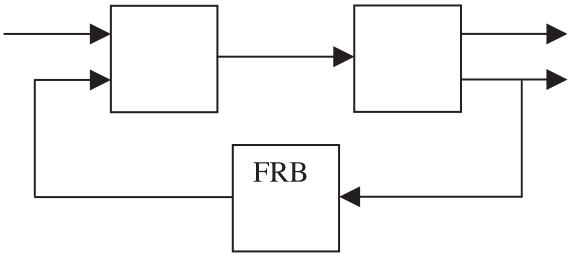

Fig. 8.3. Global feedback for rule bases in sequence

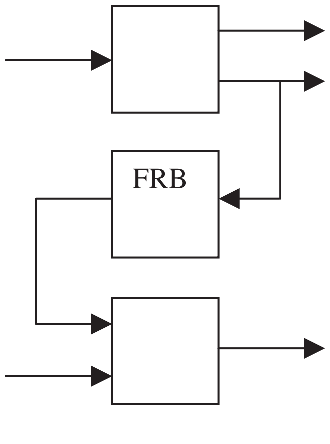

Fig. 8.4. Global downward feedback for rule bases in parallel

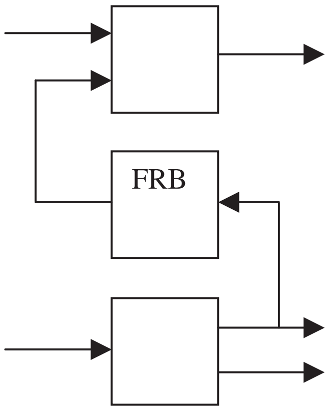

Fig. 8.5. Global upward feedback for rule bases in parallel

A fuzzy rule based system with global FB is constrained because the linguistic value of the corresponding output in each rule is mapped by a FB function onto a linguistic value of the associated input in the same rule. Depending on whether this FB function is an identity function or another type of function, these two linguistic values may be the same, as in the case of simple FB, or different, as in the case of local FB.

The notion of global FB is illustrated by six basic examples with MRB systems. In particular, Examples 8.13–8.14 show how global FB can be presented for two 1×1 rule bases standing in sequence in adjacent layers, whereas Examples 8.15–8.16 and Examples 8.17–8.18 describe downward and upward global FB, respectively, for two 1×1 rule bases standing in parallel in adjacent levels.

The procedure described in the examples below starts with a MRB system. In the case of identity FB function, the MRB system is first transformed into an equivalent SRB system by means of appropriate merging manipulations on the rule bases in the FF path. The rule base of the equivalent SRB system is then checked to ensure that the FB constraints are met, as described in Sect. 8.2. When the FB function is not an identity function, it is first replaced by a corresponding rule base in the FF path, as described in Sect. 8.3. Then, the resultant MRB system is transformed into an equivalent SRB system by means of appropriate merging manipulations. Finally, the rule base of the equivalent SRB system is checked to ensure that any FB constraints are met.

Example 8.13

A MRB system with two 1×1 rule bases in sequence and global simple feedback is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_1, i_1, o_1 & RB_2, i_2, o_2 \end{array} \tag{8.55}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1 = i_2 & o_2 = i_1 \end{array} \tag{8.56}$$

Equation (8.55) shows that both rule bases RB_1 and RB_2 have one input (i_1 and i_2, respectively) and one output (o_1 and o_2, respectively). In addition, Eq. (8.56) shows that the output o_1 from RB_1 is fed forward unchanged into the input i_2 to RB_2 whereas the output o_2 from RB_2 is fed back unchanged into the input i_1 to RB_1.

The MRB system can be transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.57).

$$RB_E = RB_1 * RB_2 \tag{8.57}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_1, o_2 \end{array} \tag{8.58}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_2 = i_1 \end{array} \tag{8.59}$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby the linguistic values of the output o_2 are fed back unchanged into the input i_1 for each of the fuzzy rules. In other words, the rule base RB_E must have the same structure as the rule base RB_{1x1} from Example 8.1, as specified by Eqs. (8.1)–(8.2).

Example 8.14

A MRB system with two 1×1 rule bases in sequence and global complex feedback is presented by the following matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_1,\ i_1, o_1 & RB_2,\ i_2, o_2 \end{array} \tag{8.60}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1 = i_2 & F(o_2) = i_1 \end{array} \tag{8.61}$$

Equation (8.60) shows that both rule bases RB_1 and RB_2 have one input (i_1 and i_2, respectively) and one output (o_1 and o_2, respectively). In addition, Eq. (8.61) shows that the output o_1 from RB_1 is fed forward unchanged into the input i_2 to RB_2 whereas the output o_2 from RB_2 is mapped by the FB function F onto the input i_1 to RB_1.

By introducing a third layer with a rule base RB_F that replaces the FB function F, the initial MRB system with complex FB is transformed into an equivalent MRB system with simple FB that embraces the three rule bases standing in sequence RB_1, RB_2 and RB_F.

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{cccc} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_1,\ i_1, o_1 & RB_2,\ i_2, o_2 & RB_F,\ i_F, o_F \end{array} \tag{8.62}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{cccc} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & o_1 = i_2 & o_2 = i_F & o_F = i_1 \end{array} \tag{8.63}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_F in the FF part of the system as well as the appearance of the two new interconnection variables i_F and o_F. In this context, Eq. (8.62) shows that the equivalent MRB system has one input (i_1) and one output (o_F). In addition, Eq. (8.63) shows that the output o_1 from RB is the same as the input i_2 to RB_2, the output o_2 from RB_2 is the same as the input i_F to RB_F, whereas the output o_F from RB_F is the same as the input i_1 to RB_1.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.64).

$$RB_E = RB_1 * RB_2 * RB_F \tag{8.64}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_1, o_F \end{array} \tag{8.65}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_F = i_1 \end{array} \tag{8.66}$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby the linguistic values of the output o_F are fed back unchanged into the input i_1 for each of the fuzzy rules. In other words, the rule base RB_E must have the same structure as the rule base RB_{1x1} from Example 8.1, as specified by Eqs. (8.1)–(8.2).

Example 8.15
A MRB system with two 1×1 rule bases in parallel and global simple downward feedback is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_1,\ i_1, o_1 \\ \text{level 2} & RB_2,\ i_2, o_2 \end{array} \tag{8.67}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_1 = i_2 \\ \text{level 2} & o_2 \end{array} \tag{8.68}$$

Equation (8.67) shows that both rule bases RB_1 and RB_2 have one input (i_1 and i_2, respectively) and one output (o_1 and o_2, respectively). In addition, Eq. (8.68) shows that the output o_1 from RB_1 is fed back unchanged into the input i_2 to RB_2.

By introducing layer 2 that replaces level 2 and moving RB_2 from its old location in level 2 to its new location in layer 2, the initial MRB system with global simple downward FB is transformed into an equivalent MRB system without FB and with the two rule bases RB_1 and RB_2 standing in sequence.

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_1,\ i_1, o_1 & RB_2,\ i_2, o_2 \end{array} \tag{8.69}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ccc} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1 = i_2 & o_2 \end{array} \tag{8.70}$$

Equation (8.69) shows that the equivalent MRB system has one input (i_1) and one output (o_2). In addition, Eq. (8.70) shows that the output o_1 from RB_1 is the same as the input i_2 to RB_2.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.71).

$$RB_E = RB_1 * RB_2 \tag{8.71}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{cc} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_1, o_2 \end{array} \tag{8.72}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{cc} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_2 \end{array} \tag{8.73}$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

Example 8.16

A MRB system with two 1×1 rule bases in parallel and global complex downward feedback is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_1,\ i_1, o_1 \\ \text{level 2} & RB_2,\ i_2, o_2 \end{array} \tag{8.74}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & F(o_1) = i_2 \\ \text{level 2} & o_2 \end{array} \tag{8.75}$$

Equation (8.74) shows that both rule bases RB_1 and RB_2 have one input (i_1 and i_2) and one output (o_1 and o_2). In addition, Eq. (8.75) shows that the output o_1 from RB_1 is mapped by the FB function F onto the input i_2 to RB_2.

By introducing layer 2 with a rule base RB_F that replaces the FB function F and layer 3 that replaces level 2 as well as moving RB_2 from its old location in level 2 to its new location in layer 3, the initial MRB system with global complex downward FB is transformed into an equivalent MRB system without FB and with the three rule bases RB_1, RB_F and RB_2 standing in sequence.

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_1,\ i_1, o_1 & RB_F,\ i_F, o_F & RB_2,\ i_2, o_2 \end{array} \tag{8.76}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & o_1 = i_F & o_F = i_2 & o_2 \end{array} \tag{8.77}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_F in the FF part of the system as well as the appearance of the two new interconnection variables i_F and o_F. In this context, Eq. (8.76) shows that the equivalent MRB system has one input (i_1)

and one output (o_2). In addition, Eq. (8.77) shows that the output o_1 from RB_1 is the same as the input i_F to RB_F whereas the output o_F from RB_F is the same as the input i_2 to RB_2.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.78).

$$RB_E = RB_1 * RB_F * RB_2 \tag{8.78}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_1, o_2 \end{array} \tag{8.79}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_2 \end{array} \tag{8.80}$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

Example 8.17

A MRB system with two 1×1 rule bases in parallel and global simple upward feedback is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_1,\ i_1, o_1 \\ \text{level 2} & RB_2,\ i_2, o_2 \end{array} \tag{8.81}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_1 \\ \text{level 2} & o_2 = i_1 \end{array} \tag{8.82}$$

Equation (8.81) shows that both rule bases RB_1 and RB_2 have one input (i_1 and i_2, respectively) and one output (o_1 and o_2, respectively). In addition, Eq. (8.82) shows that the output o_2 from RB_2 is fed back unchanged into the input i_1 to RB_1.

By introducing layer 2 and moving RB_1 from level 1of layer 1 to its new location in level 1 of layer 2 as well as moving RB_2 from level 2 of layer 1 to its new location in level 1 of layer 1 and removing level 2, the initial MRB system with global simple upward FB is transformed into an equivalent MRB system without FB and with the two rule bases RB_2 and RB_1 standing in sequence.

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_2,\ i_2, o_2 & RB_1,\ i_1, o_1 \end{array} \qquad (8.83)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_2 = i_1 & o_1 \end{array} \qquad (8.84)$$

Equation (8.83) shows that the equivalent MRB system has one input (i_2) and one output (o_1). In addition, Eq. (8.84) shows that the output o_2 from RB_2 is the same as the input i_1 to RB_1.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.85).

$$RB_E = RB_2 * RB_1 \qquad (8.85)$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_2, o_1 \end{array} \qquad (8.86)$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_1 \end{array} \qquad (8.87)$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

Example 8.18

A MRB system with two 1×1 rule bases in parallel and global complex upward feedback is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_1, i_1, o_1 \\ \text{level 2} & RB_2, i_2, o_2 \end{array} \tag{8.88}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_1 \\ \text{level 2} & F(o_2) = i_1 \end{array} \tag{8.89}$$

Equation (8.88) shows that both rule bases RB_1 and RB_2 have one input (i_1 and i_2, respectively) and one output (o_1 and o_2, respectively). In addition, Eq. (8.89) shows that the output o_2 from RB_2 is mapped by the FB function F onto the input i_1 to RB_1.

By introducing layer 3 and moving RB_1 from level 1of layer 1 to its new location in level 1 of layer 3, introducing layer 2 with a rule base RB_F that replaces the FB function F, as well as moving RB_2 from level 2 of layer 1 to its new location in level 1 of layer 1 and removing level 2, the initial MRB system with global complex upward FB is transformed into an equivalent MRB system without FB and with the three rule bases RB_2, RB_F and RB_1 standing in sequence.

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_2, i_2, o_2 & RB_F, i_F, o_F & RB_1, i_1, o_1 \end{array} \tag{8.90}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & o_2 = i_F & o_F = i_1 & o_1 \end{array} \tag{8.91}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_F in the FF part of the system as well as the appearance of the two new interconnection variables i_F and o_F. In this context, Eq. (8.90) shows that the equivalent MRB system has one input (i_2) and one output (o_1). In addition, Eq. (8.91) shows that the output o_2 from RB_2 is the same as the input i_F to RB_F whereas the output o_F from RB_F is the same as the input i_1 to RB_1.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (7.236).

$$RB_E = RB_2 * RB_F * RB_1 \tag{8.92}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{cc} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_2,\ o_1 \end{array} \tag{8.93}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{cc} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_1 \end{array} \tag{8.94}$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

8.5 Transformation of Rule Bases with Nested Feedback

Nested feedback is a type of complex FB, which is usually a combination of local and global FB. While local FB embraces only one rule base in a particular level and layer of the network structure for a MRB system and global FB embraces at least two rule bases residing in adjacent levels or layers, nested FB does both at the same time. In particular, nested FB has at least two FB loops such that either at least one of the FB outputs is from a rule base that is not a departure port for any of the other FB outputs or at least one of the FB inputs is to a rule base that is not an entry port for any of the other FB inputs. Three general cases with nested FB are presented in Figs. 8.6–8.8.

Therefore, if two or more FB loops do not satisfy the above condition, i.e. their FB outputs are from only one rule base and their FB inputs are to only one rule base, then the FB for these particular loops is not nested even though the loops may appear to be visually nested. In this case, the FB is either local or global depending on whether it embraces only one or at least two rule bases. This type of FB is a multi-output-multi-input extension of the

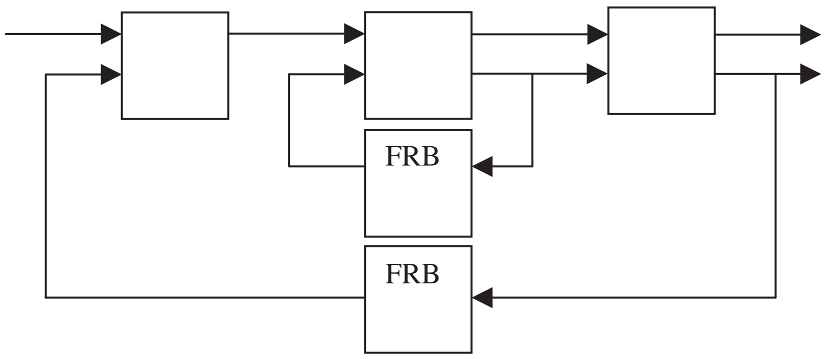

Fig. 8.6. Nested feedback for rule bases in sequence

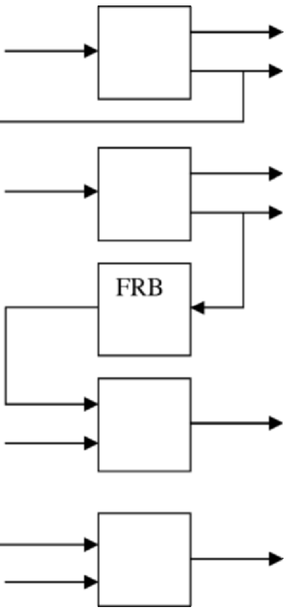

Fig. 8.7. Nested downward feedback for rule bases in parallel

single-output-single-input FB discussed in most examples from Sects. 8.3–8.4 and such an extension is usually straightforward.

A fuzzy rule based system with nested FB is constrained because the linguistic value of the corresponding outputs in each rule is mapped by a FB function onto a linguistic value of the associated input in the same rule. Depending on whether this FB function is an identity function or another type of function, these two linguistic values may be the same, as in the case of simple FB, or different, as in the case of local FB.

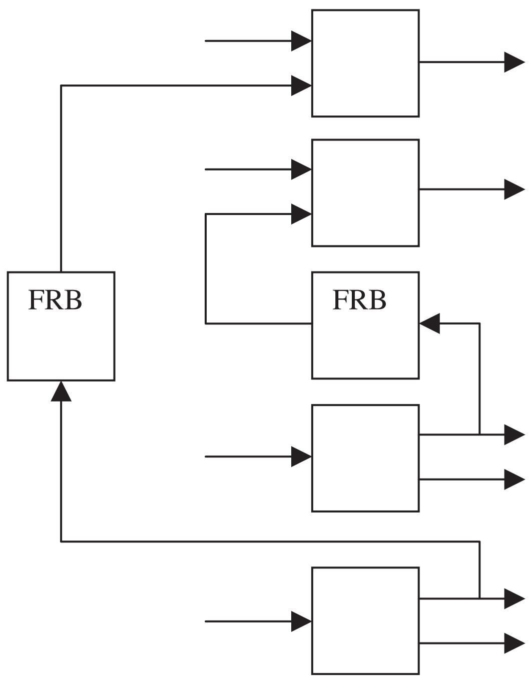

Fig. 8.8. Nested upward feedback for rule bases in parallel

The notion of nested FB is illustrated further by six basic examples with MRB systems. In particular, Examples 8.19–8.20 show how nested FB can be presented for two rule bases standing in sequence in adjacent layers, whereas Examples 8.21–8.22 and Examples 8.23–8.24 describe downward and upward nested FB, respectively, for three rule bases standing in parallel in adjacent levels. All examples consider non-identity type of FB because identity FB cases would be quite easy to deal with.

The procedure described in Examples 8.19–8.24 starts with a MRB system whose FB function is replaced by a corresponding rule base in the FF path, as described in Sect. 8.3. Then, the resultant MRB system is transformed into an equivalent SRB system by means of appropriate merging manipulations. Finally, the rule base of the equivalent SRB system is checked to ensure that any FB constraints are met.

Example 8.19

A MRB system with two rule bases in sequence and left-nested feedback is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_1,\ i_{11}, i_{12}, o_{11}, o_{12} & RB_2,\ i_2, o_2 \end{array} \quad (8.95)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & \begin{array}{l} o_{11} = i_2 \\ F1(o_{12}) = i_{12} \end{array} & F2(o_2) = i_{11} \end{array} \quad (8.96)$$

Equation (8.95) shows that RB_1 has two inputs and two outputs (i_{11}, i_{12}, o_{11}, o_{12}) whereas RB_2 has one input and one output (i_2, o_2). In addition, Eq. (8.96) shows that the output o_{11} from RB_1 is fed forward unchanged into the input i_2 to RB_2 whereas the output o_{12} from RB_1 is mapped by the FB function $F1$ onto the input i_{12} to RB_1 and the output o_2 from RB_2 is mapped by the FB function $F2$ onto the input i_{11} to RB_1.

By introducing level 1 in a new layer 3 with a rule base RB_{F2} that replaces the FB function $F2$, introducing level 2 in layer 2 with a rule base RB_{F1} that replaces the FB function $F1$, as well as introducing level 2 in layer 3 with an IRB RB_I representing a self standing output, the initial MRB system with complex FB is transformed into an equivalent MRB system with simple FB. This FB consists of two nested loops whereby the inner loop embraces the sequence of rule bases RB_1, RB_{F1}, RB_I and the outer loop embraces the sequence of rule bases RB_1, RB_2, RB_{F2}.

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_1,\ i_{11}, i_{12}, o_{11}, o_{12} & RB_2,\ i_2, o_2 & RB_{F2},\ i_{F2}, o_{F2} \\ \text{level 2} & & RB_{F1},\ i_{F1}, o_{F1} & RB_I,\ i_I, o_I \end{array} \quad (8.97)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & \begin{array}{l} o_{11} = i_2 \\ o_{12} = i_{F1} \end{array} & o_2 = i_{F2} & o_{F2} = i_{11} \\ \text{level 2} & & o_{F1} = i_I & o_I = i_{12} \end{array} \quad (8.98)$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F2}, RB_{F1} and RB_I in the FF part of the system as well as the appearance of the new interconnection variables i_{F2}, o_{F2}, i_{F1}, o_{F1}, i_I, o_I. In this context, Eq. (8.97) shows that the equivalent MRB system has two inputs (i_{11}, i_{12}) and two outputs (o_{F2}, o_I). In addition, Eq. (8.98) shows that the output o_{11} from RB_1 is the same as the input i_2 to RB_2, the output o_{12} from RB_1 is the same as the input i_{F1} to RB_{F1}, the output o_2 from RB_2 is the same as the input i_{F2} to RB_{F2}, the output o_{F1} from RB_{F1} is the same as the input i_I to RB_I, whereas the output o_{F2} from RB_{F2} is the same as the input i_{11} to RB_1 and the output o_I from RB_I is the same as the input i_{12} to RB_1.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.99).

$$RB_E = RB_1 * (RB_2 + RB_{F1}) * (RB_{F2} + RB_I) \tag{8.99}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_{11}, i_{12}, o_{F2}, o_I \end{array} \tag{8.100}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_{F2} = i_{11} \\ & o_I = i_{12} \end{array} \tag{8.101}$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby for each of the fuzzy rules the linguistic values of the output o_{F2} are fed back unchanged into the input i_{11} and the linguistic values of the output o_I are fed back unchanged into the input i_{12}. In this case, the rule base RB_E must have a structure in accordance with Eqs. (8.102)–(8.103).

RB_E: (8.102)

$i_{11}\, i_{12} / o_{F2}\, o_I$	11	12	13	21	22	23	31	32	33
11	?	0	0	0	0	0	0	0	0
12	0	?	0	0	0	0	0	0	0
13	0	0	?	0	0	0	0	0	0
21	0	0	0	?	0	0	0	0	0
22	0	0	0	0	?	0	0	0	0
23	0	0	0	0	0	?	0	0	0
31	0	0	0	0	0	0	?	0	0
32	0	0	0	0	0	0	0	?	0
33	0	0	0	0	0	0	0	0	?

$$\begin{aligned} RB_E\text{: } \{&(11, 11)_?, (12, 12)_?, (13, 13)_?, \\ &(21, 21)_?, (22, 22)_?, (23, 23)_?, \\ &(31, 31)_?, (32, 32)_?, (33, 33)_?\} \end{aligned} \tag{8.103}$$

Example 8.20

A MRB system with two rule bases in sequence and right-nested feedback is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_1,\ i_1, o_1 & RB_2,\ i_{21}, i_{22}, o_{21}, o_{22} \end{array} \tag{8.104}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1 = i_{21} & F2(o_{21}) = i_1 \\ & & F1(o_{22}) = i_{22} \end{array} \tag{8.105}$$

Equation (8.104) shows that RB_1 has one input and one output (i_1, o_1) whereas RB_2 has two inputs and two outputs (i_{21}, i_{22}, o_{21}, o_{22}). In addition, Eq. (8.105) shows that the output o_1 from RB_1 is fed forward unchanged into the input i_{21} to RB_2 whereas the output o_{21} from RB_2 is mapped by the FB function $F2$ onto the input i_1 to RB_1 and the output o_{22} from RB_2 is mapped by the FB function $F1$ onto the input i_{22} to RB_2.

By introducing level 1 in a new layer 3 with a rule base RB_{F2} that replaces the FB function $F2$, introducing level 2 in layer 1 with an IRB RB_I representing a self standing input, as well as introducing level 2 in layer 3 with a rule base RB_{F1} that replaces the FB function $F1$, the initial MRB system with complex FB is transformed into an equivalent MRB system with simple FB. This FB consists of two nested loops whereby the inner loop embraces the sequence of rule bases RB_1, RB_2, RB_{F1} and the outer loop embraces the sequence of rule bases RB_1, RB_2, RB_{F2}.

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_1,\ i_1,\ o_1 & RB_2,\ i_{21},\ i_{22},\ o_{21},\ o_{22} & RB_{F2},\ i_{F2},\ o_{F2} \\ \text{level 2} & RB_I,\ i_I,\ o_I & & RB_{F1},\ i_{F1},\ o_{F1} \end{array} \tag{8.106}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & o_1 = i_{21} & \begin{array}{l} o_{21} = i_{F2} \\ o_{22} = i_{F1} \end{array} & o_{F2} = i_1 \\ \text{level 2} & o_I = i_{22} & & o_{F1} = i_I \end{array} \tag{8.107}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F2}, RB_I and RB_{F1} in the FF part of the system as well as the appearance of the new interconnection variables i_{F2}, o_{F2}, i_I, o_I, i_{F1}, o_{F1}. In this context, Eq. (8.106) shows that the equivalent MRB system has two inputs (i_1, i_I) and two outputs (o_{F2}, o_{F1}). In addition, Eq. (8.107) shows that the output o_1 from RB_1 is the same as the input i_{21} to RB_2, the output o_I from RB_I is the same as the input i_{22} to RB_2, the output o_{21} from RB_2 is the same as the input i_{F2} to RB_{F2}, the output o_{22} from RB_2 is the same as the input i_{F1} to RB_{F1}, whereas the output o_{F2} from RB_{F2} is the same as the input i_1 to RB_1 and the output o_{F1} from RB_{F1} is the same as the input i_I to RB_I.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.108).

$$RB_E = (RB_1 + RB_I) * RB_2 * (RB_{F2} + RB_{F1}) \tag{8.108}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_1,\ i_I,\ o_{F2},\ o_{F1} \end{array} \tag{8.109}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_{F2} = i_1 \\ & o_{F1} = i_1 \end{array} \qquad (8.110)$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby for each of the fuzzy rules the linguistic values of the output o_{F2} are fed back unchanged into the input i_1 and the linguistic values of the output o_{F1} are fed back unchanged into the input i_1. In this case, the rule base RB_E must have the same structure as the rule base RB_E from Example 8.19, as specified by Eqs. (8.102)–(8.103).

Example 8.21

A MRB system with three rule bases in parallel and top-nested downward feedback is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_1,\ i_1, o_{11}, o_{12} \\ \text{level 2} & RB_2,\ i_2, o_2 \\ \text{level 3} & RB_3,\ i_3, o_3 \end{array} \qquad (8.111)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & F2(o_{11}) = i_3 \\ & F1(o_{12}) = i_2 \\ \text{level 2} & o_2 \\ \text{level 3} & o_3 \end{array} \qquad (8.112)$$

Equation (8.111) shows that RB_1 has one input and two outputs (i_1, o_{11}, o_{12}) whereas both RB_2 and RB_3 have one input and one output (i_2, o_2 and i_3, o_3, respectively). In addition, Eq. (8.112) shows that the output o_{11} from RB_1 is mapped by the FB function $F2$ onto the input i_3 to RB_3 whereas the output o_{12} from RB_1 is mapped by the FB function $F1$ onto the input i_2 to RB_2.

By introducing level 1 in a new layer 2 with a rule base RB_{F2} that replaces the FB function $F2$, introducing level 2 in layer 2 with a rule base RB_{F1} that replaces the FB function $F1$, moving RB_2 from level 2 of layer 1 to level 2 of a new layer 3 and removing level 2 from layer 1, as well as moving RB_3 from level 3 of layer 1 to level 1 of layer 3 and removing level 3 from layer 1, the initial MRB system with complex FB is transformed

into an equivalent MRB system without FB. This system consists of two groups of rule bases standing sequence, i.e. (RB_1,RB_{F2},RB_3) and (RB_1, RB_{F1},RB_2).
The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & RB_1,\ i_1, o_{11}, o_{12} & RB_{F2},\ i_{F2}, o_{F2} & RB_3,\ i_3, o_3 \\
\text{level 2} & & RB_{F1},\ i_{F1}, o_{F1} & RB_2,\ i_2, o_2
\end{array} \qquad (8.113)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & \begin{array}{l} o_{11} = i_{F2} \\ o_{12} = i_{F1} \end{array} & o_{F2} = i_3 & o_3 \\
\text{level 2} & & o_{F1} = i_2 & o_2
\end{array} \qquad (8.114)$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F2} and RB_{F1} in the FF part of the system as well as the appearance of the new interconnection variables i_{F2}, o_{F2}, i_{F1}, o_{F1}. In this context, Eq. (8.113) shows that the equivalent MRB system has one input (i_1) and two outputs (o_3, o_2). In addition, Eq. (8.114) shows that the output o_{11} from RB_1 is the same as the input i_{F2} to RB_{F2}, the output o_{12} from RB_1 is the same as the input i_{F1} to RB_{F1}, the output o_{F2} from RB_{F2} is the same as the input i_3 to RB_3, and the output o_{F1} from RB_{F1} is the same as the input i_2 to RB_2.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.115).

$$RB_E = RB_1 * (RB_{F2} + RB_{F1}) * (RB_3 + RB_2) \qquad (8.115)$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll}
\text{level/layer} & \text{layer 1} \\
\text{level 1} & RB_E,\ i_1, o_3, o_2
\end{array} \qquad (8.116)$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_3 \\ & o_2 \end{array} \tag{8.117}$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

Example 8.22
A MRB system with three rule bases in parallel and bottom-nested downward feedback is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_1,\ i_1, o_1 \\ \text{level 2} & RB_2,\ i_2, o_2 \\ \text{level 3} & RB_3,\ i_{31}, i_{32}, o_3 \end{array} \tag{8.118}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & F2(o_1) = i_{31} \\ \text{level 2} & F1(o_2) = i_{32} \\ \text{level 3} & o_3 \end{array} \tag{8.119}$$

Equation (8.118) shows that both RB_1 and RB_2 have one input and one output (i_1, o_1 and i_2, o_2, respectively) whereas RB_3 has two inputs and one output (i_{31}, i_{32}, o_3). In addition, Eq. (8.119) shows that the output o_1 from RB_1 is mapped by the FB function $F2$ onto the input i_{31} to RB_3 whereas the output o_2 from RB_2 is mapped by the FB function $F1$ onto the input i_{32} to RB_3.

By introducing level 1 in a new layer 2 with a rule base RB_{F2} that replaces the FB function $F2$, introducing level 2 in layer 2 with a rule base RB_{F1} that replaces the FB function $F1$, as well as moving RB_3 from level 3 of layer 1 to level 1 of a new layer 3 and removing level 3 from layer 1, the initial MRB system with complex FB is transformed into an equivalent MRB system without FB. This system consists of two groups of rule bases standing in sequence, i.e. (RB_1, RB_{F2}, RB_3) and (RB_2, RB_{F1}, RB_3).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_1, i_1, o_1 & RB_{F2}, i_{F2}, o_{F2} & RB_3, i_{31}, i_{32}, o_3 \\ \text{level 2} & RB_2, i_2, o_2 & RB_{F1}, i_{F1}, o_{F1} & \end{array} \tag{8.120}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & o_1 = i_{F2} & o_{F2} = i_{31} & o_3 \\ \text{level 2} & o_2 = i_{F1} & o_{F1} = i_{32} & \end{array} \tag{8.121}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F2} and RB_{F1} in the FF part of the system as well as the appearance of the new interconnection variables i_{F2}, o_{F2}, i_{F1}, o_{F1}. In this context, Eq. (8.120) shows that the equivalent MRB system has two inputs (i_1, i_2) and one output (o_3). In addition, Eq. (8.121) shows that the output o_1 from RB_1 is the same as the input i_{F2} to RB_{F2}, the output o_2 from RB_2 is the same as the input i_{F1} to RB_{F1}, the output o_{F2} from RB_{F2} is the same as the input i_{31} to RB_3, and the output o_{F1} from RB_{F1} is the same as the input i_{32} to RB_3.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.122).

$$RB_E = (RB_1 + RB_2) * (RB_{F2} + RB_{F1}) * RB_3 \tag{8.122}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_1, i_2, o_3 \end{array} \tag{8.123}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_3 \end{array} \tag{8.124}$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

Example 8.23

A MRB system with three rule bases in parallel and top-nested upward feedback is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_1,\ i_{11}, i_{12}, o_1 \\ \text{level 2} & RB_2,\ i_2, o_2 \\ \text{level 3} & RB_3,\ i_3, o_3 \end{array} \tag{8.125}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_1 \\ \text{level 2} & F1(o_2) = i_{11} \\ \text{level 3} & F2(o_3) = i_{12} \end{array} \tag{8.126}$$

Equation (8.125) shows that RB_1 has two inputs and one output (i_{11}, i_{12}, o_1) whereas both RB_2 and RB_3 have one input and one output (i_2, o_2 and i_3, o_3, respectively). In addition, Eq. (8.126) shows that the output o_2 from RB_2 is mapped by the FB function $F1$ onto the input i_{11} to RB_1 whereas the output o_3 from RB_3 is mapped by the FB function $F2$ onto the input i_{12} to RB_1.

By moving RB_1 from level 1 of layer 1 to level 2 of a new layer 3, moving RB_2 from level 2 of layer 1 to level 1 of layer 1, introducing level 2 in a new layer 2 with a rule base RB_{F2} that replaces the FB function $F2$, introducing level 1 in layer 2 with a rule base RB_{F1} that replaces the FB function $F1$, as well as moving RB_3 from level 3 of layer 1 to level 2 of layer 1 and removing level 3 from layer 1, the initial MRB system with complex FB is transformed into an equivalent MRB system without FB. This system consists of two groups of rule bases standing in sequence, i.e. (RB_2, RB_{F1}, RB_1) and (RB_3, RB_{F2}, RB_1).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_2,\ i_2, o_2 & RB_{F1},\ i_{F1}, o_{F1} & \\ \text{level 2} & RB_3,\ i_3, o_3 & RB_{F2},\ i_{F2}, o_{F2} & RB_1,\ i_{11}, i_{12}, o_1 \end{array} \tag{8.127}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & o_2 = i_{F1} & o_{F1} = i_{11} & \\ \text{level 2} & o_3 = i_{F2} & o_{F2} = i_{12} & o_1 \end{array} \tag{8.128}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F1} and RB_{F2} in the FF part of the system as well as the appearance of the new interconnection variables i_{F1}, o_{F1}, i_{F2}, o_{F2}. In this context, Eq. (8.127) shows that the equivalent MRB system has two inputs (i_2, i_3) and one output (o_1). In addition, Eq. (8.128) shows that the output o_2 from RB_2 is the same as the input i_{F1} to RB_{F2}, the output o_3 from RB_3 is the same as the input i_{F2} to RB_{F2}, the output o_{F1} from RB_{F1} is the same as the input i_{11} to RB_1 and the output o_{F2} from RB_{F2} is the same as the input i_{12} to RB_1.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.129).

$$RB_E = (RB_2 + RB_3) * (RB_{F1} + RB_{F2}) * RB_1 \tag{8.129}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_2, i_3, o_1 \end{array} \tag{8.130}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_1 \end{array} \tag{8.131}$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

Example 8.24

A MRB system with three rule bases in parallel and bottom-nested upward feedback is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_1,\ i_1,\ o_1 \\ \text{level 2} & RB_2,\ i_2,\ o_2 \\ \text{level 3} & RB_3,\ i_3,\ o_{31},\ o_{32} \end{array} \tag{8.132}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_1 \\ \text{level 2} & o_2 \\ \text{level 3} & F1(o_{31}) = i_2 \\ & F2(o_{32}) = i_1 \end{array} \tag{8.133}$$

Equation (8.132) shows that both RB_1 and RB_2 have one input and one output (i_1, o_1 and i_2, o_2, respectively) whereas RB_3 has one input and two outputs (i_3, o_{31}, o_{32}). In addition, Eq. (8.133) shows that the output o_{31} from RB_3 is mapped by the FB function $F1$ onto the input i_2 to RB_2 whereas the output o_{32} from RB_3 is mapped by the FB function $F2$ onto the input i_1 to RB_1.

By moving RB_1 from level 1 of layer 1 to level 2 of a new layer 3, moving RB_2 from level 2 of layer 1 to level 1 of layer 3, introducing level 1 in a new layer 2 with a rule base RB_{F1} that replaces the FB function $F1$, introducing level 2 in layer 2 with a rule base RB_{F2} that replaces the FB function $F2$, as well as moving RB_3 from level 3 of layer 1 to level 2 of layer 1 and removing level 3 from layer 1, the initial MRB system with complex FB is transformed into an equivalent MRB system without FB. This system consists of two groups of rule bases standing in sequence, i.e. (RB_3, RB_{F1}, RB_2) and (RB_3, RB_{F2}, RB_1).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & & RB_{F1},\ i_{F1},\ o_{F1} & RB_2,\ i_2,\ o_2 \\ \text{level 2} & RB_3,\ i_3,\ o_{31},\ o_{32} & RB_{F2},\ i_{F2},\ o_{F2} & RB_1,\ i_1,\ o_1 \end{array} \tag{8.134}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & & o_{F1} = i_2 & o_2 \\ \text{level 2} & o_{31} = i_{F1} & o_{F2} = i_1 & o_1 \\ & o_{32} = i_{F2} & & \end{array} \tag{8.135}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F1} and RB_{F2} in the FF part of the system as well as the appearance of the new interconnection variables i_{F1}, o_{F1}, i_{F2}, o_{F2}. In this context, Eq. (8.134) shows that the equivalent MRB system has one input (i_3) and two outputs (o_2, o_1). In addition, Eq. (8.135) shows that the output o_{31} from RB_3 is the same as the input i_{F1} to RB_{F1}, the output o_{32} from RB_3 is the same as the input i_{F2} to RB_{F2}, the output o_{F1} from RB_{F1} is the same as the input i_2 to RB_2 and the output o_{F2} from RB_{F2} is the same as the input i_1 to RB_1.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.136).

$$RB_E = RB_3 * (RB_{F1} + RB_{F2}) * (RB_2 + RB_1) \tag{8.136}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_3, o_2, o_1 \end{array} \tag{8.137}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_2 \\ & o_1 \end{array} \tag{8.138}$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

8.6 Transformation of Rule Bases with Overlapping Feedback

Overlapping feedback is a type of complex FB, which is usually a combination of global FB with itself that includes at least two non-nested FB loops with partial overlap. In this case, each of the FB outputs is from a rule base that is not a departure port for any of the other FB outputs and each of the FB inputs is to a rule base that is not an entry port for any of the other FB inputs. Therefore, if two or more FB loops do not satisfy the above condition, i.e. their FB outputs are from only one rule base or their FB inputs are to only one rule base, then the FB for these particular loops is nested and not overlapping. Three general cases with overlapping FB are presented in Figs. 8.9–8.11.

A fuzzy rule based system with overlapping FB is constrained because the linguistic value of the corresponding outputs in each rule is mapped by a FB function onto a linguistic value of the associated input in the same rule. Depending on whether this FB function is an identity function or another type of function, these two linguistic values may be the same, as in the case of global simple FB, or different, as in the case of global complex FB.

The notion of overlapping FB is illustrated by three basic examples with MRB systems. In particular, Examples 8.25 shows how overlapping FB can be presented for three rule bases standing in sequence in adjacent layers, whereas Examples 8.26-8.27 describe downward and upward overlapping FB, respectively, for four rule bases standing in parallel in adjacent levels. All examples consider non-identity type of FB because identity FB cases would be quite easy to deal with.

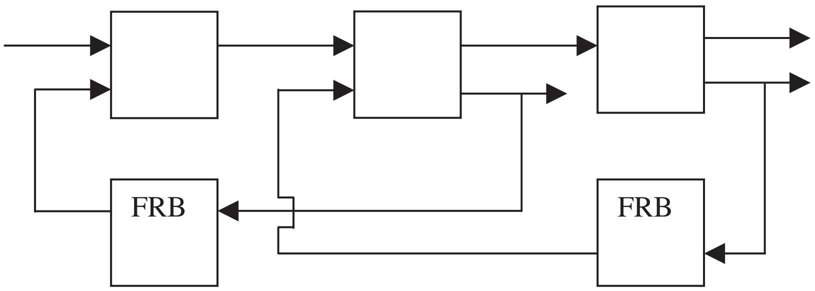

Fig. 8.9. Overlapping feedback for rule bases in sequence

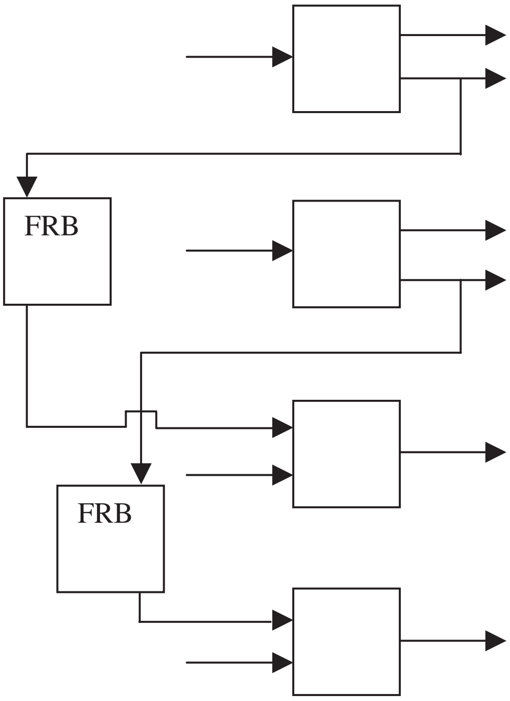

Fig. 8.10. Overlapping downward feedback for rule bases in parallel

The procedure described in Examples 8.25-8.27 starts with a MRB system whose FB function is replaced by a corresponding rule base in the FF path, as described in Sect. 8.3. Then, the resultant MRB system is transformed into an equivalent SRB system by means of appropriate merging manipulations. Finally, the rule base of the equivalent SRB system is checked to ensure that any FB constraints are met.

Example 8.25
A MRB system with three rule bases in sequence and overlapping feedback is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_1,\ i_1,\ o_1 & RB_2,\ i_{21},\ i_{22},\ o_{21},\ o_{22} & RB_3,\ i_3,\ o_3 \end{array} \qquad (8.139)$$

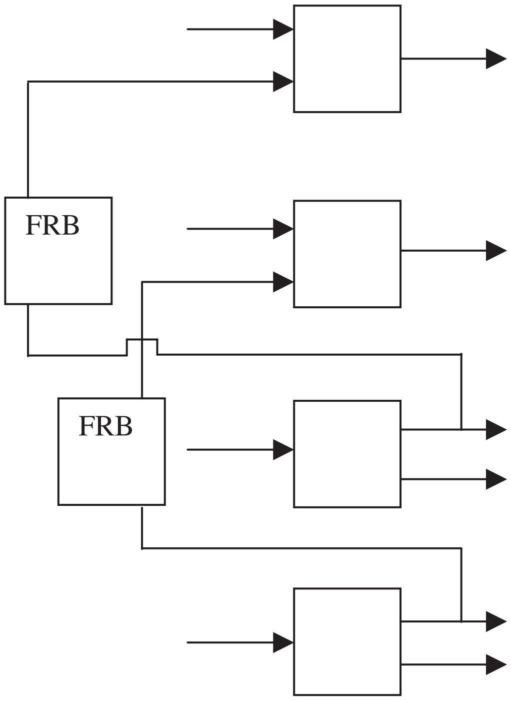

Fig. 8.11. Overlapping upward feedback for rule bases in parallel

The output-input interconnections for this MRB system are given by the matrix:

$$
\begin{array}{llll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & o_1 = i_{21} & o_{21} = i_3 & F2(o_3) = i_{22} \\
 & & & F1(o_{22}) = i_1
\end{array}
\tag{8.140}
$$

Equation (8.139) shows that both RB_1 and RB_3 have one input and one output (i_1, o_1 and i_3, o_3, respectively), whereas RB_2 has two inputs and two outputs (i_{21}, i_{22}, o_{21}, o_{22}). In addition, Eq. (8.140) shows that the output o_1 from RB_1 is fed forward unchanged into the input i_{21} to RB_2, the output o_{21} from RB_2 is fed forward unchanged into the input i_3 to RB_3, whereas the output o_{22} from RB_2 is mapped by the FB function $F1$ onto the input i_1 to RB_1 and the output o_3 from RB_3 is mapped by the FB function $F2$ onto the input i_{22} to RB_2.

By introducing level 1 in a new layer 4 with a rule base RB_{F2} that replaces the FB function $F2$, introducing level 2 in layer 3 with a rule base RB_{F1} that replaces the FB function $F1$, introducing level 2 in layer 1 with an IRB RB_{I1} representing a self standing input, as well as introducing level 2 in layer 4 with an IRB RB_{I2} representing a self standing output, the initial MRB system with complex FB is transformed into an equivalent MRB system with simple FB. This FB consists of two global simple loops embracing the two groups of rule bases standing in a sequence (RB_1, RB_2, RB_3, RB_{F2}) and (RB_{I1}, RB_2,RB_{F1}, RB_{I2}).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{lllll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} & \text{layer 4} \\
\text{level 1} & RB_1, i_1, o_1 & RB_2, i_{21}, i_{22}, o_{21}, o_{22} & RB_3, i_3, o_3 & RB_{F2}, i_{F2}, o_{F2} \\
\text{level 2} & RB_{I1}, i_{I1}, o_{I1} & & RB_{F1}, i_{F1}, o_{F1} & RB_{I2}, i_{I2}, o_{I2}
\end{array} \tag{8.141}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lllll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} & \text{layer 4} \\
\text{level 1} & o_1 = i_{21} & o_{21} = i_3 & o_3 = i_{F2} & o_{F2} = i_{I1} \\
 & & o_{22} = i_{F1} & & \\
\text{level 2} & o_{I1} = i_{22} & & o_{F1} = i_{I2} & o_{I2} = i_1
\end{array} \tag{8.142}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F2}, RB_{I2}, RB_{F1} and RB_{I2} in the FF part of the system as well as the appearance of the new interconnection variables i_{F2}, o_{F2}, i_{I1}, o_{I1}, i_{F1}, o_{F1}, i_{I2}, o_{I2}. In this context, Eq. (8.141) shows that the equivalent MRB system has two inputs (i_1, i_{I1}) and two outputs (o_{F2}, o_{I2}). In addition, Eq. (8.142) shows that the output o_1 from RB_1 is the same as the input i_{21} to RB_2, the output o_{I1} from RB_{I1} is the same as the input i_{22} to RB_2, the output o_{21} from RB_2 is the same as the input i_3 to RB_3, the output o_{22} from RB_2 is the same as the input i_{F1} to RB_{IF1}, the output o_3 from RB_3 is the same as the input i_{F2} to RB_{F2}, the output o_{F1} from RB_{F1} is the same as the input i_{I2} to RB_{I2}, whereas the output o_{F2} from RB_{F2} is the same as the input i_{I1} to RB_I and the output o_{I2} from RB_{I2} is the same as the input i_1 to RB_1.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.143).

$$RB_E = (RB_1 + RB_{I1}) * RB_2 * (RB_3 + RB_{F1}) * (RB_{F2} + RB_{I2}) \tag{8.143}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_1, i_{I1}, o_{F2}, o_{I2} \end{array} \quad (8.144)$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & \begin{array}{l} o_{F2} = i_{I1} \\ o_{I2} = i_1 \end{array} \end{array} \quad (8.145)$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby for each of the fuzzy rules the linguistic values of the output o_{F2} are fed back unchanged into the input i_{I1} and the linguistic values of the output o_{I2} are fed back unchanged into the input i_1. In other words, the rule base RB_E must have the same structure as the rule base RB_{2x2} from Example 8.6, as specified by Eqs. (8.11)–(8.12).

Example 8.26
A MRB system with four rule bases in parallel and overlapping downward feedback is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_1, i_1, o_1 \\ \text{level 2} & RB_2, i_2, o_2 \\ \text{level 3} & RB_3, i_3, o_3 \\ \text{level 4} & RB_4, i_4, o_4 \end{array} \quad (8.146)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & F1(o_1) = i_3 \\ \text{level 2} & F2(o_2) = i_4 \\ \text{level 3} & o_3 \\ \text{level 4} & o_4 \end{array} \quad (8.147)$$

Equation (8.146) shows that RB_i, $i = 1,4$ have one input and one output each (i_i, o_i, $i = 1,4$). In addition, Eq. (8.147) shows that the output o_1 from RB_1 is mapped by the FB function *F1* onto the input i_3 to RB_3 and the output o_2 from RB_2 is mapped by the FB function *F2* onto the input i_4 to RB_4.

By introducing level 1 in a new layer 2 with a rule base RB_{F1} that replaces the FB function *F1*, introducing level 2 in layer 2 with a rule base RB_{F2} that replaces the FB function *F2*, moving RB_3 from level 3 of layer 1 to level 1 of a new layer 3 and removing level 3 from layer 1, as well as moving RB_4 from level 4 of layer 1 to level 2 of layer 3 and removing level 4 from layer 1, the initial MRB system with complex FB is transformed into an equivalent MRB system without FB. This system consists of two groups of rule bases standing in sequence, i.e. (RB_1, RB_{F1}, RB_3) and (RB_2, RB_{F2}, RB_4).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & RB_1, i_1, o_1 & RB_{F1}, i_{F1}, o_{F1} & RB_3, i_3, o_3 \\
\text{level 2} & RB_3, i_3, o_3 & RB_{F2}, i_{F2}, o_{F2} & RB_4, i_4, o_4
\end{array} \qquad (8.148)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & o_1 = i_{F1} & o_{F1} = i_3 & o_3 \\
\text{level 2} & o_2 = i_{F2} & o_{F2} = i_4 & o_4
\end{array} \qquad (8.149)$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F1} and RB_{F2} in the FF part of the system as well as the appearance of the new interconnection variables i_{F1}, o_{F1}, i_{F2}, o_{F2}. In this context, Eq. (8.148) shows that the equivalent MRB system has two inputs (i_1, i_2) and two outputs (o_3, o_4). In addition, Eq. (8.149) shows that the output o_1 from RB_1 is the same as the input i_{F1} to RB_{F1}, the output o_2 from RB_2 is the same as the input i_{F2} to RB_{F2}, the output o_{F1} from RB_{F1} is the same as the input i_3 to RB_3 and the output o_{F2} from RB_{F2} is the same as the input i_4 to RB_4.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.150).

$$RB_E = (RB_1 + RB_2) * (RB_{F1} + RB_{F2}) * (RB_3 + RB_4) \qquad (8.150)$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_1, i_2, o_3, o_4 \end{array} \tag{8.151}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_3 \\ & o_4 \end{array} \tag{8.152}$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

Example 8.27
A MRB system with four rule bases in parallel and overlapping upward feedback is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_1, i_1, o_1 \\ \text{level 2} & RB_2, i_2, o_2 \\ \text{level 3} & RB_3, i_3, o_3 \\ \text{level 4} & RB_4, i_4, o_4 \end{array} \tag{8.153}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_1 \\ \text{level 2} & o_2 \\ \text{level 3} & F2(o_3) = i_1 \\ \text{level 4} & F1(o_4) = i_2 \end{array} \tag{8.154}$$

Equation (8.153) shows that RB_i, $I = 1,4$ have one input and one output each (i_i, o_i, $I = 1,4$). In addition, Eq. (8.154) shows that the output o_3 from RB_3 is mapped by the FB function $F2$ onto the input i_1 to RB_1 and the output o_4 from RB_4 is mapped by the FB function $F1$ onto the input i_2 to RB_2.

By moving RB_1 from level 1 of layer 1 to level 1 of a new layer 3, moving RB_2 from level 2 of layer 1 to level 2 of layer 3, introducing level 1 in a new layer 2 with a rule base RB_{F2} that replaces the FB function $F2$, introducing level 2 in layer 2 with a rule base RB_{F1} that replaces the FB function $F1$, moving RB_3 from level 3 of layer 1 to level 1 of layer 1 and removing level 3 from layer 1, as well as moving RB_4 from level 4 of layer 1 to level 2 of layer 1 and removing level 4 from layer 1, the initial MRB system with complex FB is transformed into an equivalent MRB system without FB. This system consists of two groups of rule bases standing in sequence, i.e. (RB_3, RB_{F2}, RB_1) and (RB_4, RB_{F1}, RB_2).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & RB_3, i_3, o_3 & RB_{F2}, i_{F2}, o_{F2} & RB_1, i_1, o_1 \\
\text{level 2} & RB_4, i_4, o_4 & RB_{F1}, i_{F1}, o_{F1} & RB_2, i_2, o_2
\end{array} \qquad (8.155)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & o_3 = i_{F2} & o_{F2} = i_1 & o_1 \\
\text{level 2} & o_4 = i_{F1} & o_{F1} = i_2 & o_2
\end{array} \qquad (8.156)$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F2} and RB_{F1} in the FF part of the system as well as the appearance of the new interconnection variables i_{F2}, o_{F2}, i_{F1}, o_{F1}. In this context, Eq. (8.155) shows that the equivalent MRB system has two inputs (i_3, i_4) and two outputs (o_1, o_2). In addition, Eq. (8.156) shows that the output o_3 from RB_3 is the same as the input i_{F2} to RB_{F2}, the output o_4 from RB_4 is the same as the input i_{F1} to RB_{F1}, the output o_{F2} from RB_{F2} is the same as the input i_1 to RB_1 and the output o_{F1} from RB_{F1} is the same as the input i_2 to RB_2.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.157).

$$RB_E = (RB_3 + RB_4) * (RB_{F2} + RB_{F1}) * (RB_1 + RB_2) \tag{8.157}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_3, i_4, o_1, o_2 \end{array} \tag{8.158}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_1 \\ & o_2 \end{array} \tag{8.159}$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

8.7 Transformation of Rule Bases with Crossed Feedback

Crossed feedback is a type of complex FB, which is an extension of overlapping FB. As opposed to overlapping FB which includes at least two overlapping loops embracing a number of layers across a single level or a number of levels across a single layer, crossed FB includes at least two crossed loops embracing at least two layers and two levels from the network structure of a MRB system. In this case, each of the FB outputs is from a rule base that is not in the same level or layer as the departure rule base for any of the other FB outputs and each of the FB inputs is to a rule base that is not in the same level or layer as the arrival rule base for any of the other FB inputs. Therefore, if two or more FB loops do not satisfy the above condition, i.e. their FB outputs are from rule bases in the same level or layer or their FB inputs are to rule base in the same level or layer, then the FB for these particular loops is overlapping and not crossed. Three general cases with crossed FB are presented in Figs. 8.12–8.14.

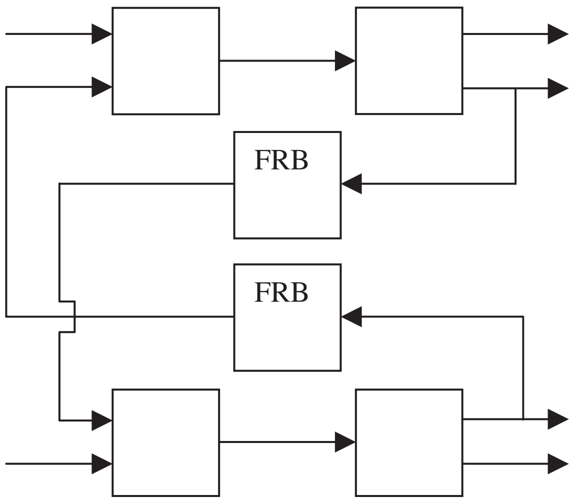

Fig. 8.12. Crossed symmetric feedback for rule bases in parallel and sequence

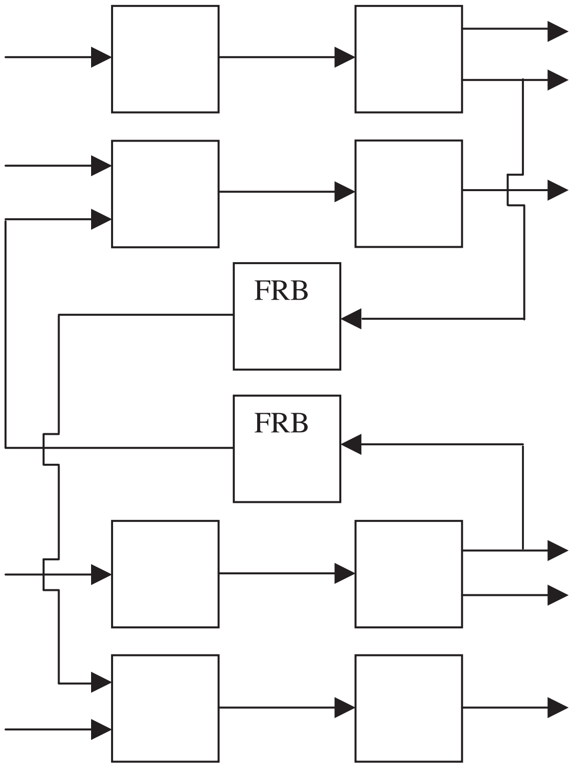

Fig. 8.13. Crossed non-symmetric mid-upward feedback for rule bases in parallel and sequence

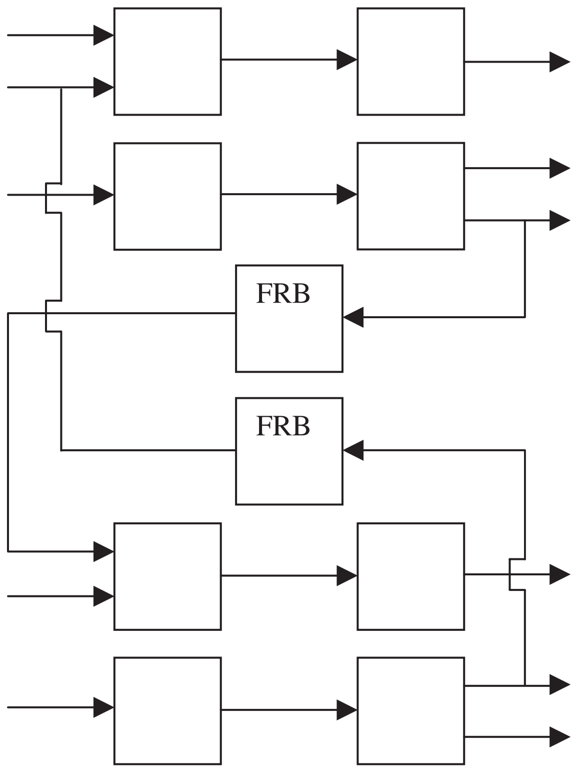

Fig. 8.14. Crossed non-symmetric mid-downward feedback for rule bases in parallel and sequence

A fuzzy rule based system with crossed FB is constrained because the linguistic value of the corresponding outputs in each rule is mapped by a FB function onto a linguistic value of the associated input in the same rule. Depending on whether this FB function is an identity function or another type of function, these two linguistic values may be the same, i.e. crossed simple FB, or different, i.e. crossed complex FB.

The notion of crossed FB is illustrated by five basic examples with MRB systems. In particular, Example 8.28 shows how crossed symmetric FB can be presented for four rule bases standing in parallel and sequence in adjacent levels and layers, whereas Examples 8.29–8.32 describe four different types of non-symmetric FB for six rule bases standing in parallel and sequence in adjacent levels and layers. All examples consider non-identity type of FB because identity FB cases would be quite easy to deal with.

The procedure described in Examples 8.28–8.32 starts with a MRB system whose FB function is replaced by a corresponding rule base in the FF path, as described in Sect. 8.3. Then, the resultant MRB system is transformed into an equivalent SRB system by means of appropriate merging manipulations. Finally, the rule base of the equivalent SRB system is checked to ensure that any FB constraints are met.

Example 8.28

A MRB system with crossed symmetric feedback involving four rule bases in parallel and sequence is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_1, i_1, o_1 & RB_2, i_2, o_2 \\ \text{level 2} & RB_3, i_3, o_3 & RB_4, i_4, o_4 \end{array} \tag{8.160}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1 = i_2 & F1(o_2) = i_3 \\ \text{level 2} & o_3 = i_4 & F2(o_4) = i_1 \end{array} \tag{8.161}$$

Equation (8.160) shows that RB_i, $i = 1,4$ have one input and one output each (i_i, o_i, $i = 1,4$). In addition, Eq. (8.161) shows that the output o_1 from RB_1 is fed forward unchanged into the input i_2 to RB_2 and the output o_3 from RB_3 is fed forward unchanged into the input i_4 to RB_4, whereas the output o_2 from RB_2 is mapped by the FB function $F1$ onto the input i_3 to RB_3 and the output o_4 from RB_4 is mapped by the FB function $F2$ onto the input i_1 to RB_1.

By introducing level 1 in a new layer 3 with a rule base RB_{F1} that replaces the FB function $F1$ and introducing level 2 in layer 3 with a rule base RB_{F2} that replaces the FB function $F2$, the initial MRB system with complex FB is transformed into an equivalent MRB system with simple FB. This FB consists of two crossed simple loops embracing the two groups of rule bases standing in sequence (RB_1, RB_2, RB_{F1}) and (RB_3, RB_4, RB_{F2}).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_1, i_1, o_1 & RB_2, i_2, o_2 & RB_{F1}, i_{F1}, o_{F1} \\ \text{level 2} & RB_3, i_3, o_3 & RB_4, i_4, o_4 & RB_{F2}, i_{F2}, o_{F2} \end{array} \tag{8.162}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{cccc} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & o_1 = i_2 & o_2 = i_{F1} & o_{F1} = i_3 \\ \text{level 2} & o_3 = i_4 & o_4 = i_{F2} & o_{F2} = i_1 \end{array} \tag{8.163}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F1} and RB_{F2} in the FF part of the system as well as the appearance of the new interconnection variables i_{F1}, o_{F1}, i_{F2}, o_{F2}. In this context, Eq. (8.162) shows that the equivalent MRB system has two inputs (i_1, i_3) and two outputs (o_{F1}, o_{F2}). In addition, Eq. (8.163) shows that the output o_1 from RB_1 is the same as the input i_2 to RB_2, the output o_3 from RB_3 is the same as the input i_4 to RB_4, the output o_2 from RB_2 is the same as the input i_{F1} to RB_{F1}, the output o_4 from RB_4 is the same as the input i_{F2} to RB_{F2}, whereas the output o_{F1} from RB_{F1} is the same as the input i_3 to RB_3 and the output o_{F2} from RB_{F2} is the same as the input i_1 to RB_1.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.164).

$$RB_E = (RB_1 + RB_3) * (RB_2 + RB_4) * (RB_{F1} + RB_{F2}) \tag{8.164}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{cc} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_1, i_3, o_{F1}, o_{F2} \end{array} \tag{8.165}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{cc} \text{level/layer} & \text{layer 1} \\ \text{level 1} & \begin{array}{c} o_{F1} = i_3 \\ o_{F2} = i_1 \end{array} \end{array} \tag{8.166}$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby for each of the fuzzy rules the linguistic values of the output o_{F1} are fed back unchanged into the input i_3 and the linguistic values of the output o_{F2} are fed back unchanged into the input i_1. In this case, the rule base RB_E must have the same structure as the rule base RB_E from Example 8.6, as specified by Eqs. (8.11)–(8.12).

Example 8.29
A MRB system with crossed non-symmetric top-upward feedback involving six rule bases in parallel and sequence is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_1, i_1, o_1 & RB_2, i_2, o_2 \\ \text{level 2} & RB_3, i_3, o_3 & RB_4, i_4, o_4 \\ \text{level 3} & RB_5, i_5, o_5 & RB_6, i_6, o_6 \end{array} \tag{8.167}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1 = i_2 & F1(o_2) = i_5 \\ \text{level 2} & o_3 = i_4 & F2(o_4) = i_1 \\ \text{level 3} & o_5 = i_6 & o_6 \end{array} \tag{8.168}$$

Equation (8.167) shows that RB_i, $i=1,6$ have one input and one output each (i_i, o_i, $i=1,6$). In addition, Eq. (8.168) shows that the output o_1 from RB_1 is fed forward unchanged into the input i_2 to RB_2, the output o_3 from RB_3 is fed forward unchanged into the input i_4 to RB_4, the output o_5 from RB_5 is fed forward unchanged into the input i_6 to RB_6, whereas the output o_2 from RB_2 is mapped by the FB function $F1$ onto the input i_5 to RB_5 and the output o_4 from RB_4 is mapped by the FB function $F2$ onto the input i_1 to RB_1.

By introducing level 1 in a new layer 3 with a rule base RB_{F1} that replaces the FB function $F1$, introducing level 2 in layer 3 with a rule base RB_{F2} that replaces the FB function $F2$, as well as introducing level 3 in layer 3 with an IRB RB_I representing a self standing output, the initial MRB system with complex FB is transformed into an equivalent MRB system with simple FB. This FB consists of two crossed simple loops embracing the three groups of rule bases standing in sequence (RB_1, RB_2, RB_{F1}), (RB_3, RB_4, RB_{F2}) and (RB_5, RB_6, RB_I).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_1,\ i_1, o_1 & RB_2,\ i_2, o_2 & RB_{F1},\ i_{F1}, o_{F1} \\ \text{level 2} & RB_3,\ i_3, o_3 & RB_4,\ i_4, o_4 & RB_{F2},\ i_{F2}, o_{F2} \\ \text{level 3} & RB_5,\ i_5, o_5 & RB_6,\ i_6, o_6 & RB_I,\ i_I, o_I \end{array} \quad (8.169)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & o_1 = i_2 & o_2 = i_{F1} & o_{F1} = i_5 \\ \text{level 2} & o_3 = i_4 & o_4 = i_{F2} & o_{F2} = i_1 \\ \text{level 3} & o_5 = i_6 & o_6 = i_I & o_I \end{array} \quad (8.170)$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F1}, RB_{F2} and RB_I in the FF part of the system as well as the appearance of the new interconnection variables i_{F1}, o_{F1}, i_{F2}, o_{F2}, i_I, o_I. In this context, Eq. (8.169) shows that the equivalent MRB system has three inputs (i_1, i_3, i_5) and three outputs (o_{F1}, o_{F2}, o_I). In addition, Eq. (8.170) shows that the output o_1 from RB_1 is the same as the input i_2 to RB_2, the output o_3 from RB_3 is the same as the input i_4 to RB_4, the output o_5 from RB_5 is the same as the input i_6 to RB_6, the output o_2 from RB_2 is the same as the input i_{F1} to RB_{F1}, the output o_4 from RB_4 is the same as the input i_{F2} to RB_{F2}, the output o_6 from RB_6 is the same as the input i_I to RB_I, whereas the output o_{F1} from RB_{F1} is the same as the input i_5 to RB_5 and the output o_{F2} from RB_{F2} is the same as the input i_1 to RB_1.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.171).

$$RB_E = (RB_1 + RB_3 + RB_5) * (RB_2 + RB_4 + RB_6) * (RB_{F1} + RB_{F2} + RB_I) \quad (8.171)$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_1, i_3, i_5, o_{F1}, o_{F2}, o_I \end{array} \quad (8.172)$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_{F1} = i_5 \\ & o_{F2} = i_1 \\ & o_1 \end{array} \tag{8.173}$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby for each of the fuzzy rules the linguistic values of the output o_{F1} are fed back unchanged into the input i_5 and the linguistic values of the output o_{F2} are fed back unchanged into the input i_1. In this case, the rule base RB_E must have a structure in accordance with Eqs. (8.174)–(8.175).

RB_E: (8.174)

$i_1\, i_3\, i_5 \,/\, o_{F1}\, o_{F2}\, o_1$	111	112	121	122	211	212	221	222
111	?	?	0	0	0	0	0	0
112	0	0	0	0	?	?	0	0
121	?	?	0	0	0	0	0	0
122	0	0	0	0	?	?	0	0
211	0	0	?	?	0	0	0	0
212	0	0	0	0	0	0	?	?
221	0	0	?	?	0	0	0	0
222	0	0	0	0	0	0	?	?

$$\begin{aligned} RB_E\text{: } \{&(111, 111)_?, (111, 112)_?, (112, 211)_?, (112, 212)_?, \\ &(121, 111)_?, (121, 112)_?, (122, 211)_?, (122, 212)_?, \\ &(211, 121)_?, (211, 122)_?, (212, 221)_?, (212, 222)_?, \\ &(221, 121)_?, (221, 122)_?, (222, 221)_?, (222, 222)_?\} \end{aligned} \tag{8.175}$$

Example 8.30

A MRB system with crossed non-symmetric bottom-upward feedback involving six rule bases in parallel and sequence is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_1, i_1, o_1 & RB_2, i_2, o_2 \\ \text{level 2} & RB_3, i_3, o_3 & RB_4, i_4, o_4 \\ \text{level 3} & RB_5, i_5, o_5 & RB_6, i_6, o_6 \end{array} \tag{8.176}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1 = i_2 & F1(o_2) = i_5 \\ \text{level 2} & o_3 = i_4 & o_4 \\ \text{level 3} & o_5 = i_6 & F2(o_6) = i_3 \end{array} \tag{8.177}$$

Equation (8.176) shows that RB_i, $i = 1,6$ have one input and one output each (i_i, o_i, $i = 1,6$). In addition, Eq. (8.187) shows that the output o_1 from RB_1 is fed forward unchanged into the input i_2 to RB_2, the output o_3 from RB_3 is fed forward unchanged into the input i_4 to RB_4, the output o_5 from RB_5 is fed forward unchanged into the input i_6 to RB_6, whereas the output o_2 from RB_2 is mapped by the FB function $F1$ onto the input i_5 to RB_5 and the output o_6 from RB_6 is mapped by the FB function $F2$ onto the input i_3 to RB_3.

By introducing level 1 in a new layer 3 with a rule base RB_{F1} that replaces the FB function $F1$, introducing level 2 in layer 3 with an IRB RB_I representing a self standing output, as well as introducing level 3 in layer 3 with a rule base RB_{F2} that replaces the FB function $F2$, the initial MRB system with complex FB is transformed into an equivalent MRB system with simple FB. This FB consists of two crossed simple loops embracing the three groups of rule bases standing in sequence (RB_1, RB_2, RB_{F1}), (RB_3, RB_4, RB_I) and (RB_5, RB_6, RB_{F2}).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_1, i_1, o_1 & RB_2, i_2, o_2 & RB_{F1}, i_{F1}, o_{F1} \\ \text{level 2} & RB_3, i_3, o_3 & RB_4, i_4, o_4 & RB_I, i_I, o_I \\ \text{level 3} & RB_5, i_5, o_5 & RB_6, i_6, o_6 & RB_{F2}, i_{F2}, o_{F2} \end{array} \tag{8.178}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & o_1 = i_2 & o_2 = i_{F1} & o_{F1} = i_5 \\ \text{level 2} & o_3 = i_4 & o_4 = i_I & o_I \\ \text{level 3} & o_5 = i_6 & o_6 = i_{F2} & o_{F2} = i_3 \end{array} \tag{8.179}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F1}, RB_I and RB_{F2} in the FF part of the system as well as the appearance of the new interconnection variables i_{F1}, o_{F1}, i_I, o_I, i_{F2}, o_{F2}. In this context, Eq. (8.178) shows that the equivalent MRB system has three inputs (i_1, i_3, i_5) and three outputs (o_{F1}, o_I, o_{F2}). In addition, Eq. (8.179) shows that the output o_1 from RB_1 is the same as the input i_2 to RB_2, the output o_3 from RB_3 is the same as the input i_4 to RB_4, the output o_5 from RB_5 is the same as the input i_6 to RB_6, the output o_2 from RB_2 is the same as the input i_{F1} to RB_{F1}, the output o_4 from RB_4 is the same as the input i_I to RB_I, the output o_6 from RB_6 is the same as the input i_{F2} to RB_{F2}, whereas the output o_{F1} from RB_{F1} is the same as the input i_5 to RB_5, and the output o_{F2} from RB_{F2} is the same as the input i_3 to RB_3.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.180).

$$RB_E = (RB_1 + RB_3 + RB_5) * (RB_2 + RB_4 + RB_6) * (RB_{F1} + RB_I + RB_{F2}) \quad (8.180)$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_1, i_3, i_5, o_{F1}, o_I, o_{F2} \end{array} \quad (8.181)$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_{F1} = i_5 \\ & o_I \\ & o_{F2} = i_3 \end{array} \quad (8.182)$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby for each of the fuzzy rules the linguistic values of the output o_{F1} are fed back unchanged into the input i_5 and the linguistic values of the output o_{F2} are fed back unchanged into the input i_3. In this case, the rule base RB_E must have a structure in accordance with Eqs. (8.183)–(8.184).

(8.183)

RB_E: $i_1\, i_3\, i_5 \,/\, o_{F1}\, o_I\, o_{F2}$	111	112	121	122	211	212	221	222
111	?	0	?	0	0	0	0	0
112	0	0	0	0	?	0	?	0
121	0	?	0	?	0	0	0	0
122	0	0	0	0	0	?	0	?
211	?	0	?	0	0	0	0	0
212	0	0	0	0	?	0	?	0
221	0	?	0	?	0	0	0	0
222	0	0	0	0	0	?	0	?

$$\begin{aligned} RB_E\text{: } \{&(111, 111)_?, (111, 121)_?, (112, 211)_?, (112, 221)_?, \\ &(121, 112)_?, (121, 122)_?, (122, 212)_?, (122, 222)_?, \\ &(211, 111)_?, (211, 121)_?, (212, 211)_?, (212, 221)_?, \\ &(221, 112)_?, (221, 122)_?, (222, 212)_?, (222, 222)_?\} \end{aligned} \tag{8.184}$$

Example 8.31

A MRB system with crossed non-symmetric top-downward feedback involving six rule bases in parallel and sequence is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_1, i_1, o_1 & RB_2, i_2, o_2 \\ \text{level 2} & RB_3, i_3, o_3 & RB_4, i_4, o_4 \\ \text{level 3} & RB_5, i_5, o_5 & RB_6, i_6, o_6 \end{array} \tag{8.185}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1 = i_2 & F1(o_2) = i_3 \\ \text{level 2} & o_3 = i_4 & o_4 \\ \text{level 3} & o_5 = i_6 & F2(o_6) = i_1 \end{array} \tag{8.186}$$

Equation (8.185) shows that RB_i, $i = 1,6$ have one input and one output each (i_i, o_i, $i = 1,6$). In addition, Eq. (8.186) shows that the output o_1 from RB_1 is fed forward unchanged into the input i_2 to RB_2, the output o_3 from RB_3 is fed forward unchanged into the input i_4 to RB_4, the output o_5 from RB_5 is fed forward unchanged into the input i_6 to RB_6, whereas the output o_2 from RB_2 is mapped by the FB function $F1$ onto the input i_3 to RB_3 and the output o_6 from RB_6 is mapped by the FB function $F2$ onto the input i_1 to RB_1.

By introducing level 1 in a new layer 3 with a rule base RB_{F1} that replaces the FB function $F1$, introducing level 2 in layer 3 with an IRB RB_I representing a self standing output, as well as introducing level 3 in layer 3 with a rule base RB_{F2} that replaces the FB function $F2$, the initial MRB system with complex FB is transformed into an equivalent MRB system with simple FB. This FB consists of two crossed simple loops embracing the three groups of rule bases standing in a sequence (RB_1, RB_2, RB_{F1}), (RB_3, RB_4, RB_I) and (RB_5, RB_6, RB_{F2}).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & RB_1,\, i_1,\, o_1 & RB_2,\, i_2,\, o_2 & RB_{F1},\, i_{F1},\, o_{F1} \\
\text{level 2} & RB_3,\, i_3,\, o_3 & RB_4,\, i_4,\, o_4 & RB_I,\, i_I,\, o_I \\
\text{level 3} & RB_5,\, i_5,\, o_5 & RB_6,\, i_6,\, o_6 & RB_{F2},\, i_{F2},\, o_{F2}
\end{array} \tag{8.187}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & o_1 = i_2 & o_2 = i_{F1} & o_{F1} = i_3 \\
\text{level 2} & o_3 = i_4 & o_4 = i_I & o_I \\
\text{level 3} & o_5 = i_6 & o_6 = i_{F2} & o_{F2} = i_1
\end{array} \tag{8.188}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_{F1}, RB_I and RB_{F2} in the FF part of the system as well as the appearance of the new interconnection variables i_{F1}, o_{F1}, i_I, o_I, i_{F2}, o_{F2}. In this context, Eq. (8.187) shows that the equivalent MRB system has three inputs (i_1, i_3, i_5) and three outputs (o_{F1}, o_I, o_{F2}). In addition, Eq. (8.188) shows that the output o_1 from RB_1 is the same as the input i_2 to RB_2, the output o_3 from RB_3 is the same as the input i_4 to RB_4, the output o_5 from RB_5 is the same as the input i_6 to RB_6, the output o_2 from RB_2 is the same as the input i_{F1} to RB_{F1}, the output o_4 from RB_4 is the same as the input i_I to RB_I, the output o_6 from RB_6 is the same as the input i_{F2} to RB_{F2}, whereas the output o_{F1} from RB_{F1} is the same as the input i_3 to RB_3, and the output o_{F2} from RB_{F2} is the same as the input i_1 to RB_1.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.189).

$$RB_E = (RB_1 + RB_3 + RB_5) * (RB_2 + RB_4 + RB_6) * (RB_{F1} + RB_I + RB_{F2}) \quad (8.189)$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_1, i_3, i_5, o_{F1}, o_I, o_{F2} \end{array} \quad (8.190)$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_{F1} = i_3 \\ & o_I \\ & o_{F2} = i_1 \end{array} \quad (8.191)$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby for each of the fuzzy rules the linguistic values of the output o_{F1} are fed back unchanged into the input i_3 and the linguistic values of the output o_{F2} are fed back unchanged into the input i_1. In this case, the rule base RB_E must have a structure in accordance with Eqs. (8.192)–(8.193).

(8.192)

RB_E: $i_1\, i_3\, i_5 \,/\, o_{F1}\, o_I\, o_{F2}$	111	112	121	122	211	212	221	222
111	?	0	?	0	0	0	0	0
112	?	0	?	0	0	0	0	0
121	0	0	0	0	?	0	?	0
122	0	0	0	0	?	0	?	0
211	0	?	0	?	0	0	0	0
212	0	?	0	?	0	0	0	0
221	0	0	0	0	0	?	0	?
222	0	0	0	0	0	?	0	?

$$\begin{aligned} RB_E\!: \{&(111, 111)_?, (112, 111)_?, (111, 121)_?, (112, 121)_?, \\ &(121, 211)_?, (122, 211)_?, (121, 221)_?, (122, 221)_?, \\ &(211, 112)_?, (212, 112)_?, (211, 122)_?, (212, 122)_?, \\ &(221, 212)_?, (222, 212)_?, (221, 222)_?, (222, 222)_?\} \end{aligned} \quad (8.193)$$

Example 8.32

A MRB system with crossed non-symmetric bottom-downward feedback involving six rule bases in parallel and sequence is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_1, i_1, o_1 & RB_2, i_2, o_2 \\ \text{level 2} & RB_3, i_3, o_3 & RB_4, i_4, o_4 \\ \text{level 3} & RB_5, i_5, o_5 & RB_6, i_6, o_6 \end{array} \tag{8.194}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & o_1 = i_2 & o_2 \\ \text{level 2} & o_3 = i_4 & F1(o_4) = i_5 \\ \text{level 3} & o_5 = i_6 & F2(o_6) = i_1 \end{array} \tag{8.195}$$

Equation (8.194) shows that RB_i, $i = 1,6$ have one input and one output each (i_i, o_i, $i = 1,6$). In addition, Eq. (8.195) shows that the output o_1 from RB_1 is fed forward unchanged into the input i_2 to RB_2, the output o_3 from RB_3 is fed forward unchanged into the input i_4 to RB_4, the output o_5 from RB_5 is fed forward unchanged into the input i_6 to RB_6, whereas the output o_4 from RB_4 is mapped by the FB function $F1$ onto the input i_5 to RB_5 and the output o_6 from RB_6 is mapped by the FB function $F2$ onto the input i_1 to RB_1.

By introducing level 1 in a new layer 3 with an IRB RB_I representing a self standing output, introducing level 2 in layer 3 with a rule base RB_{F1} that replaces the FB function $F1$, as well as introducing level 3 in layer 3 with a rule base RB_{F2} that replaces the FB function $F2$, the initial MRB system with complex FB is transformed into an equivalent MRB system with simple FB. This FB consists of two crossed simple loops embracing the three groups of rule bases standing in sequence (RB_1, RB_2, RB_I), (RB_3, RB_4, RB_{F1}) and (RB_5, RB_6, RB_{F2}).

The equivalent MRB system is presented by the following matrix:

level/layer	layer 1	layer 2	layer 3
level 1	RB_1, i_1, o_1	RB_2, i_2, o_2	RB_I, i_I, o_I
level 2	RB_3, i_3, o_3	RB_4, i_4, o_4	RB_{F1}, i_{F1}, o_{F1}
level 3	RB_5, i_5, o_5	RB_6, i_6, o_6	RB_{F2}, i_{F2}, o_{F2}

(8.196)

The output-input interconnections for this MRB system are given by the matrix:

level/layer	layer 1	layer 2	layer 3
level 1	$o_1 = i_2$	$o_2 = i_I$	o_I
level 2	$o_3 = i_4$	$o_4 = i_{F1}$	$o_{F1} = i_5$
level 3	$o_5 = i_6$	$o_6 = i_{F2}$	$o_{F2} = i_1$

(8.197)

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_I, RB_{F1} and RB_{F2} in the FF part of the system as well as the appearance of the new interconnection variables i_I, o_I, $_{F1}$, o_{F1}, i_{F2}, o_{F2}. In this context, Eq. (8.196) shows that the equivalent MRB system has three inputs (i_1, i_3, i_5) and three outputs (o_I, o_{F1}, o_{F2}). In addition, Eq. (8.197) shows that the output o_1 from RB_1 is the same as the input i_2 to RB_2, the output o_3 from RB_3 is the same as the input i_4 to RB_4, the output o_5 from RB_5 is the same as the input i_6 to RB_6, the output o_2 from RB_2 is the same as the input i_I to RB_I, the output o_4 from RB_4 is the same as the input i_{F1} to RB_{F1}, the output o_6 from RB_6 is the same as the input i_{F2} to RB_{F2}, whereas the output o_{F1} from RB_{F1} is the same as the input i_5 to RB_5 and the output o_{F2} from RB_{F2} is the same as the input i_1 to RB_1.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.198).

$$RB_E = (RB_1 + RB_3 + RB_5) * (RB_2 + RB_4 + RB_6) * (RB_I + RB_{F1} + RB_{F2}) \quad (8.198)$$

The equivalent SRB system is presented by the following matrix:

level/layer	layer 1
level 1	$RB_E, i_1, i_3, i_5, o_I, o_{F1}, o_{F2}$

(8.199)

The output-input interconnections for this SRB system are given by the matrix:

level/layer	layer 1
level 1	o_1
	$o_{F1} = i_5$
	$o_{F2} = i_1$

(8.200)

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby for each of the fuzzy rules the linguistic values of the output o_{F1} are fed back unchanged into the input i_5 and the linguistic values of the output o_{F2} are fed back unchanged into the input i_1. In this case, the rule base RB_E must have a structure in accordance with Eqs. (8.201)–(8.202).

(8.201)

RB_E: $i_1\, i_3\, i_5\, /\, o_1\, o_{F1}\, o_{F2}$	111	112	121	122	211	212	221	222
111	?	0	0	0	?	0	0	0
112	0	0	?	0	0	0	?	0
121	?	0	0	0	?	0	0	0
122	0	0	?	0	0	0	?	0
211	0	?	0	0	0	?	0	0
212	0	0	0	?	0	0	0	?
221	0	?	0	0	0	?	0	0
222	0	0	0	?	0	0	0	?

$$
\begin{aligned}
RB_E\text{: } \{&(111, 111)_?, (112, 121)_?, (111, 211)_?, (112, 221)_?,\\
&(121, 111)_?, (122, 121)_?, (121, 211)_?, (122, 221)_?,\\
&(211, 112)_?, (212, 122)_?, (211, 212)_?, (212, 222)_?,\\
&(221, 112)_?, (222, 122)_?, (221, 212)_?, (222, 222)_?\}
\end{aligned}
\tag{8.202}
$$

8.8 Transformation of Rule Bases with Multiple Feedback

Multiple feedback is an extension of all other types of FB considered so far. In this sense, the latter are assumed to have only single-output-single-input FB loops whereas multiple FB has single-output-multiple-input, multiple-output-single-input or multiple-output-multiple-input FB loops. Therefore, multiple FB always comes from or goes to at least two rule bases in parallel or sequence in the network structure of a MRB system. Three general cases with multiple FB are presented in Figs. 8.15–8.17.

A fuzzy rule based system with multiple FB is constrained because the linguistic value of the corresponding outputs in each rule is mapped by a FB function onto a linguistic value of the associated input in the same rule. Depending on whether this FB function is an identity function or another

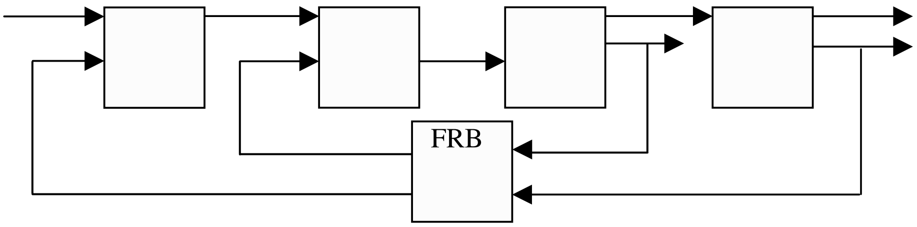

Fig. 8.15. Multiple feedback for rule bases in a single level

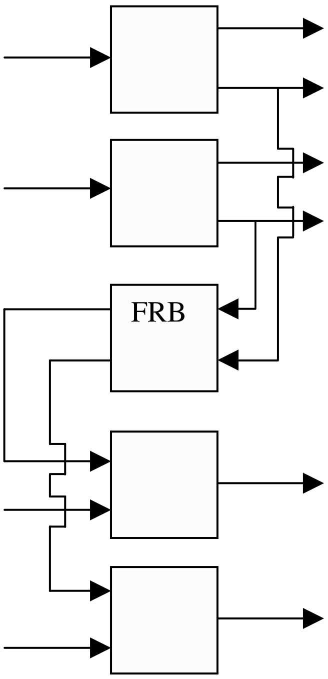

Fig. 8.16. Multiple feedback for rule bases in a single layer

type of function, these two linguistic values may be the same, i.e. multiple simple FB, or different, i.e. multiple complex FB.

The notion of multiple FB is illustrated by three basic examples with MRB systems. In particular, Example 8.33 illustrates multiple FB from one to another couple of rule bases such that all rule bases are standing in sequence in a single level. Example 8.34 describes multiple FB from one to another couple of rule bases such that all rule bases are standing in parallel in a single layer. Finally, Example 8.35 describes multiple FB from one to another couple of rule bases such that the rule bases in each couple are

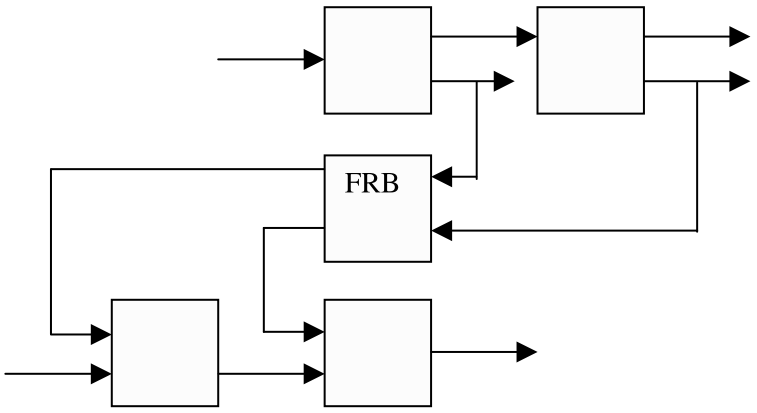

Fig. 8.17. Multiple feedback for rule bases in different levels and layers

standing in sequence in a level different from the level of the other couple but also in parallel with respect to the rule bases in the other couple.

All examples consider non-identity type of FB because identity FB cases would be quite easy to deal with. Also, all examples consider two-output-two-input FB, which is a straightforward extension of single-output-two-input or two-output-single-input FB.

The procedure described in Examples 8.33–8.35 starts with a MRB system whose FB function is replaced by a corresponding rule base in the FF path, as described in Sect. 8.3. Then, the resultant MRB system is transformed into an equivalent SRB system by means of appropriate merging manipulations. Finally, the rule base of the equivalent SRB system is checked to ensure that any FB constraints are met.

Example 8.33

A MRB system with multiple FB from and to rule bases in sequence in a single level is presented by the following matrix:

$$\begin{array}{lllll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} & \text{layer 4} \\ \text{level 1} & RB_1, i_1, o_1 & RB_2, i_{21}, i_{22}, o_2 & RB_3, i_3, o_{31}, o_{32} & RB_4, i_4, o_4 \end{array} \tag{8.203}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lllll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} & \text{layer 4} \\
\text{level 1} & o_1 = i_{21} & o_2 = i_3 & o_{31} = i_4 & F(o_4) = i_1 \\
 & & & F(o_{32}) = i_1 & F(o_4) = i_{22} \\
 & & & F(o_{22}) = i_{32} &
\end{array} \qquad (8.204)$$

Equation (8.203) shows that RB_1 has one input and one output (i_1, o_1), RB_2 has two inputs and one output (i_{21}, i_{22}, o_2), RB_3 has one input and two outputs (i_3, o_{31}, o_{32}), whereas RB_4 has one input and one output (i_4, o_4). In addition, Eq. (8.204) shows that the output o_1 from RB_1 is fed forward unchanged into the input i_{21} to RB_2, the output o_2 from RB_2 is fed forward unchanged into the input i_3 to RB_3, the output o_{31} from RB_3 is fed forward unchanged into the input i_4 to RB_4, whereas the outputs o_{32} from RB_3 and the output o_4 from RB_4 are mapped by the FB function F onto the input i_1 to RB_1 and the input i_{22} to RB_2.

By introducing level 1 in a new layer 5 with a rule base RB_F that replaces the FB function F, introducing level 2 in layer 1 with a rule base RB_{I1} representing a self standing input, as well as introducing level 2 in layer 4 with a rule base RB_{I2} representing a partial identity line, the initial MRB system with complex FB is transformed into an equivalent MRB system with simple FB. This FB consists of two nested simple loops whereby the inner loop embraces the sequence of rule bases RB_{I1}, RB_2,RB_3, RB_{I2}, RB_F and the outer loop embraces the sequence of rule bases RB_1, RB_2,RB_3, RB_4, RB_F.

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llllll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} & \text{layer 4} & \text{layer 5} \\
\text{level 1} & RB_1,i_1,o_1 & RB_2,i_{21},i_{22},o_2 & RB_3,i_3,o_{31},o_{32} & RB_4,i_4,o_4 & RB_F,i_{F1},o_{F2},i_{F1},o_{F2} \\
\text{level 2} & RB_{I1},i_{I1},o_{I1} & & & RB_{I2},i_{I2},o_{I2} &
\end{array} \qquad (8.205)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llllll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} & \text{layer 4} & \text{layer 5} \\
\text{level 1} & o_1 = i_{21} & o_2 = i_3 & o_{31} = i_4 & o_4 = i_{F1} & o_{F1} = i_1 \\
 & & & o_{32} = i_{I2} & & o_{F2} = i_{I1} \\
\text{level 2} & o_{I1} = i_{22} & & & o_{I2} = i_{F2} &
\end{array} \qquad (8.206)$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_F, RB_{I1} and RB_{I2} in the FF part of the system as well as the appearance of the new interconnection variables i_{F1}, i_{F2}, o_{F1}, o_{F2}, i_{I1}, o_{I1}, i_{I2}, o_{I2}. In this context, Eq. (8.205) shows that the equivalent

MRB system has two inputs (i_I, i_{II}) and two outputs (o_{F1}, o_{F2}). In addition, Eq. (8.206) shows that the output o_I from RB_I is the same as the input i_{21} to RB_2, the output o_{II} from RB_{II} is the same as the input i_{22} to RB_2, the output o_2 from RB_2 is the same as the input i_3 to RB_3, the output o_{31} from RB_3 is the same as the input i_4 to RB_4, the output o_{32} from RB_3 is the same as the input i_{I2} to RB_{I2}, the output o_4 from RB_4 is the same as the input i_{F1} to RB_F, the output o_{I2} from RB_{I2} is the same as the input i_{F2} to RB_F, whereas the output o_{F1} from RB_F is the same as the input i_I to RB_I and the output o_{F2} from RB_F is the same as the input i_{II} to RB_{II}.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.207).

$$RB_E = (RB_I + RB_{II}) * RB_2 * RB_3 * (RB_4 + RB_{I2}) * RB_F \qquad (8.207)$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_I, i_{II}, o_{F1}, o_{F2} \end{array} \qquad (8.208)$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & \begin{array}{l} o_{F1} = i_I \\ o_{F2} = i_{II} \end{array} \end{array} \qquad (8.209)$$

The rule base RB_E must satisfy the constraints imposed by the simple FB whereby for each of the fuzzy rules the linguistic values of the output o_{F1} are fed back unchanged into the input i_I and the linguistic values of the output o_{F2} are fed back unchanged into the input i_{II}. In other words, the rule base RB_E must have the same structure as the rule base RB_E from Example 8.19, as specified by Eqs. (8.102)–(8.103).

Example 8.34

A MRB system with multiple FB from and to rule bases in parallel in a single layer is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_1,\ i_1, o_1 \\ \text{level 2} & RB_2,\ i_2, o_2 \\ \text{level 3} & RB_3,\ i_3, o_3 \\ \text{level 4} & RB_4,\ i_4, o_4 \end{array} \qquad (8.210)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & F(o_1) = i_3 \\ & F(o_1) = i_4 \\ \text{level 2} & F(o_2) = i_3 \\ & F(o_2) = i_4 \\ \text{level 3} & o_3 \\ \text{level 4} & o_4 \end{array} \qquad (8.211)$$

Equation (8.210) shows that RB_i, $i = 1,4$ have one input and one output each (i_i, o_i, $i = 1,4$). In addition, Eq. (8.211) shows that the output o_1 from RB_1 and the output o_2 from RB_2 are mapped by the FB function F onto the input i_3 to RB_3 and the input i_4 to RB_4.

By moving RB_3 from level 3 of layer 1 to level 1 of a new layer 3 and removing level 3 from layer 1, moving RB_4 from level 4 of layer 1 to level 2 of layer 3 and removing level 4 from layer 1, as well as introducing level 1 in a new layer 2 with a rule base RB_F that replaces the FB function F, the initial MRB system with complex FB is transformed into an equivalent MRB system without FB. This system consists of two groups of rule bases standing in sequence, i.e. (RB_1, RB_F, RB_3) and (RB_2, RB_F, RB_4).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & RB_1, i_1, o_1 & RB_F, i_{F1}, i_{F2}, o_{F1}, o_{F2} & RB_3, i_3, o_3 \\ \text{level 2} & RB_2, i_2, o_2 & & RB_4, i_4, o_4 \end{array} \qquad (8.212)$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & o_1 = i_{F1} & o_{F1} = i_3 & o_3 \\ & & o_{F2} = i_4 & \\ \text{level 2} & o_2 = i_{F2} & & o_4 \end{array} \qquad (8.213)$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_F in the FF part of the system as well as the appearance of the new interconnection variables $i_{F1}, i_{F2}, o_{F1}, o_{F2}$. In this context, Eq. (8.212) shows that the equivalent MRB system has two inputs (i_1, i_2) and two outputs (o_3, o_4). In addition, Eq. (8.213) shows that the output o_1 from RB_1 is the same as the input i_{F1} to RB_F, the output o_2 from RB_2 is the same as the input i_{F2} to RB_F, the output o_{F1} from RB_F is the same as the input i_3 to RB_3 and the output o_{F2} from RB_F is the same as the input i_4 to RB_4.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.214).

$$RB_E = (RB_1 + RB_2) * RB_F * (RB_3 + RB_4) \tag{8.214}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E, i_1, i_2, o_3, o_4 \end{array} \tag{8.215}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{ll} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_3 \\ & o_4 \end{array} \tag{8.216}$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

Example 8.35

A MRB system with multiple FB from rule bases in sequence in one level to rule bases in sequence in another level is presented by the following matrix:

$$\begin{array}{llll} \text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\ \text{level 1} & & RB_1, i_1, o_{11}, o_{12} & RB_2, i_2, o_2 \\ \text{level 2} & RB_3, i_3, o_3 & RB_4, i_{41}, i_{42}, o_4 & \end{array} \tag{8.217}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} \\
\text{level 1} & & o_{11} = i_2 & F(o_2) = i_3 \\
 & & F(o_{12}) = i_3 & F(o_2) = i_{41} \\
 & & F(o_{12}) = i_{41} & \\
\text{level 2} & o_3 = i_{42} & o_4 &
\end{array} \tag{8.218}$$

Equation (8.217) shows that RB_1 has one self standing input and two outputs (i_1, o_{11}, o_{12}), RB_2 has one input and one output (i_2, o_2), RB_3 has one input and one output (i_3, o_3), whereas RB_4 has two inputs and one self standing output (i_{41}, i_{42}, o_4). In addition, Eq. (8.218) shows that the output o_3 from RB_3 is fed forward unchanged into the input i_{42} to RB_4, the output o_{11} from RB_1 is fed forward unchanged into the input i_2 to RB_2, whereas the output o_{12} from RB_1 and the output o_2 from RB_2 are mapped by the FB function F onto the input i_3 to RB_3 and the input i_{41} to RB_4.

By moving RB_1 from level 1 of layer 2 to level 1 of layer 1, moving RB_2 from level 1 of layer 3 to level 1 of layer 2, moving RB_3 from level 2 of layer 1 to level 2 in a new layer 4, moving RB_4 from level 2 of layer 2 to level 1 in a new layer 5, introducing in level 1 of layer 3 a rule base RB_F that replaces the FB function F, as well as introducing a rule base RB_{I1} in level 2 of layer 2 and a rule base RB_{I2} in level 1 of layer 4 representing partial identity lines, the initial MRB system with complex FB is transformed into an equivalent MRB system without FB. This system consists of two groups of rule bases standing in sequence, i.e. (RB_1, RB_2, RB_F, RB_{I2}, RB_4) and (RB_1, RB_{I1}, RB_F, RB_3, RB_4).

The equivalent MRB system is presented by the following matrix:

$$\begin{array}{llllll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} & \text{layer 4} & \text{layer 5} \\
\text{level 1} & RB_1, i_1, o_{11}, o_{12} & RB_2, i_2, o_2 & RB_F, i_{F1}, i_{F2}, o_{F1}, o_{F2} & RB_{I2}, i_{I2}, o_{I2} & RB_4, i_{41}, i_{42}, o_4 \\
\text{level 2} & & RB_{I1}, i_{I1}, o_{I1} & & RB_3, i_3, o_3 &
\end{array} \tag{8.219}$$

The output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{llllll}
\text{level/layer} & \text{layer 1} & \text{layer 2} & \text{layer 3} & \text{layer 4} & \text{layer 5} \\
\text{level 1} & o_{11} = i_2 & o_2 = i_{F1} & o_{F1} = i_{I2} & o_{I2} = i_{41} & o_4 \\
 & o_{12} = i_{I1} & & o_{F2} = i_3 & & \\
\text{level 2} & & o_{I1} = i_{F2} & & o_3 = i_{42} &
\end{array} \tag{8.220}$$

The transformation of the initial MRB system into an equivalent MRB system has led to the appearance of RB_F, RB_{I2} and RB_{I1} in the FF part of the system as well as the appearance of the new interconnection variables i_{F1}, i_{F2}, o_{F1}, o_{F2}, i_{I2}, o_{I2}, i_{I1}, o_{I1}. In this context, Eq. (8.219) shows that the equivalent MRB system has one inputs (i_1) and one output (o_4). In addition, Eq. (8.220) shows that the output o_{11} from RB_1 is the same as the input i_2 to RB_2, the output o_{12} from RB_1 is the same as the input i_{I1} to RB_{I1}, the output o_2 from RB_2 is the same as the input i_{F1} to RB_F, the output o_{I1} from RB_{I1} is the same as the input i_{F2} to RB_F, the output o_{F1} from RB_F is the same as the input i_{I2} to RB_{I2}, the output o_{F2} from RB_F is the same as the input i_3 to RB_3, the output o_{I2} from RB_{I2} is the same as the input i_{41} to RB_4 and the output o_3 from RB_3 is the same as the input i_{42} to RB_4.

The equivalent MRB system can be further transformed into an equivalent SRB system with a rule base RB_E which is derived from Eq. (8.221).

$$RB_E = RB_1 * (RB_2 + RB_{I1}) * RB_F * (RB_{I2} + RB_3) * RB_4 \tag{8.221}$$

The equivalent SRB system is presented by the following matrix:

$$\begin{array}{cc} \text{level/layer} & \text{layer 1} \\ \text{level 1} & RB_E,\ i_1,\ o_4 \end{array} \tag{8.222}$$

The output-input interconnections for this SRB system are given by the matrix:

$$\begin{array}{cc} \text{level/layer} & \text{layer 1} \\ \text{level 1} & o_4 \end{array} \tag{8.223}$$

Due to the lack of FB in the equivalent SRB system, its rule base RB_E may have an arbitrary structure.

8.9 Feedback Rule Base Design

Sections 8.2–8.8 consider FB interconnections in a MRB system in the context of analysis whereby all FB loops are given and we have to study the system for the purpose of formal transformation. In this case, we have to ensure that the rule base RB_E of the equivalent SRB system does not violate the constraints imposed by the FB function F.

However, in the context of design we may have to build a MRB system with specific features from scratch or by expanding an existing fuzzy system. In either case, we may have to derive a FB rule base RB_F representing the FB interconnections. Therefore, RB_F must guarantee that the rule base RB_E of the equivalent SRB system does not violate the constraints imposed by the FB.

In general, the FRB design problem can be presented by the equation

$$RB_G * RB_F = RB_E \tag{8.224}$$

where RB_G is the given rule base for an existing fuzzy system, RB_E is the ERB for the expanded fuzzy system and RB_F is the FB rule base, as shown in Fig. 8.18. In this case, RB_G is known and it reflects the features of the existing fuzzy system that we want to expand, RB_E is also known and it reflects the features of the target fuzzy system into which we want to expand the existing system, whereas RB_F is unknown and it reflects the FB interconnections in the target fuzzy system.

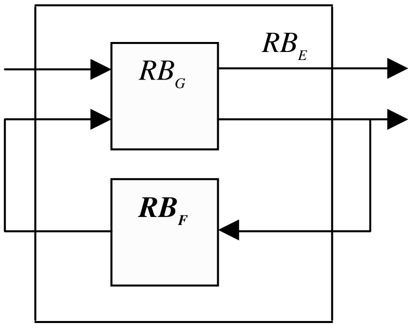

Fig. 8.18. Feedback rule base design

In other words, we have to solve the Boolean matrix equation (8.224) with respect to the unknown rule base RB_F. In this case, the three rule bases will have the form

$$RB_E\!: \quad \begin{array}{l} e_{11}\ e_{12} \dots e_{1c} \\ e_{21}\ e_{22} \dots e_{2c} \\ \dots\dots\dots\dots \\ e_{a1}\ e_{a2} \dots e_{ac} \end{array} \tag{8.225}$$

$$RB_G\!: \quad \begin{array}{l} g_{11}\ g_{12} \dots g_{1b} \\ g_{21}\ g_{22} \dots g_{2b} \\ \dots\dots\dots\dots \\ g_{a1}\ g_{a2} \dots g_{ab} \end{array} \tag{8.226}$$

$$RB_F:\quad \begin{array}{l} f_{11}\ f_{12} \ldots f_{1c} \\ f_{21}\ f_{22} \ldots f_{2c} \\ \ldots\ldots\ldots\ldots \\ f_{b1}\ f_{b2} \ldots f_{bc} \end{array} \tag{8.227}$$

where a is the number of permutations of linguistic values of inputs for RB_G and RB_E, c is the number of permutations of linguistic values of outputs for RB_F and RB_E, whereas b is the number of permutations of linguistic values of outputs for RB_G and the number of permutations of linguistic values of inputs for RB_F.

On the basis of Eqs. (8.225)–(8.227), the Boolean matrix equation (8.224) can be presented in the expanded form

$$\begin{array}{l} g_{11}\cdot f_{11} + g_{12}\cdot f_{21} + \ldots + g_{1b}\cdot f_{b1} = e_{11} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ g_{a1}\cdot f_{11} + g_{a2}\cdot f_{21} + \ldots + g_{ab}\cdot f_{b1} = e_{a1} \\ \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \\ g_{11}\cdot f_{1c} + g_{12}\cdot f_{2c} + \ldots + g_{1b}\cdot f_{bc} = e_{1c} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ g_{a1}\cdot f_{1c} + g_{a2}\cdot f_{2c} + \ldots + g_{ab}\cdot f_{bc} = e_{ac} \end{array} \tag{8.228}$$

with c systems of Boolean equations whereby each of these systems consists of a equations with b unknowns.

In this case, a unique solution to Eq. (8.228) would imply the existence of only one FB rule base RB_F whereas a non-unique solution would imply the existence of more than one FB rule bases. Also, the lack of solution would imply the non-existence of a FB rule base.

The derivation of a FB rule base RB_F with a specific structure is required only when the initial system with complex FB is transformed into an equivalent system with simple FB. Otherwise, i.e. when the initial system with complex FB is transformed into an equivalent system without FB, the derivation procedure is irrelevant because the FB rule base RB_F may have an arbitrary structure.

The concept of FB rule base design is illustrated further by Examples 8.36–8.41 which are based on Examples 8.7–8.12. The examples are fairly simple but quite representative and therefore an extension to more complex cases would be straightforward.

Example 8.36

This example is based on the 1×1 rule base *RB* from Example 8.7. In this case, *RB* is the given rule base for the existing fuzzy system. This rule base is known and it is represented by Eqs. (8.13)–(8.16). RB_E is the ERB for the expanded fuzzy system, which is also known. This rule base is represented by Eqs. (8.18)–(8.19) and its structure is in accordance with Eqs. (8.1)–(8.2) from Example 8.1. RB_F is the FB rule base, which is unknown and can be derived from the following Boolean matrix equation:

$$RB * RB_F = RB_E \tag{8.229}$$

Equation (8.229) is of the same type as Eq. (8.224) and can be solved using Eqs. (8.225)–(8.228).

Example 8.37

This example is based on the 2×1 rule base *RB* from Example 8.8. In this case, *RB* is the given rule base for the existing fuzzy system. This rule base is known and it is represented by Eqs. (8.20)–(8.23). RB_E is the ERB for the expanded fuzzy system, which is also known. This rule base is represented by Eqs. (8.25)–(8.26) and its structure is in accordance with Eqs. (8.3)–(8.4) from Example 8.2. RB_F is the FB rule base, which is unknown and can be derived from the following Boolean matrix equation:

$$RB * RB_F = RB_E \tag{8.230}$$

Equation (8.230) is of the same type as Eq. (8.224) and can be solved using Eqs. (8.225)–(8.228).

Example 8.38

This example is based on the 1x2 rule base *RB* from Example 8.9. In this case, *RB* is the given rule base for the existing fuzzy system. This rule base is known and it is represented by Eqs. (8.27)–(8.30). RB_E is the ERB for the expanded fuzzy system, which is also known. This rule base is represented by Eqs. (8.32)–(8.33) and its structure is in accordance with Eqs. (8.5)–(8.6) from Example 8.3. RB_F is the FB rule base, which is unknown and is part of the following Boolean matrix equation:

$$\mathrm{RB} * (RB_I + RB_F) = RB_E \tag{8.231}$$

Equation (8.231) is not of the same type as Eq. (8.224) and has to be simplified by separating the FB loop before it can be solved. In particular, Eq. (8.231) can be decomposed into the following two Boolean matrix equations

$$RB_1 * RB_I = RB_{E1} \tag{8.232}$$

$$RB_2 * RB_F = RB_{E2} \tag{8.233}$$

where the rule bases RB_1 and RB_2 are the product rule bases from the output slitting of RB, i.e. $RB = RB_1{:}RB_2$.

Equations (8.232)–(8.233) show that the ERB RB_E from Eq. (8.231) has been replaced by the two rule bases RB_{E1} and RB_{E2} such that the FB rule base RB_F appears only in Eq. (8.233). In this case, RB_{E1} is obtainable from Eq. (8.232) by merging RB_1 and RB_I horizontally, whereas RB_{E2} has a structure in accordance with Eqs. (8.1)–(8.2) from Example 8.1 and is used for the derivation of RB_F from Eq. (8.233). The latter is of the same type as Eq. (8.224) and therefore can be solved using Eqs. (8.225)–(8.228).

Example 8.39

This example is based on the 2×2 rule base RB from Example 8.10. In this case, RB is the given rule base for the existing fuzzy system. This rule base is known and it is represented by Eqs. (8.34)–(8.37). RB_E is the ERB for the expanded fuzzy system, which is also known. This rule base is represented by Eqs. (8.39)–(8.40) and its structure is in accordance with Eqs. (8.7)–(8.8) from Example 8.4. RB_F is the FB rule base, which is unknown and is part of the following Boolean matrix equation:

$$\mathrm{RB} * (RB_F + RB_I) = RB_E \tag{8.234}$$

Equation (8.234) is not of the same type as Eq. (8.224) and has to be simplified by separating the FB loop before it can be solved. In particular, Eq. (8.234) can be decomposed into the following two Boolean matrix equations

$$RB_F * RB_I = RB_{E1} \tag{8.235}$$

$$RB_2 * RB_I = RB_{E2} \tag{8.236}$$

where the rule bases RB_1 and RB_2 are the product rule bases from the output slitting of RB, i.e. $RB = RB_1{:}RB_2$.

Equations (8.235)–(8.236) show that the ERB RB_E from Eq. (8.234) has been replaced by the two rule bases RB_{E1} and RB_{E2} such that the FB rule base RB_F appears only in Eq. (8.235). In this case, RB_{E2} is obtainable from Eq. (8.236) by merging RB_2 and RB_I horizontally, whereas RB_{E1} has a structure in accordance with Eqs. (8.3)–(8.4) from Example 8.2 and is used for the derivation of RB_F from Eq. (8.235). The latter is of the same type as Eq. (8.224) and therefore can be solved using Eqs. (8.225)–(8.228).

Example 8.40

This example is based on the 2×2 rule base *RB* from Example 8.11. In this case, *RB* is the given rule base for the existing fuzzy system. This rule base is known and it is represented by Eqs. (8.41)–(8.44). RB_E is the ERB for the expanded fuzzy system, which is also known. This rule base is represented by Eqs. (8.46)–(8.47) and its structure is in accordance with Eqs. (8.9)–(8.10) from Example 8.5. RB_F is the FB rule base, which is unknown and is part of the following Boolean matrix equation:

$$\mathrm{RB} * (RB_I + RB_F) = RB_E \tag{8.237}$$

Equation (8.237) is not of the same type as Eq. (8.224) and has to be simplified by separating the FB loop before it can be solved. In particular, Eq. (8.237) can be decomposed into the following two Boolean matrix equations

$$RB_1 * RB_I = RB_{E1} \tag{8.238}$$

$$RB_2 * RB_F = RB_{E2} \tag{8.239}$$

where the rule bases RB_1 and RB_2 are the product rule bases from the output slitting of *RB*, i.e. $RB = RB_1{:}RB_2$.

Equations (8.238)–(8.239) show that the ERB RB_E from Eq. (8.237) has been replaced by the two rule bases RB_{E1} and RB_{E2} such that the FB rule base RB_F appears only in Eq. (8.239). In this case, RB_{E1} is obtainable from Eq. (8.238) by merging RB_1 and RB_I horizontally, whereas RB_{E2} has a structure in accordance with Eqs. (8.240)–(8.241) further below and is used for the derivation of RB_F from Eq. (8.239). The latter is of the same type as Eq. (8.224) and therefore can be solved using Eqs. (8.225)–(8.228).

$$\begin{array}{cccc}
RB_{E2}\text{: } i_1\, i_2 / o_F & 1 & 2 & 3 \\
11 & ? & 0 & 0 \\
12 & ? & 0 & 0 \\
13 & ? & 0 & 0 \\
 & & & \\
21 & 0 & ? & 0 \\
22 & 0 & ? & 0 \\
23 & 0 & ? & 0 \\
 & & & \\
31 & 0 & 0 & ? \\
32 & 0 & 0 & ? \\
33 & 0 & 0 & ?
\end{array} \tag{8.240}$$

$$\begin{aligned} RB_{E2}\text{: } \{&(11, 1)_?, (12, 1)_?, (13, 1)_?, \\ &(21, 2)_?, (22, 2)_?, (23, 2)_?, \\ &(31, 3)_?, (32, 3)_?, (33, 3)_?\} \end{aligned} \tag{8.241}$$

Example 8.41

This example is based on the 2×2 rule base *RB* from Example 8.12. In this case, *RB* is the given rule base for the existing fuzzy system. This rule base is known and it is represented by Eqs. (8.48)–(8.51). RB_E is the ERB for the expanded fuzzy system, which is also known. This rule base is represented by Eqs. (8.53)–(8.54) and its structure is in accordance with Eqs. (8.11)–(8.12) from Example 8.6. RB_F is the FB rule base, which is unknown and is part of the following Boolean matrix equation:

$$\text{RB} * (RB_{F1} + RB_{F2}) = RB_E \tag{8.242}$$

Equation (8.242) is not of the same type as Eq. (8.224) and has to be simplified by separating the two FB loops from each other before it can be solved. In particular, Eq. (8.242) can be decomposed into the following two Boolean matrix equations

$$RB_1 * RB_{F1} = RB_{E1} \tag{8.243}$$

$$RB_2 * RB_{F2} = RB_{E2} \tag{8.244}$$

where the rule bases RB_1 and RB_2 are the product rule bases from the output slitting of *RB*, i.e. $RB = RB_1{:}RB_2$.

Equations (8.243)–(8.244) show that the ERB RB_E from Eq. (8.242) has been replaced by the two rule bases RB_{E1} and RB_{E2} such that the first FB rule base RB_{F1} appears only in Eq. (8.243) and the second FB rule base RB_{F2} appears only in Eq. (8.244). In this case, RB_{E1} has the same structure as the rule base RB_{E1} from Example 8.39 and is used for the derivation of RB_{F1} from Eq. (8.243), whereas RB_{E2} has the same structure as the rule base RB_{E2} from Example 8.40 and is used for the derivation of RB_{F2} from Eq. (8.244). Eqs. (8.243)–(8.244) are both of the same type as Eq. (8.224) and therefore each of them can be solved using Eqs. (8.225)–(8.228).

8.10 Canonical Rule Base Networks

The formal transformation process of MRB systems with FB considered in this chapter includes the following two main stages:

- transformation of the initial MRB system with complex FB into an equivalent MRB system with simple FB or without FB,
- transformation of the equivalent MRB system into an equivalent SRB system with simple FB or without FB.

In this context, the initial MRB system is in the form of an *arbitrary rule base network* (ARBN) such that individual rule bases reside in particular layers and levels of this network. The ARBN may include FF interconnections to subsequent layers as well as simple or complex FB interconnections to the same or preceding layers.

As opposed to the initial MRB system, the equivalent MRB system is in the form of a *canonical rule base network* (CRBN), which may include only FF interconnections and global simple FB interconnections from the last to the first layer. In this case, individual rule bases also reside in particular layers and levels of the CRBN but their location there is usually different from the associated location in the corresponding ARBN due to the rearrangement of layers and levels during the first stage of the transformation process.

The second stage of the transformation process is based on repetitive manipulations for vertical and horizontal merging of rule bases. However, in the case of more complex topology of the FF interconnections in the CRBN, a few additional steps may be required before the equivalent MRB system can be transformed into an equivalent SRB system. This process is illustrated by Example 8.42 and Algorithm 8.1.

Example 8.42
A MRB system is presented by the following matrix:

$$\begin{array}{lll} \text{level/layer} & \text{layer 1} & \text{layer 2} \\ \text{level 1} & RB_1, i_1, o_{11}, o_{12} & RB_2, i_{21}, i_{22}, o_2 \\ \text{level 2} & RB_3, i_3, o_{31}, o_{32} & RB_4, i_{41}, i_{42}, o_4 \end{array} \qquad (8.245)$$

The output-input interconnections for this MRB system are given by the matrix:

level/layer	layer 1	layer 2
level 1	$o_{11} = i_{21}$ $o_{12} = i_{41}$	o_2
level 2	$o_{31} = i_{22}$ $o_{32} = i_{42}$	o_4

(8.246)

The MRB system is in ARBN form, as shown in Fig. 8.19.

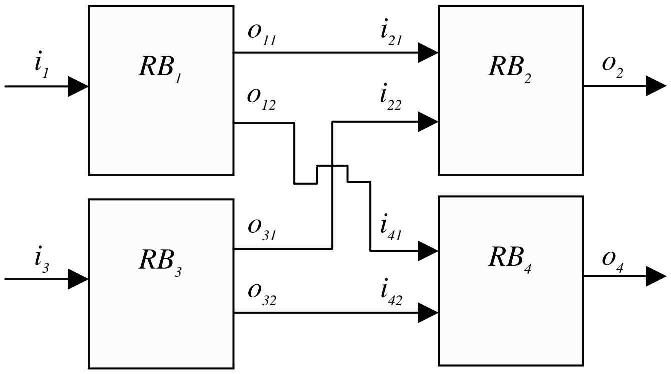

Fig. 8.19. Multiple rule base system in arbitrary rule base network form

This MRB system is with complex topology because of the crossing of the FF interconnections $o_{12} = i_{41}$ and $o_{31} = i_{22}$. In this case, it is not possible to merge horizontally the rule bases in the first layer RB_1 and RB_3 with the rule bases in the second layer RB_1 and RB_3. Therefore, it is necessary to convert this *multiple rule base output* (MRBO) system into a collection of equivalent *single rule base output* (SRBO) systems. If any of these SRBO systems are MO systems, i.e. with more than one output, they can be easily represented by a collection of equivalent SO systems, as described in Sect. 2.3.

The following algorithm describes the process of converting the MRBO system from Eqs. (8.245)–(8.246) into a collection of equivalent SRBO systems:

Algorithm 8.1

1. Find rule bases RB_{11} and RB_{12} such that $RB_1 = RB_{11}:RB_{12}$.
2. Find rule bases RB_{31} and RB_{32} such that $RB_3 = RB_{31}:RB_{32}$.
3. Find a rule base RB_{E1} such that $(RB_{11} + RB_{31})*RB_2 = RB_{E1}$.
4. Find a rule base RB_{E2} such that $(RB_{12} + RB_{32})*RB_4 = RB_{E2}$.
5. Find the rule base RB_E such that $RB_{E1};RB_{E2} = RB_E$.

In the above algorithm, RB_E is the ERB for the MRBO system from Eqs. (8.245)–(8.246), whereas RB_{E1} and RB_{E1} are the ERBs for the two SRBO systems which are obtained as a result of the conversion of the MRBO system. In particular, steps 1-2 illustrate the output splitting of the rule bases RB_1 and RB_3 into the two couples of rule bases (RB_{11}, RB_{12}) and (RB_{31}, RB_{32}), steps 3-4 show the derivation of the two ERBs RB_{E1} and RB_{E2} by vertical and horizontal merging manipulations on the four new and two of the initial rule bases, whereas step 5 describes the output merging of RB_{E1} and RB_{E2} into RB_E.

On the basis of Algorithm 8.1, the MRB system from Eqs. (8.245)–(8.246) can be presented by the following matrix:

$$\begin{array}{lll}
\text{level/layer} & \text{layer 1} & \text{layer 2} \\
\text{level 1} & RB_{11}, i_1, o_{11} & RB_2, i_{21}, i_{22}, o_2 \\
\text{level 2} & RB_{12}, i_1, o_{12} & \\
\text{level 3} & RB_{31}, i_3, o_{31} & \\
\text{level 4} & RB_{32}, i_3, o_{32} & RB_4, i_{41}, i_{42}, o_4
\end{array} \tag{8.247}$$

In this case, the output-input interconnections for this MRB system are given by the matrix:

$$\begin{array}{lll}
\text{level/layer} & \text{layer 1} & \text{layer 2} \\
\text{level 1} & o_{11} = i_{21} & o_2 \\
\text{level 2} & o_{12} = i_{41} & \\
\text{level 3} & o_{31} = i_{22} & \\
\text{level 4} & o_{32} = i_{42} & o_4
\end{array} \tag{8.248}$$

The MRB system is in CRBN form, as shown in Fig. 8.20.

The topology of CRBNs is described by a number of indicators, which provide useful information for the analysis and design of such networks. These indicators are as follows:

- in-degree and out-degree for a node, i.e. the number of inputs to and outputs from an individual rule base in the CRBN,
- overall in-degree and out-degree for a layer, i.e. the number of inputs to and outputs from the rule bases in a particular layer of the CRBN,

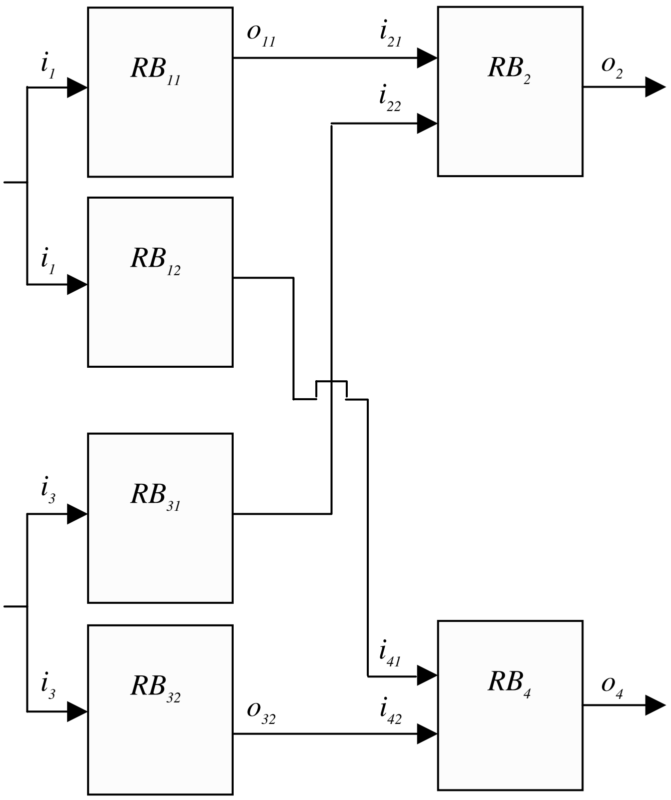

Fig. 8.20. Multiple rule base system in canonical rule base network form

- overall in-degree and out-degree for a level, i.e. the number of inputs to and outputs from the rule bases in a particular level of the CRBN,
- degree of completeness for a layer, i.e. the number of occupied level positions in a particular layer of the CRBN as a proportion of the overall number of level positions in this layer,
- degree of completeness for a level, i.e. the number of occupied layer positions in a particular level of the CRBN as a proportion of the overall number of layer positions in this level,
- overall degree of completeness for a CRBN, i.e. the number of occupied positions as a proportion of the overall number of positions in the network.

In general, FF interconnections in CRBNs can be classified as either local, i.e. to rule bases in the adjacent subsequent layer, or global, i.e. to rule bases in non-adjacent subsequent layers. However, global FF interconnections must be converted into local FF interconnections by introducing IRBs in all unoccupied positions, which are part of self standing inputs and outputs as well as partial or total identity lines. This type of conversion is required for the derivation of the rule base for the equivalent SRB system and it can be done by horizontal splitting of any rule base into the same rule base and an IRB.

8.11 Analysis of Transformation Techniques for Feedback Rule Bases

Examples 8.1-8.42 from the previous sections provide a detailed insight into how FB interconnections in MRB systems can be dealt with for the purpose of formal transformation. In this case, the ultimate goal is to transform a fairly complex ARBN form of the initial system into a simpler CRBN form, which can be further transformed into an equivalent SRB system. Depending on the type of transformation approach used, i.e. analysis or synthesis, the equivalent SRB system is either checked to ensure that any constraints imposed by the FB interconnections are not violated or derived such that the structure of the FB interconnections is determined in advance with the same purpose, i.e. to ensure that any FB constraints are met.

As far as the overall study of formal transformation of MRB systems is concerned, Boolean matrices are a more suitable tool than binary relations and that is why the latter are marginalised in this chapter. For example, it is much easier to transform a MRB system from an ARBN form into a CRBN form and then transform the latter further into an equivalent SRB system using Boolean matrices rather than the corresponding binary relations.

The transformation techniques with FF and FB rule bases have been demonstrated in the previous and the current chapter only in the context of qualitative complexity, i.e. for the purpose of improving the transparency and facilitating the interpretation of fuzzy systems. However, these transformation techniques can be also used in the context of quantitative complexity, i.e. to reduce the overall number of computations during the fuzzy inference process. A detailed study on this subject is presented in the next chapter.

9 Formal Simplification of Fuzzy Rule Based Systems

9.1 Preliminaries on Rule Base Simplification

As opposed to Chapters 4–8, which deal mainly with qualitative complexity in fuzzy systems, the current chapter is dedicated to the problem of quantitative complexity. However, the material presented here is a natural extension of the concepts introduced earlier in relation to qualitative complexity. In this context, the overall complexity management process in fuzzy systems has to be seen as a general framework whereby we first use formal presentation, manipulation and transformation techniques to address the qualitative complexity in fuzzy systems and then use formal simplification techniques to deal with the quantitative complexity.

The underlying idea of formal simplification of fuzzy rule based systems is to remove all redundant operations during the stages of fuzzification, inference and defuzzification. This redundancy is usually caused by inconsistent and non-monotonic rules. Therefore, the formal simplification process has to identify such rules and remove them safely, i.e. without affecting the output from the fuzzy system. This approach is quite different from the available rule base reduction methods in the sense that it is aimed at reducing the quantitative complexity in fuzzy systems without compromising the quality of the solution.

In order to identify the redundancy in a fuzzy rule based system, it is necessary to consider the stages of fuzzification, inference and defuzzification, as shown in Fig. 9.1. This consideration is done further in the current section whereby the inference stage is represented as a sequence of three substages – application, implication and aggregation. The considerations presented are about SRB systems, which are either given as such or have been transformed from a MRB system using the techniques introduced in some of the previous chapters. These considerations describe MISO systems, which are either given as such or are a logically equivalent collection of MISO systems representing a MIMO system, as described in Sect. 2.3.

The fuzzification stage in a fuzzy system maps the crisp value of each input to the system onto a fuzzy value by means of a fuzzy membership degree [58, 66]. This degree can be obtained from the fuzzy membership functions for the inputs to the fuzzy system. The considerations presented here are based on normal triangular fuzzy membership functions, which

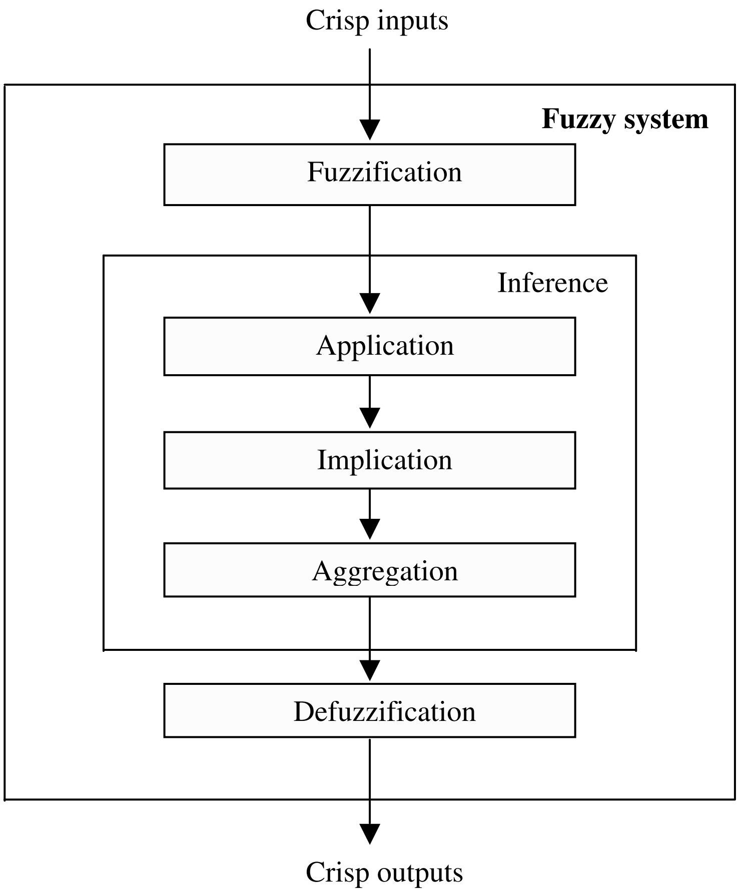

Fig. 9.1. Main operation stages and substages in a fuzzy system

have a unique maximum equal to 1 and appear to be most commonly used for the fuzzification stage in fuzzy systems due to their simplicity.

The fuzzy membership degree f_{ps} for an input can be obtained by the formula

$$\begin{aligned} f_{ps} &= 0, \text{ if } x_{ps} \le a_{ps} \\ f_{ps} &= (x_{ps} - a_{ps}) / (b_{ps} - a_{ps}), \text{ if } a_{ps} \le x_{ps} \le b_{ps} \\ f_{ps} &= (c_{ps} - x_{ps}) / (c_{ps} - b_{ps}), \text{ if } b_{ps} \le x_{ps} \le c_{ps} \\ f_{ps} &= 0, \text{ if } c_{ps} \le x_{ps} \end{aligned} \tag{9.1}$$

where x_{ps}, $p=1,..,m,\ s=1,..,r$ is the continuous crisp value of the p-th input in the s-th rule of the fuzzy system and a_{ps}, b_{ps}, c_{ps} are the parameters of the triangular fuzzy membership function used for the fuzzification of this input. The fuzzy membership degree f_{ps} can take only values in the interval [0, 1] whereas x_{ps} can take any values within the crisp variation range for the input, i.e. the continuous universe of discourse. The values of the parameters a_{ps}, b_{ps}, c_{ps} must be also within this universe of discourse. In particular, a_{ps} is the point at which the membership function becomes greater than 0, b_{ps} is the point at which the membership function reaches its maximum at 1 and c_{ps} is the point at which the membership function becomes equal to 0 again. The symbol '/' in Eq. (9.1) denotes arithmetic division and it is used mainly in this context in the current chapter.

The application substage in a fuzzy system maps the fuzzy membership degrees of the inputs in each rule in the system onto a firing strength for this rule [58, 66]. The considerations presented here are based on CON fuzzy rule bases, i.e. rule bases with CON antecedents. Such rule bases appear to be most commonly used in fuzzy systems as they allow the consideration of all possible permutations of linguistic values of inputs.

The firing strength g_s for a rule can be obtained by the formula

$$g_1 = min\ (f_{11},\dots,f_{m1}) \tag{9.2}$$

$$\dots\dots\dots\dots\dots\dots$$

$$g_r = min\ (f_{1r},\dots,f_{mr})$$

where f_{ps}, $p = 1,..,m$, $s = 1,..,r$ is the fuzzy membership degree for the p-th input in the s-th rule of the fuzzy system. Obviously, the firing strength g_s can take only values in the interval [0, 1].

The implication substage in a fuzzy system maps the firing strength for each rule of the system onto a fuzzy membership function for the output in this rule [58, 66]. The considerations presented here are based on horizontal truncation, which usually cuts the normal fuzzy triangular membership function for the output in each rule to a subnormal fuzzy trapezoidal membership function whose maximum is equal to the firing strength for this rule. This type of truncation is most commonly used for the implication substage in fuzzy systems due to its simplicity.

The fuzzy membership function F_{sq} for an output is usually defined by

$$F_{sq} = \{ f_{1sq} / y_{1sq},\dots, f_{tsq} / y_{tsq} \} \tag{9.3}$$

where f_{ksq}, $k = 1,..,t$, $s = 1,..,r$, $q=1,..,n$ is the fuzzy membership degree for the k-th element from the discrete universe of discourse for the q-th output in the s-th rule of the fuzzy system, whereas y_{ksq} is the associated element from this universe of discourse. As an exception, the 'forward slash' symbol '/' in Eq. (9.3) denotes binary association, i.e. the fuzzy membership degree f_{ksq} is associated with the element y_{ksq} from the universe of discourse.

When any of the subscripts in Eq. (9.3) are not quite relevant, they will be omitted in any further considerations of f_{ksq} and y_{ksq} for simplicity. This applies here to the subscript k and therefore the elements y_{sq} are mapped onto their fuzzy membership degrees f_{sq} by the formula

$$\begin{aligned} f_{sq} &= 0,\ if\ y_{sq} \le a_{sq} \\ f_{sq} &= (y_{sq} - a_{sq}) / (b_{sq} - a_{sq}),\ if\ a_{sq} \le y_{sq} \le b_{sq} \\ f_{sq} &= g_s,\ if\ b_{sq} \le y_{sq} \le c_{sq} \\ f_{sq} &= (d_{sq} - y_{sq}) / (c_{sq} - b_{sq}),\ if\ c_{sq} \le y_{sq} \le d_{sq} \\ f_{sq} &= 0,\ if\ d_{sq} \le y_{sq} \end{aligned} \tag{9.4}$$

where y_{sq}, $s = 1,..,r,\ q = 1,..,n$ is the discrete crisp value of the q-th output in the s-th rule of the fuzzy system and a_{sq}, b_{sq}, c_{sq}, d_{sq} are the parameters of the trapezoidal fuzzy membership function for this output obtained during the implication substage from the initial triangular fuzzy membership function for the output. The fuzzy membership degree f_{sq} can take only values in the interval [0, 1] whereas y_{sq} can take any values within the crisp variation range for the output, i.e. the discrete universe of discourse. The values of the parameters a_{sq}, b_{sq}, c_{sq}, d_{sq} must be also within this universe of discourse. In particular, a_{sq} is the point at which the membership function becomes greater than 0, b_{sq} is the point at which the membership function becomes equal to its maximum g_s, c_{sq} is the point at which the membership function becomes less than its maximum at g_s and d_{ps} is the point at which the membership function becomes equal to 0 again.

The aggregation substage in a fuzzy system maps the fuzzy membership functions for all rules in the system onto an aggregated fuzzy membership function representing the overall output for all the rules [58, 66]. The considerations presented here are based on DIS fuzzy rule bases, i.e. rule bases with DIS rules. Such rule bases appear to be most commonly used in fuzzy systems as they are more realistic, i.e. they only assume that at least one rule is satisfied at a time.

The aggregated fuzzy membership function F_q for an output can be obtained by the formula

$$F_q = F_{1q} \cup \ldots \cup F_{rq} \tag{9.5}$$

where F_{sq}, $s = 1,..,r$, $q = 1,..,n$ is the fuzzy membership function for the q-th output in the s-th rule of the fuzzy system. In this case, the symbol '$\cup$' denotes a 'fuzzy set union operation' that is applied by taking the minimum of the fuzzy membership degrees from the fuzzy membership functions for all rules. This minimum is taken with respect to the fuzzy membership degrees for all the elements from the discrete universe of discourse for this output.

The defuzzification stage in a fuzzy system maps the aggregated fuzzy membership function for an output in the system onto a crisp value from the universe of discourse for this output [58, 66]. This value is usually of continuous type, which implies that the associated discrete universe of discourse is mapped onto its continuous image. The considerations presented here are based the so-called 'centroid method' whereby the defuzzified value of the output is the centre of gravity for the aggregated fuzzy membership function for this output. This defuzzification method is most commonly used in fuzzy systems due to its applicability for any shape or type of aggregated fuzzy membership function for the output.

The defuzzified value D_q for an output can be obtained by the formula

$$D_q = (f_{1q} . y_{1q} + \dots + f_{tq} . y_{tq}) / (f_{1q} + \dots + f_{tq}) \tag{9.6}$$

where f_{kq}, $k = 1,..,t$, $q = 1,..,n$ is the aggregated fuzzy membership degree for the k-th element from the discrete universe of discourse for the q-th output of the fuzzy system, whereas y_{kq} is the associated element from this universe of discourse. In this case, Eq. (9.6) represents f_{ksq} and y_{ksq} from Eq. (9.3) with the index s being omitted due to the fact that the fuzzy rules do not appear as variables during the defuzzification stage. Obviously, D_q can take any values within the crisp variation range for the output, i.e. the so-called continuous image of the discrete universe of discourse for this output. In Eq. (9.6), the symbol '.' denotes arithmetic multiplication, the symbol '+' denotes arithmetic addition and the symbol '/' denotes again arithmetic division.

Two formal simplification techniques are introduced in the following sections of this chapter. Both techniques are based on the abstract considerations of the stages of fuzzification, inference and defuzzification in fuzzy systems made so far. The techniques are illustrated by examples with SRB systems, which are either given as such or have been transformed from a MRB system using the techniques introduced in some of the previous chapters.

The examples in the next section describe SISO systems, which are either given as such or are part of a logically equivalent collection of SISO systems representing a SIMO system, as described in Sect. 2.3. The examples can be extended to incorporate MISO systems. This type of extension is straight-forward and therefore is not considered here. In the case of a MISO system, the antecedent part of the rule base represents more than one input and this is the only difference in comparison to a rule base with one input.

What makes the examples distinctive is the number of fuzzy membership functions for the output and the number of rules in the fuzzy rule base. In this context, Examples 9.1–9.2 consider in some detail inconsistent rule bases whose outputs have three fuzzy membership functions each whereas Examples 9.3–9.12 deal briefly with inconsistent rule bases whose outputs have five fuzzy membership functions each. Similarly, Example 9.13 considers in some detail a non-monotonic rule base with 20 rules whereas Example 9.14 deals briefly with a non-monotonic rule base with 27 rules.

9.2 Rule Base Simplification by Aggregation of Inconsistent Rules

The formal simplification technique introduced here is based on the idea of removing the inherent redundancy in an inconsistent fuzzy rule base. This type of redundancy is expressed by the presence of inconsistent rules and it can be removed by aggregating such rules with the aim of making the rule base consistent.

The overall process of aggregating inconsistent rules is illustrated by the following algorithm:

Algorithm 9.1

1. Put all inconsistent rules in groups whereby the rules in each group have the same permutation of linguistic values of inputs and different permutations of linguistic values of outputs.
2. For each group of rules, find a single equivalent rule whose effect on the defuzzified output is the same as the effect of all rules.
3. If it is not possible to apply step 2, then find a subset of equivalent rules whose effect on the defuzzified output is the same as the effect of the whole set of rules for this group.
4. For each group of rules, keep either the single equivalent rule or the subset of equivalent rules and remove all other rules.

It follows from Algorithm 9.1 that there may still be some inconsistency left in a fuzzy rule base after the completion of the aggregation process. It is suggested that such a rule base is left as it is rather than being generated again through a new modelling process. In other words, inconsistency is something natural in a fuzzy rule base and although it would be desirable to remove it, sometimes we may only be able to reduce it. Moreover, the quality compromising effect of any residual inconsistency on the defuzzified output would possibly be negligible compared to the time consuming effect of a new modelling process which may lead again to an inconsistent rule base.

Algorithm 9.1 describes the aggregation process for inconsistent rules but it does not say when this process can be applied with full success, i.e. without any residual inconsistency being left. In other words, the question is when it would be possible to aggregate all inconsistent rules from each group into a single equivalent rule. This would be possible if the following conditions are fulfilled with respect to the fuzzy membership functions for the output:

- the number of these fuzzy membership functions is odd, i.e. there is a fuzzy membership function in the middle,
- the fuzzy membership function in the middle is symmetrical, i.e. it has an axis of symmetry,
- each of the remaining fuzzy membership functions has a symmetrical image with respect to the axis of symmetry of another symmetrical fuzzy membership function.

The above three conditions guarantee that the aggregation process will lead to a single equivalent rule for each group of inconsistent rules. In this case, the relative contribution of the single equivalent rule in each group to the defuzzified output would be the same as the relative contribution of the associated inconsistent rules in the group. Although the conditions may appear to be restrictive, they are actually not as most fuzzy systems meet these conditions anyway as part of the requirements for spreading the fuzzy membership functions for the output uniformly across its universe of discourse.

The implementation of Algorithm 9.1 can be done easily using Boolean matrices or binary relations, as shown by Algorithms 9.2–9.3.

Algorithm 9.2

1. Go through all the rows of the Boolean matrix from top to bottom, as described by the steps 2-6.
2. Count the number of non-zero elements in the current row.
3. If there is more than one non-zero element in the current row, go to step 4, otherwise move to the next row and go back to step 2.
4. Moving from left to right in the current row, find all non-zero elements in odd columns and aggregate these elements into a single non-zero equivalent element, if possible, or alternatively, into a set of non-zero equivalent elements.
5. Moving from left to right in the current row, find all non-zero elements in even columns and aggregate these elements into a single non-zero equivalent element, if possible, or alternatively, into a set of non-zero equivalent elements.
6. If there are any rows left, move to the next row and go back to step 2, otherwise stop.

Algorithm 9.3

1. Put all maplets from the binary relation into groups whereby the first element in all maplets from each group is the same but also different from the first element in any maplet from any other group.
2. Go through all groups of maplets in an increasing order with respect to the first elements in the associated maplets, as described by the steps 3-7.
3. Count the number of maplets in the current group of maplets.
4. If there is more than one maplet in the current group, go to step 5, otherwise move to the next group of maplets and go back to step 3.
5. Moving from left to right in the current group, find all maplets with an odd second element and aggregate these maplets into a single equivalent maplet, if possible, or alternatively, into a set of equivalent maplets.
6. Moving from left to right in the current group, find all maplets with an even second element and aggregate these maplets into a single equivalent maplet, if possible, or alternatively, into a set of equivalent maplets.
7. If there are any groups left, move to the next group and go back to step 3, otherwise stop.

It is obvious from Algorithms 9.2–9.3 that a single non-zero equivalent element in a Boolean matrix or a single equivalent maplet in a binary relation represent a single equivalent rule in a fuzzy rule base. Likewise, a set of non-zero equivalent elements in a Boolean matrix or a set of equivalent maplets in a binary relation represent a set of equivalent rules in the rule base.

Algorithms 9.2–9.3 show that the process of aggregating inconsistent rules in a fuzzy system is equivalent to representing a one-to-many mapping as a one-to-one mapping. A theoretical justification of this representation based on the laws of Boolean logic is shown below. By definition, each ‘if-then’ rule in a fuzzy rule base is a logical implication whereby if the antecedent and the consequent part in the rule are true then the whole rule must be true.

Therefore, a group of inconsistent rules may be represented in the form

$$\begin{gathered}\text{If } (A_{1s} \text{ and } \ldots \text{ and } A_{ms}) \text{ then } C_{q1} \\ \text{or} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \text{or} \\ \text{If } (A_{1s} \text{ and } \ldots \text{ and } A_{ms}) \text{ then } C_{qz}\end{gathered} \tag{9.7}$$

where $A_{ps} = (i_p \text{ is } v_{ip,s})$, $p = 1,..,m$ and $C_{qz} = (o_q \text{ is } v_{oq,z})$, $q = 1,..,n$ are the logical propositions describing the antecendent terms for the p-th input and the consequent term for the q-th output, respectively. In this case, s is a label for the group being considered whereas z is the number of inconsistent rules in this group.

Equation (9.7) may be rewritten in the following equivalent forms:

$$[(A_{1s} \text{ and } \ldots \text{ and } A_{ms}) \text{ imp } C_{q1}] \text{ or } \ldots \text{ or } [(A_{1s} \text{ and } \ldots \text{ and } A_{ms}) \text{ imp } C_{qz}] \qquad (9.8)$$

$$[\text{not}\,(A_{1s} \text{ and } \ldots \text{ and } A_{ms}) \text{ or } C_{q1}] \text{ or } \ldots \text{ or } [\text{not}\,(A_{1s} \text{ and } \ldots \text{ and } A_{ms}) \text{ or } C_{qz}] \qquad (9.9)$$

$$[\text{not } (A_{1s} \text{ and } \ldots \text{ and } A_{ms})] \text{ or } \ldots \text{ or } [\text{not } (A_{1s} \text{ and } \ldots \text{ and } A_{ms})] \text{ or } (C_{q1} \text{ or } \ldots \text{ or } C_{qz}) \qquad (9.10)$$

$$[\text{not } (A_{1s} \text{ and } \ldots \text{ and } A_{ms})] \text{ or } (C_{q1} \text{ or } \ldots \text{ or } C_{qz}) \qquad (9.11)$$

$$[\text{not } (A_{1s} \text{ and } \ldots \text{ and } A_{ms})] \text{ imp } (C_{q1} \text{ or } \ldots \text{ or } C_{qz}) \qquad (9.12)$$

$$\text{If } (A_{1s} \text{ and } \ldots \text{ and } A_{ms}) \text{ then } (C_{q1} \text{ or } \ldots \text{ or } C_{qz}) \qquad (9.13)$$

So, the one-to-many mapping described by Eq. (9.7) has been represented equivalently as a one-to-one mapping described by Eq. (9.13). In this case, the z simple logical propositions $C_{q1} \ldots C_{qz}$ in the consequent part of the inconsistent rules in Eq. (9.7) have been represented by a single compound proposition (C_{q1} or … or C_{qz}) in the aggregated consequent part of the single equivalent rule in Eq. (9.13).

Example 9.1

A SISO system has the following group of two inconsistent rules

$$\begin{array}{c} \text{If } i_1 \text{ is P then } o_1 \text{ is S} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is B} \end{array} \qquad (9.14)$$

where the simple linguistic terms P, S and B denote the linguistic values *positive*, *small* and *big*, respectively.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is M} \qquad (9.15)$$

in which the simple linguistic term M denoting the linguistic value *medium* has replaced the compound term (S or B).

For clarity, the fuzzy system from Eq. (9.14) will be called ‘conventional’ whereas the fuzzy system from Eq. (9.15) will be referred to as ‘aggregated’. In order to show the equivalence of these two systems, we will need to go for each of them through the implication substage, the aggregation substage and the defuzzification stage. The fuzzification stage and the application substage are assumed to have been done in advance for each of the two systems because they would lead to the same results due to the identical antecendent parts for the input, as shown by Eq. (9.14)–(9.15).

As the antecedent parts of the two rules in the *conventional system* (CS) are identical, the firing strength g_S for the first rule and the firing strength g_B for the second rule in this system are assumed to have been found to be equal to 0.66. Likewise, due to the identity between the antecedent part of the single rule in the *aggregated system* (AS) and the antecedent parts of the two rules in the CS, the firing strength g_M for this single rule must also have been found to be equal to 0.66.

At the implication substage, the fuzzy membership functions F_S and F_B for the output from the CS are obtained as

$$F_S = \{0/0, 0.33/1, 0.66/2, 0.66/3, 0.66/4, 0.33/5, 0/6, 0/7, 0/8, 0/9, 0/10, 0/11, 0/12\} \tag{9.16}$$

$$F_B = \{0/0 , 0/1, 0/2, 0/3, 0/4, 0/5, 0/6, 0.33/7, 0.66/8, 0.66/9, 0.66/10, 0.33/11, 0/12\} \tag{9.17}$$

where F_S and F_B represent the linguistic values S and B, respectively.

Due to the trapezoidal shape F_S and F_B, the associated fuzzy membership degrees f_S and f_B for any element y from the discrete universe of discourse for the output will be mapped by

$$\begin{gathered} f_S = 0,\ if\ y \le a_S \\ f_S = (y - a_S) / (b_S - a_S),\ if\ a_S \le y \le b_S \\ f_S = 0.66,\ if\ b_S \le y \le c_S \\ f_S = (d_S - y) / (d_S - c_S),\ if\ c_S \le y \le d_S \\ f_S = 0,\ if\ d_S \le y \end{gathered} \tag{9.18}$$

$$f_B = 0,\ if\ y \le a_B \tag{9.19}$$
$$f_B = (\ y - a_B\ )\ /\ (\ b_B - a_B),\ if\ a_B \le\ y \le b_B$$
$$f_B = 0.66,\ if\ b_B \le\ y \le c_B$$
$$f_B = (\ d_B - y\ )\ /\ (\ d_B - c_B),\ if\ c_B \le\ y \le d_B$$
$$f_B = 0,\ if\ d_B \le y$$

where the parameters of the membership functions F_S and F_B are the following

$$a_S = 0,\ b_S = 2,\ c_S = 4,\ d_S = 6 \tag{9.20}$$

$$a_B = 6,\ b_B = 8,\ c_B = 10,\ d_B = 12 \tag{9.21}$$

At the aggregation substage, the aggregated fuzzy membership functions F_{SB} for the output from the CS is obtained as follows

$$F_{SB} = F_S \cup F_B = \tag{9.22}$$
$$\{0/0,\ 0.33/1,\ 0.66/2,\ 0.66/3,\ 0.66/4,\ 0.33/5,\ 0/6,$$
$$0.33/7,\ 0.66/8,\ 0.66/9,\ 0.66/10,\ 0.33/11,\ 0/12\}$$

At the defuzzification stage, the defuzzified value D_{SB} for the output from the CS is obtained as follows

$$D_{SB} = [(0\ .\ 0) + (0.33\ .\ 1) + (0.66\ .\ 2) + (0.66\ .\ 3) + (0.66\ .\ 4) \tag{9.23}$$
$$+ (0.33\ .\ 5) + (0\ .\ 6) + (0.33\ .\ 7) + (0.66\ .\ 8)$$
$$+ (0.66\ .\ 9) + (0.66\ .\ 10) + (0.33\ .\ 11) + (0\ .\ 12)]\ /$$
$$(0 + 0.33 + 0.66 + 0.66 + 0.66 + 0.33$$
$$+ 0 + 0.33 + 0.66 + 0.66 + 0.66 + 0.33 + 0\ ) = 32\ /\ 5.33 = 6$$

At the implication substage, the fuzzy membership function F_M for the output from the AS is obtained as

$$F_M = \{0/0,\ 0/1,\ 0/2,\ 0/3,\ 0.33/4,\ 0.66/5,\ 0.66/6,\ 0.66/7,\ 0.33/8,\ 0/9,\ 0/10,\ 0/11,\ 0/12\} \tag{9.24}$$

where F_M represents the linguistic value M.

Due to the trapezoidal shape of F_M, the associated fuzzy membership degree f_M for any element y from the discrete universe of discourse for the output will be mapped by

$$\begin{gathered} f_M = 0,\ if\ y \le a_M \\ f_M = (\ y - a_M\ ) / (\ b_M - a_M),\ if\ a_M \le\ y \le b_M \\ f_M = 0.66,\ if\ b_M \le\ y \le c_M \\ f_M = (\ d_M - y\ ) / (\ d_M - c_M),\ if\ c_M \le\ y \le d_M \\ f_M = 0,\ if\ d_M \le y \end{gathered} \tag{9.25}$$

where the parameters of the membership functions F_M and F_B are the following

$$a_M = 3,\ b_M = 5,\ c_M = 7,\ d_M = 9 \tag{9.26}$$

At the aggregation substage, the aggregated fuzzy membership function for the output from the AS is equal to F_M because there is only one rule in this system.

At the defuzzification stage, the defuzzified value D_M for the output from the AS is obtained as follows

$$\begin{aligned} D_M = {} & [(0\ .\ 0) + (0\ .\ 1) + (0\ .\ 2) + (0\ .\ 3) + (0.33\ .\ 4) + (0.66\ .\ 5) \\ & + (0.66\ .\ 6) + (0.66\ .\ 7) + (0.33\ .\ 8) + (0\ .\ 9) + (0\ .\ 10) \\ & + (0\ .\ 11) + (0\ .\ 12)]\ / \\ & (0 + 0 + 0 + 0 + 0.33 + 0.66 \\ & + 0.66 + 0.66 + 0.33 + 0 + 0 + 0 + 0) = 16 / 2.66 = 6 \end{aligned} \tag{9.27}$$

It follows from Eq. (9.23) and Eq. (9.27) that the defuzzified value D_{SB} for the output from the CS is equal to the defuzzified value D_M for the same output from the AS. This shows that the two systems from Eqs. (9.14)–(9.15) are equivalent in terms of their behaviour. In other words, the inconsistent rule base has been simplified to a consistent rule base by aggregating the inconsistent rules in a way, which removes the inherent redundancy without any effect on the final result.

The considerations above can be generalised easily using Boolean matrices and binary relations. If the input i_1 can take the linguistic values *negative* (N), *zero* (Z) and *positive* (P) and the output o_1 can take the linguistic values *small* (S), *medium* (M) and *big* (B), then we could make the substitutions N = 1, Z = 2 and P = 3 and S = 1, M = 2 and B = 3.

If we assume that there is only one inconsistent group in the CS, its Boolean matrix and binary relation RB_{SB} will look like

$$RB_{SB}: \quad \begin{array}{cccc} i_1/o_1 & 1 & 2 & 3 \\ 1 & ? & ? & ? \\ 2 & ? & ? & ? \\ 3 & 1 & 0 & 1 \end{array} \tag{9.28}$$

$$RB_{SB}: \{?, (3, 1), (3, 3)\} \tag{9.29}$$

where only the elements reflecting the considerations in this example are designated with valid values. The remaining elements are designated with the symbol '?' to avoid confusion.

The Boolean matrix and binary relation RB_M for the consistent AS will look like

$$RB_{M}: \quad \begin{array}{cccc} i_1/o_1 & 1 & 2 & 3 \\ 1 & ? & ? & ? \\ 2 & ? & ? & ? \\ 3 & 0 & 1 & 0 \end{array} \tag{9.30}$$

$$RB_{M}: \{?, (3, 2)\} \tag{9.31}$$

Here again, only the elements reflecting the considerations in this example are designated with valid values whereas the remaining elements are designated with the symbol '?'.

Example 9.2

A SISO system has the following group of three inconsistent rules

$$\begin{array}{c} \text{If } i_1 \text{ is P then } o_1 \text{ is S} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is M} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is B} \end{array} \tag{9.32}$$

where the simple linguistic terms P, S, M and B denote the linguistic values *positive*, *small*, *medium* and *big*, respectively.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is M} \tag{9.33}$$

in which the simple linguistic term M has replaced the compound term (S or M or B).

If we repeat the considerations from Example 9.1, we will see that the defuzzified outputs for the fuzzy rule bases described by Eq. (9.32) and Eq. (9.33) have the same value.

If we assume that there is only one inconsistent group in the CS, its Boolean matrix and binary relation RB_{SMB} will look like

$$RB_{SMB}: \begin{array}{cccc} i_1/o_1 & 1 & 2 & 3 \\ 1 & ? & ? & ? \\ 2 & ? & ? & ? \\ 3 & 1 & 1 & 1 \end{array} \tag{9.34}$$

$$RB_{SMB}: \{?, (3, 1), (3, 2), (3, 3)\} \tag{9.35}$$

The Boolean matrix and binary relation RB_M for the consistent AS will look like

$$RB_{M}: \begin{array}{cccc} i_1/o_1 & 1 & 2 & 3 \\ 1 & ? & ? & ? \\ 2 & ? & ? & ? \\ 3 & 0 & 1 & 0 \end{array} \tag{9.36}$$

$$RB_{M}: \{?, (3, 2)\} \tag{9.37}$$

Example 9.3

A SISO system with the set of fuzzy output membership functions {VS, S, M, B, VB} has the following group of inconsistent rules

$$\begin{array}{c} \text{If } i_1 \text{ is P then } o_1 \text{ is VS} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is M} \end{array} \tag{9.38}$$

where the simple linguistic terms P, VS, S, M, B and VB denote the linguistic values *positive*, *very small*, *small*, *medium*, *big* and *very big*, respectively.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is S} \qquad (9.39)$$

in which the simple linguistic term S has replaced the compound term (VS or M).

Example 9.4

A SISO system with the set of fuzzy output membership functions {VS, S, M, B, VB} has the following group of inconsistent rules

$$\begin{array}{c} \text{If } i_1 \text{ is P then } o_1 \text{ is VS} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is S} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is M} \end{array} \qquad (9.40)$$

where the simple linguistic terms P, VS, S, M, B and VB denote the same linguistic values as the ones in Example 9.3.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is S} \qquad (9.41)$$

in which the simple linguistic term S has replaced the compound term (VS or S or M).

Example 9.5

A SISO system with the set of fuzzy output membership functions {VS, S, M, B, VB} has the following group of inconsistent rules

$$\begin{array}{c} \text{If } i_1 \text{ is P then } o_1 \text{ is M} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is VB} \end{array} \qquad (9.42)$$

where the simple linguistic terms P, VS, S, M, B and VB denote the same linguistic values as the ones in Example 9.3.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is B} \tag{9.43}$$

in which the simple linguistic term B has replaced the compound term (M or VB).

Example 9.6

A SISO system with the set of fuzzy output membership functions {VS, S, M, B, VB} has the following group of inconsistent rules

$$\begin{gathered} \text{If } i_1 \text{ is P then } o_1 \text{ is M} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is B} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is VB} \end{gathered} \tag{9.44}$$

where the simple linguistic terms P, VS, S, M, B and VB denote the same linguistic values as the ones in Example 9.3.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is B} \tag{9.45}$$

in which the simple linguistic term B has replaced the compound term (M or B or VB).

Example 9.7

A SISO system with the set of fuzzy output membership functions {VS, S, M, B, VB} has the following group of inconsistent rules

$$\begin{gathered} \text{If } i_1 \text{ is P then } o_1 \text{ is S} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is B} \end{gathered} \tag{9.46}$$

where the simple linguistic terms P, VS, S, M, B and VB denote the same linguistic values as the ones in Example 9.3.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is M} \tag{9.47}$$

in which the simple linguistic term M has replaced the compound term (S or B).

Example 9.8

A SISO system with the set of fuzzy output membership functions {VS, S, M, B, VB} has the following group of inconsistent rules

$$\begin{array}{c} \text{If } i_1 \text{ is P then } o_1 \text{ is S} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is M} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is B} \end{array} \tag{9.48}$$

where the simple linguistic terms P, VS, S, M, B and VB denote the same linguistic values as the ones in Example 9.3.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is M} \tag{9.49}$$

in which the simple linguistic term M has replaced the compound term (S or M or B).

Example 9.9

A SISO system with the set of fuzzy output membership functions {VS, S, M, B, VB} has the following group of inconsistent rules

$$\begin{array}{c} \text{If } i_1 \text{ is P then } o_1 \text{ is VS} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is VB} \end{array} \tag{9.50}$$

where the simple linguistic terms P, VS, S, M, B and VB denote the same linguistic values as the ones in Example 9.3.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is M} \tag{9.51}$$

in which the simple linguistic term M has replaced the compound term (VS or VB).

Example 9.10

A SISO system with the set of fuzzy output membership functions {VS, S, M, B, VB} has the following group of inconsistent rules

$$\begin{gathered}\text{If } i_1 \text{ is P then } o_1 \text{ is VS}\\ \text{or}\\ \text{If } i_1 \text{ is P then } o_1 \text{ is M}\\ \text{or}\\ \text{If } i_1 \text{ is P then } o_1 \text{ is VB}\end{gathered} \tag{9.52}$$

where the simple linguistic terms P, VS, S, M, B and VB denote the same linguistic values as the ones in Example 9.3.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is M} \tag{9.53}$$

in which the simple linguistic term M has replaced the compound term (VS or M or VB).

Example 9.11

A SISO system with the set of fuzzy output membership functions {VS, S, M, B, VB} has the following group of inconsistent rules

$$\begin{gathered}\text{If } i_1 \text{ is P then } o_1 \text{ is VS}\\ \text{or}\\ \text{If } i_1 \text{ is P then } o_1 \text{ is S}\\ \text{or}\\ \text{If } i_1 \text{ is P then } o_1 \text{ is B}\\ \text{or}\\ \text{If } i_1 \text{ is P then } o_1 \text{ is VB}\end{gathered} \tag{9.54}$$

where the simple linguistic terms P, VS, S, M, B and VB denote the same linguistic values as the ones in Example 9.3.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is M} \tag{9.55}$$

in which the simple linguistic term M has replaced the compound term (VS or S or B or VB).

Example 9.12

A SISO system with the set of fuzzy output membership functions {VS, S, M, B, VB} has the following group of inconsistent rules

$$\begin{array}{c} \text{If } i_1 \text{ is P then } o_1 \text{ is VS} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is S} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is M} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is B} \\ \text{or} \\ \text{If } i_1 \text{ is P then } o_1 \text{ is VB} \end{array} \tag{9.56}$$

where the simple linguistic terms P, VS, S, M, B and VB denote the same linguistic values as the ones in Example 9.3.

In accordance with Eqs. (9.7)–(9.13), this system can be represented with the single equivalent rule

$$\text{If } i_1 \text{ is P then } o_1 \text{ is M} \tag{9.57}$$

in which the simple linguistic term M has replaced the compound term (VS or S or M or B or VB).

9.3 Rule Base Simplification by Filtration of Non-monotonic Rules

The formal simplification technique introduced here is based on the idea of removing the inherent redundancy in a non-monotonic fuzzy rule base. This type of redundancy is expressed by the presence of non-monotonic rules and it can be removed by filtering such rules with the aim of making the rule base monotonic.

The overall process of filtering non-monotonic rules is illustrated by the following algorithm:

Algorithm 9.4

1. Put all non-monotonic rules in groups sorted in an increasing order with respect to the permutations of linguistic values of outputs, whereby the rules in each group have the same permutation of linguistic values of outputs and different permutations of linguistic values of inputs.

2. For each group of rules, find a single equivalent rule whose effect on the defuzzified output is the same as the effect of all rules.
3. For each group of rules, keep the single equivalent rule and remove all other rules.

Algorithm 9.4 guarantees that there will be only monotonic rules left in a fuzzy rule base after the completion of the filtering process. In this case, the number of monotonic rules is equal to the number of groups and the number of different permutations of linguistic values of outputs. Therefore, the filtering process can be always applied with full success, i.e. without any residual non-monotonousness being left.

As opposed to the aggregation process for inconsistent rules which can be carried out entirely off-line, i.e. before the fuzzification of inputs, the filtering process for non-monotonic rules must be carried out partially on-line, i.e. after the fuzzification of inputs. In this case, step 1 in Algorithm 9.4 can still be applied off-line but steps 2-3 can only be applied on-line. This is due to the fact that the single equivalent rule in Algorithm 9.4 is like a dominant rule which can be found only after the completion of the fuzzification stage and the application substage. This dominancy is expressed in terms of the rule with the maximal firing strength for each group as a result of which the effect of all other rules from the group on the defuzzified output will be completely neutralised. Obviolusly, when there is more than one dominant rule in a group, i.e. two or more rules with maximal firing strength for the group, one of these rules should be selected arbitrarily as a single equivalent rule.

The implementation of Algorithm 9.4 can be done easily using Boolean matrices or binary relations, as shown by Algorithms 9.5–9.6.

Algorithm 9.5

1. Sort the rows of the Boolean matrix in groups such that the rows in each group have a non-zero element in the same column and this column is to the left of any other columns with non-zero elements from any subsequent groups.
2. Filter the rows in each group such that only the row representing the dominant rule is left in the group.

Algorithm 9.6

1. Sort the maplets from the binary relation in groups such that the second elements in the maplets from each group are the same with respect to each other and smaller than the second elements in the maplets from any subsequent groups.
2. Filter the maplets in each group such that only the maplet representing the dominant rule is left in the group.

Algorithms 9.5–9.6 show that the process of filtering non-monotonic rules in a fuzzy system is equivalent to representing a many-to-one mapping as a one-to-one mapping. A theoretical justification of this representation based on the laws of Boolean logic is shown below. Here again, each 'if-then' rule in a fuzzy rule base is a logical implication whereby if the antecedent and the consequent part in the rule are true then the whole rule must be true.

Therefore, a group of non-monotonic rules may be represented in the form

$$\begin{array}{c} \text{If } (A_{11} \text{ and } \ldots \text{ and } A_{m1}) \text{ then } C_q \\ \text{or} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \text{or} \\ \text{If } (A_{1z} \text{ and } \ldots \text{ and } A_{mz}) \text{ then } C_q \end{array} \tag{9.58}$$

where $A_{pj} = (i_p \text{ is } v_{ip,j})$, $p = 1,..,m$, $j = 1,..,z$ and $C_q = (o_q \text{ is } v_{oq})$, $q = 1,..,n$ are the logical propositions describing the antecendent terms for the p-th input in the j-th rule and the consequent term for the q-th output, respectively. In this case, q is also a label for the group being considered whereas z is the number of non-monotonic rules in this group.

Equation (9.58) may be rewritten in the following equivalent forms:

$$[(A_{11} \text{ and } \ldots \text{ and } A_{m1}) \text{ imp } C_q] \text{ or } \ldots \text{ or } [(A_{1z} \text{ and } \ldots \text{ and } A_{mz}) \text{ imp } C_q] \tag{9.59}$$

$$[\text{not}(A_{11} \text{ and } \ldots \text{ and } A_{m1}) \text{ or } C_q] \text{ or } \ldots \text{ or } [\text{not}(A_{11} \text{ and } \ldots \text{ and } A_{m1}) \text{ or } C_q] \tag{9.60}$$

$$[\text{not } (A_{11} \text{ and } \ldots \text{ and } A_{m1})] \text{ or } \ldots \text{ or } [\text{not } (A_{1z} \text{ and } \ldots \text{ and } A_{mz})] \text{ or } (C_q \text{ or } \ldots \text{ or } C_q) \tag{9.61}$$

$$\text{not } [(A_{11} \text{ and } \ldots \text{ and } A_{m1})] \text{ and } \ldots \text{ and } (A_{1z} \text{ and } \ldots \text{ and } A_{mz})] \text{ or } C_q \tag{9.62}$$

$$[(A_{11} \text{ and } \ldots \text{ and } A_{m1})] \text{ and } \ldots \text{ and } (A_{1z} \text{ and } \ldots \text{ and } A_{mz})] \text{ imp } C_q \tag{9.63}$$

$$\text{If } [(A_{11} \text{ and } \ldots \text{ and } A_{m1})] \text{ and } \ldots \text{ and } (A_{1z} \text{ and } \ldots \text{ and } A_{mz})] \text{ then } C_q \tag{9.64}$$

So, the many-to-one mapping described by Eq. (9.58) has been represented equivalently as a one-to-one mapping described by Eq. (9.64). In this case, the z compound logical propositions $(A_{11}$ and … and $A_{m1})$ … $(A_{1z}$ and … and $A_{mz})$ in the antecedent part of the non-monotonic rules in Eq. (9.58) have been represented by a single compound proposition $[(A_{11}$ and … and $A_{m1})$ and … and $(A_{1z}$ and … and $A_{mz})]$ in the filtered antecedent part of the single equivalent rule in Eq. (9.64).

Example 9.13

A fuzzy system for aircraft landing control is described by the inputs i_1, i_2 and the output o_1 where i_1 is the *relative height* (h) of the aircraft in *feet* (ft), i_2 is the *vertical velocity* (v) of the aircraft in *feet per second* (ft/s) and o_1 is the *control effort* (e) in libras (lb) that must be applied to the aircraft [66]. In this case, i_1 can take the four linguistic values *near zero* (NZ), *small* (S), *medium* (M) and *large* (L), whereas both i_2 and o_1 can take the five linguistic values *down large* (DL), *down small* (DS), *zero* (Z), *up small* (US) and *up large* (UL).

By making the substitutions NZ = 1, S = 2, M = 3, L = 4 for i_1 as well as the substitutions DL = 1, DS = 2, Z = 3, US = 4, UL = 5 for both i_1 and o_1, we can construct the integer table for the fuzzy rule base of this CS, as shown in Table 9.1. Then, by applying step 1 from Algorithm 9.4, we can construct the integer table for the rule base of the *sorted system* (SS), as shown in Table 9.2. The empty rows in these tables are used only for visual separation of the rules in different groups, which facilitates the analysis of the contents of the tables.

Table 9.1. Integer table for the rule base of the conventional system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of o_1
1	1	1	5
2	1	2	5
3	1	3	3
4	1	4	2
5	1	5	2
6	2	1	5
7	2	2	4
8	2	3	3
9	2	4	2
10	2	5	1
11	3	1	4
12	3	2	3
13	3	3	2
14	3	4	1
15	3	5	1
16	4	1	3
17	4	2	2
18	4	3	1
19	4	4	1
20	4	5	1

Table 9.2. Integer table for the rule base of the sorted system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of o_1
10	2	5	1
14	3	4	1
15	3	5	1
18	4	3	1
19	4	4	1
20	4	5	1
4	1	4	2
5	1	5	2
9	2	4	2
13	3	3	2
17	4	2	2
3	1	3	3
8	2	3	3
12	3	2	3
16	4	1	3
7	2	2	4
11	3	1	4
1	1	1	5
2	1	2	5
6	2	1	5

By applying steps 2-3 from Algorithm 9.4, we can now construct the integer table for the fuzzy rule base of the *filtered system* (FS), as shown in Table 9.3.

The integer table for the fuzzy rule base of the FS contains only the single equivalent rule from each of the sorted five groups of rules. The process leading to these single equivalent rules is described further in the current section and therefore the contents of this integer table must be taken for granted at this stage.

Table 9.3. Integer table for the rule base of the filtered system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of o_1
10	2	5	1
17	4	2	2
12	3	2	3
11	3	1	4
1	1	1	5

Algorithm 9.4 can be implemented much easier using Boolean matrices or binary relations, as described by Algorithms 9.5–9.6. This is shown briefly by Eqs. (9.65)–(9.70) which are associated with the integer tables from Tables 9.1–9.3. In particular, Eqs. (9.65)–(9.66) relate to Table 9.1, Eqs. (9.67)–(9.68) relate to Table 9.2, whereas Eqs. (9.69)–(9.70) relate to Table 9.3.

The Boolean matrix and the binary relation for the fuzzy rule base RB_{CS} of the CS are given by

$$
RB_{CS}: \quad \begin{array}{c|ccccc}
i_1\, i_2 / o_1 & 1 & 2 & 3 & 4 & 5 \\
11 & 0 & 0 & 0 & 0 & 1 \\
12 & 0 & 0 & 0 & 0 & 1 \\
13 & 0 & 0 & 1 & 0 & 0 \\
14 & 0 & 1 & 0 & 0 & 0 \\
15 & 0 & 1 & 0 & 0 & 0 \\
 & & & & & \\
21 & 0 & 0 & 0 & 0 & 1 \\
22 & 0 & 0 & 0 & 1 & 0 \\
23 & 0 & 0 & 1 & 0 & 0 \\
24 & 0 & 1 & 0 & 0 & 0 \\
25 & 1 & 0 & 0 & 0 & 0 \\
 & & & & & \\
31 & 0 & 0 & 0 & 1 & 0 \\
32 & 0 & 0 & 1 & 0 & 0 \\
33 & 0 & 1 & 0 & 0 & 0 \\
34 & 1 & 0 & 0 & 0 & 0 \\
35 & 1 & 0 & 0 & 0 & 0 \\
 & & & & & \\
41 & 0 & 0 & 1 & 0 & 0 \\
42 & 0 & 1 & 0 & 0 & 0 \\
43 & 1 & 0 & 0 & 0 & 0 \\
44 & 1 & 0 & 0 & 0 & 0 \\
45 & 1 & 0 & 0 & 0 & 0
\end{array} \tag{9.65}
$$

$$
\begin{aligned}
RB_{CS}: \{&(11, 5), (12, 5), (13, 3), (14, 2), (15, 2), \\
&(21, 5), (22, 4), (23, 3), (24, 2), (25, 1), \\
&(31, 4), (32, 3), (33, 2), (34, 1), (35, 1), \\
&(41, 3), (42, 2), (43, 1), (44, 1), (45, 1)\}
\end{aligned} \tag{9.66}
$$

The Boolean matrix and the binary relation for the fuzzy rule base RB_{SS} of the SS are given by

RB_{SS}: (9.67)

$i_1\, i_2 / o_1$	1	2	3	4	5
25	1	0	0	0	0
34	1	0	0	0	0
35	1	0	0	0	0
43	1	0	0	0	0
44	1	0	0	0	0
45	1	0	0	0	0
14	0	1	0	0	0
15	0	1	0	0	0
24	0	1	0	0	0
33	0	1	0	0	0
42	0	1	0	0	0
13	0	0	1	0	0
23	0	0	1	0	0
32	0	0	1	0	0
41	0	0	1	0	0
22	0	0	0	1	0
31	0	0	0	1	0
11	0	0	0	0	1
12	0	0	0	0	1
21	0	0	0	0	1

$$RB_{SS}: \{(25, 1), (34, 1), (35, 1), (43, 1), (44, 1), (45, 1), \\ (14, 2), (15, 2), (24, 2), (33, 2), (42, 2), \\ (13, 3), (23, 3), (32, 3), (41, 3), \\ (22, 4), (31, 4), \\ (11, 5), (12, 5), (21, 5)\} \quad (9.68)$$

The Boolean matrix and the binary relation for the fuzzy rule base RB_{FS} of the FS are given by

RB_{FS}: (9.69)

$i_1\, i_2 / o_1$	1	2	3	4	5
25	1	0	0	0	0
42	0	1	0	0	0
32	0	0	1	0	0
31	0	0	0	1	0
11	0	0	0	0	1

$$RB_{FS}\text{: } \{(25, 1), (42, 2), (32, 3), (31, 4), (11, 5)\} \tag{9.70}$$

The next steps in the current example consider the fuzzification, inference and defuzzification stages for the CS and the FS. The aim of this consideration is to show the behavioural equivalence of the two systems, i.e. that the defuzzified output is the same for any crisp values of the inputs. In the current example, these values are taken as h = 980 and v = –14.2. Therefore, when we have to deal with an arbitrarily complex CS represented by an integer table, Boolean matrix or binary relation, we can formally simplify this system to a fairly simple FS and use the associated integer table, Boolean matrix or binary relation instead.

At the fuzzification stage for the CS, we have to consider all possible linguistic values for each input. In this context, we need to specify how the fuzzy membership degree for a particular linguistic value of a given input can be obtained. This specification is based on the parameters of the fuzzy membership functions for the fuzzification of the inputs, as shown further below.

The fuzzy membership degree f_h^{NZ} for the linguistic value *near zero* of the input *height* can be obtained by the formula

$$\begin{gathered}
f_h^{NZ} = 0,\ if\, h < a_h^{NZ} \\
f_h^{NZ} = 1,\ if\, a_h^{NZ} \le h \le b_h^{NZ} \\
f_h^{NZ} = (\,c_h^{NZ} - h\,)\,/\,(\,c_h^{NZ} - b_h^{NZ}\,),\ if\, b_h^{NZ} \le h \le c_h^{NZ} \\
f_h^{NZ} = 0,\ if\, c_h^{NZ} \le h
\end{gathered} \tag{9.71}$$

where a_h^{NZ}, b_h^{NZ}, c_h^{NZ} are the parameters of the associated triangular fuzzy membership function whose values are given by

$$a_h^{NZ} = 0,\ b_h^{NZ} = 0,\ c_h^{NZ} = 500 \tag{9.72}$$

Equation (9.71) differs slightly from Eq. (9.1). In particular, the inequality sign in the first line of Eq. (9.1) has been strengthened in Eq. (9.71) to account for the vertical left shoulder of the membership function. Also, the arithmetic division in the second line of Eq. (9.1) has been removed from Eq. (9.71) to avoid a division by zero due to the equality of the parameters a_h^{NZ} and b_h^{NZ}.

The fuzzy membership degree f_h^S for the linguistic value *small* of the input *height* can be obtained by the formula

$$
\begin{gathered}
f_h^S = 0,\ if\ h \le a_h^S \\
f_h^S = (h - a_h^S)/(b_h^S - a_h^S),\ if\ a_h^S \le h \le b_h^S \\
f_h^S = (c_h^S - h)/(c_h^S - b_h^S),\ if\ b_h^S \le h \le c_h^S \\
f_h^S = 0,\ if\ c_h^S \le h
\end{gathered}
\tag{9.73}
$$

where a_h^S, b_h^S, c_h^S are the parameters of the associated triangular fuzzy membership function whose values are given by

$$a_h^S = -200,\ b_h^S = 300,\ c_h^S = 800 \tag{9.74}$$

The fuzzy membership degree f_h^M for the linguistic value *medium* of the input *height* can be obtained by the formula

$$
\begin{gathered}
f_h^M = 0,\ if\ h \le a_h^M \\
f_h^M = (h - a_h^M)/(b_h^M - a_h^M),\ if\ a_h^M \le h \le b_h^M \\
f_h^M = (c_h^M - h)/(c_h^M - b_h^M),\ if\ b_h^M \le h \le c_h^M \\
f_h^M = 0,\ if\ c_h^M \le h
\end{gathered}
\tag{9.75}
$$

where a_h^M, b_h^M, c_h^M are the parameters of the associated triangular fuzzy membership function whose values are given by

$$a_h^M = 300,\ b_h^M = 800,\ c_h^M = 1300 \tag{9.76}$$

The fuzzy membership degree f_h^L for the linguistic value *large* of the input *height* can be obtained by the formula

$$
\begin{gathered}
f_h^L = 0,\ if\ h \le a_h^L \\
f_h^L = (h - a_h^L)/(b_h^L - a_h^L),\ if\ a_h^L \le h \le b_h^L \\
f_h^L = 1,\ if\ b_h^L \le h \le c_h^L \\
f_h^L = 0,\ if\ c_h^L < h
\end{gathered}
\tag{9.77}
$$

where a_h^L, b_h^L, c_h^L are the parameters of the associated triangular fuzzy membership function whose values are given by

$$a_h^L = 500,\ b_h^L = 1000,\ c_h^L = 1000 \tag{9.78}$$

Equation (9.77) differs slightly from Eq. (9.1). In particular, the arithmetic division in the third line of Eq. (9.1) has been removed from Eq. (9.77) to avoid a division by zero due to the equality of the parameters b_h^L and c_h^L. Also, the inequality sign in the fourth line of Eq. (9.1) has been strengthened in Eq. (9.77) to account for the vertical right shoulder of the membership function.

The parameters of the fuzzy membership functions for the first input to the fuzzy system are summarised in Table 9.4.

Table 9.4. Fuzzy membership function parameters for the first input

Linguistic value / Input	Relative height
Near zero	[0 0 500]
Small	[-200 300 800]
Medium	[300 800 1300]
Large	[500 1000 1000]

The fuzzy membership degree f_v^{DL} for the linguistic value *down large* of the input *velocity* can be obtained by the formula

$$\begin{gathered} f_v^{DL} = 0,\ if\ v < a_v^{DL} \\ f_v^{DL} = 1,\ if\ a_v^{DL} \leq v \leq b_v^{DL} \\ f_v^{DL} = 1,\ if\ b_v^{DL} \leq v \leq c_v^{DL} \\ f_v^{DL} = (d_v^{DL} - v) / (d_v^{DL} - c_v^{DL}),\ if\ c_v^{DL} \leq v \leq d_v^{DL} \\ f_v^{DL} = 0,\ if\ d_v^{DL} \leq v \end{gathered} \tag{9.79}$$

where a_v , b_v^{DL}, c_v^{DL}, d_v^{DL} are the parameters of the associated trapezoidal fuzzy membership function whose values are given by

$$a_v^{DL} = -30,\ b_v^{DL} = -30,\ c_v^{DL} = -20,\ d_v^{DL} = -10 \tag{9.80}$$

Equation (9.79) differs slightly from the standard formula for a trapezoidal fuzzy membership. In particular, the inequality sign in the first line of the standard formula has been strengthened in Eq. (9.79) to account for the vertical left shoulder of the membership function. Also, the arithmetic division in the second line of the standard formula has been removed from Eq. (9.79) to avoid a division by zero due to the equality of the parameters a_v^{DL} and b_v^{DL}.

The fuzzy membership degree f_v^{DS} for the linguistic value *down small* of the input *velocity* can be obtained by the formula

$$\begin{gathered} f_v^{DS} = 0,\ if\ v \le a_v^{DS} \\ f_v^{DS} = (\ v - a_v^{DS}\ )\ /\ (\ b_v^{DS} - a_v^{DS}\ ),\ \ if\ a_v^{DS} \le\ v \le b_v^{DS} \\ f_v^{DS} = (\ c_v^{DS} - v\ )\ /\ (\ c_v^{DS} - b_v^{DS}\ ),\ if\ b_v^{DS} \le\ v \le c_v^{DS} \\ f_v^{DS} = 0,\ if\ c_v^{DS} \le v \end{gathered} \tag{9.81}$$

where a_v^{DS}, b_v^{DS}, c_v^{DS} are the parameters of the associated triangular fuzzy membership function whose values are given by

$$a_v^{DS} = -20,\ b_v^{DS} = -10,\ c_v^{DS} = 0 \tag{9.82}$$

The fuzzy membership degree f_v^{Z} for the linguistic value *zero* of the input *velocity* can be obtained by the formula

$$\begin{gathered} f_v^{Z} = 0,\ if\ v \le a_v^{Z} \\ f_v^{Z} = (\ v - a_v^{Z}\ )\ /\ (\ b_v^{Z} - a_v^{Z}\ ),\ \ if\ a_v^{Z} \le\ v \le b_v^{Z} \\ f_v^{Z} = (\ c_v^{Z} - v\ )\ /\ (\ c_v^{Z} - b_v^{Z}\ ),\ if\ b_v^{Z} \le\ v \le c_v^{Z} \\ f_v^{Z} = 0,\ if\ c_v^{Z} \le v \end{gathered} \tag{9.83}$$

where a_v^{Z}, b_v^{Z}, c_v^{Z} are the parameters of the associated triangular fuzzy membership function whose values are given by

$$a_v^{Z} = -10,\ b_v^{Z} = 0,\ c_v^{Z} = 10 \tag{9.84}$$

The fuzzy membership degree f_v^{US} for the linguistic value *up small* of the input *velocity* can be obtained by the formula

$$\begin{gathered} f_v^{US} = 0,\ if\ v \le a_v^{US} \\ f_v^{US} = (\ v - a_v^{US}\ )\ /\ (\ b_v^{US} - a_v^{US}\ ),\ \ if\ a_v^{US} \le\ v \le b_v^{US} \\ f_v^{US} = (\ c_v^{US} - v\ )\ /\ (\ c_v^{US} - b_v^{US}\ ),\ if\ b_v^{US} \le\ v \le c_v^{US} \\ f_v^{US} = 0,\ if\ c_v^{US} \le v \end{gathered} \tag{9.85}$$

where a_v^{US}, b_v^{US}, c_v^{US} are the parameters of the associated triangular fuzzy membership function whose values are given by

$$a_v^{US} = 0,\ b_v^{US} = 10,\ c_v^{US} = 20 \tag{9.86}$$

The fuzzy membership degree f_v^{UL} for the linguistic value *up large* of the input *velocity* can be obtained by the formula

$$\begin{gathered} f_v^{UL} = 0,\ if\ v \leq a_v^{UL} \\ f_v^{UL} = (\ v - a_v^{UL}\ )\ /\ (\ b_v^{UL} - a_v^{UL}\ ),\ if\ a_v^{UL} \leq\ v \leq b_v^{UL} \\ f_v^{UL} = 1,\ if\ b_v^{UL} \leq\ v \leq c_v^{UL} \\ f_v^{UL} = 1,\ if\ c_v^{UL} \leq\ v \leq d_v^{UL} \\ f_v^{UL} = 0,\ if\ d_v^{UL} < v \end{gathered} \tag{9.87}$$

where a_v^{UL}, b_v^{UL}, c_v^{UL}, d_v^{UL} are the parameters of the associated trapezoidal fuzzy membership function whose values are given by

$$a_v^{UL} = 10,\ b_v^{UL} = 20,\ c_v^{UL} = 30,\ d_v^{UL} = 30 \tag{9.88}$$

Equation (9.87) differs slightly from the standard formula for a trapezoidal fuzzy membership function. In particular, the arithmetic division in the fourth line of the standard formula has been removed from Eq. (9.87) to avoid a division by zero due to the equality of the parameters c_v^{UL} and d_v^{UL}. Also, the inequality sign in the fifth line of the standard formula has been strengthened in Eq. (9.87) to account for the vertical right shoulder of the membership function.

The parameters of the fuzzy membership functions for the second input to the fuzzy system are summarised in Table 9.5.

Table 9.5. Fuzzy membership function parameters for the second input

Linguistic value / Input	Vertical velocity
Down large	[-30 -30 -20 -10]
Down small	[-20 -10 0]
Zero	[-10 0 10]
Up small	[0 10 20]
Up large	[10 20 30 30]

At the application substage of the inference stage for the CS, we have to find the firing strength for each rule. For this purpose, we assume to have converted the crisp values of the inputs into fuzzy membership degrees during the fuzzification stage. The result of this conversion is used in the application substage which is applied to each rule, as shown by Eqs. (9.89)–(9.108).

$$g_1^{UL} = \min\,(f_h^{NZ}, f_v^{DL}) = \min\,(0, 0) = 0 \tag{9.89}$$

$$g_2^{UL} = \min\,(f_h^{NZ}, f_v^{DS}) = \min\,(0, 0) = 0 \tag{9.90}$$

$$g_3^{Z} = \min\,(f_h^{NZ}, f_v^{Z}) = \min\,(0, 0) = 0 \tag{9.91}$$

$$g_4^{DS} = \min\,(f_h^{NZ}, f_v^{US}) = \min\,(0, 0) = 0 \tag{9.92}$$

$$g_5^{DS} = \min\,(f_h^{NZ}, f_v^{UL}) = \min\,(0, 0) = 0 \tag{9.93}$$

$$g_6^{UL} = \min\,(f_h^{S}, f_v^{DL}) = \min\,(0, 0) = 0 \tag{9.94}$$

$$g_7^{US} = \min\,(f_h^{S}, f_v^{DS}) = \min\,(0, 0) = 0 \tag{9.95}$$

$$g_8^{Z} = \min\,(f_h^{S}, f_v^{Z}) = \min\,(0, 0) = 0 \tag{9.96}$$

$$g_9^{DS} = \min\,(f_h^{S}, f_v^{US}) = \min\,(0, 0) = 0 \tag{9.97}$$

$$g_{10}^{DL} = \min\,(f_h^{S}, f_v^{UL}) = \min\,(0, 0) = 0 \tag{9.98}$$

$$g_{11}^{US} = \min\,(f_h^{M}, f_v^{DL}) = \min\,(0.64, 0.42) = 0.42 \tag{9.99}$$

$$g_{12}^{Z} = \min\,(f_h^{M}, f_v^{DS}) = \min\,(0.64, 0.58) = 0.58 \tag{9.100}$$

$$g_{13}^{DS} = \min\,(f_h^{M}, f_v^{Z}) = \min\,(0, 0) = 0 \tag{9.101}$$

$$g_{14}^{DL} = \min\,(f_h^{M}, f_v^{US}) = \min\,(0, 0) = 0 \tag{9.102}$$

$$g_{15}^{DL} = \min\,(f_h^{M}, f_v^{UL}) = \min\,(0, 0) = 0 \tag{9.103}$$

$$g_{16}^{Z} = \min\,(f_h^{L}, f_v^{DL}) = \min\,(0.96, 0.42) = 0.42 \tag{9.104}$$

$$g_{17}^{DS} = \min\,(f_h^{L}, f_v^{DS}) = \min\,(0.96, 0.58) = 0.58 \tag{9.105}$$

$$g_{18}^{DL} = \min\,(f_h^{L}, f_v^{Z}) = \min\,(0, 0) = 0 \tag{9.106}$$

$$g_{19}^{DL} = \min\,(f_h^{L}, f_v^{US}) = \min\,(0, 0) = 0 \tag{9.107}$$

$$g_{20}^{DL} = \min\,(f_h^{L},\ f_v^{UL}) = \min\,(0, 0) = 0 \tag{9.108}$$

It is obvious that only four rules have a firing strength greater than zero for the considered crisp values of the inputs. These rules are 11, 12, 16 and 17, as shown by Eq. (9.99), Eq. (9.100), Eq. (9.104) and Eq. (9.105), respectively. In this context, Eq. (9.99) shows that in rule 11 the linguistic value *medium* of the input *height* contributes with a fuzzy membership degree of 0.64 to the linguistic value *up small* of the output whereas the linguistic value *down large* of the input *velocity* contributes with a fuzzy membership degree of 0.42 to the same linguistic value of the output. Similar considerations apply to the other rules.

At the implication substage of the inference stage for the CS, we have to find the truncated fuzzy membership function for the output in each rule. For this purpose, we assume to have represented the output with fuzzy membership functions, which happen to be the same as the ones for the second input. The only difference is in the physical meaning of the universe of discourse, which is not continuous but discrete and represents not *vertical velocity* but *control effort*. The implication substage is described by Eqs. (9.109)–(9.123).

The fuzzy membership degree $f_{e,s}^{DL}$, $s = 10,14,15,18,19,20$ for the linguistic value *down large* of the output *effort* in the *s*-th rule can be obtained by the formula

$$\begin{gathered}
f_{e,s}^{DL} = 0,\ if\ e < a_{e,s}^{DL} \\
f_{e,s}^{DL} = g_s^{DL},\ if\ a_{e,s}^{DL} \le e \le b_{e,s}^{DL} \\
f_{e,s}^{DL} = g_s^{DL},\ if\ b_{e,s}^{DL} \le e \le c_{e,s}^{DL} \\
f_{e,s}^{DL} = (\,d_{e,s}^{DL} - e\,)\,/\,(\,d_{e,s}^{DL} - c_{e,s}^{DL}\,),\ if\ c_{e,s}^{DL} \le e \le d_{e,s}^{DL} \\
f_{e,s}^{DL} = 0,\ if\ d_{e,s}^{DL} \le e
\end{gathered} \tag{9.109}$$

where $a_{e,s}^{DL}$, $b_{e,s}^{DL}$, $c_{e,s}^{DL}$, $d_{e,s}^{DL}$, $s = 10,14,15,18,19,20$ are the parameters of the truncated trapezoidal fuzzy membership function for the output in the *s*-th rule and g_s^{DL}, $s = 10,14,15,18,19,20$ is firing strength of the *s*-th rule. The values of these parameters can be obtained by the formula

$$\begin{gathered}
a_{e,s}^{DL} = a_e^{DL} \\
b_{e,s}^{DL} = a_e^{DL} + (\,b_e^{DL} - a_e^{DL}\,)\,.\,g_s^{DL} \\
c_{e,s}^{DL} = d_e^{DL} - (\,d_e^{DL} - c_e^{DL}\,)\,.\,g_s^{DL} \\
d_{e,s}^{DL} = d_e^{DL}
\end{gathered} \tag{9.110}$$

where a_e^{DL}, b_e^{DL}, c_e^{DL}, d_e^{DL} are the parameters of the original trapezoidal fuzzy membership function for the linguistic value *down large* of the output. The values of these parameters are given by

$$a_e^{DL} = -30,\ b_e^{DL} = -30,\ c_e^{DL} = -20,\ d_e^{DL} = -10 \tag{9.111}$$

Equation (9.109) differs slightly from Eq. (9.4). In particular, the inequality sign in the first line of Eq. (9.4) has been strengthened in Eq. (9.109) to account for the vertical left shoulder of the truncated membership function. Also, the arithmetic division in the second line of Eq. (9.4) has been removed from Eq. (9.109) to avoid a division by zero due to the equality of the parameters $a_{e,s}^{DL}$ and b_{DS}^{DL}.

The fuzzy membership degree $f_{e,s}^{e,s}$, $s = 4,5,9,13,17$ for the linguistic value *down small* of the output *effort* in the *s*-th rule can be obtained by the formula

$$\begin{gathered}
f_{e,s}^{DS} = 0,\ if\ e \le a_{e,s}^{DS} \\
f_{e,s}^{DS} = (\,e - a_{e,s}^{DS}\,)\,/\,(\,b_{e,s}^{DS} - a_{e,s}^{DS}\,),\ if\ a_{e,s}^{DS} \le e \le b_{e,s}^{DS} \\
f_{e,s}^{DS} = g_s^{DS},\ if\ b_{e,s}^{DS} \le e \le c_{e,s}^{DS} \\
f_{e,s}^{DS} = (\,d_{e,s}^{DS} - e\,)\,/\,(\,d_{e,s}^{DS} - c_{e,s}^{DS}\,),\ if\ c_{e,s}^{DS} \le e \le d_{e,s}^{DS} \\
f_{e,s}^{DS} = 0,\ if\ d_{e,s}^{DS} \le e
\end{gathered} \tag{9.112}$$

where $a_{e,s}^{DS}$, $b_{e,s}^{DS}$, $c_{e,s}^{DS}$, $d_{e,s}^{DS}$, $s = 4,5,9,13,17$ are the parameters of the truncated trapezoidal fuzzy membership function for the output in the *s*-th rule and g_s^{DS}, $s = 4,5,9,13,17$ is firing strength of the *s*-th rule. The values of these parameters can be obtained by the formula

$$\begin{gathered}
a_{e,s}^{DS} = a_e^{DS} \\
b_{e,s}^{DS} = a_e^{DS} + (\,b_e^{DS} - a_e^{DS}\,)\,.\,g_s^{DS} \\
c_{e,s}^{DS} = c_e^{DS} - (\,c_e^{DS} - b_e^{DS}\,)\,.\,g_s^{DS} \\
d_{e,s}^{DS} = c_e^{DS}
\end{gathered} \tag{9.113}$$

where a_e^{DS}, b_e^{DS}, c_e^{DS} are the parameters of the original triangular fuzzy membership function for the linguistic value *down small* of the output. The values of these parameters are given by

$$a_e^{DS} = -20,\ b_e^{DS} = -10,\ c_e^{DS} = 0 \tag{9.114}$$

The fuzzy membership degree $f_{e,s}^{Z}$, $s = 3,8,12,16$ for the linguistic value *zero* of the output *effort* in the *s*-th rule can be obtained by the formula

$$\begin{gathered}
f_{e,s}^{Z} = 0,\ if\ e \leq a_{e,s}^{Z} \\
f_{e,s}^{Z} = (\,e - a_{e,s}^{Z}\,)\,/\,(\,b_{e,s}^{Z} - a_{e,s}^{Z}\,),\ if\ a_{e,s}^{Z} \leq e \leq b_{e,s}^{Z} \\
f_{e,s}^{Z} = g_{s}^{Z},\ if\ b_{e,s}^{Z} \leq e \leq c_{e,s}^{Z} \\
f_{e,s}^{Z} = (\,d_{e,s}^{Z} - e\,)\,/\,(\,d_{e,s}^{Z} - c_{e,s}^{Z}\,),\ if\ c_{e,s}^{Z} \leq e \leq d_{e,s}^{Z} \\
f_{e,s}^{Z} = 0,\ if\ d_{e,s}^{Z} \leq e
\end{gathered} \tag{9.115}$$

where $a_{e,s}^{Z}$, $b_{e,s}^{Z}$, $c_{e,s}^{Z}$, $d_{e,s}^{Z}$, $s = 3,8,12,16$ are the parameters of the truncated trapezoidal fuzzy membership function for the output in the *s*-th rule and g_{s}^{Z}, $s = 3,8,12,16$ is firing strength of the *s*-th rule. The values of these parameters can be obtained by the formula

$$\begin{gathered}
a_{e,s}^{Z} = a_{e}^{Z} \\
b_{e,s}^{Z} = a_{e}^{Z} + (\,b_{e}^{Z} - a_{e}^{Z}\,)\,.\,g_{s}^{Z} \\
c_{e,s}^{Z} = c_{e}^{Z} - (\,c_{e}^{Z} - b_{e}^{Z}\,)\,.\,g_{s}^{Z} \\
d_{e,s}^{Z} = c_{e}^{Z}
\end{gathered} \tag{9.116}$$

where a_{e}^{Z}, b_{e}^{Z}, c_{e}^{Z} are the parameters of the original triangular fuzzy membership function for the linguistic value *zero* of the output. The values of these parameters are given by

$$a_{e}^{Z} = -10,\ b_{e}^{Z} = 0,\ c_{e}^{Z} = 10 \tag{9.117}$$

The fuzzy membership degree $f_{e,s}^{US}$, $s = 7,11$ for the linguistic value *up small* of the output *effort* in the *s*-th rule can be obtained by the formula

$$\begin{gathered}
f_{e,s}^{US} = 0,\ if\ e \leq a_{e,s}^{US} \\
f_{e,s}^{US} = (\,e - a_{e,s}^{US}\,)\,/\,(\,b_{e,s}^{US} - a_{e,s}^{US}\,),\ if\ a_{e,s}^{US} \leq e \leq b_{e,s}^{US} \\
f_{e,s}^{US} = g_{s}^{US},\ if\ b_{e,s}^{US} \leq e \leq c_{e,s}^{US} \\
f_{e,s}^{US} = (\,d_{e,s}^{US} - e\,)\,/\,(\,d_{e,s}^{US} - c_{e,s}^{US}\,),\ if\ c_{e,s}^{US} \leq e \leq d_{e,s}^{US} \\
f_{e,s}^{US} = 0,\ if\ d_{e,s}^{US} \leq e
\end{gathered} \tag{9.118}$$

where $a_{e,s}^{US}$, $b_{e,s}^{US}$, $c_{e,s}^{US}$, $d_{e,s}^{US}$, $s = 7,11$ are the parameters of the truncated trapezoidal fuzzy membership function for the output in the *s*-th rule and g_s^{US}, $s = 7,11$ is firing strength of the *s*-th rule. The values of these parameters can be obtained by the formula

$$\begin{aligned} a_{e,s}^{US} &= a_e^{US} \\ b_{e,s}^{US} &= a_e^{US} + (b_e^{US} - a_e^{US}) \,.\, g_s^{US} \\ c_{e,s}^{US} &= c_e^{US} - (c_e^{US} - b_e^{US}) \,.\, g_s^{US} \\ d_{e,s}^{US} &= c_e^{US} \end{aligned} \tag{9.119}$$

where a_e^{US}, b_e^{US}, c_e^{US} are the parameters of the original triangular fuzzy membership function for the linguistic value *up small* of the output. The values of these parameters are given by

$$a_e^{US} = 0,\ b_e^{US} = 10,\ c_e^{US} = 20 \tag{9.120}$$

The fuzzy membership degree $f_{e,s}^{UL}$, $s = 1,2,6$ for the linguistic value *up large* of the output *effort* in the *s*-th rule can be obtained by the formula

$$\begin{aligned} f_{e,s}^{UL} &= 0,\ if\ e \le a_{e,s}^{UL} \\ f_{e,s}^{UL} &= (e - a_{e,s}^{UL}) / (b_{e,s}^{UL} - a_{e,s}^{UL}),\ if\ a_{e,s}^{UL} \le e \le b_{e,s}^{UL} \\ f_{e,s}^{UL} &= g_s^{UL},\ if\ b_{e,s}^{UL} \le e \le c_{e,s}^{UL} \\ f_{e,s}^{UL} &= g_s^{UL},\ if\ c_{e,s}^{UL} \le e \le d_{e,s}^{UL} \\ f_{e,s}^{UL} &= 0,\ if\ d_{e,s}^{UL} < e \end{aligned} \tag{9.121}$$

where $a_{e,s}^{UL}$, $b_{e,s}^{UL}$, $c_{e,s}^{UL}$, $d_{e,s}^{UL}$, $s = 1,2,6$ are the parameters of the truncated trapezoidal fuzzy membership function for the output in the *s*-th rule and g_s^{UL}, $s = 1,2,6$ is firing strength of the *s*-th rule. The values of these parameters can be obtained by the formula

$$\begin{aligned} a_{e,s}^{UL} &= a_e^{UL} \\ b_{e,s}^{UL} &= a_e^{UL} + (b_e^{UL} - a_e^{UL}) \,.\, g_s^{UL} \\ c_{e,s}^{UL} &= d_e^{UL} - (d_e^{UL} - c_e^{UL}) \,.\, g_s^{UL} \\ d_{e,s}^{UL} &= d_e^{UL} \end{aligned} \tag{9.122}$$

where a_e^{UL}, b_e^{UL}, c_e^{UL}, d_e^{UL} are the parameters of the original trapezoidal fuzzy membership function for the linguistic value *up large* of the output. The values of these parameters are given by

$$a_e^{UL} = 10,\ b_e^{UL} = 20,\ c_e^{UL} = 30,\ d_e^{UL} = 30 \tag{9.123}$$

Equation (9.121) differs slightly from Eq. (9.4). In particular, the arithmetic division in the fourth line of Eq. (9.4) has been removed from Eq. (9.121) to avoid a division by zero due to the equality of the parameters $c_{e,s}^{UL}$ and $d_{e,s}^{UL}$. Also, the inequality sign in the fifth line of Eq. (9.4) has been strengthened in Eq. (9.121) to account for the vertical right shoulder of the truncated membership function.

The parameters of the fuzzy membership functions for the output from the fuzzy system are summarised in Table 9.6.

Table 9.6. Fuzzy membership function parameters for the output

Linguistic value / Output	Control effort
Down large	[-30 -30 -20 -10]
Down small	[-20 -10 0]
Zero	[-10 0 10]
Up small	[0 10 20]
Up large	[10 20 30 30]

At the aggregation substage of the inference stage for the CS, we have to find the aggregated fuzzy membership function representing the overall output for all the rules. For this purpose, we assume to have represented the fuzzy membership function for the output in each rule during the implication substage, as shown by Eqs. (9.124)–(9.130).

$$F_s = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0/\text{-}15, 0/\text{-}10, 0/\text{-}5, 0/0, 0/5, 0/10, 0/15, 0/20, 0/25, 0/30\},\quad s = 1,2,3,4,5,6,7,8,9,10 \tag{9.124}$$

$$F_{11} = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0/\text{-}15, 0/\text{-}10, 0/\text{-}5, 0/0, 0.42/5, 0.42/10, 0.42/15, 0/20, 0/25, 0/30\} \tag{9.125}$$

$$F_{12} = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0/\text{-}15, 0/\text{-}10, 0.5/\text{-}5, 0.58/0, 0.5/5, 0/10, 0/15, 0/20, 0/25, 0/30\} \tag{9.126}$$

$$F_s = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0/\text{-}15, 0/\text{-}10, 0/\text{-}5, 0/0, 0/5, 0/10, 0/15, 0/20, 0/25, 0/30\}, \quad s = 13,14,15 \tag{9.127}$$

$$F_{16} = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0/\text{-}15, 0/\text{-}10, 0.42/\text{-}5, 0.42/0, 0.42/5, 0/10, 0/15, 0/20, 0/25, 0/30\} \tag{9.128}$$

$$F_{17} = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0.5/\text{-}15, 0.58/\text{-}10, 0.5/\text{-}5, 0/0, 0/5, 0/10, 0/15, 0/20, 0/25, 0/30\} \tag{9.129}$$

$$F_s = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0/\text{-}15, 0/\text{-}10, 0/\text{-}5, 0/0, 0/5, 0/10, 0/15, 0/20, 0/25, 0/30\}, \quad s = 18,19,20 \tag{9.130}$$

Therefore, the aggregated fuzzy membership function F for the output for the CS can be obtained by the formula

$$F = F_1 \cup \ldots \cup F_{20} = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0.5/\text{-}15, 0.58/\text{-}10, 0.5/\text{-}5, 0.58/0, 0.5/5, 0.42/10, 0.42/15, 0/20, 0/25, 0/30\} \tag{9.131}$$

At the defuzzification stage for the CS, we have to find the crisp value for the output, as shown by Eq. (9.132).

$$\begin{aligned} D = (f_1 . y_1 + \ldots + f_{13} . y_{13}) / (f_1 + \ldots + f_{13}) = \\ [(0 . -30) + (0 . -25) + (0 . -20) + (0.5 . -15) + (0.58 . -10) \\ + (0.5 . -5) + (0.58 . 0) + (0.5 . 5) + (0.42 . 10) + (0.42 . 15) \\ + (0 . 20) + (0 . 25) + (0 . 30)] / \\ (0 + 0 + 0 + 0.5 + 0.58 + 0.5 \\ + 0.58 + 0.5 + 0.42 + 0.42 + 0 + 0 + 0) = -2.8 / 3.5 = -0.8 \end{aligned} \tag{9.132}$$

Having found the defuzzified output for the CS, we have to find the defuzzified output for the FS as well. For this purpose, we first put the rules of the CS in groups in order to obtain the SS. Then, we go through the fuzzification stage and the application substage for the SS. The results from these will be the same as the ones already obtained for the CS because neither the fuzzification stage nor the application substage are affected by the rearrangement of the rules. Next, we identify the single equivalent rules for all groups and remove all other rules from the rule base, as shown below.

The firing strength for each rule in the groups of the SS is given by Eqs. (9.133)–(9.137).

$$Group\ DL{:}\ g_{10}^{DL} = 0,\ g_{14}^{DL} = 0,\ g_{15}^{DL} = 0,\ g_{18}^{DL} = 0,\ g_{19}^{DL} = 0,\ g_{20}^{DL} = 0 \quad (9.133)$$

$$Group\ DS{:}\ g_{4}^{DS} = 0,\ g_{5}^{DS} = 0,\ g_{9}^{DS} = 0,\ g_{13}^{DS} = 0,\ g_{17}^{DS} = 0.58 \quad (9.134)$$

$$Group\ Z{:}\ g_{3}^{Z} = 0,\ g_{8}^{Z} = 0,\ g_{12}^{Z} = 0.58,\ g_{16}^{Z} = 0.42 \quad (9.135)$$

$$Group\ US{:}\ g_{7}^{US} = 0,\ g_{11}^{US} = 0.42 \quad (9.136)$$

$$Group\ UL{:}\ g_{1}^{UL} = 0,\ g_{2}^{UL} = 0,\ g_{6}^{UL} = 0 \quad (9.137)$$

The single equivalent rules for the above groups are given by Eqs. (9.138)–(9.142).

$$Group\ DL{:}\ g_{10}^{DL} = 0 \quad (9.138)$$

$$Group\ DS{:}\ g_{17}^{DS} = 0.58 \quad (9.139)$$

$$Group\ Z{:}\ g_{12}^{Z} = 0.58 \quad (9.140)$$

$$Group\ US{:}\ g_{11}^{US} = 0.42 \quad (9.141)$$

$$Group\ UL{:}\ g_{1}^{UL} = 0 \quad (9.142)$$

The single equivalent rules for the first and the last group could have been chosen arbitrarily because all rules from these two groups have firing strength 0. However, we have decided to choose the first rule, i.e. the rule with the lowest number, as the single equivalent rule in each of these two groups.

Therefore, the FS will be represented by the five rules with numbers 10, 17, 12, 11 and 1. Due the reduced number of rules, the subsequent implication and aggregation substages are significantly simplified, as shown by Eqs. (9.143)–(9.148).

$$F_{10} = \{0/\text{-}30,\ 0/\text{-}25,\ 0/\text{-}20,\ 0/\text{-}15,\ 0/\text{-}10,\ 0/\text{-}5,\ 0/0,\ 0/5,\ 0/10,\ 0/15,\ 0/20,\ 0/25,\ 0/30\} \quad (9.143)$$

$$F_{17} = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0.5/\text{-}15, 0.58/\text{-}10, 0.5/\text{-}5, 0/0, 0/5, 0/10, 0/15, 0/20, 0/25, 0/30\} \tag{9.144}$$

$$F_{12} = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0/\text{-}15, 0/\text{-}10, 0.5/\text{-}5, 0.58/0, 0.5/5, 0/10, 0/15, 0/20, 0/25, 0/30\} \tag{9.145}$$

$$F_{11} = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0/\text{-}15, 0/\text{-}10, 0/\text{-}5, 0/0, 0.42/5, 0.42/10, 0.42/15, 0/20, 0/25, 0/30\} \tag{9.146}$$

$$F_{1} = \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0/\text{-}15, 0/\text{-}10, 0/\text{-}5, 0/0, 0/5, 0/10, 0/15, 0/20, 0/25, 0/30\} \tag{9.147}$$

$$\begin{aligned} F &= F_{10} \cup F_{17} \cup F_{12} \cup F_{11} \cup F_{11} \\ &= \{0/\text{-}30, 0/\text{-}25, 0/\text{-}20, 0.5/\text{-}15, 0.58/\text{-}10, 0.5/\text{-}5, 0.58/0, \\ &\quad 0.5/5, 0.42/10, 0.42/15, 0/20, 0/25, 0/30\} \end{aligned} \tag{9.148}$$

The aggregated fuzzy membership function F for the output of the FS is the same as the one for the output of the CS, as can be seen from Eq. (9.131) and Eq. (9.148). Therefore, the five rules of the FS will yield a defuzzified output which is the same as the twenty rules of the CS, as shown by Eq. (9.132). This also shows that like the fuzzification stage and the application substage of the inference stage, the defuzzification stage is not affected by the reduced number of rules either. In spite of that, the efficiency gained as a result of the removed redundant operations during the implication and aggregation substages of the inference stage significantly outweighs the complexity added by the selection process for single equivalent rules, as shown further in this chapter.

The overall behavioural equivalence of the CS and the FS is illustrated in Figs. 9.2–9.3. These figures show the output surfaces for the two systems, which are identical. The surfaces are generated with 4×5 equally spaced points in order to facilitate the analogy with the rule bases for the two systems which are with 4×5 linguistic values for the inputs.

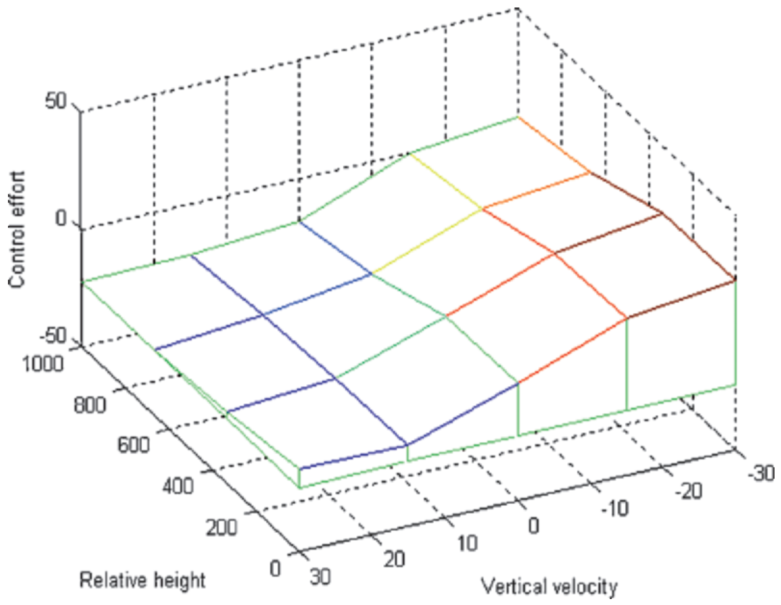

Fig. 9.2. Output surface with 4×5 points for the conventional system

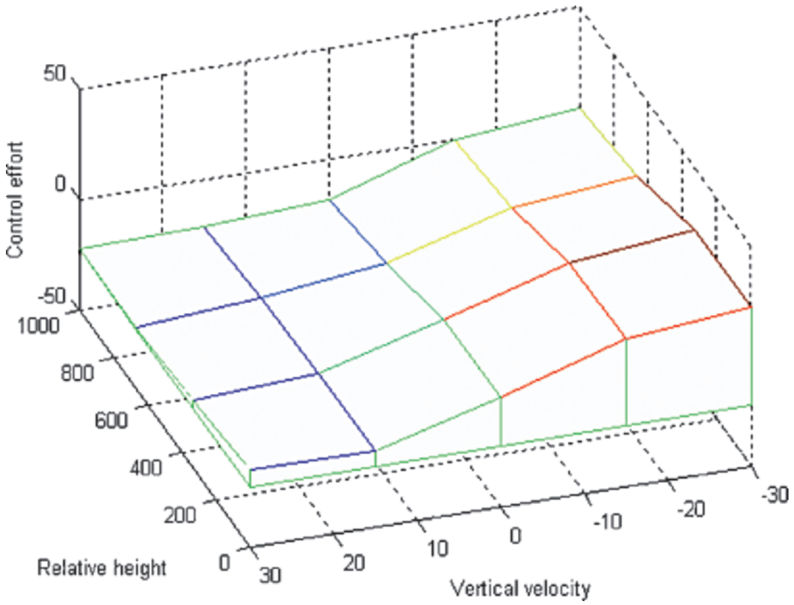

Fig. 9.3. Output surface with 4×5 points for the filtered system

The evidence for the overall behavioural equivalence of the CS and the FS is given in Table 9.7. This table shows the numerical values of the 4×5 point output surfaces for the two systems, which are equal for each permutation of crisp values of the inputs.

Table 9.7. Numerical values of the 4×5 point output surface for the two systems

Point number / Point component	Input 1 (Relative height)	Input 2 (Vertical velocity)	CS output (Control effort)	FS output (Control effort)
1	0.0	-30.0	22.4	22.4
2	0.0	-15.0	16.9	16.9
3	0.0	0.0	0.0	0.0
4	0.0	15.0	-15.6	-15.6
5	0.0	30.0	-14.7	-14.7
6	333.3	-30.0	21.5	21.5
7	333.3	-15.0	15.4	15.4
8	333.3	0.0	-0.9	-0.9
9	333.3	15.0	-16.4	-16.4
10	333.3	30.0	-19.4	-19.4
11	666.7	-30.0	10.0	10.0
12	666.7	-15.0	5.0	5.0
13	666.7	0.0	-11.3	-11.3
14	666.7	15.0	-17.9	-17.9
15	666.7	30.0	-21.8	-21.8
16	1000.0	-30.0	4.5	4.5
17	1000.0	-15.0	0.0	0.0
18	1000.0	0.0	-18.3	-18.3
19	1000.0	15.0	-21.3	-21.3
20	1000.0	30.0	-22.4	-22.4

The dominant rules for all permutations of crisp values of the inputs for the FS are presented in Table 9.8. This table shows the corresponding rule numbers whereby each of these numbers defines uniquely a rule from the rule base for the CS in Table 9.1.

The overall behavioural equivalence of the CS and the FS is illustrated further in Figs. 9.4–9.5. These figures show the output surfaces with 40×50 equally spaced points for the two systems. As in the case of Figs. 9.2–9.3, the two surfaces are identical but the corresponding numerical values and dominant rules are not shown due to the high space requirements for representing this large number of points. A brief visual inspection of Figs. 9.2–9.5 shows that the 10×10 fold increase of the number of points

Table 9.8. Dominant rules for the filtered system

Point number / Point component	Input 1 (Relative height)	Input 2 (Vertical velocity)	Dominant rules (Numbers)
1	0.0	-30.0	10, 4, 3, 7, 1
2	0.0	-15.0	10, 4, 3, 7, 2
3	0.0	0.0	10, 4, 3, 7, 1
4	0.0	15.0	10, 5, 3, 7, 1
5	0.0	30.0	10, 5, 3, 7, 1
6	333.3	-30.0	10, 4, 3, 11, 6
7	333.3	-15.0	10, 4, 12, 7, 6
8	333.3	0.0	10, 13, 8, 7, 1
9	333.3	15.0	10, 9, 3, 7, 1
10	333.3	30.0	10, 5, 3, 7, 1
11	666.7	-30.0	10, 4, 16, 11, 6
12	666.7	-15.0	10, 17, 12, 11, 6
13	666.7	0.0	18, 13, 8, 7, 1
14	666.7	15.0	15, 9, 3, 7, 1
15	666.7	30.0	15, 4, 3, 7, 1
16	1000.0	-30.0	10, 4, 16, 11, 1
17	1000.0	-15.0	10, 17, 16, 11, 1
18	1000.0	0.0	18, 13, 3, 7, 1
19	1000.0	15.0	20, 4, 3, 7, 1
20	1000.0	30.0	20, 4, 3, 7, 1

improves significantly the precision of the graphical presentation of the output surfaces but at the same time the complexity of this presentation is substantially increased.

Example 9.14

A fuzzy system for the operation of a service center for spare parts is described by the inputs i_1, i_2, i_3 and the output o_1 where i_1 is the *repair utilisation factor*, i_2 is the *number of servers*, i_3 is the *mean delay of service* and o_1 is the *number of spare parts* [58]. In this case, i_1 can take the three linguistic values *low* (L), *medium* (M) and *high* (H), i_2 can take the three linguistic values *small* (S), *medium* (M) and *large* (L), i_3 can take the three linguistic values *very short* (VS), *short* (S) and *medium* (M), whereas o_1 can take the seven linguistic values *very small* (VS), *small* (S), *rather small* (RS), *medium* (M) *rather large* (RL), *large* (L) and *very large* (VL).

The linguistic values for the three inputs are presented by fuzzy membership functions on a normalised continuous universe of discourse [0, 1]. As far as the output is concerned, its linguistic values are presented by fuzzy membership functions on a normalised discrete universe of discourse [0, 1]. For simplicity, these membership functions are not given here explicitly.

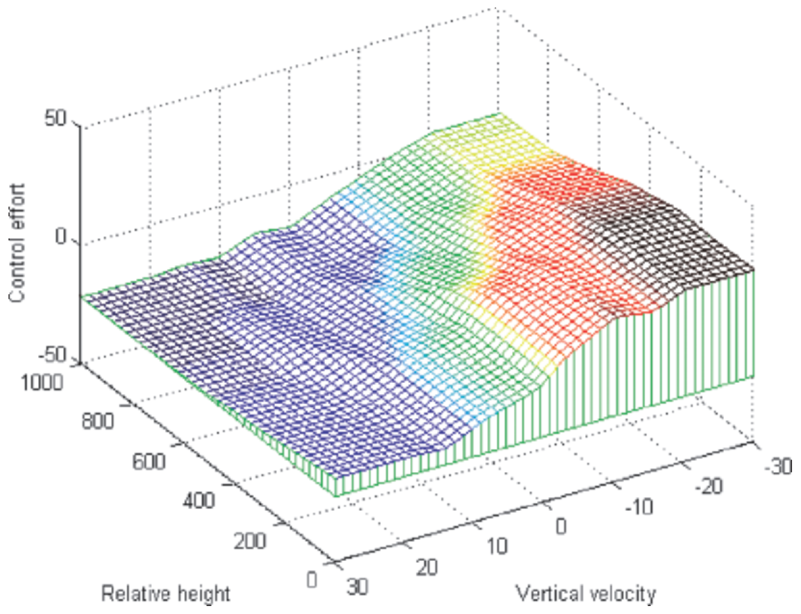

Fig. 9.4. Output surface with 40×50 points for the conventional system

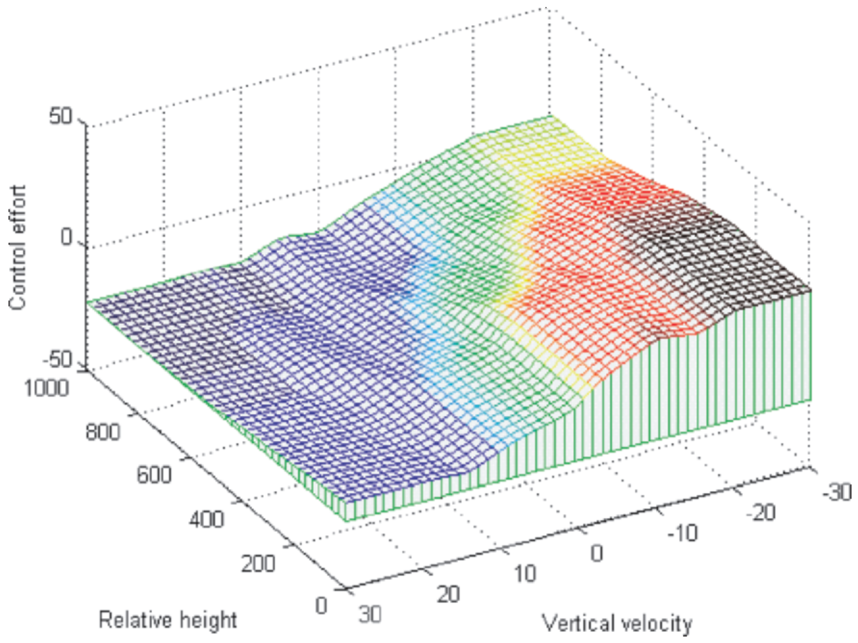

Fig. 9.5. Output surface with 40×50 points for the filtered system

By making the substitutions L = 1, M = 2, H = 3 for i_1, S = 1, M = 2, L = 3 for i_2, VS = 1, S = 2, M = 3 for i_3, as well as the substitutions VS = 1, S = 2, RS = 3, M = 4, RL = 5, L = 6, VL = 7 for o_1, we can construct the integer table for the fuzzy rule base of this CS, as shown in Table 9.9. Then, by applying step 1 from Algorithm 9.4, we can construct the integer table for the rule base of the SS, as shown in Table 9.10. The empty rows in these tables are used only for visual separation of the rules in different groups, which facilitates the analysis of the contents of the tables.

Table 9.9. Integer table for the rule base of the conventional system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of i_3	Linguistic value of o_1
1	1	1	1	1
2	1	1	2	1
3	1	1	3	1
4	1	2	1	1
5	1	2	2	1
6	1	2	3	1
7	1	3	1	2
8	1	3	2	2
9	1	3	3	1
10	2	1	1	2
11	2	1	2	1
12	2	1	3	1
13	2	2	1	3
14	2	2	2	2
15	2	2	3	1
16	2	3	1	4
17	2	3	2	3
18	2	3	3	2
19	3	1	1	7
20	3	1	2	6
21	3	1	3	4
22	3	2	1	4
23	3	2	2	4
24	3	2	3	2
25	3	3	1	5
26	3	3	2	4
27	3	3	3	3

Table 9.10. Integer table for the rule base of the sorted system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of i_3	Linguistic value of o_1
1	1	1	1	1
2	1	1	2	1
3	1	1	3	1
4	1	2	1	1
5	1	2	2	1
6	1	2	3	1
9	1	3	3	1
11	2	1	2	1
12	2	1	3	1
15	2	2	3	1
7	1	3	1	2
8	1	3	2	2
10	2	1	1	2
14	2	2	2	2
18	2	3	3	2
24	3	2	3	2
13	2	2	1	3
17	2	3	2	3
27	3	3	3	3
16	2	3	1	4
21	3	1	3	4
22	3	2	1	4
23	3	2	2	4
26	3	3	2	4
25	3	3	1	5
20	3	1	2	6
19	3	1	1	7

By applying steps 2-3 from Algorithm 9.4, we can now construct the integer table for the fuzzy rule base of the FS, as shown in Table 9.11.

The integer table for the fuzzy rule base of the FS contains only the single equivalent rule from each of the sorted seven groups of rules. The process leading to these single equivalent rules is not described here because it is the same as the corresponding process from Example 9.13. For this reason, some of the elements in Table 9.11 representing rule numbers and linguistic values of inputs are shown only in a general form, i.e. as compound expressions of DIS terms and not as atomic terms.

Table 9.11. Integer table for the rule base of the filtered system

Rule number	Linguistic value of i_1	Linguistic value of i_2	Linguistic value of i_3	Linguistic value of o_1
1 or 2 or 3 or 4 or 5 or 6 or 9 or 11 or 12 or 15	1 or 2	1 or 2 or 3	1 or 2 or 3	1
7 or 8 or 10 or 14 or 18 or 24	1 or 2 or 3	1 or 2 or 3	1 or 2 or 3	2
13 or 17 or 27	2 or 3	2 or 3	1 or 2 or 3	3
16 or 21 or 22 or 23 or 26	2 or 3	1 or 2 or 3	1 or 2 or 3	4
25	3	3	1	5
20	3	1	2	6
19	3	1	1	7

As in Example 9.13, Algorithm 9.4 can be implemented here much easier using Boolean matrices or binary relations, as described by Algorithms 9.5–9.6 This is shown briefly by Eqs. (9.149)–(9.154) which are associated with the integer tables from Tables 9.9–9.11. In particular, Eqs. (9.149)–(9.150) relate to Table 9.9, Eqs. (9.151)–(9.152) relate to Table 9.10, whereas Eqs. (9.153)–(9.154) relate to Table 9.11.

The Boolean matrix and the binary relation for the fuzzy rule base RB$_{CS}$ of the CS are given by

$$RB_{CS}: \quad \begin{array}{c|ccccccc} i_1\, i_2 i_3 \,/\, o_1 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 111 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 112 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 113 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & \\ 121 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 122 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 123 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & \\ 131 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 132 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 133 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & \\ 211 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 212 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 213 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & \\ 221 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 222 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 223 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \qquad (9.149)$$

231	0	0	0	1	0	0	0
232	0	0	1	0	0	0	0
233	0	1	0	0	0	0	0
311	0	0	0	0	0	0	1
312	0	0	0	0	0	1	0
313	0	0	0	1	0	0	0
321	0	0	0	1	0	0	0
322	0	0	0	1	0	0	0
323	0	1	0	0	0	0	0
331	0	0	0	0	1	0	0
332	0	0	0	1	0	0	0
333	0	0	1	0	0	0	0

$$\begin{aligned} RB_{CS}\text{: } \{&(111, 1), (112, 1), (113, 1),\\ &(121, 1), (122, 1), (123, 1),\\ &(131, 2), (132, 2), (133, 1),\\ &(211, 2), (212, 1), (213, 1),\\ &(221, 3), (222, 2), (223, 1),\\ &(231, 4), (232, 3), (233, 2),\\ &(311, 7), (312, 6), (313, 4),\\ &(321, 4), (322, 4), (323, 2),\\ &(331, 5), (332, 4), (333, 3)\} \end{aligned} \tag{9.150}$$

The Boolean matrix and the binary relation for the fuzzy rule base RB$_{SS}$ of the SS are given by

(9.151)

RB_{SS}: $i_1\, i_2\, i_3 / o_1$	1	2	3	4	5	6	7
111	1	0	0	0	0	0	0
112	1	0	0	0	0	0	0
113	1	0	0	0	0	0	0
121	1	0	0	0	0	0	0
122	1	0	0	0	0	0	0
123	1	0	0	0	0	0	0
133	1	0	0	0	0	0	0
212	1	0	0	0	0	0	0
213	1	0	0	0	0	0	0
223	1	0	0	0	0	0	0

131	0	1	0	0	0	0	0
132	0	1	0	0	0	0	0
211	0	1	0	0	0	0	0
222	0	1	0	0	0	0	0
233	0	1	0	0	0	0	0
323	0	1	0	0	0	0	0
221	0	0	1	0	0	0	0
232	0	0	1	0	0	0	0
333	0	0	1	0	0	0	0
231	0	0	0	1	0	0	0
313	0	0	0	1	0	0	0
321	0	0	0	1	0	0	0
322	0	0	0	1	0	0	0
332	0	0	0	1	0	0	0
331	0	0	0	0	1	0	0
312	0	0	0	0	0	1	0
311	0	0	0	0	0	0	1

$$
\begin{aligned}
RB_{SS}\colon \{&(111, 1), (112, 1), (113, 1), (121, 1), (122, 1),\\
&(123, 1), (133, 1), (212, 1), (213, 1), (223, 1),\\
&(131, 2), (132, 2), (211, 2), (222, 2), (233, 2), (323, 2),\\
&(221, 3), (232, 3), (333, 3),\\
&(231, 4), (313, 4), (321, 4), (322, 4), (332, 4),\\
&(331, 5),\\
&(312, 6),\\
&(311, 7)\}
\end{aligned}
\tag{9.152}
$$

The Boolean matrix and the binary relation for the fuzzy rule base RB_{FS} of the FS are given by

(9.153)

RB_{FS}: $i_1 i_2 i_3 / o_1$	1	2	3	4	5	6	7
111 or 112 or 113 or 121 or 122 or 123 or 133 or 212 or 213 or 223	1	0	0	0	0	0	0
131 or 132 or 211 or 222 or 233 or 323	0	1	0	0	0	0	0
221 or 232 or 333	0	0	1	0	0	0	0
231 or 313 or 321 or 322 or 332	0	0	0	1	0	0	0
331	0	0	0	0	1	0	0
312	0	0	0	0	0	1	0
311	0	0	0	0	0	0	1

RB_{FS}: {(111 or 112 or 113 or 121 or 122 or 123 or 133 or 212 or 213 or 223, 1), (9.154)
(131 or 132 or 211 or 222 or 233 or 323, 2),
(221 or 232 or 333, 3),
(231 or 313 or 321 or 322 or 332, 4),
(331, 5),
(312, 6),
(311, 7)}

Therefore, when we have to deal with an arbitrarily complex CS represented by an integer table, Boolean matrix or binary relation, we can formally simplify this system to a fairly simple FS and use the associated integer table, Boolean matrix or binary relation instead. As shown in Table 9.11, some of the elements in Eqs. (9.153)–(9.154) representing rule labels and permutations of linguistic values of inputs are given only in a general form, i.e. as compound expressions of DIS terms and not as atomic terms.

If we consider the fuzzification, inference and defuzzification stages for the CS and the FS, we will see the behavioural equivalence of the two fuzzy systems, i.e. that the defuzzified output is the same for any crisp values of the inputs. In order to prove this equivalence, we need to specify first the parameters of the fuzzy membership functions for the inputs and the output, as shown in Tables 9.12–9.15.

Table 9.12. Fuzzy membership function parameters for the first input

Linguistic value / Input	Repair utilisation factor
Low	[0.0 0.0 0.4 0.6]
Medium	[0.4 0.6 0.8]
High	[0.6 0.8 1.0 1.0]

Table 9.13. Fuzzy membership function parameters for the second input

Linguistic value / Input	Number of servers
Small	[0.0 0.0 0.15 0.35]
Medium	[0.3 0.5 0.7]
Large	[0.6 0.8 1.0 1.0]

Table 9.14. Fuzzy membership function parameters for the third input

Linguistic value / Input	Mean delay of service
Very short	[0.0 0.0 0.1 0.3]
Short	[0.1 0.3 0.5]
Medium	[0.4 0.6 1.0 1.0]

Table 9.15. Fuzzy membership function parameters for the output

Linguistic value / Output	Number of spare parts
Very small	[0.0 0.0 0.1 0.3]
Small	[0.0 0.2 0.4]
Rather small	[0.25 0.35 0.45]
Medium	[0.3 0.5 0.7]
Rather large	[0.55 0.65 0.75]
Large	[0.6 0.8 1.0]
Very large	[0.7 0.9 1.0 1.0]

As the output surface can show explicitly only two inputs, we have to fix one of the inputs to a particular value in order to generate this surface. In this context, it would be sensible to fix the first input (repair utilisation factor) to three different values – the bottom, the middle and the top of its range. In this case, three separate output surfaces will be generated for both the CS and the FS. These output surfaces will be with 3×3 equally spaced points in analogy with the rule bases for the two systems which are with 3×3×3 linguistic values for the inputs. For consistency with Example 9.13, additional output surfaces with a 10×10 fold increase of the number of points, i.e. with 30×30 points, will be generated to show the overall behavioural equivalence of the two systems.

The overall behavioural equivalence of the CS and the FS is illustrated below. In particular, Figs. 9.6–9.11 show the 3×3 point output surfaces for the two systems which are pairwise identical for each fixed value of input 1.

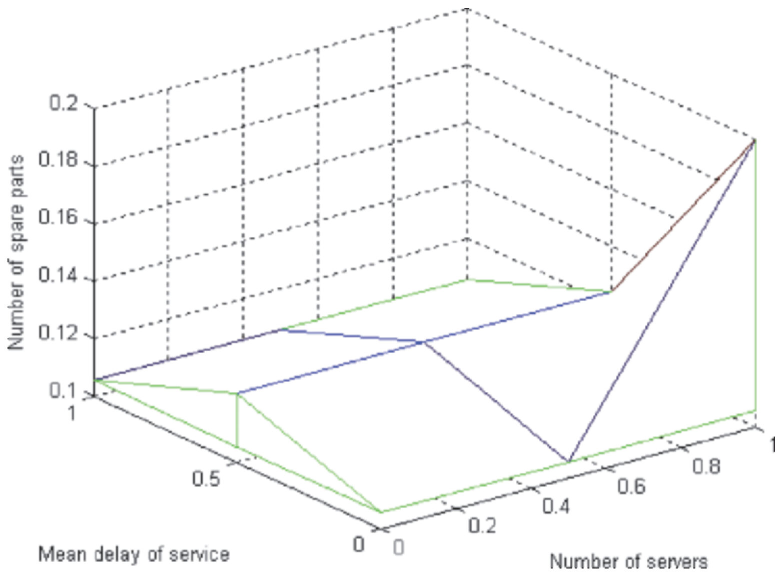

Fig. 9.6. Output surface with 3×3 points for the conventional system (repair utilisation factor = 0)

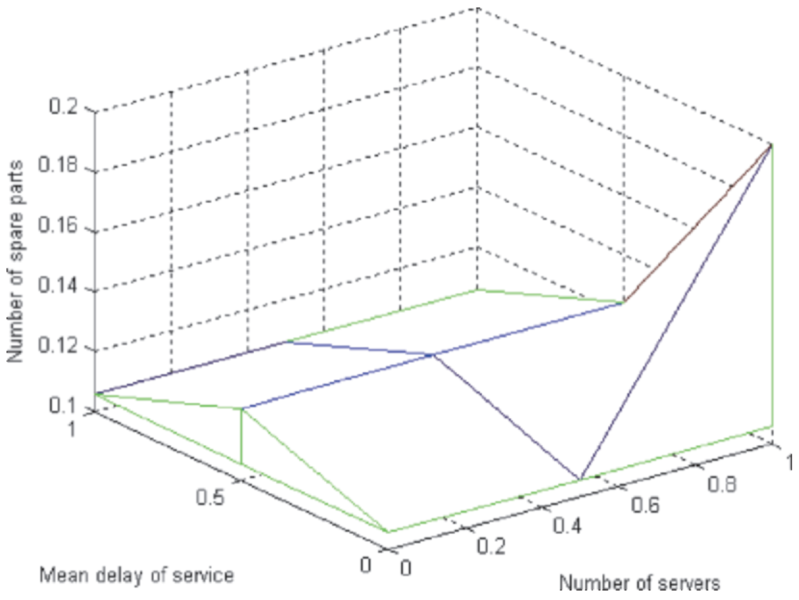

Fig. 9.7. Output surface with 3×3 points for the filtered system (repair utilisation factor = 0)

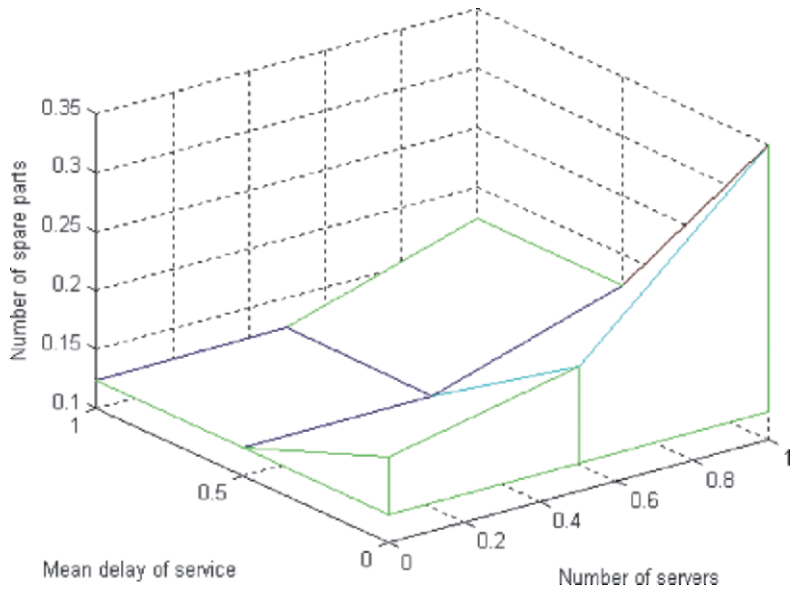

Fig. 9.8. Output surface with 3×3 points for the conventional system (repair utilisation factor = 0.5)

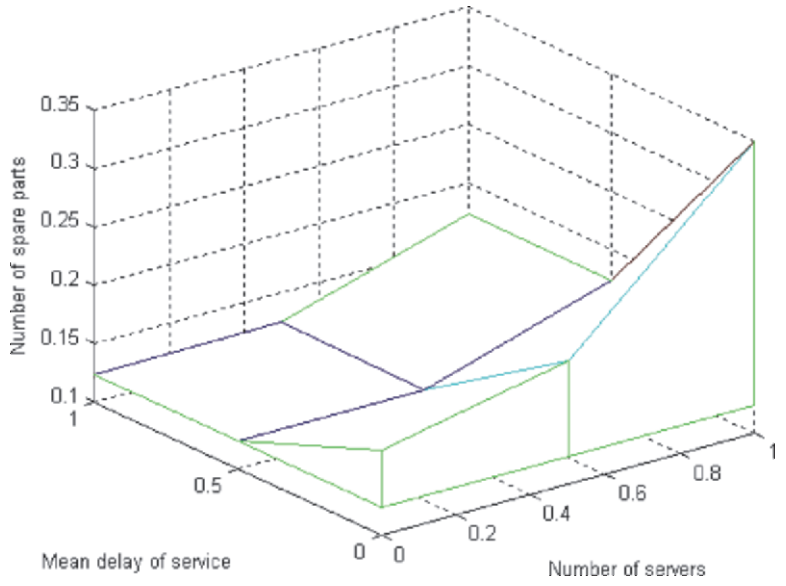

Fig. 9.9. Output surface with 3×3 points for the filtered system (repair utilisation factor = 0.5)

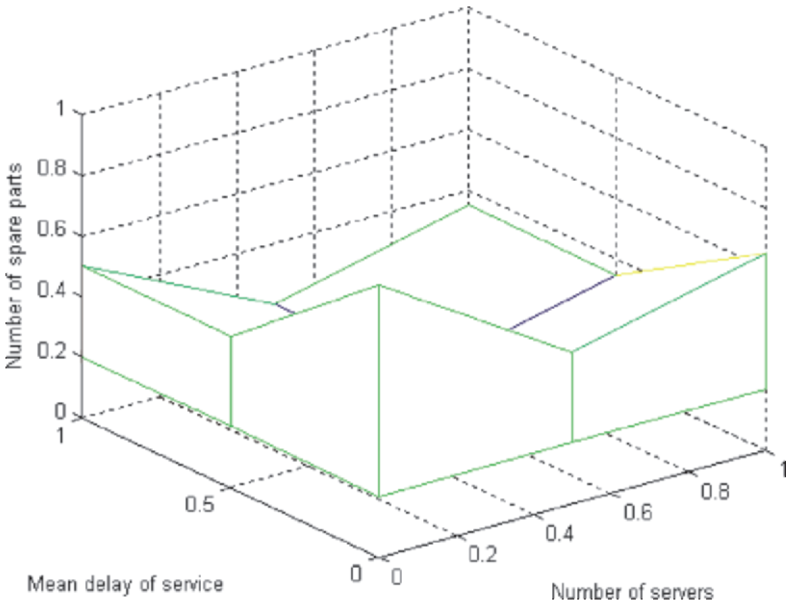

Fig. 9.10. Output surface with 3×3 points for the conventional system (repair utilisation factor = 1)

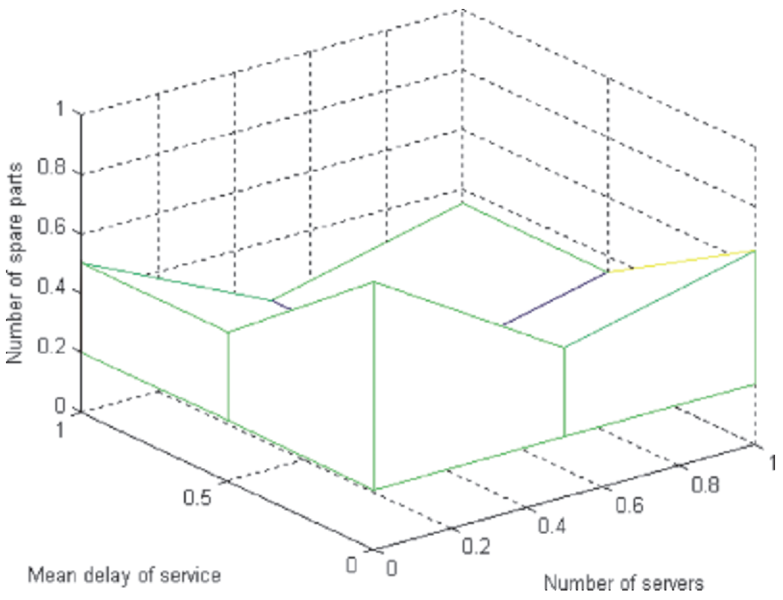

Fig. 9.11. Output surface with 3×3 points for the filtered system (repair utilisation factor = 1)

The evidence for the overall behavioural equivalence of the CS and the FS is given in Tables 9.16–9.18. These tables show the numerical values of the 3×3 point output surfaces for the two systems which are pairwise equal for each permutation of crisp values of the inputs.

Table 9.16. Numerical values for the 3×3 point output surfaces (repair utilisation factor = 0)

Point number / Point component	Input 2 (Number of servers	Input 3 (Mean delay of service)	CS output (Number of spare parts)	FS output (Number of spare parts
1	0.00	0.00	0.10	0.10
2	0.00	0.50	0.12	0.12
3	0.00	1.00	0.10	0.10
4	0.50	0.00	0.10	0.10
5	0.50	0.50	0.12	0.12
6	0.50	1.00	0.10	0.10
7	1.00	0.00	0.20	0.20
8	1.00	0.50	0.12	0.12
9	1.00	1.00	0.10	0.10

Table 9.17. Numerical values for the 3×3 point output surfaces (repair utilisation factor = 0.5)

Point number / Point component	Input 2 (Number of servers	Input 3 (Mean delay of service)	CS output (Number of spare parts)	FS output (Number of spare parts
1	0.00	0.00	0.17	0.17
2	0.00	0.50	0.12	0.12
3	0.00	1.00	0.12	0.12
4	0.50	0.00	0.20	0.20
5	0.50	0.50	0.12	0.12
6	0.50	1.00	0.12	0.12
7	1.00	0.00	0.35	0.35
8	1.00	0.50	0.17	0.17
9	1.00	1.00	0.17	0.17

The dominant rules for all permutations of crisp values of the inputs for the FS are presented in Tables 9.19–9.21. These tables show the corresponding rule numbers whereby each of these numbers defines uniquely a rule from the rule base for the CS in Table 9.9.

Table 9.18. Numerical values for the 3×3 point output surfaces (repair utilisation factor = 1)

Point number / Point component	Input 2 (Number of servers	Input 3 (Mean delay of service)	CS output (Number of spare parts)	FS output (Number of spare parts
1	0.00	0.00	0.89	0.89
2	0.00	0.50	0.50	0.50
3	0.00	1.00	0.50	0.50
4	0.50	0.00	0.50	0.50
5	0.50	0.50	0.20	0.20
6	0.50	1.00	0.20	0.20
7	1.00	0.00	0.65	0.65
8	1.00	0.50	0.35	0.35
9	1.00	1.00	0.35	0.35

Table 9.19. Dominant rules for the filtered system (repair utilisation factor = 0)

Point number / Point component	Input 2 (Number of servers	Input 3 (Mean delay of service)	Dominant rules (Numbers)
1	0.00	0.00	1, 7, 13, 16, 25, 20, 19
2	0.00	0.50	1, 7, 13, 16, 25, 20, 19
3	0.00	1.00	3, 7, 13, 16, 25, 20, 19
4	0.50	0.00	4, 7, 13, 16, 25, 20, 19
5	0.50	0.50	6, 7, 13, 16, 25, 20, 19
6	0.50	1.00	6, 7, 13, 16, 25, 20, 19
7	1.00	0.00	1, 7, 13, 16, 25, 20, 19
8	1.00	0.50	9, 7, 13, 16, 25, 20, 19
9	1.00	1.00	9, 7, 13, 16, 25, 20, 19

Table 9.20. Dominant rules for the filtered system (repair utilisation factor = 0.5)

Point number / Point component	Input 2 (Number of servers	Input 3 (Mean delay of service)	Dominant rules (Numbers)
1	0.00	0.00	1, 10, 13, 16, 25, 20, 19
2	0.00	0.50	12, 7, 13, 16, 25, 20, 19
3	0.00	1.00	12, 7, 13, 16, 25, 20, 19
4	0.50	0.00	4, 7, 13, 16, 25, 20, 19
5	0.50	0.50	6, 7, 13, 16, 25, 20, 19
6	0.50	1.00	6, 7, 13, 16, 25, 20, 19
7	1.00	0.00	1, 7, 13, 16, 25, 20, 19
8	1.00	0.50	9, 18, 13, 16, 25, 20, 19
9	1.00	1.00	9, 18, 13, 16, 25, 20, 19

Table 9.21. Dominant rules for the filtered system (repair utilisation factor = 1)

Point number / Point component	Input 2 (Number of servers	Input 3 (Mean delay of service)	Dominant rules (Numbers)
1	0.00	0.00	1, 7, 13, 16, 25, 20, 19
2	0.00	0.50	1, 7, 13, 21, 25, 20, 19
3	0.00	1.00	1, 7, 13, 21, 25, 20, 19
4	0.50	0.00	1, 7, 13, 22, 25, 20, 19
5	0.50	0.50	1, 24, 13, 16, 25, 20, 19
6	0.50	1.00	1, 24, 13, 16, 25, 20, 19
7	1.00	0.00	1, 7, 13, 16, 25, 20, 19
8	1.00	0.50	1, 7, 27, 16, 25, 20, 19
9	1.00	1.00	9, 7, 27, 16, 25, 20, 19

The overall behavioural equivalence of the CS and the FS is illustrated further below. In particular, Figs. 9.12–9.17 show the 30×30 point output surfaces for the two systems which are pairwise identical for each fixed value of input 1.

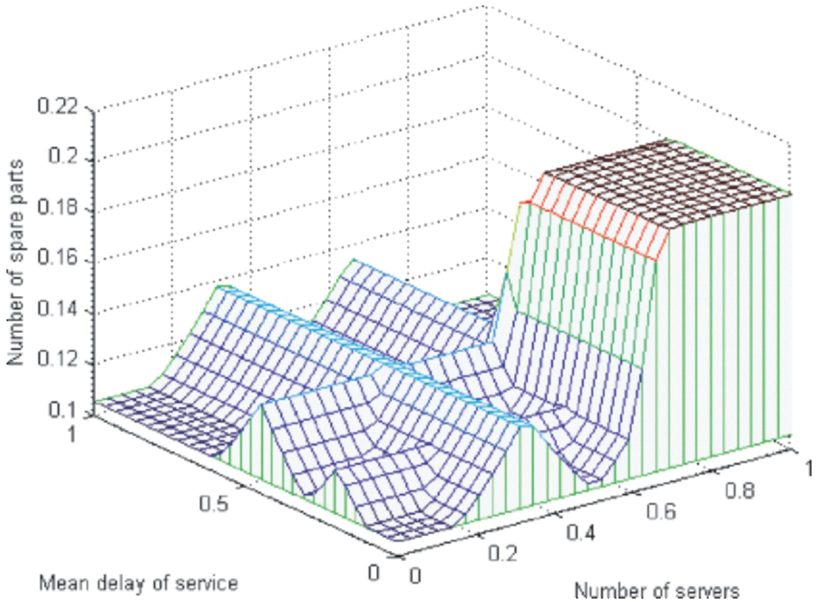

Fig. 9.12. Output surface with 30×30 points for the conventional system (repair utilisation factor = 0)

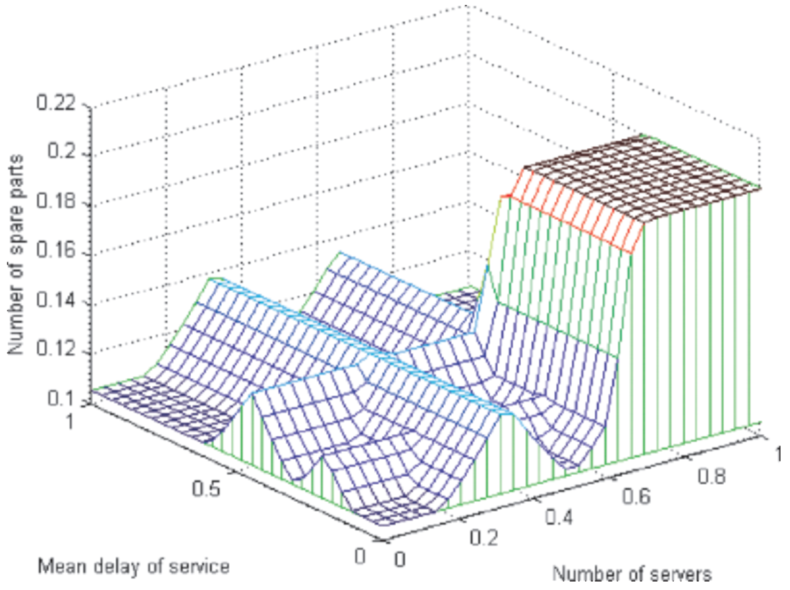

Fig. 9.13. Output surface with 30×30 points for the filtered system (repair utilisation factor = 0)

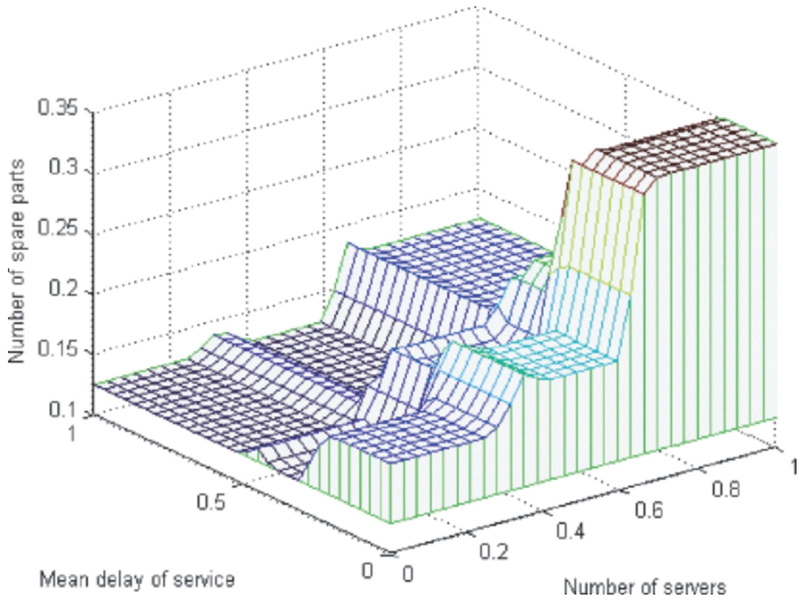

Fig. 9.14. Output surface with 30×30 points for the conventional system (repair utilisation factor = 0.5)

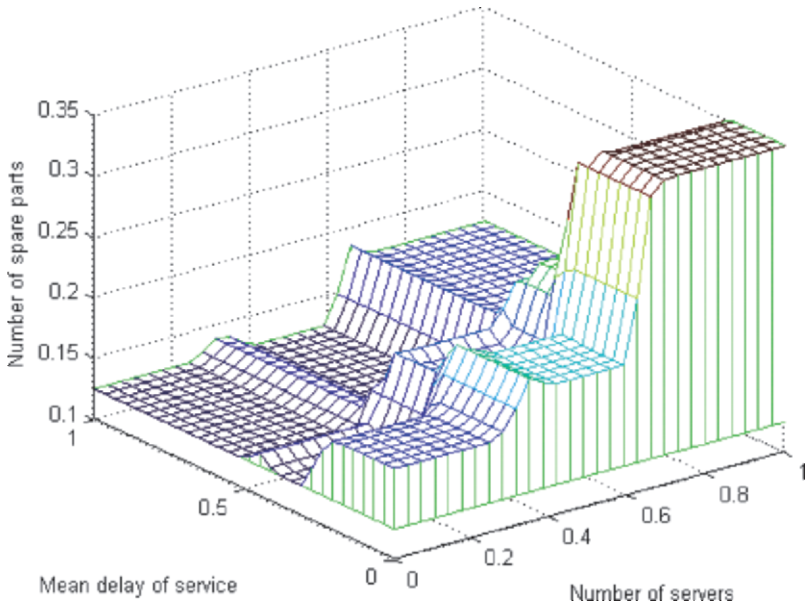

Fig. 9.15. Output surface with 30×30 points for the filtered system (repair utilisation factor = 0.5)

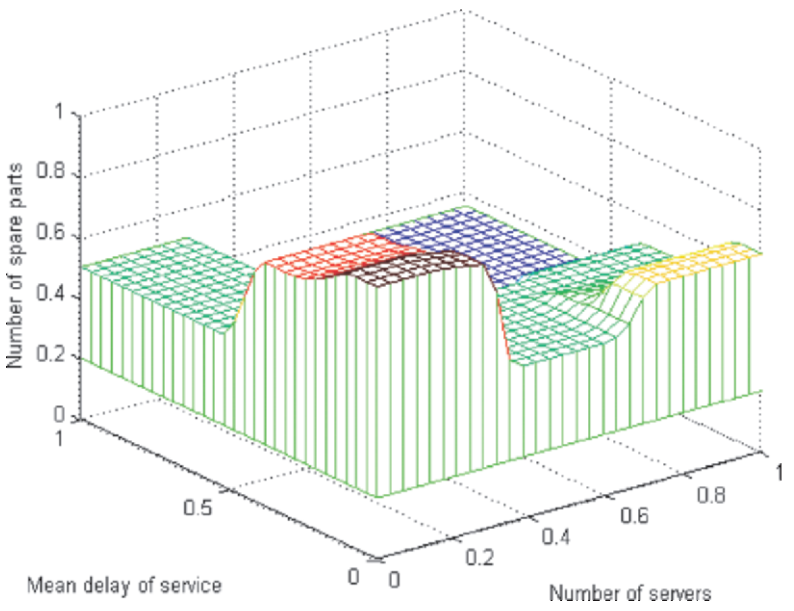

Fig. 9.16. Output surface with 30×30 points for the conventional system (repair utilisation factor = 1)

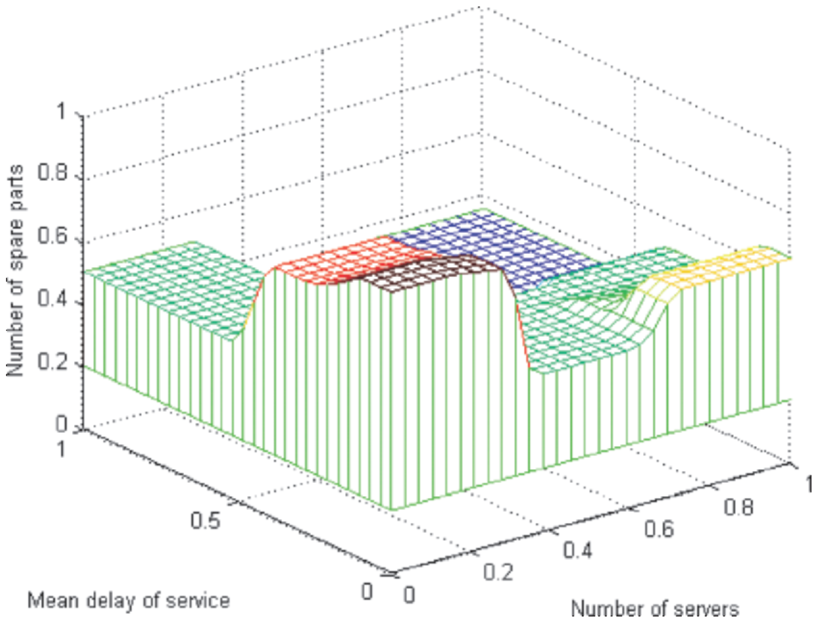

Fig. 9.17. Output surface with 30×30 points for the filtered system (repair utilisation factor = 1)

9.4 Complexity Evaluation of Formal Simplification Techniques

This section evaluates the quantitative complexity of the two formal simplification techniques. The evaluation approach used is based on precise calculations and it is superior to the so called ‘BIG(O)’ approach. The latter is based on approximate calculations and may lead to big errors in the presence of multiple dominant terms.

In particular, the ASs from Examples 9.1–9.12 and the FSs from Examples 9.13–9.14 are compared to the corresponding CSs in terms of the exact amount of on-line operations. For the aggregation method, which can be implemented entirely off-line, this task is quite simple because the comparison can be expressed as the ratio between the number of rules in the AS and the CS in Examples 9.1–9.12. As far as the filtration method is concerned, most of its implementation has to be done on-line. Also, the evaluation of its complexity is more difficult as it is expressed as a sum of the complexity in all stages and substages in Examples 9.13–9.14 such as fuzzification, inference, i.e. application, implication and aggregation, and defuzzification.

Apart from the specific context of Examples 9.13–9.14, the complexity of the filtration method and the associated FS is compared in general to the complexity of the conventional method and the associated CS, as well as to the complexity of the hierarchical method from Chapter 3 and the associated *hierarchical system* (HS). The latter is by far the best of all available fuzzy rule base reduction methods due to its wide applicability.

The parameters used for evaluating the quantitative complexity of the three methods are: m – number of inputs, w – number of linguistic values per input, n – number of outputs, t – number of elements in the discrete universe of discourse for the output, h – number of simulation cycles. In some cases, e.g. in Example 9.13, the number of linguistic values per inputs may vary and therefore the associated complexity evaluation formulas will be modified accordingly to reflect this.

The exact amount of on-line operations for the separate stages and substages in a fuzzy system is determined by the overall number of *elementary operations* (EO) such as addition, subtraction, multiplication, division and comparison. For simplicity, we assume that each of these operations is equal to one computational time unit and we will quantify each stage and substage in the fuzzy system by means of the overall number of these units. In this case, the term on-line refers to operations carried out after the measurement of the crisp values of the inputs and their quantitative complexity often has time-critical implications. All other operations, i.e. the ones carried out before the measurement of the crisp values of the inputs, are referred to as ‘off-line operations’ and their quantitative complexity usually does not have any time-critical implications.

The quantitative complexity for the calculation of the fuzzy membership degree f_{ps}, $p = 1,..,m$, $s = 1,..,r$ during the fuzzification stage can be obtained by the formula

$$f_{ps} = \max\{ \min[(x_{ps}-a_{ps})/(b_{ps}-a_{ps}), (c_{ps}-x_{ps})/(c_{ps}-b_{ps})], 0\} \qquad (9.155)$$

which follows from Eq. (9.1). On the basis of this formula, the overall number of EO_{FU}^{CS}, EO_{FU}^{HS} and EO_{FU}^{FS} for a CS, HS and FS will be given by Eq. (9.156), Eq. (9.157) and Eq. (9.158), respectively.

$$EO_{FU}^{CS} = 6\,.\,m\,.\,w\,.\,n\,.\,h \qquad (9.156)$$

$$EO_{FU}^{HS} = (m-1)\,.\,(12\,.\,w\,.\,n\,.\,h) \qquad (9.157)$$

$$EO_{FU}^{FS} = 6\,.\,m\,.\,w\,.\,n\,.\,h \qquad (9.158)$$

Equations (9.156)–(9.158) show that in the fuzzification stage the complexity of the FS is equal to the complexity of the CS whereas the complexity of the HS is lower.

The quantitative complexity for the calculation of the firing strength g_s, $s = 1,..,r$ during the application substage can be obtained from Eq. (9.2). On the basis of this equation, the overall number of EO_{AP}^{CS}, EO_{AP}^{HS} and EO_{AP}^{FS} for a CS, HS and FS will be given by Eq. (9.159), Eq. (9.160) and Eq. (9.161), respectively.

$$EO_{AP}^{CS} = (w+m-2)\,.\,w^{m-1}.\,n\,.\,h \qquad (9.159)$$

$$EO_{AP}^{HS} = (m-1)\,.\,w^{2}.\,n\,.\,h \qquad (9.160)$$

$$EO_{AP}^{FS} = (w+m-2)\,.\,w^{m-1}.\,n\,.\,h \qquad (9.161)$$

Equations (9.159)–(9.161) show that in the application substage the complexity of the FS is equal to the complexity of the CS whereas the complexity of the HS is lower.

The quantitative complexity for the calculation of the fuzzy membership function F_{sq}, $s = 1,..,r$, $q = 1,..,n$ during the implication substage is based on Eq. (9.3) and can be obtained by the formula

$$f_{sq} = \max\{ \min[(y_{sq}-a_{sq})/(b_{sq}-a_{sq}), g_s, (d_{sq}-y_{sq})/(d_{sq}-c_{sq})], 0\} \qquad (9.162)$$

which follows from Eqs. (9.4). On the basis of this formula, the overall number of EO_{IM}^{CS}, EO_{IM}^{HS} and EO_{IM}^{FS} for a CS, HS and FS will be given by Eq. (9.163), Eq. (9.164) and Eq. (9.165), respectively.

$$EO_{IM}^{CS} = 7 . w^{m} . t . n . h \tag{9.163}$$

$$EO_{IM}^{HS} = (m - 1) . (7 . w^{2} . t . n . h) \tag{9.164}$$

$$EO_{IM}^{FS} = 7 . w . t . n . h \tag{9.165}$$

Equations (9.163)–(9.165) show that in the implication substage the complexity of the HS is lower than the complexity of the CS but the complexity of the FS is even lower than the complexity of the HS.

The quantitative complexity for the calculation of the aggregated fuzzy membership function F_{sq}, $s = 1,..,r$, $q = 1,..,n$ during the aggregation substage can be obtained from Eq. (9.5). On the basis of this equation, the overall number of EO_{AG}^{CS}, EO_{AG}^{HS} and EO_{AG}^{FS} for a CS, HS and FS will be given by Eq. (9.166), Eq. (9.167) and Eq. (9.168), respectively.

$$EO_{AG}^{CS} = (w^{m} - 1) . t . n . h \tag{9.166}$$

$$EO_{AG}^{HS} = (m - 1) . (w^{2} - 1) . t . n . h \tag{9.167}$$

$$EO_{AG}^{FS} = (w - 1) . t . n . h \tag{9.168}$$

Equations (9.166)–(9.168) show that in the aggregation substage the complexity of the HS is lower than the complexity of the CS whereas the complexity of the FS is even lower than the complexity of the HS.

The quantitative complexity for the calculation of the defuzzified value D_q, $q = 1,.., n$ during the defuzzification stage can be obtained from Eq. (9.6). On the basis of this equation, the overall number of EO_{DE}^{CS}, EO_{DE}^{HS} and EO_{DE}^{FS} for a CS, HS and FS will be given by Eq. (9.169), Eq. (9.170) and Eq. (9.171), respectively.

$$EO_{DE}^{CS} = (3 . t - 1) . n . h \tag{9.169}$$

$$EO_{DE}^{HS} = (m - 1) . (3 . t - 1) . n . h \tag{9.170}$$

$$EO_{DE}^{FS} = (3 . t - 1) . n . h \tag{9.171}$$

Equations (9.169)–(9.171) show that in the defuzzification stage the complexity of the FS is equal to the complexity of the CS whereas the complexity of the HS is higher.

Before we can proceed further, we must also take into account the quantitative complexity for the identification of the single equivalent rules for the FS, as described by step 2 in Algorithm 9.4. For consistency, these operations may be assumed to be part of a comparison stage as the single equivalent rules are actually identified by comparing the firing strength of the rules in each group. So, the overall number of EO_{CO}^{FS} for a FS will be given by Eq. (9.172).

$$EO_{CO}^{FS} = (w^{m} - w) \,.\, n \,.\, h \tag{9.172}$$

Chronologically, the comparison stage for the FS must be placed after the application substage and before the implication substage. However, it has been considered here last to avoid confusion because this stage is only to be found in the FS.

The next step is to find the overall number of EO for the three systems under consideration, i.e. for the CS, the HS and the FS. This is done by adding the number of operations for the separate stages and substages for each of the three systems, as shown by Eq. (9.173), Eq. (9.174) and Eq. (9.175), respectively.

$$\begin{aligned} EO^{CS} = {} & EO_{FU}^{CS} + EO_{AP}^{CS} + EO_{IM}^{CS} + EO_{AG}^{CS} + EO_{DE}^{CS} \\ & = (6 \,.\, m \,.\, w \,.\, n \,.\, h\,) + (w + m - 2) \,.\, w^{m-1} .\, n \,.\, h + (7 \,.\, w^{m} .\, t \,.\, n \,.\, h\,) \\ & \quad + (w^{m} - 1) \,.\, t .\, n \,.\, h + (3 \,.\, t - 1) \,.\, n \,.\, h = \\ & = (6 \,.\, m \,.\, w) \,.\, n \,.\, h + (w + m - 2) \,.\, w^{m-1} .\, n \,.\, h + (7 \,.\, w^{m} .\, t\,) .\, n \,.\, h \\ & \quad + (w^{m} .\, t - t) \,.\, n \,.\, h + (3 \,.\, t - 1) \,.\, n \,.\, h \\ & = (6 \,.\, m \,.\, w + w^{m} + m \,.\, w^{m-1} - 2 \,.\, w^{m-1} + 7 \,.\, w^{m} .\, t \\ & \quad + w^{m} .\, t - t + 3 \,.\, t - 1) \,.\, n \,.\, h \\ & = [w^{m} + 7 \,.\, t \,.\, w^{m} + t \,.\, w^{m} + (m - 2) \,.\, w^{m-1} + 6 \,.\, w \,.\, m \\ & \quad + 2 \,.\, t - 1] \,.\, n \,.\, h \\ & = [(8 \,.\, t + 1) .\, w^{m} + (m - 2) \,.\, w^{m-1} + 6 \,.\, m \,.\, w + 2 \,.\, t - 1] \,.\, n \,.\, h \end{aligned} \tag{9.173}$$

We can use Eqs. (9.173)–(9.175) to evaluate comparatively the quantitative complexity of the three fuzzy systems when applied to Examples 9.13–9.14. As far as Example 9.13 is concerned, the two inputs take 4 and 5 linguistic values, respectively. For this reason, the value of *w* has been set equal to the average of the two numbers, i.e. 4.5, and this should lead to a good approximation of the exact number of on-line operations. The results for one simulation step in these examples are presented in Table 9.22.

$$\begin{aligned} EO^{HS} &= EO_{FU}^{HS} + EO_{AP}^{HS} + EO_{IM}^{HS} + EO_{AG}^{HS} + EO_{DE}^{HS} \\ &= (m-1) . (12 . w . n . h) + (m-1) . w^{2} . n . h \\ &\quad + (m-1) . (7 . w^{2} . t . n . h) + (m-1) . (w^{2} - 1) . t . n . h \\ &\quad + (m-1) . (3 . t - 1) . n . h \\ &= (m-1) . [(12 . w . n . h) + w^{2} . n . h + (7 . w^{2} . t . n . h) \\ &\quad + (w^{2} - 1) . t . n . h + (3 . t - 1) . n . h] = \\ &= (m-1) . [12 . w . n . h + w^{2} . n . h + 7 . w^{2} . t . n . h + \\ &\quad + (w^{2} - 1) . t . n . h + (3 . t - 1) . n . h] \\ &= (m-1) . (12 . w + w^{2} + 7. t . w^{2} + t . w^{2} - t + 3 . t - 1) . n . h \\ &= (m-1) . [(8 . t + 1) . w^{2} + 12 . w + 2 . t - 1] . n . h \end{aligned} \tag{9.174}$$

$$\begin{aligned} EO^{FS} &= EO_{FU}^{FS} + EO_{AP}^{FS} + EO_{IM}^{FS} + EO_{AG}^{FS} + EO_{DE}^{FS} + EO_{CO}^{FS} \\ &= (6 . m . w . n . h) + (w + m - 2) . w^{m-1} . n . h + (7 . w . t . n . h) \\ &\quad + (w - 1) . t . n . h + (3 . t - 1) . n . h + (w^{m} - w) . n . h \\ &= (6 . m . w) . n . h + (w + m - 2) . w^{m-1} . n . h + (7 . w . t). n . h \\ &\quad + (w . t - t). n . h + (3 . t - 1) . n . h + (w^{m} - w) . n . h \\ &= (6 . m . w + w^{m} + m . w^{m-1} - 2 . w^{m-1} + 7 . w . t \\ &\quad + w . t - t + 3 . t - 1 + w^{m} - w) . n . h \\ &= [2 . w^{m} + (m-2) . w^{m-1} + (6 . m - 1) . w + (8 . w + 2). t - 1] . n . h \end{aligned} \tag{9.175}$$

Table 9.22 shows that the FS is almost 4 times more efficient than the CS and the HS for Example 9.13. Also, the FS is more than 6 times more efficient than the CS and more than 4 times more efficient than the HS for

Table 9.22. Quantitative complexity of the three fuzzy systems in Examples 9.13–9.14

Fuzzy system / Example	Example 9.13	Example 9.14
EO^{CS}	2205	2487
EO^{HS}	2205	1716
EO^{FS}	583	399

Example 9.14. As in terms of behaviour the HS is only an approximation of the CS whereas the FS is equivalent to the CS, it is obvious that the FS outperforms significantly the other two systems.

Equations (9.173)–(9.175) are general formulas for the overall number of EO for the three fuzzy systems and can therefore be used for the evaluation of these systems in a wide context, i.e. without looking at specific examples. By varying some of the parameters in these formulas, it would be possible to see the dependency between the quantitative complexity for each system and the values of its parameters.

As the increase of the parameters *n* and *h* would always lead to a similar linear increase of the quantitative complexity for the three systems, it would be reasonable to keep these parameters fixed. As far as the other parameters *m*, *w* and *t* are concerned, it would be necessary to vary them because their increase would usually lead to a different exponential increase of the quantitative complexity of the three systems. Therefore, we will assume that the parameters have the following fixed values and variation ranges:

$$n = 1;\ h = 1;\ m = 2,3,4;\ w = 3,5,7,9,11;\ \ t = 7,13,19,25,31 \qquad (9.176)$$

In order to reduce the number of possible permutations of values for *m*, *w* and *t*, we will assume that the variation of the parameters w and t is fixed by the formula:

$$t = 3\ .\ w - 2 \qquad (9.177)$$

Equations (9.176)–(9.177) define a fairly wide and reasonable scope for evaluating the quantitative complexity in fuzzy systems. In particular, most fuzzy systems are initially considered for one simulation step of one output before more simulation steps of this output or simulations of other outputs are considered. Also, fuzzy systems are usually represented with up to 4 inputs as the number of rules for more inputs would be almost unmanageable. In addition, the inputs and outputs of fuzzy systems are often described by an odd number of linguistic values as this provides better coverage of the associated universes of discourse. And finally, the number of elements in the discrete universe of discourse for an output is often between 2 and 3 times greater than the number of linguistic values for this output, whose number is often close or even equal to the number of linguistic values that each input can take.

Table 9.23 presents the results from a general comparative evaluation of the quantitative complexity of the CS, the HS and the FS. This evaluation is made using Eqs. (9.173)–(9.175) and in accordance with the assumptions made for the values of all relevant parameters, as shown by Eqs. (9.176)–(9.177).

Table 9.23. Quantitative complexity of the three fuzzy systems

Number of rules / Fuzzy system	EO^{CS}	EO^{HS}	EO^{FS}
$3^2 = 9$	562	562	232
$3^3 = 27$	1,615	1,124	295
$3^4 = 81$	4,756	1,686	466
$5^2 = 25$	2,710	2,710	650
$5^3 = 125$	13,265	5,420	905
$5^4 = 625$	66,020	8,130	2,160
$7^2 = 49$	7,618	7,618	1,276
$7^3 = 343$	52,691	15,236	1,955
$7^4 = 2401$	368,244	22,854	6,750
$9^2 = 81$	16,438	16,438	2,110
$9^3 = 729$	146,821	32,876	3,541
$9^4 = 6,561$	1,320,484	49,314	16,636
$11^2 = 121$	30,322	30,322	3,152
$11^3 = 1,331$	331,779	60,644	5,759
$11^4 = 14,641$	3,648,596	90,966	34,986

The figures for the HS and the FS in Table 9.23 are also illustrated graphically in Fig. 9.18. The CS is not shown because it is very inefficient and would obstruct the graphical interpretation of the other two systems.

Table 9.23 and Fig. 9.18 show that the FS is superior for all considered permutations of values for the relevant parameters. In order to find the margin and the extent of this superiority, we have to compare each of the two inferior systems to the FS by subtracting and dividing the amount of corresponding operations, as shown in Tables 9.24–9.25.

It can be seen from Tables 9.24–9.25 that in most cases the margin and the extent of superiority of the FS with respect to the CS and the HS increases with the increase of the number of inputs for a fixed number of linguistic values per input as well as with the increase of the linguistic values per input for a fixed number of inputs. This increase is with a fairly big magnitude with respect the CS and with a more moderate magnitude with respect to the HS.

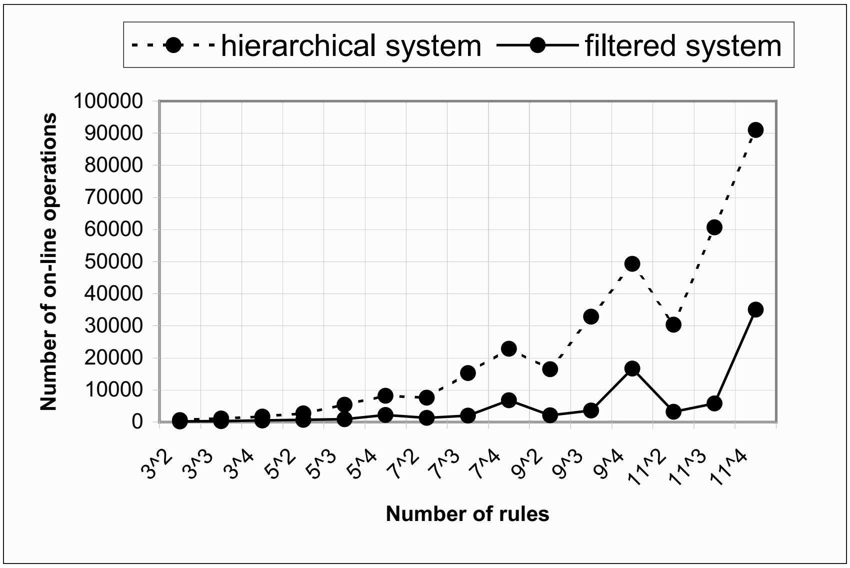

Fig. 9.18. Quantitative complexity of the hierarchical system and the filtered system

Table 9.24. Margin of superiority of the filtered system

Number of rules / Comparison	EO^{CS} - EO^{FS}	EO^{HS} - EO^{FS}
$3^2 = 9$	562 - 232 = 330	562 - 232 = 330
$3^3 = 27$	1,615 - 295 = 1,320	1,124 - 295 = 829
$3^4 = 81$	4,756 - 466 = 4,290	1,686 - 466 = 1,220
$5^2 = 25$	2,710 - 650 = 2,060	2,710 - 650 = 2,060
$5^3 = 125$	13,265 - 905 = 12,360	5,420 - 905 = 4,515
$5^4 = 625$	66,020 - 2,160 = 63,860	8,130 - 2,160 = 5,970
$7^2 = 49$	7,618 - 1,276 = 7,342	7,618 - 1,276 = 7,342
$7^3 = 343$	52,691 - 1,955 = 50,736	15,236 - 1,955 = 13,281
$7^4 = 2,401$	368,244 - 6,750 = 361,494	22,854 - 6,750 = 16,104
$9^2 = 81$	16,438 - 2,110 = 14,328	16,438 - 2,110 = 14,328
$9^3 = 729$	146,821 - 3,541 = 143,280	32,876 - 3,541 = 29,335
$9^4 = 6,561$	1,320,484 - 16,636 = 1,303,848	49,314 - 16,636 = 32,678
$11^2 = 121$	30,322 - 3,152 = 27,170	30,322 - 3,152 = 27,170
$11^3 = 1,331$	331,799 - 5,759 = 326,040	60,644 - 5,759 = 54,885
$11^4 = 14,641$	3,648,596 - 34,986 = 3,613,610	90,966 - 34,986 = 55,980

Table 9.25. Extent of superiority of the filtered system

Number of rules / Comparison	EO^{CS} / EO^{FS}	EO^{HS} / EO^{FS}
$3^2 = 9$	562 / 232 = 2.42	562 / 232 = 2.42
$3^3 = 27$	1,615 / 295 = 5.47	1,124 / 295 = 3.81
$3^4 = 81$	4,756 / 466 = 10.20	1,686 / 466 = 3.61
$5^2 = 25$	2,710 / 650 = 4.16	2,710 / 650 = 4.16
$5^3 = 125$	13,265 / 905 = 14.65	5,420 / 905 = 5.98
$5^4 = 625$	66,020 / 2,160 = 30.56	8,130 / 2,160 = 3.76
$7^2 = 49$	7,618 / 1,276 = 5.97	7,618 / 1,276 = 5.97
$7^3 = 343$	52,691 / 1,955 = 26.95	15,236 / 1,955 = 7.79
$7^4 = 2,401$	368,244 / 6,750 = 54.55	22,854 / 6,750 = 3.38
$9^2 = 81$	16,438 / 2,110 = 7.79	16,438 / 2,110 = 7.79
$9^3 = 729$	146,821 / 3,541 = 41.46	32,876 / 3,541 = 9.28
$9^4 = 6,561$	1,320,484 / 16,636 = 79.37	49,314 / 16,636 = 2.96
$11^2 = 121$	30,322 / 3,152 = 9.61	30,322 / 3,152 = 9.61
$11^3 = 1,331$	331,799 / 5,759 = 57.61	60,644 / 5,759 = 10.53
$11^4 = 14,641$	3,648,596 / 34,986 = 104.28	90,966 / 34,986 = 2.60

Another thing that may be interest for the comparative complexity evaluation of the three systems is the average margin and the average extent of superiority of the FS with respect to the other systems for a varying number of inputs and a fixed number of linguistic values per input. For this purpose, we have to consider the FS in a separate pair with each of the other two systems, as shown in Table 9.26–9.27.

Tables 9.26–9.27 show in all cases that the average margin and the average extent of superiority of the FS with respect to the CS and the HS increases with the increase of the linguistic values per input for a fixed number of inputs. This increase is close to linear with respect to both the CS and the HS. The magnitude of this increase is fairly big with respect the CS and more moderate magnitude with respect to the HS.

Table 9.26. Average margin of superiority of the filtered system

Parameter values / Pair	EO^{CS} vs EO^{FS}	EO^{HS} vs EO^{FS}
m = 2,3,4; w = 3	(330 + 1,320 + 4,290) / 3 = 1,980	(330 + 829 + 1,220) / 3 = 793
m = 2,3,4; w = 5	(2,060 + 12,360 + 63,860) / 3 =26,093	(2,060 + 4,515 + 5,970) / 3 = 4,181
m = 2,3,4; w = 7	(7,342 + 50,736 + 361,494) / 3 =139,857	(7,342 + 13,281 + 16,104) / 3 = 12,424
m = 2,3,4; w = 9	(14,328 + 143,280 + 1,303,848) / 3 =487,152	(14,328 + 29,335 + 32,678) / 3 =25,447
m = 2,3,4; w = 11	(27,170 + 326,040 + 3,613,610) / 3 =1,322,273	(27,170 + 54,885 + 55,980) / 3 = 46,011

Table 9.27. Average extent of superiority of the filtered system

Parameter values / Pair	EO^{CS} vs EO^{FS}	EO^{HS} vs EO^{FS}
m = 2,3,4; w = 3	(2.42 + 5.47 + 10.20) / 3 = 6.03	(2.42 + 3.81 + 3.61) / 3 = 3.28
m = 2,3,4; w = 5	(4.16 + 14.65 + 30.56) / 3 = 16.45	(4.16 + 5.98 + 3.76) /3 = 4.63
m = 2,3,4; w = 7	(5.97 + 26.95 + 54.55) / 3 = 29.15	(5.97 + 7.79 + 3.38) / 3 = 5.71
m = 2,3,4; w = 9	(7.79 + 41.46 + 79.37) / 3 = 42.87	(7.79 + 9.28 + 2.96) / 3 = 6.67
m = 2,3,4; w = 11	(9.61 + 57.61 + 104.28) / 3 = 57.16	(9.61 + 10.53 + 2.60) / 3 = 7.58

The last thing that may be interest for the comparative complexity evaluation of the three systems is the overall margin and the overall extent of superiority of the FS with the respect to the other systems for a varying number of inputs and a varying number of linguistic values per input. For this purpose, we have to consider the FS again in a separate pair with each of the other two systems, as shown in Tables 9.28–9.29.

Table 9.28. Overall margin of superiority of the filtered system

Parameter values / Pair	EO^{CS} vs EO^{FS}	EO^{HS} vs EO^{FS}
m = 2,3,4; w = 3,5,7,9,11	(1,980 + 26,093 + 139,857 + + 487,152 + 1,322,273) / 5 = 395,471	(793 + 4,181 + 12,424 + + 25,447 + 46,011) / 5 = 17,771

Table 9.29. Overall extent of superiority of the filtered system

Parameter values / Pair	EO^{CS} vs EO^{FS}	EO^{HS} vs EO^{FS}
m = 2,3,4; w = 3,5,7,9,11	(6.03 + 16.45 + 29.15 + + 42.87 + 57.16) / 5 = 30.33	(3.28 + 4.63 + 5.71 + + 6.67 + 7.58) / 5 = 5.57

It can be seen from Table 9.28 that overall the FS is with more than 395,000 operations more efficient than the CS and with more than 17,000 operations more efficient than the HS. Table 9.29 show that that overall the FS is more than 30 times more efficient than the CS and more than 5 times more efficient than the HS.

As in terms of behaviour the HS is only an approximation of the CS whereas the FS is equivalent to the CS, it is obvious that the FS outperforms significantly the other two systems. Moreover, this superiority is valid for fuzzy systems whose number of rules exceeds 14,000, as shown in Table 9.23.

9.5 Comparative Analysis of Formal Simplification Techniques

The two formal simplification techniques introduced in Sects. 9.2–9.3 are a powerful tool for reducing the quantitative complexity in fuzzy systems. In particular, the aggregation of inconsistent rules and the filtration of non-monotonic rules can be used for reducing the number of rules in SRB systems or MRB systems represented with equivalent SRB systems. This leads to a reduction of the overall amount of operations during the operational stages and substages in fuzzy systems such as fuzzification, inference, i.e. application, implication, aggregation, and defuzification.

From the two formal simplification techniques, aggregation has a fairly limited effect because fuzzy rule bases are rarely inconsistent but if that is the case then there is usually a small number of inconsistent rules. As opposed to this, filtration has a much wider effect because fuzzy rule bases almost always have a big number of non-monotonic rules. The effect of the two techniques determines to a great extent their impact on a fuzzy system which is fairly moderate for aggregation but quite big for filtration. As far as the solution for an AS and a FS is concerned, it is equivalent to the one for a CS, i.e. a system whose inconsistent rules are not aggregated or a system whose non-monotonic rules are not filtered.

The considerations presented above on formal simplification techniques for fuzzy rule bases provide essential information about the main characteristics of these techniques. These characteristics are summarised in Table 9.30.

Table 9.30. Characteristics of formal simplification techniques for fuzzy rule bases

Technique / Characteristic	Effect	Impact	Solution
Aggregation of inconsistent rules	limited	moderate	equivalent
Filtration of non-monotonic rules	wide	big	equivalent

9.6 Application Range of Formal Simplification Techniques

The formal simplification techniques introduced in this chapter are applicable to a wide range of SRB systems. These techniques can be applied to Mamdami, Sugeno and Tsukamoto systems, CON and DIS systems, MO and SO systems, as well as FF and FB systems.

Examples 9.1–9.14 describe explicitly a fuzzy system of Mamdami type. In order to apply the associated rule base simplification algorithms to Sugeno and Tsukamoto systems, the substages of implication and aggregation within the inference stage have to be modified accordingly. In particular, the outputs from the implication substage will not be membership functions for the output but its defuzzified values for the separate rules. These values will then have to be aggregated by a weighted average method to give the overall defuzzified value of the output for all the rules. In this case, inconsistent and non-monotonic rules can still be aggregated and filtered but some adjustments have to be made in order to guarantee that the overall defuzzified value for the output by using the weighted average method will be the same as the one for a CS.

As far as CON and DIS systems are concerned, the formal simplification techniques are directly applicable to all of them. In this case, depending on the particular type of system, i.e. CADR, DADR, CACR or DACR, appropriate modifications have to be made for the application and aggregation substages of the inference stage.

The formal simplification techniques can be extended easily for other types of fuzzification, implication or defuzzification. For example, instead of triangular functions in the fuzzification stage, it is possible to use trapezoidal, Gaussian or other types of membership functions. Also, instead of a truncation type of implication, it is possible to use a scaling type of implication or another implication. And finally, instead of centroid defuzzification it is possible to use other types of defuzzification such as maximum, weighted average, etc. In all these cases, appropriate adjustments have to be made to ensure that the overall defuzzified value of the output is the same as the one for a CS.

The formal simplification techniques have been illustrated only for single simulation cycles of SO systems. However, they can be easily extended to multiple simulation cycles and MO systems. In this case, all procedures presented in this chapter should be applied in exactly the same way to each simulation cycle of each output. This would obviously lead to a linear increase of the associated quantitative complexity, which would be proportional to the number of simulation cycles and the number of outputs.

Although the formal simplification techniques have been demonstrated only for SRB systems, they can be indirectly applied to MRB systems, which can be of either FF or FB type. In this case, the MRB system must be transformed first into an equivalent SRB system.

The formal simplification techniques introduced in this chapter facilitate the complexity management in fuzzy systems. These techniques allow the information contained in the inconsistent or non-monotonic rule base of a SRB system to be compressed in a non-lossy manner by removing the redundancy in the rule base. As a result this compression, the rule base can be reduced and represented equivalently as a consistent and monotonic rule base.

The formal simplification techniques presented here are the last building block in a series of complexity management techniques for fuzzy systems introduced in the preceding chapters. However, all these techniques have been discussed fairly independently from each other so far. Therefore, it would be useful to know how we can make these building blocks stick together in the context of a general framework for complexity management in fuzzy systems. A detailed discussion on this issue is presented in the next concluding chapter.

10 Conclusion

10.1 Formal Approach for Fuzzy Rule Base Compression

This book treats in detail complexity management aspects in fuzzy systems. In particular, Chapters 4–9 are dedicated to different techniques for complexity management. However, there is a common feature uniting most of these techniques in that complexity management in fuzzy systems is usually implemented by compression of the rule base.

In the case of formal presentation, the integer table of a rule base is compressed by a Boolean matrix or binary relation. In formal manipulation, the Boolean matrices or binary relations of two rule bases are compressed into a single Boolean matrix or binary relation by merging. Formal transformation is an extension of formal manipulation whereby the Boolean matrices or binary relations describing a MRB system are compressed into a single Boolean matrix or binary relation describing the equivalent SRB system. And finally, in the case of formal simplification, the Boolean matrix or binary relation of a SRB system is compressed by aggregation of inconsistent rules or filtration of non-monotonic rules.

In this context, formal presentation, manipulation and transformation techniques deal mainly with qualitative aspects of complexity whereby the fuzzy rule bases become more transparent and easier for interpretation. At the same time, formal simplification techniques are focused predominantly on quantitative aspects of complexity whereby the amount of on-line operations and the associated computational times in fuzzy systems are reduced.

10.2 Theoretical Significance of Fuzzy Rule Base Compression

The formal approach of fuzzy rule base compression introduced in this book has a big theoretical significance. It makes good use of mathematics and its power to formalise and facilitate a particular course of action as well as to justify and guarantee the result from this action. In this context, a number of algorithms and numerous examples are given for illustration of the underlying theoretical concepts.

Another interesting feature of the formal approach used is that it leads to non-lossy compression of the rule base. In particular, the compression of the IFS is done in a way which allows its reconstruction, if necessary, and ensures that the solution is uncompromised, i.e. the defuzzified output from the RFS is the same as the one from the IFS. In this sense, the approach is very suitable for both time-critical and safety-critical applications.

Also, this formal approach undoubtedly improves the important attributes of computer speed and intelligence. These attributes are crucial for the successful treatment of time-critical and safety-critical applications of fuzzy systems characterised by quantitative and qualitative complexity.

10.3 Application Framework for Fuzzy Rule Base Compression

The formal approach of fuzzy rule base compression described in this book is applicable to a wide range of fuzzy systems. It would not be an overstatement to say that it can be applied almost universally to any type of fuzzy system irrespective of the number of its inputs, rules, etc. In this context, the algorithm below describes briefly an application framework for the formal approach.

Algorithm 10.1

```
Off-line
1. Present formally a fuzzy system.
2. For a MRB system, go to step 3; for a SRB system, go to step 5.
3. Manipulate formally the constituent SRB systems of the MRB system.
4. Transform formally the MRB system into an equivalent SRB system.
5. For a MO SRB system, go to step 6; for a SO SRB system, go to step 7.
6. Convert the MO SRB system into an equivalent collection of SO SRB systems.
7. For each SO SRB system, sort the inconsistent rules in groups.
8. For each SO SRB system, aggregate the inconsistent rules from each group.
9. For each SO SRB system, sort the non-monotonic rules in groups.

On-line
1. For each SO SRB system, apply the fuzzification stage.
2. For each SO SRB system, apply the application substage of the inference stage.
3. For each SO SRB system, filter the non-monotonic rules from each group.
4. For each SO SRB system, apply the implication substage of the inference stage.
```

5. For each SO SRB system, apply the aggregation substage of the inference stage.
6. For each SO SRB system, apply the defuzzification stage.

It is obvious from Algorithm 10.1 that almost all formal compression techniques from this book are applied in off-line steps 1-9. The only exception is the filtration of non-monotonic rules, which is applied in on-line step 3. The remaining on-line steps, i.e. steps 1-2 and 4-6, are occupied by standard processes in fuzzy systems such as fuzzification, application, implication, aggregation and defuzzification. This algorithm shows that the formal approach for rule base compression fits very well within the established general application framework for fuzzy systems.

The on-line steps in Algorithm 10.1 reflect only one simulation cycle of a fuzzy system. In the case of more simulation cycles, all on-line steps must be applied for each new cycle. In this case, the computations for each SO SRB system may be done in parallel, which would reduce the overall computational time.

10.4 Future Directions for Related Research in Fuzzy Systems

The formal approach of fuzzy rule base compression presented in this book is expected to encourage and stimulate new research in fuzzy systems and related areas. This expectation is based on the fact that this formal approach has a natural overlap with some recent research trends in other types of complex systems, e.g. deterministic and probabilistic systems.

One possibility in this respect would be the extension of fuzzy systems to fuzzy networks, which has already been initiated by the techniques of formal transformation of a MRB system into an equivalent SRB system. In this context, MRB systems can be viewed as networks whose nodes are in the form of SRB systems and whose connections are of FF or FB type.

Another possibility would be the extension of fuzzy systems to fuzzy multi-agent systems. This has also been initiated by the techniques of formal transformation of fuzzy systems whereby the transformation process exhibits features similar to the ones of multi-agent systems. In this context, MRB systems can be viewed as multi-agent systems whose agents are in the form of SRB systems and whose migration is in the form of their relocation during the transformation process.

A third possibility would be the development of a new type of adaptive fuzzy systems by means of formal simplification techniques. In this case, the aggregation of inconsistent rules and the filtration of non-monotonic rules can be viewed as off-line and on-line adaptation of the associated rule base, respectively.

10.5 Overall Book Evaluation

To the best knowledge of the author, this book is a first attempt for addressing the growing problem of complexity in fuzzy systems at a monographic level. As such, the book possibly has some drawbacks, e.g. theoretical bias, simplistic illustrations, partial software implementations, etc. For this reason, the author would be very thankful to colleagues from the international academic community who may be interested in any further theoretical developments or practical applications of the research results presented here.

In spite of the possible drawbacks of this work, the author believes that the timeliness for its publication is more important than a comprehensive coverage. Moreover, the book appears to have dealt successfully with almost all questions raised in Chapter 1. Some of these questions are discussed in the preceding sections of this chapter whereas others are considered briefly below.

For example, the formal approach used can be classified as fundamental and applied science on the basis of its theoretical foundations and suitability for applications. In this sense, it is neither an abstract theory nor empirical practice and it can hopefully stand the potential criticism from opponents of fuzzy logic. Also, the book has managed to open new horizons and suggest viable alternatives to the existing ‘status quo’ in the field of fuzzy systems by successfully replacing the established approach of complexity reduction with the more advanced approach of complexity management. As such, the book has helped the better understanding and analysis of complexity in fuzzy systems which will hopefully make these systems easier to interpret and more enjoyable to work with.

The author believes that this book will help fuzzy logic move a bit closer to the place that it deserves – as a main subject in university curricula and a key area for scientific research. Because undoubtedly fuzzy systems have a great potential that has been only partially explored. And maybe one day fuzzy logic will replace binary logic not only in the world of computing but also far beyond.

References

1. Alcala R, Cano J, Cordon O, Herrera F, Villar P, Zwir I (2003) Linguistic modelling with hierarchical systems of weighted linguistic rules. International Journal of Approximate Reasoning 32 : 187–215
2. Ascia G, Catania V, Russo M (1999) VLSI hardware architecture for complex fuzzy systems. IEEE Transactions on Fuzzy Systems 7/5 : 553–570
3. Baboshin N, Naryshkin D (1990) On identification of multidimensional fuzzy systems. Fuzzy Sets and Systems 35 : 325–331
4. Babuska R (1998) Fuzzy modelling for control. Kluwer, Boston
5. Bolognani S, Zigliotto M (1998) Hardware and software effective configurations for multi-input fuzzy logic controllers. IEEE Transactions on Fuzzy Systems 6/1 : 173–179
6. Boverie S, Demaya B, Lequellec J, Titli A (1993) Fuzzy control of high order systems using a parallel structure of second order blocks. In: IFAC world congress, pp 573–576
7. Bucolo M, Fortuna L, La Rosa M (2004) Complex dynamics through fuzzy chains. IEEE Transactions on Fuzzy Systems 12/3 : 289–295
8. Burke D, Rattan K (1993) A multi-layer motion controller for a mobile robot implemented with fuzzy logic. In: American control conference, pp 2248–2251
9. Chaudhury S, Singh T, Goswami P (2004) Distributed fuzzy case based reasoning. Applied Soft Computing 4 : 323–343
10. Chen S, Yu F, Chung H (2002) Decoupled fuzzy controller design with single-input fuzzy logic. Fuzzy Sets and Systems 129 : 335–342
11. Chung F, Duan J (2000) On multistage fuzzy neural network modelling. IEEE Transactions on Fuzzy Systems 8/2 : 125–142
12. Combs W, Andrews J (1998) Combinatorial rule explosion eliminated by a fuzzy rule configuration. IEEE Transactions on Fuzzy Systems 6/1 : 1–11
13. Cordon O, Herrera F, Zwir I (2002) Linguistic modelling by hierarchical systems of linguistic rules. IEEE Transactions on Fuzzy Systems 10/1 : 2–20
14. Cordon O, Herrera F, Zwir I (2003) Fuzzy modelling by hierarchically built fuzzy rule bases. International Journal of Approximate Reasoning 27 : 61–93
15. Cordon O, Herrera F, Zwir I (2003) A hierarchical knowledge-based environment for linguistic modelling: models and iterative methodology. Fuzzy Sets and Systems 138 : 307–341
16. De Oliveira J, Gomide F (2001) Formal methods for fuzzy modelling and control. Fuzzy Sets and Systems 121 : 1–2
17. De Silva C (1993) Knowledge base decoupling in fuzzy logic control systems. In: American control conference, pp 760–764
18. Dubois D, Prade H, Ughetto L (1997) Checking the coherence and redundancy of fuzzy knowledge rule bases. IEEE Transactions on Fuzzy Systems 5/3 : 398–417

19. Endo Y, Horiuchi K (2004) Risk analysis of fuzzy control systems with (n+1)-inputs and 1-output FLC. Fuzzy Sets and Systems 148 : 341–361
20. Galichet S, Boukezzoula R, Foulloy L (2004) Explicit analytical formulation and exact inversion of decomposable fuzzy systems with singleton consequents. Fuzzy Sets and Systems 146 : 421–436
21. Gegov A (1994) Multilevel intelligent fuzzy control of oversaturated urban traffic networks. International Journal of Systems Science 25/6 : 967–978
22. Gegov A (1995) Multilayer fuzzy control of multivariable systems by passive decomposition. Applied Mathematics and Computer Science 5/4 : 615–633
23. Gegov A (1995) Decentralised fuzzy control of multivariable systems by direct decomposition. Engineering Applications of Artificial Intelligence 8/2 : 201–209
24. Gegov A (1996) Distributed fuzzy control of multivariable systems. Kluwer, Dordrecht
25. Gegov A (1997) Multilayer fuzzy control of multivariable systems by active decomposition. International Journal of Intelligent Systems 12/1 : 83–103
26. Gegov A (1998) Multilayer fuzzy control of multivariable systems by direct decomposition. International Journal of Systems Science 29/8 : 851–862
27. Gegov A (1998) Linguistic decomposition of MIMO fuzzy systems. Research report, Delft University of Technology
28. Gegov A, Frank M (1994) Decentralised fuzzy control of multivariable systems by passive decomposition. Intelligent Systems Engineering 3/4 : 194–200
29. Gegov A, Frank M (1995) Hierarchical fuzzy control of multivariable systems. Fuzzy Sets and Systems 72 : 299–310
30. Gegov A, Frank M (1995) Decomposition of multivariable systems for distributed fuzzy control. Fuzzy Sets and Systems 73 : 329–340
31. Gegov A, Frank M (1995) Reduction of multidimensional relational relations in fuzzy control systems. Systems and Control Letters 25 : 307–313
32. Gegov A, Maketas D (2005) Formal presentation of fuzzy systems with multiple sensor inputs. Sensors and Transducers Magazine 366–373
33. Gegov A, Babuska R, Verbruggen H (1998) Linguistic decoupling in MIMO fuzzy systems. In: European congress on intelligent techniques and soft computing, pp 1620–1624
34. Gegov A, Babuska R, Verbruggen H (1999) Linguistic analysis of interactions in MIMO fuzzy systems. In: IFAC world congress, pp 249–254
35. Golob M (2001) Decomposed fuzzy proportional-integral-derivative controllers. Applied Soft Computing 1 : 201–214
36. Gupta M, Kiszka J, Trojan G (1986) Multivariable structure of fuzzy control systems. IEEE Transactions on Systems, Man and Cybernetics 16/5 : 638–655
37. Guven M, Passino K (2001) Avoiding exponential parameter growth in fuzzy systems. IEEE Transactions on Fuzzy Systems 9/1 : 194–199
38. Hall L (2001) Rule chaining in fuzzy expert systems. IEEE Transactions on Fuzzy Systems 9/6 : 822–828
39. Henderson M, Gill K (1993) Robotic control using fuzzy logic and parallel processing. In: European control conference, pp 481–485
40. Hirota K, Pedrycz W (2002) Data compression with fuzzy relational equations. Fuzzy Sets and Systems 126 : 325–335

41. Huwendiek O, Brockmann W (1999) Function approximation with decomposed fuzzy systems. Fuzzy Sets and Systems 101 : 273–286
42. Jamshidi M (1997) Large scale systems: modelling, control and fuzzy logic. Prentice Hall, Upper Saddle River
43. Jang J, Sun C, Mizutani E (1997) Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Prentice Hall, Upper Saddle River
44. Jia L, Zhang X (1993) Identification of multivariable fuzzy systems through fuzzy cell mapping. In: IFAC World Congress, pp 389–393
45. Joo M, Lee J (2002) Universal approximation by hierarchical fuzzy system with constraints on the fuzzy rule. Fuzzy Sets and Systems 130 : 175–188
46. Joo M, Lee J (2005) A class of hierarchical fuzzy systems with constraints on the fuzzy rules. IEEE Transactions on Fuzzy Systems 13/2 : 194–203
47. Kim K (1982) Boolean matrix theory and applications. Marcel Dekker, New York
48. Kim Y, Ahn S, Kwon W (2000) Computational complexity of general fuzzy logic control and its simplification for a loop controller. Fuzzy Sets and Systems 111 : 215–224
49. Koczy L, Hirota K (1993) Modular rule bases in fuzzy control. In: European congress on fuzzy and intelligent technologies, pp 606–610
50. Lacrose, V (1997) Complexity reduction of fuzzy controllers: application to multivariable control. PhD thesis, Toulouse Laboratory for Systems Analysis and Architecture
51. Lazzerini B, Marcelloni F (2000) Reducing computation overhead in MISO fuzzy systems. Fuzzy Sets and Systems 113 : 485–496
52. Lee M, Chung H, Yu F (2003) Modelling of hierarchical fuzzy systems. Fuzzy Sets and Systems 138 : 343–361
53. Lehmke S, Temme K, Thiele H (1998) Reducing the number of inference steps for multiple-stage fuzzy if-then rule bases. Research report, University of Dortmund
54. Li H, Tso S (1999) Higher order fuzzy control structure for higher order or time-delay systems. IEEE Transactions on Fuzzy Systems 7/5 : 540–552
55. Mar J, Lin HT (2005) A car-following collision prevention control device based on the cascaded fuzzy inference system. Fuzzy Sets and Systems 150 : 457–473
56. Mendel J, Liang Q (1999) Comments on "combinatorial rule explosion eliminated by a fuzzy rule configuration". IEEE Transactions on Fuzzy Systems 7/3 : 369–371
57. Mollov, S (2002) Fuzzy Control of multiple-input multiple-output processes. PhD thesis, Delft University of Technology
58. Negnevitsky M (2002) Artificial intelligence: a guide to intelligent systems. Pearson Education, Harlow
59. Nurnberger A (2003) A hierarchical recurrent neuro-fuzzy model for system identification. International Journal of Approximate Reasoning 32 : 153–170
60. Pal N, Eluri V, Mandal G (2002) Fuzzy logic approaches to structure preserving dimensionality reduction. IEEE Transactions on Fuzzy Systems 10/3 : 277–286
61. Passino K, Yurkovich S (1998) Fuzzy Control. Addison-Wesley, Menlo Park

62. Pedrycz W, Reformat M (1997) Rule-based models of multivariable functions. Fuzzy Sets and Systems 90 : 235–253
63. Qi X, Chin T (1997) Genetic algorithms based fuzzy controller for high order systems. Fuzzy Sets and Systems 91 : 279–284
64. Raju G, Zhou J, Kisner R (1991) Hierarchical fuzzy control. International Journal of Control 54/5 : 1201–1216
65. Rigatos G, Tzafestas S (2002) Parallelization of a fuzzy control algorithm using quantum computation. IEEE Transactions on Fuzzy Systems 10/4 : 451–460
66. Ross T (2004) Fuzzy logic with engineering applications. Wiley, Chichester
67. Roubos H, Setnes M (2001) Compact and transparent fuzzy models and classifiers through iterative complexity reduction. IEEE Transactions on Fuzzy Systems 9/4 : 516–524
68. Setnes M (1995) Fuzzy rule base simplification using similarity measures. MSc thesis, Delft University of Technology
69. Setnes M, Babuska R, Verbruggen H (1998) Rule-based modelling: precision and transparency. IEEE Transactions on Systems, Man and Cybernetics 28/1 : 165–169
70. Simon D (2000) Design and rule base reduction of a filter for the estimation of motor currents. International Journal of Approximate Reasoning 25 : 145–167
71. Sun Q, Li R, Zhang P (2003) Stable and optimal adaptive fuzzy control of complex systems using fuzzy dynamic model. Fuzzy Sets and Systems 133 : 1–17
72. Tao C (2001) Comments on “reduction of fuzzy rule base via singular value decomposition”. IEEE Transactions on Fuzzy Systems 9/4 : 675–676
73. Trillas E, Alsina C (2002) On the law [if (p and q) then r] = [(if p then r) or (if q then r)] in fuzzy logic. IEEE Transactions on Fuzzy Systems 10/1 : 84–88
74. Tu K, Lee T, Wang W (2000) Design of a multi-layer fuzzy logic controller for multi-input multi-output systems. Fuzzy Sets and Systems 111 : 199–214
75. Wan F, Shang H, Wang L, Sun Y (2005) How to determine the minimum number of fuzzy rules to achieve given accuracy: a computational geometric approach to SISO case. Fuzzy Sets and Systems 150 : 199–209
76. Wang L (1999) Analysis and design of hierarchical fuzzy systems. IEEE Transactions on Fuzzy Systems 7/5 : 617–624
77. Xiong N, Litz L (2002) Reduction of fuzzy control rules by means of premise learning – method and case study. Fuzzy Sets and Systems 132 : 217–231
78. Xu C (1991) Linguistic decoupling control of fuzzy multivariable processes. Fuzzy Sets and Systems 44 : 209–217
79. Xu C, Lu Y (1989) Decoupling in fuzzy systems: a cascade compensation approach. Fuzzy Sets and Systems 29 : 177–185
80. Yam Y, Baranyi P, Yang C (1999) Reduction of fuzzy rule base via singular value decomposition. IEEE Transactions on Fuzzy Systems 7/2 : 120–132
81. Yan J, Ryan M, Power J (1994) Using fuzzy logic. Prentice Hall, New York
82. Yeh Z (1998) A cross-coupled bistage fuzzy controller for biaxis servomechanism control. Fuzzy Sets and Systems 97 : 265–275
83. Yeh Z, Li K (2004) A systematic approach for designing multistage fuzzy control systems. Fuzzy Sets and Systems 143 : 251–273
84. Zeng X, Keane J (2005) Approximation capabilities of hierarchical fuzzy systems. IEEE Transactions on Fuzzy Systems 13/5 : 659–672

Index