Fuzzy Logic in Control

René Jager

Fuzzy Logic in Control

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College van Dekanen aangewezen,
op maandag 26 juni 1995 te 10:30 uur
door

René JAGER

elektrotechnisch ingenieur,
geboren te Amsterdam

Dit proefschrift is goedgekeurd door de promotor:
Prof. ir. H.B. Verbruggen

Ir. P.M. Bruijn heeft als begeleider in belangrijke mate
aan het totstandkomen van het proefschrift bijgedragen.

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Jager, René

Fuzzy logic in control / René Jager. - [S.l. : s.n.]. - Ill.
Thesis Technische Universiteit Delft. - With index, ref. -
With summary in Dutch.
ISBN 90-9008318-9
NUGI 832
Subject headings: fuzzy logic / fuzzy control.

Contents

1 Introduction

Nowadays, fuzzy control is a hot topic. Journals and books on fuzzy set theory are published, symposia on fuzzy modeling and control are organized and software packages for fuzzy control are released. Despite all the publications on fuzzy sets and fuzzy control this thesis has been written, because, in our opinion, there exists much misunderstanding about fuzzy control. This thesis aims at providing an analysis of fuzzy control to clear this up this misunderstanding.

The first section of this chapter describes the reasons for writing this thesis in more detail. The second section briefly addresses the place of fuzzy control within the field of control. In this thesis a controller is considered as a (static) function. This "functional approach" to control is discussed in section 1.3.1. Stability issues are briefly addressed in section 1.3.2. The place of fuzzy logic and fuzzy control within the field of artificial, or computational, intelligence is addressed in section 1.4. The last section can be used as a road-map for this thesis: it helps you to avoid areas in the field of fuzzy control that you are already familiar with, and it can serve as a guide to direct you to areas of your own interest.

1.1 Why yet another work on fuzzy control?

When reaching for this thesis, one might ask: "Why yet another thesis on fuzzy control?". Considering the available amount of books, journals and conference proceedings, this question might seem legitimate. However, on one hand, many books and journals are very mathematical and are not focused on fuzzy control. On the other hand, many books and articles only focus on simple fuzzy control problems and applications, and are incomplete with respect to the theoretical framework in which fuzzy control resides. The contributions that do focus on fuzzy control, including the contributions to the enormous number of symposia and workshops on fuzzy systems, do, in our opinion, not place fuzzy control in the right perspective. There is still a lot of vagueness and misunderstanding around the topic, which, we think, is not necessary. Thus, this more or less answers the question why this thesis is written: a *demystification of fuzzy control* and on the same time a *profilation of fuzzy control*. The aim of this is to place fuzzy control in the right perspective. For this reason, only half of this thesis focuses on fuzzy control and the other half is used for underlying concepts, necessary for a good understanding of fuzzy control, and more general topics related to fuzzy logic and fuzzy control.

Nowadays books are being published which are written for people not yet or barely familiar with fuzzy logic and fuzzy control. Fuzzy control is treated as such, without questioning the underlying concepts. However, those underlying concepts are very important, as will be shown in this thesis, and can lead to conclusions that suggest the elimination of fuzzy calculus within fuzzy control and use well-known interpolation techniques: in that case the fuzzy part is limited to user-interfacing. In the field of fuzzy control many software tools are available and are mostly promoted with slogans which state that fuzzy control is the solution to all our problems. Considering this hype around fuzzy control and the resulting misconception is reason enough to take a close look at what a fuzzy controller actually is and how it works.

Approximate reasoning, based on fuzzy set theory and possibility theory, provides several techniques to reason with fuzzy and uncertain concepts in knowledge-based systems. Applying fuzzy techniques in knowledge-based systems can provide a knowledge representation and inference which is closer to the way humans express their knowledge and reason with it than in the case of conventional knowledge-based systems based on "classical logic". This comes very close to the field of natural language understanding and processing. It is hard to imagine oneself talking with someone else which cannot distinguish between "fairly true" and "very true", although both qualifications are rather vague. Fuzzy logic and approximate reasoning enable us to (partly) model human reasoning by means of computer implementations. When we take a look into the future and imagine humans communicating with computers on a, compared to nowadays, high level of intelligence, then this requires the modeling of human reasoning and natural language. Approximate reasoning provides a theoretical framework to perform this modeling. Fuzzy control can be regarded as a small part within the framework of approximate reasoning. For this reason a chapter is dedicated to fuzzy logic in knowledge-based systems, extending the "narrow" view used in fuzzy control literature to the "broader" framework where fuzzy control resides in.

1.2 Why fuzzy control and where does it fit in?

Today, a lot of interest from industry in fuzzy systems can be noticed. In western countries, this is currently mostly limited to an orientation to the field of fuzzy control. One of the reasons for this orientation towards fuzzy control is because competing companies (mostly Japanese) are using or starting to use fuzzy control in competing products and advantages of doing so are reported in literature. Many examples of fuzzy control applications exist in consumer product.
Hence, the growing interest in fuzzy control is understandable, but the question then rises why do the (competing) companies use fuzzy control? Considering the discussions in literature and on newsgroups[*] and mailing lists[†] the following reasons can be extracted:

1. Fuzzy control is a "new technology" and therefore can be used to avoid patent-claims of similar solutions for technical problems, which are based on a different technique.
2. Nowadays, in Japan, fuzzy is "wanted" by consumers, since it represents "high-tech". In this case fuzzy techniques are mostly used as a marketing tool.
3. The development of fuzzy controllers is easier to learn and requires less skilled personnel than the development of conventional controllers. This results in cheaper production.
4. Fuzzy controllers provide more robustness than conventional control.
5. Fuzzy controllers are more appropriate to control nonlinear processes.

For academia, reasons 1 and 2 should not play a role, so in the following we will focus on the last three reasons.

Fuzzy controllers are represented by if-then rules and thus can provide a user-friendly and understandable knowledge representation. One can see this as a (very) high-level programming language, where the program consists of if-then rules and the compiler and/or interpreter results in a nonlinear control algorithm. Hence, programming by means of qualitative statements, represented by means of if-then statements, to obtain a program working on quantitative domains, provided by sensor and actuator signals. Intuitively, this entails loss of information, because there is no unique translation from a qualitative entity to a quantitative representation except for some special cases. For example, there

[*]The main newsgroup on fuzzy systems is `comp.ai.fuzzy`.

[†]A major mailing list in this field is `fuzzy-mail@vexpert.dbai.tuwien.at`, which also mirrors newsgroup `comp.ai.fuzzy`.

is no unique translation from "large voltage" to a real-valued voltage and vice versa.
Because in control the results of a controller are expected to be precise quantities since those results are signals for actuators of motors, valves, pumps, heaters, etc., special additional techniques are necessary for the translation of qualitative information to quantitative information. It is an advantage though, that (complex) control strategies which are "known" by operators or process engineers in the form of experience and/or domain knowledge, can be "programmed" and maintained in a user-friendly and understandable way.

It is often claimed that fuzzy control provides more robustness. However, no research results has been found that prove that fuzzy controllers are more robust than conventional controllers in general. As will be shown in this thesis, a fuzzy controller is in fact a static nonlinearity and whether this is more robust than a conventional controller depends on the rules defining this static nonlinearity.
However, when the variations of process parameters are (partly) known, a fuzzy controller can be designed to be less sensitive for those parameter changes and thus be more robust than a comparable linear controller. Indeed, this can be compared to gain-scheduling (Åström and Wittenmark, 1984) with the difference that a fuzzy controller implicitly provides bumpless transfers from one set of controller parameters to another.
So, *fuzzy controllers are more robust* should be interpreted as *fuzzy controllers can be more robust to known parameter changes*. How to device a fuzzy controller which is more robust still remains a problem since it depends mainly on knowledge of the process to be controlled.

Another claim often made about fuzzy control is that fuzzy control is more appropriate to control nonlinear processes. Whether a fuzzy controller, or a nonlinear controller in general, is in principle able to control a nonlinear process sufficiently, depends primarily on the chosen inputs of the controller.
Fuzzy controllers are often said to be superior to their corresponding linear controller to control nonlinear processes. For fuzzy PID-like controllers, this is only true for a small set of problems, namely when the nonlinearity of the process can be written as a function of the error and its derivatives, being the input signals of the controller. This is normally not the case, because the error and error change are not only determined by the process, but also by the externally defined reference signal. If it is desired that a controller is able to "capture" nonlinearities of the process to be controlled, the controller should not be based on the error and its derivatives: one could use, for example, the reference signal or the process output (and their derivatives) as additional inputs of the controller.
In general it can be stated, based on the same controller inputs, that: *a fuzzy controller can control a nonlinear process as least as good as its corresponding linear controller can do, just because a fuzzy controller can control a linear process as least as good as its corresponding linear controller can do*. Thus, in principle, a fuzzy controller is more

capable of controlling a nonlinear process, but additional knowledge of the nonlinearities of the process is needed.

Summarizing the previous discussion, we can state the following: fuzzy control provides a method to construct controller algorithms in a user-friendly way and provides the ability to capture the nonlinear control behavior of humans which has proven to be appropriate for many complex tasks. Having a design method for controllers which is closer to human thinking and perception can reduce development time and requires less skilled personnel to design controllers. The economical benefit of this is trivial. It should be noted that the robustness of human controllers is primarily due to their ability to adapt to a changing environment and their learning capability. Building this ability into fuzzy controllers is beyond today's application of fuzzy control in consumers products, but research on adaptive fuzzy control has been done for quite some time.

When the current state of fuzzy control is considered, it can be stated that the main areas in which fuzzy control can be applied, are the following:

1. Processes which can be adequately controlled by humans and the controller to be designed has sensors to provide similar information used by humans to control the process. Examples are the application of fuzzy logic in automatic transmission for cars, washing machines, etc. Nowadays, there are many applications of fuzzy logic in consumer products.

2. Processes which are currently controlled by (basically) linear control algorithms and need further development resulting in nonlinear control algorithms which are known by operators or process engineers. Mamdani (1994) states that:

   > *"Fuzzy logic is successful because it replaces the classical PID controller. When tuned, the parameters of a PID controller affect the shape of the entire control surface. Because fuzzy logic control is a rule-based controller, the shape of the control surface can be individually manipulated for the different regions of the state space, thus limiting possible effects to neighboring regions only."*

   As a starting point for a fuzzy controller, the linear controller that is currently used to control the process in question can be used, because, under certain conditions (section 4.5.2), a fuzzy controller can be designed to "emulate" a linear controller.

A more or less critical point of view on fuzzy control is given by Elkan (1994), who states that fuzzy controllers are characterized by the following properties:

- fuzzy controllers use typically fewer than 100 rules; often even fewer than 20 rules;

- the knowledge within a fuzzy controller is usually shallow, both statically and dynamically;
- the knowledge within a fuzzy controller typically reflects correlations between controller inputs and outputs;
- the numerical parameters of a fuzzy controller are tuned in a learning process;
- fuzzy controllers use fuzzy logic operators.

Elkan (1994) states that the success of fuzzy control is mainly because of the first four properties and that the use of fuzzy logic is not essential. In our opinion these statements about fuzzy controllers are mostly correct, but it should be expressed that these properties do not have to be interpreted as being negative, since it allows user-friendly development of (nonlinear) controllers. Although the rest of Elkan's provocative article does not show a deep understanding of fuzzy logic and related topics in our opinion, it started a discussion on fuzzy logic and fuzzy control which contributes to further discussion of the strengths and limitations of fuzzy logic (Zadeh, 1994b).

1.3 Fuzzy control and control systems theory

In this section we describe the view on control that is used in this thesis. An important issue is the way a controller is considered. This is described in the first subsection. The second subsection addresses briefly the aspect of stability.

1.3.1 Controllers as static functions

Today, most controllers are implemented by computer algorithms. This implies that the controller inputs are measured at certain sampling rates. For example, the linear part of a classical PID[*] controller can be represented by:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt} \tag{1.1}$$

where $u(t)$ is the control signal fed to the process to be controlled and $e(t)$ is the error signal: the difference between the desired and measured process output. A computer implementation of a PID controller can be expressed as a difference equation:

$$u_{\mathrm{PID}}[k] = u_{\mathrm{PID}}[k-1] + k_I e[k] + k_P \Delta e[k] + k_D \Delta^2 e[k] \tag{1.2}$$

[*]Proportional-Integral-Differential.

with:

$$\begin{aligned}\Delta e[k] &= e[k] - e[k-1] \\ \Delta^2 e[k] &= \Delta e[k] - \Delta e[k-1]\end{aligned}$$

When we consider a PI or PD controller, the following difference equations can be derived:

$$u_{\mathrm{PI}}[k] = u_{\mathrm{PI}}[k-1] + k_{\mathrm{I}} e[k] + k_{\mathrm{P}} \Delta e[k] \tag{1.3a}$$

$$u_{\mathrm{PD}}[k] = k_{\mathrm{P}} e[k] + k_{\mathrm{D}} \Delta e[k] \tag{1.3b}$$

Equations (1.2), (1.3a) and (1.3b) can be compared with the algebraic representations of a (hyper)plane:

$$y = a_0 + \sum_{i=1}^{n} a_i x_i \tag{1.4}$$

A schematic representation for (1.3a) and (1.3b) is shown in figure 1.1. When we consider a mapping from controller inputs to controller outputs in general (MIMO[*] system), the controller function is represented by a mapping:

$$\boldsymbol{y} = f(\boldsymbol{x}) \tag{1.5}$$

This is the way controllers are considered in this thesis: controller outputs are static functions (mappings) of the controller inputs. Dynamical behavior of a controller, like differential or integral action, are "emulated" by extending the controller function to more inputs. Those inputs are delays or differences of other inputs and outputs. Hence, a controller is considered to consist of a static controller function and additional "prefiltering" and "postfiltering" parts to obtain delayed inputs, input differences, integrations, limited signals, etc. Note that this approach is generally also used in the field of neural networks.

To be consistent with the above described "functional view" of controllers, the variable naming in the rest of this thesis will be conform (1.5). Hence, controller output(s) will be addressed as y's and input(s) will be addressed as x's. Moreover, this notation is used for fuzzy systems in general in this thesis. Using the same variable naming convention for both models of processes and controllers is because the functional behavior of a model or a controller is similar in our opinion; they both have to fit a certain (non)linear mapping of inputs to outputs, meeting a number of predefined criteria.

[*]Multiple-Inputs-Multiple-Outputs.

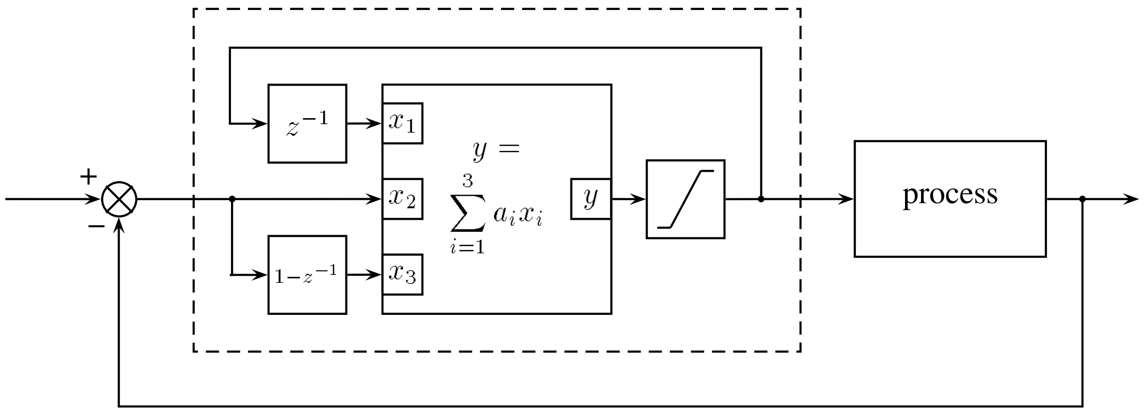


Figure 1.1: *Example of PI and PD controllers regarded as a static mapping using "prefiltering" and "postfiltering" blocks. Parameters are chosen as follows:* $a_1 = 1$, $a_2 = k_\mathrm{I}$ *and* $a_3 = k_\mathrm{P}$ *for a PI controller, and* $a_1 = 0$, $a_2 = k_\mathrm{P}$ *and* $a_3 = k_\mathrm{D}$ *for a PD controller.*

1.3.2 Stability issues

Leaving the aspect of stability out of a thesis on control is hardly possible. Therefore the stability issue is addressed in this section, but it is also the only place where it is addressed. This is based on two reasons. Firstly, fuzzy controllers can be regarded as nonlinear controllers and for this reason it is difficult to obtain general results on the analysis and design of fuzzy controllers (Driankov et al., 1993). The second reason is clearly expressed by Mamdani (1993):

> *"Industry has never put forward a view that the mathematical stability analysis is a necessary and sufficient requirement for the acceptance of a well designed control system. That is merely the view that control system scientists wished to put forward, but it has never gained currency outside academic circles. . . . Prototype testing is more important than stability analysis; stability analysis by itself can never be considered a* ***sufficient*** *test. Moreover, in any practically useful methodology, a stability analysis step would need to be made a desirable but an optional step; it cannot be a* ***necessary*** *step."*

These statements might seem rather strong and indeed Mamdani was much criticized for these statements, but in our opinion his statements contain a lot of truth. The *stability proofs for fuzzy controllers* found in literature are restricted to the cases where fuzzy controllers are simple, for example PID-like fuzzy controllers, and where the process to be controlled is stable itself; see for example the article of Malki, Li and Chen (1994). In most cases the stability "proofs" are trivial due to the simplicity of the controller and process. If the process cannot be modeled mathematically, for example the control of cement kilns (Östergaard, 1990), then stability proofs cannot be given at all. Because the first applications of fuzzy control were controllers for processes which could not (and cannot) be modeled mathematically, like cement kilns, the criticism that fuzzy control does not allow stability analysis was not valid since stability analysis is based on a mathematical model of the process and such models were not available (Mamdani, 1993). Considering the numerous applications of fuzzy control in consumer products, one can question the need for a mathematical stability analysis for these, in this case rather simple, control problems. We conclude with another quote from Mamdani (1993), because his paper *"Twenty years of fuzzy control: experiences gained and lessons learnt"* contains a view on control which closely resemblances ours, including the following statement:

> *"Stability is still an important issue but a different way has to be found to study it. In the final analysis all one may be able to do is to build prototypes for the purpose of approval certification. This is a well tried and tested approach used in industry and there is no reason why it may not suffice with control systems as well."*

1.4 Relation to artificial and computational intelligence

Fuzzy logic is regarded as one of the artificial intelligence (AI) techniques, from which "conventional" expert systems, neural networks and genetic algorithms are well known. It can be disputed whether, for example, neural networks and genetic algorithms should be considered as artificial intelligence techniques. Zadeh (1994a) proposes the denomination *soft computing* to address the field of neural networks, genetic algorithms, fuzzy logic and combinations of those. Today, the field of fuzzy control and modeling is in many publications considered to overlap with the field of neural networks. Many publications on these "neuro-fuzzy systems" or "fuzzy neural networks" can be found in literature. In our opinion this overlap of neural networks and fuzzy systems is purely based on functional equivalence and not based on the underlying ideas. However, considering this functional equivalence and the many publications addressing the merging of these techniques, it seems valid to address those techniques by one name.

An important concept of fuzzy set theory and fuzzy logic is the linguistic variable (Zadeh, 1994a). In their survey, Dubois and Prade (1991) state that *"the main motivation of fuzzy set theory is apparently the desire to build up a formal, quantitative framework that* *captures the vagueness of human knowledge as it is expressed via natural languages"*. From this point of view, fuzzy logic can be regarded as part of the field of artificial intelligence, because fuzzy logic and approximate reasoning (see chapter 6) can provide a framework for natural language understanding and processing, and modeling of the way humans reason and communicate. Although many authors claim fuzzy logic to be a suitable framework for dealing with uncertainty in expert systems, one can hardly find a real application of such a system (Elkan, 1993). As opposed to the large number of theoretical papers on fuzzy logic in expert systems, there exist only a small number of reported prototypes of such systems (Graham, 1991).

But what about the large number of reported applications of fuzzy control in consumer products? In our opinion these applications are not successful because they are applications of AI, but because they provide a user-friendly method to implement nonlinear controller functions. In other words: *"Fuzzy controllers have encountered great success by providing an efficient way of implementing an interpolative mechanism, not only in small, but also in very large and complex problems"* (Dubois et al., 1994). Zadeh (1994b) expressed the comparison between (fuzzy) control and more general knowledge-based systems as follows: *"Basically, what differentiates control applications from knowledge-based systems applications is that in control the main problem that has to be addressed is that of imprecision. By contrast, in the case of knowledge-based systems, one has to come to grip with both imprecision and uncertainty"*. This becomes also clear when one notices that fuzzy control can be considered as only a small part of the theoretical framework of approximate reasoning.

1.5 What to expect: a road-map for this thesis

Before providing a road-map for this thesis, it should be stated that this thesis does not contain parts which describe applications of fuzzy control for a specific industrial or laboratory set-up. This is considered not necessary since there exist numerous commercial applications of fuzzy control in consumer products as already pointed out in section 1.2. When appropriate, small examples are used to clarify described methods or algorithms. The aim of this thesis is to investigate the fundamentals of fuzzy control and to provide an insight in the underlying theory, not to show the success in some application areas.

In the remainder of this section a road-map to this thesis is presented. This makes it possible for the reader to exclude parts through which he or she already is already familiar with, or directly go to parts of personal interest. Before giving a short description of the contents of following chapters in this thesis, it should be noticed that each chapter has a final section in which conclusions and/or a short summary are given. Most chapters also contain a section which addresses practical aspects, mainly based on issues concerning computer implementation.

The next chapter describes partly the theory of fuzzy sets. This includes the basic notion of a fuzzy set, properties of fuzzy sets and operations on fuzzy sets. The aim of this chapter is not to be complete, but to provide the parts of fuzzy set theory which are necessary to understand the remainder of this thesis.

Chapter 3 will address the basics of fuzzy logic and reasoning. This assumes that the reader is familiar with fuzzy set theory (chapter 2). Different logical operations will be described and discussed, including logical connectives and implications. Reasoning with fuzzy logic is described for single rules and sets of fuzzy rules (rule bases).

Fuzzy control is described and discussed in chapter 4. A detailed description is given as well as advantages and disadvantages of different approaches. In this chapter fuzzy control is considered to be based on fuzzy rules which directly "connect" controller inputs to controller outputs ("flat" rule base). This type of fuzzy control is mostly applied and chaining of fuzzy rules is not considered. An analysis of fuzzy controllers with respect to different controller parameters is provided.

Adaptive fuzzy control is an extension of fuzzy control and is described in chapter 5. Two main approaches within the field of adaptive fuzzy control can be distinguished: approaches based on the self-organizing controller introduced by Procyk and Mamdani (1979) and approaches based on gradient-descent adaptation. Adaptive fuzzy controllers or models based on the latter approach are often referred to as *"fuzzy neural networks"* or *"neuro-fuzzy systems"*. Both types are addressed in this chapter.

In chapter 6 approximate reasoning is addressed. Approximate reasoning provides a framework to model natural language understanding and human reasoning. Within this field of research one can distinguish several points of view and approaches, which will be described in this chapter. The practical application of approximate reasoning is not always straightforward and difficulties encountered are also discussed in this chapter. The basics of possibility theory, as part of the approximate reasoning framework, are explained in this chapter, but a good understanding of fuzzy set theory and fuzzy logic (chapters 2 and 3) is assumed.

The last chapter gives conclusions based on the contents of this thesis, general remarks on addressed topics and suggestions for further research.

2

Fuzzy sets and relations

This chapter contains the basics of fuzzy set theory that are necessary for a correct understanding of the rest of this thesis. If the reader is already familiar with the field of fuzzy set theory, this chapter will probably contain nothing new. If the reader is not familiar with the field it will serve as an introduction to fuzzy set theory. This chapter will start with a section about what fuzzy sets are and how they are related to classical (ordinary) set theory. After this, a number of properties of fuzzy sets are given in section 2.1.2. A special type of fuzzy set, referred to as fuzzy numbers, is described in section 2.1.3. The extension principle is one of the basic concepts in fuzzy set theory and allows mathematical concepts to be extended for use with fuzzy sets, and is addressed in section 2.1.4. The union and intersection of fuzzy sets, and the complement of fuzzy sets, are presented in section 2.3. Linguistic modifiers, usually referred to as hedges, are addressed in section 2.2. Section 2.4 deals with fuzzy relations. A final section summarizes this chapter.

2.1 Fuzzy sets

Zadeh (1965) introduced fuzzy sets, although the underlying idea or ideas close to it had already been recognized earlier by others, mainly by philosophers. A comprehensive overview is given in the introduction of the readings in “Fuzzy Sets for Intelligent Systems”, edited by Dubois, Prade and Yager (1993). Fuzzy sets as defined by Zadeh are described in this section. In addition to a basic description and some examples, properties of fuzzy sets are addressed (section 2.1.2). Section 2.1.3 describes a special type of fuzzy set: fuzzy numbers and intervals. Extending mathematical operations to operate on fuzzy sets is possible by applying the extension principle, which is described in 2.1.4.

2.1.1 What are fuzzy sets?

Classical set theory is well known, and in the field of fuzzy set theory is usually called “classical set theory” instead of just “set theory”. The membership $\mu_A(x)$ of x of a classical set A, as subset of the universe X, is defined by:

$$\mu_A(x) = \begin{cases} 1, & \text{iff} \quad x \in A \\ 0, & \text{iff} \quad x \notin A \end{cases} \tag{2.1}$$

This means that an element x is either a member of set A $(\mu_A(x) = 1)$ or not $(\mu_A(x) = 0)$. Classical sets are also referred to as *crisp* sets. For many classifications, however, it is not quite clear whether x belongs to a set A or not. For example, if set A represents PCs which are too expensive for a student’s budget, then it is obvious that this set has no clear boundaries. Of course, it could be said that a PC priced at \$2500 is too expensive, but what about PCs priced at \$2495 or \$2502? Are those PCs too expensive or not? Clearly, a boundary could be determined above which a PC is too expensive for the average student, say \$2500, and a boundary below which a PC is certainly not too expensive, say \$1000. Between those boundaries, however, there remains a vague interval in which it is not quite clear whether a PC is too expensive or not. In this interval, a grade could be used to classify the price as partly too expensive. This is where fuzzy sets come in: sets of which the membership has grades in the interval [0,1].

A fuzzy set, introduced by (Zadeh, 1965), is a set with graded membership in the real interval: $\mu_A(x) \in [0, 1]$. A fuzzy set A, a fuzzy subset of X, is denoted by:

$$A = \sum_{i=1}^{m} \mu_A(x_i)/x_i \tag{2.2a}$$

$$= \mu_A(x_1)/x_1 + \cdots + \mu_A(x_m)/x_m \tag{2.2b}$$

where $\mu_A(x)$ is known as the membership function, and where X is known as the *universe of discourse*. When X is not finite, a fuzzy set A is defined by:

$$A = \int_X \mu_A(x)/x \tag{2.2c}$$

In this thesis the latter is primarily used to denote fuzzy sets. A simple example of a fuzzy set is one representing PCs *too expensive* for a student's budget. This fuzzy set is depicted in figure 2.1. One can see that if the price is below \$1000 the PC is certainly not too expensive, and if the price is above \$2500 the PC is fully classified as too expensive. In between, an increasing membership of the fuzzy set *too expensive* can be seen. It is not necessary that the membership linearly increases with the price, nor that there is a discontinuous transient for \$1000 and \$2500, as will be seen later in this thesis. The choice of the membership function of the fuzzy set is arbitrary.

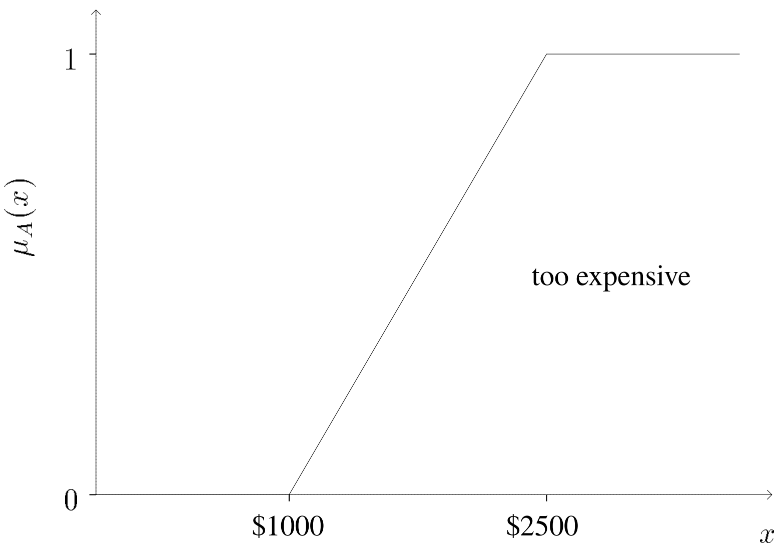


Figure 2.1: *Fuzzy set A representing PCs too expensive for a student's budget. The price is represented by variable x.*

2.1.2 Properties of fuzzy sets

In this section a number of properties of fuzzy sets is given. The aim of this section is not to be complete, but to provide those parts of fuzzy set theory necessary to make the rest of this thesis understandable. It is mostly a list of commonly used fuzzy set properties.

The *height* of a fuzzy set A, *hgt*(A), is defined by:

$$\mathrm{hgt}(A) = \sup_{x \in X} \mu_A(x) \tag{2.3}$$

and fuzzy sets with a height equal to 1 are called *normal*. Fuzzy sets called *subnormal* are characterized by *hgt*$(A) < 1$. The *core* of a fuzzy set, also referred to as *kernel* or *nucleus*, is a crisp subset of X:

$$\mathrm{core}(A) = \{x \in X \mid \mu_A(x) = 1\} \tag{2.4}$$

The *support* of a fuzzy set is also a crisp subset of X:

$$\mathrm{supp}(A) = \{x \in X \mid \mu_A(x) > 0\} \tag{2.5}$$

If the support of a fuzzy set is finite, it is called *compact support*. Figure 2.2 shows schematically the height, core and support of a fuzzy set.

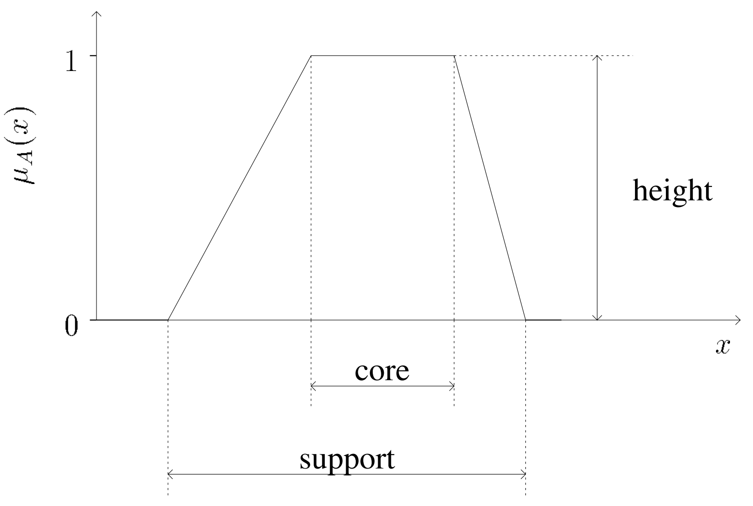


Figure 2.2: *Height, core and support of a fuzzy set.*

The elements of x where $\mu_A(x) = \frac{1}{2}$ are called *crossover points*. The α*-cut* of a fuzzy set is defined by:

$$\alpha\text{-}\mathrm{cut}(A) = \{x \in X \mid \mu_A(x) \geq \alpha\} \tag{2.6}$$

An α*-cut* of a fuzzy set is often referred to as a *level set*. A *strong* α*-cut* is defined by:

$$\overline{\alpha}\text{-}\mathrm{cut}(A) = \{x \in X \mid \mu_A(x) > \alpha\} \tag{2.7}$$

Thus the core of a fuzzy set can be defined by an α*-cut*, with $\alpha = 1$:

$$\mathrm{core}(A) = 1\text{-}\mathrm{cut}(A)$$

and the support of a fuzzy set can be defined by a strong α*-cut* ($\overline{\alpha}$*-cut*), with $\alpha = 0$:

$$\mathrm{supp}(A) = \overline{0}\text{-}\mathrm{cut}(A)$$

Another property of fuzzy sets used in this thesis is whether a fuzzy set is *convex* or not. A convex fuzzy set is characterized by:

$$\forall\, x_1, x_2, x_3 \in X,\;\; x_1 \leq x_2 \leq x_3 \;\rightarrow\; \mu_A(x_2) \geq \min(\mu_A(x_1), \mu_A(x_3)) \tag{2.8}$$

where x_1, x_2 and x_3 are values in X. Hence, the fuzzy set in figure 2.2 is convex. Convexity of a fuzzy set can play an important role when analysing a fuzzy controller in combination with another important property of fuzzy sets: whether the fuzzy sets form a fuzzy partition. When N_A fuzzy sets A_j are fuzzy subsets of universe X, such a tuple of fuzzy sets $(A_1, \ldots, A_j, \ldots, A_{N_A})$ is called a *fuzzy partition* when:

$$\forall x \in X,\; \sum_{j=1}^{N_A} \mu_{A_j}(x) = 1 \tag{2.9}$$

provided that $A_j \neq \emptyset$ and $A_j \neq X$. An example of a fuzzy partition is given in figure 2.3. A fuzzy partition formed by fuzzy sets which are normal and convex, does not contain more than two overlapping fuzzy sets.

2.1.3 Fuzzy numbers and intervals

A *fuzzy number* is a special type of fuzzy set. A fuzzy set F is a fuzzy number, usually a fuzzy subset of $\mathbb{R}$, if it meets the following criteria:

- the fuzzy set is convex, as defined by (2.8);
- the fuzzy set is normalized: *hgt*$(F) = 1$;
- the membership function of the fuzzy set is piecewise continuous;
- the core of the fuzzy set consists of one value only.

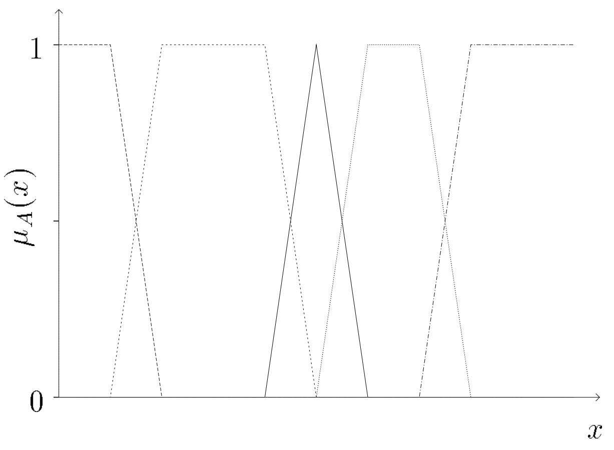


Figure 2.3: *A fuzzy partition.*

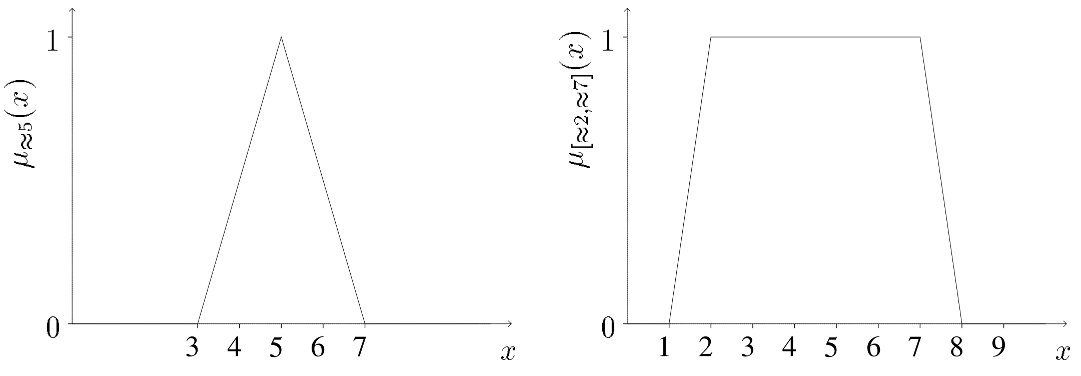


Figure 2.4: *Fuzzy sets representing fuzzy number "about 5" and fuzzy interval "from about 2 to about 7".*

Thus a fuzzy number is always a fuzzy set, but a fuzzy set is not always a fuzzy number. An example of a fuzzy number *"about 5"* is shown in figure 2.4a. Mathematical operations like addition, subtraction, etc. can be extended for use with fuzzy numbers by means of the *extension principle*, which is addressed in section 2.1.4.

In addition to fuzzy numbers one can consider *fuzzy intervals* (Dubois and Prade, 1988). A fuzzy interval is a fuzzy set with the same restrictions as these defined for fuzzy numbers, but with the exception that the core is no longer restricted to one point only. An example of the fuzzy interval *"from about 2 to about 7"* is shown in figure 2.4b.

2.1.4 The extension principle

The *extension principle* was introduced by Zadeh (1975) and is one of the most important elements of fuzzy set theory. As Dubois and Prade (1980) put it: *it provides a general method for extending non-fuzzy mathematical concepts in order to deal with fuzzy quantities*. The extension principle allows the extension of a mapping f from points in X to fuzzy subsets of X:

$$f(A) = f(\mu_1/x_1 + \cdots + \mu_n/x_n) \tag{2.10a}$$

$$\triangleq \mu_1/f(x_1) + \cdots \mu_n/f(x_n) \tag{2.10b}$$

A simple example is given by the following. Consider the fuzzy set *"about 5"* with a discrete universe, $x_i \in \mathbb{Z}$ and mapping f representing the square. Then the application of the extension principle results in:

$$(\text{“}\textit{about 5}\text{”})^2 = (\tfrac{1}{2}/4 + 1/5 + \tfrac{1}{2}/6)^2 = (\tfrac{1}{2}/16 + 1/25 + \tfrac{1}{2}/36)$$

The extension principle applied to a function or mathematical operation $f(x_1, \ldots, x_n)$ is defined by:

$$\mu_B(y) = \sup_{\substack{x_1,\ldots,x_n \\ y=f(x_1,\ldots,x_n)}} \mu_A(x_1, \ldots, x_n) \tag{2.11a}$$

$$= \sup_{\substack{x_1,\ldots,x_n \\ y=f(x_1,\ldots,x_n)}} \min(\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n)) \tag{2.11b}$$

where the Cartesian product $A_1 \times \cdots \times A_n$ is used to represent the multi-dimensional fuzzy set A, because the fuzzy set A is usually not available. This implies A to be the largest set whose projections on $X_1, \ldots, X_n$ are $A_1, \ldots, A_n$, respectively (Dubois et al., 1993a). See section 2.4.1 for more details on projections.

Hence, the extension principle allows the derivation of a fuzzy set B on y by $B = f(A_1, \ldots, A_n)$, where A_i are fuzzy sets in X_i. In other words: the extension principle can be used to extend normal mathematical operations to operations with fuzzy sets. Another way to write (2.11a) is:

$$B = f(A_1, \ldots, A_n) \tag{2.11c}$$

$$= \int_{X_1 \times \ldots \times X_n} \min(\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n))/f(x_1, \ldots, x_n) \tag{2.11d}$$

To clear things up, here is a small example. Suppose the addition of two fuzzy numbers is desired, then the extension principle can be used to derive a resulting fuzzy set for the outcome of the addition. Let us take the fuzzy number also used in section 2.1.3, *"about 5"* with membership function $\mu_{\approx 5}(x_1) = \max(1 - \frac{1}{2}|x_1 - 5|, 0)$, shown in figure 2.7a, and a fuzzy number to add to this, *"about 2"*, with membership function $\mu_{\approx 2}(x_2) = \max(1 - |x_2 - 2|, 0)$, shown in figure 2.7b.

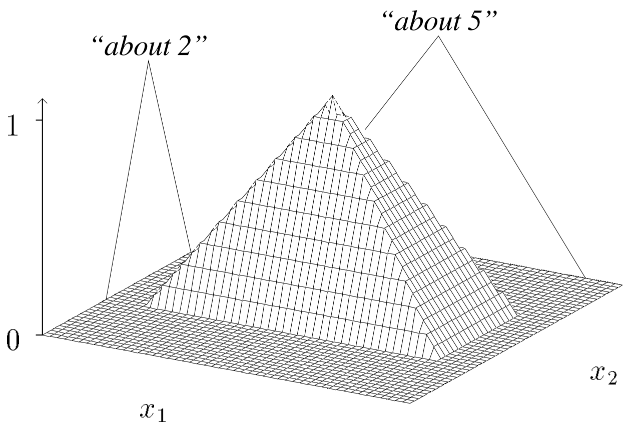


Figure 2.5: *Cartesian product of "about 5" and "about 2".*

The Cartesian product of *about 5*, projected on an x-axis, and *about 2*, projected on a y-axis, is shown in figure 2.5. In figure 2.6, a contour plot of the Cartesian product space $X_1 \times X_2$

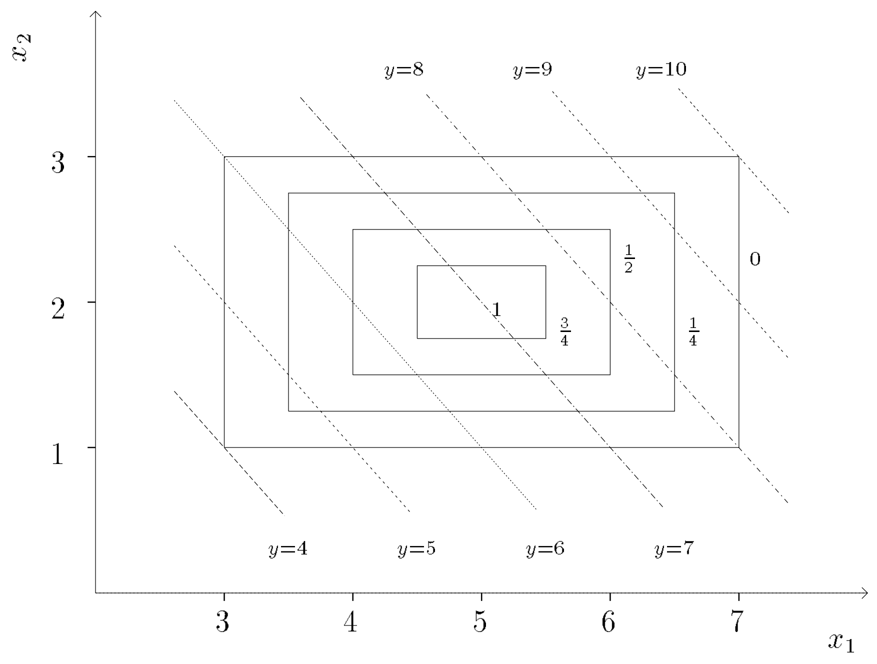


Figure 2.6: *Contour plot for product space* $X_1 \times X_2$ *with fuzzy sets* $\mu_{\approx 5}(x_1) = \max(1 - \frac{1}{2}|x_1 - 5|, 0)$ *and* $\mu_{\approx 2}(x_2) = \max(1 - |x_2 - 2|, 0)$. *Isolines for* $y = 4, 5, \ldots, 10$ *and isocurves for* $\max(\mu_{\approx 5}(x_1), \mu_{\approx 2}(x_2)) = 0, \frac{1}{2}, \ldots, 1$ *are shown.*

is shown with isolines for y. The rectangles represent isocurves of *min*$(\mu_{\approx 5}(x_1), \mu_{\approx 2}(x_2))$ for a number of membership values: 0, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$ and 1 (which is only one point). The next step in applying the extension principle is to take the supremum of *min*$(\mu_{\approx 5}(x_1), \mu_{\approx 2}(x_2))$ for each isoline in order to obtain the membership value for $\mu_B(y)$ with $y = x_1 + x_2$. In figure 2.6 several of those lines are shown. The resulting membership function $\mu_B(y)$ is *max*$(1 - \frac{1}{3}|y - 7|, 0)$, shown in figure 2.7c. In figure 2.7, the calculus of the fuzzy numbers is shown schematically. When the universes are discrete, $x_i \in \mathbb{Z}$, the example simplifies to the following:

$$\begin{aligned} \text{“\textit{about 5}”} + \text{“\textit{about 2}”} &= (\tfrac{1}{2}/4 + 1/5 + \tfrac{1}{2}/6) + (1/2) \\ &= \tfrac{1}{2}/6 + 1/7 + \tfrac{1}{2}/8 \end{aligned}$$

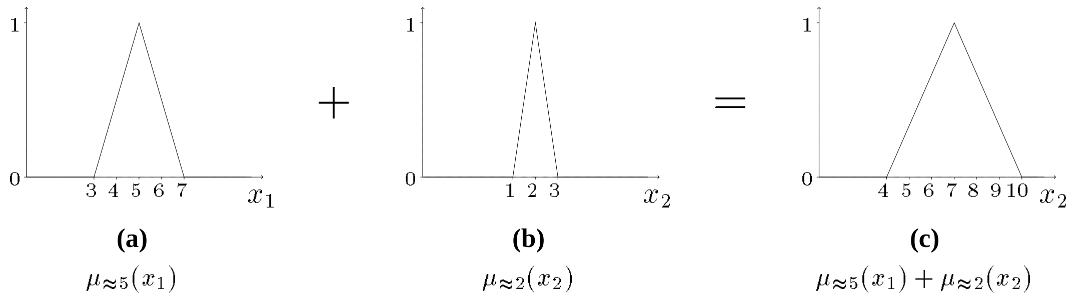


Figure 2.7: *Addition of two fuzzy numbers using extension principle.*

Using triangular-shaped membership functions, the determination of the outcome is more or less trivial for addition and subtraction. However, in the case of more complexly shaped membership functions, the application of the extension principle can involve a rather severe calculational load due to calculus on product spaces. Also in the case of more complex functions or mathematical operations, the application of the extension principle often cannot be simplified. For example, calculation of the product instead of the addition of two fuzzy numbers will change the isolines for y in figure 2.6 into hyperbolic isocurves. The product of two fuzzy numbers with triangular-shaped membership functions will therefore not result in a triangular-shaped membership function for the outcome. From the extension principle it can be derived that an operation $f(\cdot)$ on fuzzy sets which are fuzzy subsets of the same universe can be written as (Kandel, 1986):

$$f(A_1, A_2) = \bigcup_{\alpha \in \langle 0,1]} f(\alpha\text{-cut}(A_1), \alpha\text{-cut}(A_2)) \tag{2.12}$$

Since the level sets are crisp intervals (or combinations of them), the operation $f(\cdot)$ can be applied straightforwardly. However, in practice $\alpha \in \langle 0, 1]$ forces a discretization of α for many cases. Only in specific cases is an analytical solution "easy" to obtain, for example the addition of two fuzzy numbers as shown in figure 2.7.

2.1.5 Fuzzy set representations

In fuzzy set theory most operations are defined for continuous universes. The definitions include operations on discrete universes as special cases. In practice fuzzy sets are stored in computer memory by data structures and operations on fuzzy set are implemented by computer algorithms. Since most applications of fuzzy set theory are computer-based, it is necessary to consider fuzzy set representations (Yamamoto, 1994). The following three types of fuzzy set representation can be distinguished:

1. **Functional representation**. This type of fuzzy set representation uses functional descriptions to represent fuzzy sets:

   $$\mu_A(x) = f(x) \tag{2.13}$$

   An example is the functional description of a triangular-shaped fuzzy set:

   $$\mu_A(x) = \frac{|x - a|}{b}$$

   Using functions to represent fuzzy sets in practice poses difficulties when combinations with other fuzzy sets are being made. The use of symbolic calculus is only limited to simple cases due to the complexity of the operations. Although it is possible to "store" operations in a symbolic way, the results of operations cannot be derived symbolically and thus an approximation of the result of an operation has to be made. Hence, discretizations are necessary in practical applications.

2. **Paired representation**, defines a fuzzy set by:

   $$\mu_A(x) = \mu_1/x_1 + \mu_2/x_2 + \cdots + \mu_n/x_n \tag{2.14}$$

   This representation is natural for fuzzy sets on discrete domains. For example, consider a fuzzy set "friends of John", which is a set of persons identified by their name (x_i) and a grade (μ_i) attached to them, representing the degree of friendship.

3. **Level-set representation** describes a fuzzy set by its level sets (α-cuts):

$$\mu_A(x) = \sup_{\alpha \in \langle 0,1]} \alpha\text{-cut}(\mu_A(x)) \tag{2.15}$$

   Since $\alpha \in \langle 0, 1]$, this representation method is only applicable if α is discretized.

From the above listed types of fuzzy set representation it can be derived that for discrete universes of discourse the paired representation is the obvious choice for computer implementation. For continuous domains the function representation is only useful in case of simple operations. Using approximations of the fuzzy set, several choices are possible, each with its own advantages and disadvantages:

1. **Vector representation** represents a fuzzy set on a continuous domain by a paired fuzzy set with an equidistant discretized domain. This type of representation is called vector representation, since only the membership values are stored as a vector. This type of approximation of a fuzzy set can lead to a problem when a numerical value has to be represented. For example, a fuzzy set represented by only one point (singleton), which does not match one of the discretizations, cannot be represented using a vector representation. As an approximation of a singleton membership function $\mu_A(x)$ it is possible to assign values to the neighboring two membership grades μ_l and μ_r, which fulfill:

$$\frac{\mu_l x_l + \mu_r x_r}{\mu_l + \mu_r} = \mu_A^{-1}(1) \tag{2.16a}$$

$$\mu_l + \mu_r = 1 \tag{2.16b}$$

   which would result in:

$$\mu_l = \frac{\mu_A^{-1}(1) - x_r}{x_l - x_r} \tag{2.17a}$$

$$\mu_r = 1 - \mu_l \tag{2.17b}$$

   Note, however, that this approximation fuzzifies the originally crisp set (numerical value).

2. **Point-wise representation** is a paired representation on a nonequidistant discretized domain. Intermediate points are obtained by means of interpolation. This type of representation can eliminate the representation problem due to equidistant discretizations (see vector representation). This point-wise representation is often used

in commercial software for fuzzy control. A simple example is a triangular-shaped fuzzy set which only needs three points μ_i/x_i to represent the membership function of the fuzzy set. More complex shaped fuzzy sets can be approximated to any degree which provides a trade-off between memory requirements and accuracy of the fuzzy set representation.

3. **Discrete level-set representation**. Discretizations of the levels results in a discrete approximation of a fuzzy set, which can be stored as a set of intervals (each interval is a classical set) with a grade attached to it. The support of the fuzzy set can be included as an open interval. In case non-convex fuzzy sets are involved, a level set can consist of more than one interval. Operations on fuzzy sets which are subsets of the same universe can be defined as operations on their level sets as shown by (2.12) in section 2.1.4. In case of operations on product spaces the use of the level-set representation can be quite complicated and the number of operations on classical set can depend exponentially on the number of fuzzy sets involved in the operation.

Summarizing, we can state that in computer implementation it is in most cases unavoidable to use discretized approximations of fuzzy sets. When analytical solutions exist for certain operations it is advisable to use the analytical descriptions of the fuzzy sets involved. Therefore it is desired to have fuzzy sets stored in such a way that it can provide different types of representations when needed, because for different operations different fuzzy set representations are more appropriate.

2.2 Hedges: linguistic modifiers

Linguistic modifiers can be used to modify the meaning of a fuzzy set. For example, the linguistic modifier *very* can be used to change the meaning of *big* to *very big*. Several authors (Hellendoorn, 1990; Zimmermann, 1985) have addressed linguistic modifiers for fuzzy sets, also known as *hedges*. Examples of hedges are: *very*, *slightly*, *more-or-less*, etc. The following two approaches to hedges can be distinguished:

- powered hedges (Zimmermann, 1985)
- shifted hedges (Lakoff, 1973)

These two approaches to hedges are described in the next two sections. Section 2.2.3 addresses a type of shifted hedge, referred to as *scaled hedge*, that has some of the characteristic properties of powered hedges.

2.2.1 Powered hedges

The powered hedges operate on grades of membership and are represented by (Zimmermann, 1985):

$$m_p(A) = \int_X \mu_A^p(x)/x \tag{2.18}$$

where m_p is the linguistic modifier and p is a parameter specific to a certain linguistic modifier and $p \in \langle 0, 1]$. For most hedges default values for p are chosen, like 2 for *very*, $\frac{1}{2}$ for *more-or-less*, etc. For example:

$$\text{very}(A) = \int_X \mu_A^2(x)/x$$

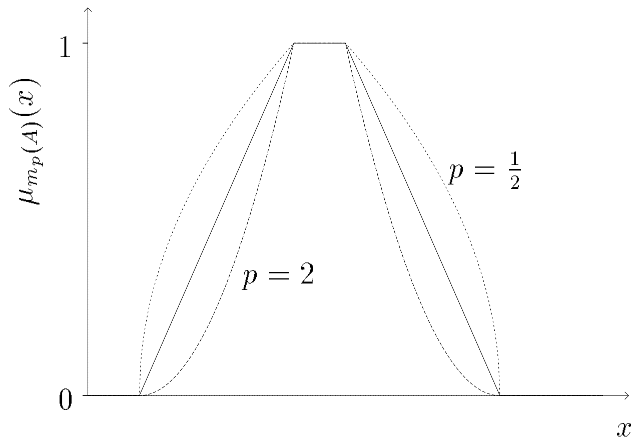


Figure 2.8: *Fuzzy set* A *(solid) and resulting fuzzy sets* very$(A) = A^2$ *(dashed, inner set,* $p = 2$*) and* more-or-less$(A) = \sqrt{A}$ *(dotted, outer set,* $p = \frac{1}{2}$*) using powered hedge operations.*

In figure 2.8, an example is given for the linguistic modifiers *very* and *more-or-less*. The advantage of the powered-hedges approach is that for each hedge a standard operation can be defined by choosing a standard value for p. Note the following properties of m_p for values of p:

$$0 < p < 1 : \text{fuzzy set is dilated: } m_p(A) \supset A$$

$p = 1$: fuzzy set is not modified: $m_p(A) = A$

$p > 1$: fuzzy set is concentrated: $m_p(A) \subset A$

Other characteristic properties of powered hedges are the fact that the support and core of a fuzzy set are not changed by the operation defined for powered hedges, since $1^p = 1$ and $0^p = 0$ for allowed values for p.

2.2.2 Shifted hedges

The shifted hedges (Lakoff, 1973; Hellendoorn, 1990) are defined by:

$$m_s(A) = \int_X \mu_A(x - s)/x \tag{2.19}$$

where m_s is a linguistic modifier and s represents the magnitude of the shift. The value of s has different values within one hedge. Several authors, among which Pétieau et al. (1990), have provided schemes for applying a shifted hedge to a fuzzy set. When applying the shifted-hedge approach of the linguistic modifier *very* on a trapezoidal-shaped fuzzy set, the value of s will be positive on the left side of the center of the fuzzy set and negative on the right side. In this way the fuzzy set is concentrated, resulting in $m_s(A) \subset A$. Dilation of a fuzzy set, for example the hedge *more or less*, is obtained by an inverse operation: $m_s(A) \supset A$.

The method described by Pétieau et al. (1990) applied to a fuzzy set A is shown in figure 2.9. In the case of the hedge *very*, the method consists of making the support of the fuzzy set equal to the core and reducing the core by the same quantity, if possible. For example, in the case of triangular-shaped membership functions, such a reduction of the core is not possible since the core consists of only one point. For the linguistic modifier *more-or-less* a complementary method is applied, since it is considered dual to *very* (see figure 2.9). For different types (shapes) of fuzzy sets, different schemes can be defined.

2.2.3 Scaled hedges

To combine the advantageous property of powered hedges, namely that a standard operation on a fuzzy set can be defined for a specific hedge, with the property of the shifted hedges approach that the “shape” of the fuzzy set is contained after applying the hedge,

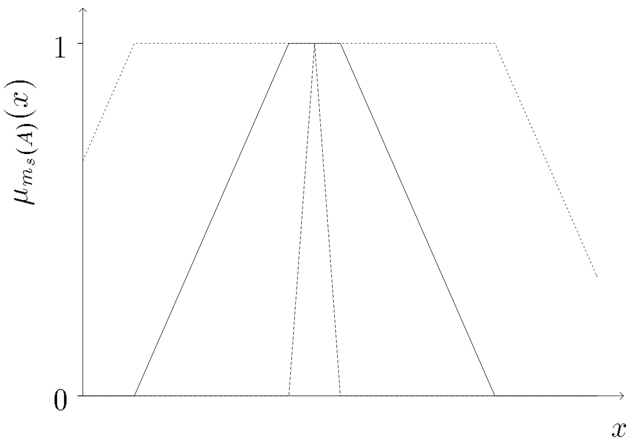


Figure 2.9: *Fuzzy set A (solid) and resulting fuzzy sets* very(A) *(dashed, inner set) and* more-or-less(A) *(dotted, outer set) using shifted hedge operations according to Pétieau, Moreau and Willaeys (1990).*

let us define *scaled hedges*. The general form of the modifier function used with scaled hedges is:

$$m_c(A) = \int_X \mu(c(x - r_A) + r_A)/x = \int_X \mu(cx + (1 - c)r_A)/x \tag{2.20}$$

where:

c : scaling factor of modifier m_c
r_A : reference point for fuzzy set A

The scaling factor c can be standard for a specific linguistic modifier. The same values as those for the powered hedges can be chosen; for example, using the linguistic modifier *very* ($c = 2$) can be defined by:

$$\text{very}(A) = \int_X \mu(2(x - r_A) + r_A)/x = \int_X \mu(2x - r_A)/x$$

The reference point r_A is a characteristic point of fuzzy set A. In the case of convex membership functions, the value of this reference point can be chosen to the center of the core of the membership function. In the case of monotonic membership functions, this reference point could be chosen equal to the supremum of X, if it exists. If the supremum of X does not exist or is not usable in practice, an arbitrary reference point has to be chosen, which acts as a virtual supremum.

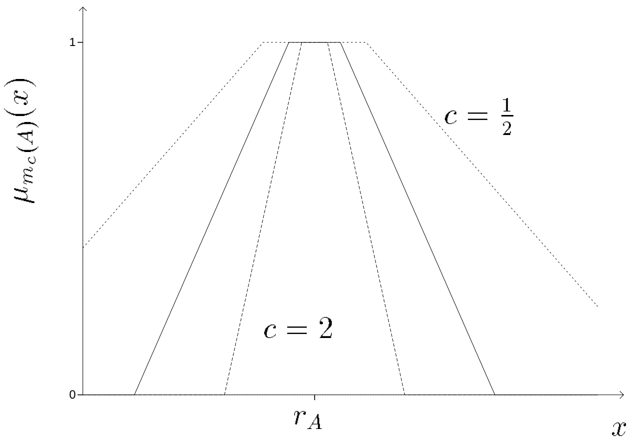


Figure 2.10: *Fuzzy set A (solid) and resulting fuzzy sets* very(A) *(dashed, inner set, $c = 2$) and* more-or-less(A) *(dotted, outer set, $c = \frac{1}{2}$) using scaled hedge operations relative to r_A.*

In figure 2.10, a fuzzy set, together with the results of the linguistic modifiers *very* and *more or less* are shown. As can be seen, the modifiers preserve the original shape of the basic membership function. This is also the case in the shifted-hedges approach, but is not the case in the powered-hedges approach. Applying the scaled-hedges approach as a special case of the shifted-hedges approach, the operation on fuzzy sets is defined by the hedge which is used and not by the hedge in combination with the type or shape of fuzzy set. For example, the method shown in figure 2.9 for application of the shifted-hedges approach in the case of trapezoidal-shaped fuzzy sets cannot be applied in the same way if the fuzzy set is crisp, although the membership function of a crisp set can be seen as a special case of a trapezoidal-shaped membership function.

2.3 Operations on fuzzy sets

As operations are defined on classical sets, similar operations are defined on fuzzy sets. The intersection and union of two sets and the complement of a set are known from classical set theory. Set-theoretic operations like intersection, union and complement are uniquely defined for classical sets and are shown in table 2.1.

Table 2.1: *Set-theoretic operations in classical set theory.*

A	B	$A \wedge B$	$A \vee B$	$\neg A$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

These operations are also defined in fuzzy set theory. However, due to the fact that membership values are no longer restricted to $\{0, 1\}$ but can have any value in the interval $[0, 1]$, these operators cannot be uniquely defined. In the following subsection we go further into detail with respect to union and intersection of fuzzy sets (section 2.3.1) and the complement of a fuzzy set (section 2.3.2).

2.3.1 Union and intersection

The extension of the intersection and union of two classical sets to the intersection and union of two fuzzy sets is not uniquely defined. It is clear that intersection and union operations for fuzzy sets should be subject to the intersection and union of classical sets, because a classical set can be seen as a special case of a fuzzy set. Zadeh (1965) proposed to use the following definitions:

$$\begin{aligned} \mu_{A\cap B} &= \min(\mu_A(x), \mu_B(x)) && \text{(intersection)} \\ \mu_{A\cup B} &= \max(\mu_A(x), \mu_B(x)) && \text{(union)} \end{aligned}$$

If we restrict $\mu_A(x)$ and $\mu_B(x)$ to values in $\{0, 1\}$, then indeed these operators reduce to the intersection and union as defined for classical sets. However, an infinite number of possible definitions can be chosen to implement intersection and union. General forms of intersection and union are represented by *triangular norms* (T-norms) and *triangular conorms* (T-conorms or S-norms), respectively.

A T-norm is a two-place function from $[0,1] \times [0,1]$ to $[0,1]$ satisfying the following criteria:

T-1 $\mathrm{T}(a,1) = a$

T-2 $\mathrm{T}(a,b) \leq \mathrm{T}(c,d)$, whenever $a \leq c$, $b \leq d$

T-3 $\mathrm{T}(a,b) = \mathrm{T}(b,a)$

T-4 $\mathrm{T}(\mathrm{T}(a,b),c) = \mathrm{T}(a,\mathrm{T}(b,c))$

Triangular norms satisfy the restriction:

$$\mathrm{T_W}(a,b) \leq \mathrm{T}(a,b) \leq \min(a,b) \tag{2.21}$$

where T_W is the T-norm according to Weber (1983), also known as *drastic product* and defined by:

$$\mathrm{T}_W(a,b) = \begin{cases} a, & \text{if } b = 1 \\ b, & \text{if } a = 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.22}$$

The upper and lower boundary for a general T-norm are shown in figure 2.11a and 2.11c, respectively. The conditions defining a T-conorm (S-norm) are, besides **T-2**, **T-3** and **T-4**:

S-1 $\mathrm{S}(a,0) = a$

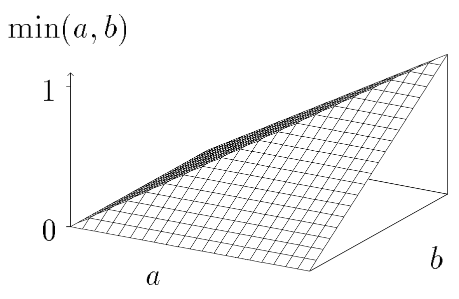
These criteria restrict a general T-conorm by:

$$\max(a,b) \leq \mathrm{S}(a,b) \leq \mathrm{S_W}(a,b) \tag{2.23}$$

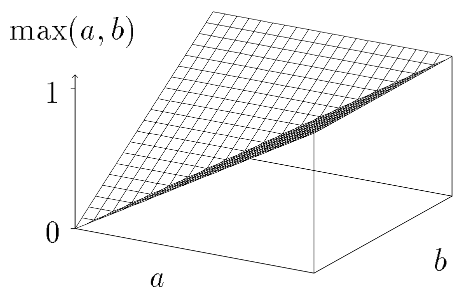
where S_W is the S-norm (T-conorm) according to Weber (1983), also known as *drastic sum* and defined by:

$$\mathrm{S}_W(a,b) = \begin{cases} a, & \text{if } b = 0 \\ b, & \text{if } a = 0 \\ 1, & \text{otherwise} \end{cases} \tag{2.24}$$

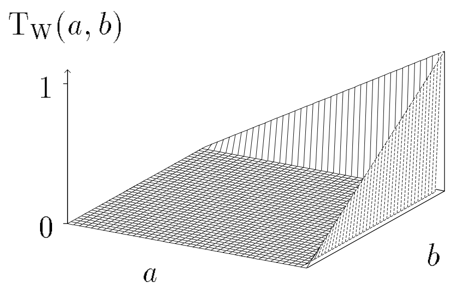
and is shown in figure 2.11d. Figure 2.11b shows the lower boundary for a general S-norm: the *max* operator. A fairly complete overview of T- and S-norms is given by Bellman and



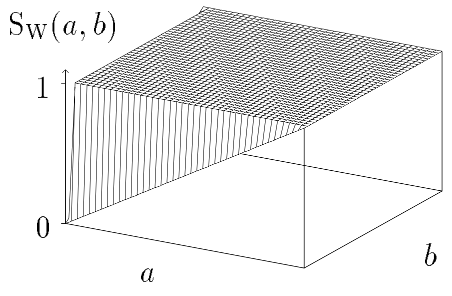
(a)
Upper boundary for T-norm.



(b)
Lower boundary for T-conorm.



(c)
Lower boundary for T-norm, given by (2.22).



(d)
Upper boundary for T-conorm, given by (2.24).

Figure 2.11: *Upper and lower bounds for T-norms and T-conorms.*

Giertz (1973) and Gupta and Qi (1991b). In table A.2 in appendix A a large number of T-norms and -conorms are given.

A property of the combination of a T-norm and an S-norm which is used in the rest of this thesis is the *c*-duality of a T-norm and -conorm. A T-norm and an S-norm are *c*-dual if the following holds:

$$\mathrm{T}(a,b) = \mathrm{c}(\mathrm{S}(\mathrm{c}(a),\mathrm{c}(b))) \tag{2.25}$$

where *c* is a fuzzy complement. The complement of a fuzzy set is described in the following section.

2.3.2 Complement of fuzzy sets

The complement $\overline{A}$ of a fuzzy set A is defined by:

c-1 $\mathrm{c}(0) = 1$

c-2 $\mathrm{c}(a) < \mathrm{c}(b)$, whenever $a > b$

c-3 $\mathrm{c}(\mathrm{c}(a)) = a$

Besides $c(a) = 1 - a$, which is the original representation according to Zadeh, the λ-complement according to Sugeno (1977) could be used:

$$\mu_{\overline{A}^{\lambda}}(x) = \frac{1 - \mu_A(x)}{1 + \lambda \mu_A(x)} \tag{2.26}$$

with $\lambda > 0$. An example of the λ-complement of a fuzzy set for several values of λ is given in figure 2.12.

However, if more severe requirements are introduced, for example, by Gaines (1976):

c-4a $\mu_{\overline{A}}(x_1) + \mu_{\overline{A}}(x_2) = 1$, whenever $\mu_A(x_1) + \mu_A(x_2) = 1$

requiring that if the sum of two fuzzy sets is 1, the sum of the complements of those sets also equals 1, or by Bellman and Giertz (1973):

c-4b $\mu_A(x_1) - \mu_A(x_2) = \mu_{\overline{A}}(x_1) - \mu_{\overline{A}}(x_2)$

requiring that a change in the membership function has the opposite change in the complement of the membership function, then $c(a) = 1 - a$ is the only solution. Table A.1 in appendix A presents a summary of commonly known fuzzy complement operators.

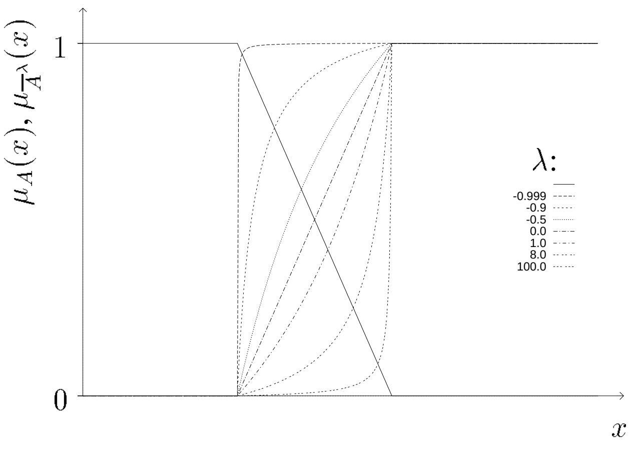


Figure 2.12: *Example of λ-complement of fuzzy set for different values of λ.*

2.4 Fuzzy relations

So far only fuzzy sets with membership functions of one variable have been considered. Fuzzy sets can, however, be extended to have higher dimensional membership functions. These multi-dimensional fuzzy sets are normally referred to as *fuzzy relations*. An n-ary fuzzy relation R in $X_1 \times \cdots \times X_n$ is a (multi-dimensional) fuzzy subset of $X_1 \times \cdots \times X_n$ and is denoted by:

$$R = \{\mu_R(x_1, \ldots, x_n)/(x_1, \ldots, x_n) \mid x_1 \in X_1, \ldots, x_n \in X_n\} \tag{2.27}$$

or using a notation analog to (2.2c):

$$R = \int_{X_1} \cdots \int_{X_n} \mu_R(x_1, \ldots, x_n)/(x_1, \ldots, x_n) \tag{2.28a}$$

$$= \int_{X_1 \times \cdots \times X_n} \mu_R(x_1, \ldots, x_n)/(x_1, \ldots, x_n) \tag{2.28b}$$

Such a fuzzy relation can represent an association or correlation between elements of the product space. An example of such an association is the linguistic statement *approximately equal*, of which the fuzzy relation $R_{\approx}(x, y)$ is shown in figure 2.13.

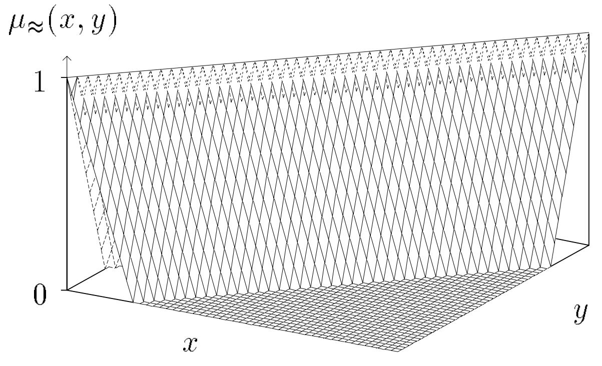


Figure 2.13: *Fuzzy relation representing* approximately equal *with membership function* $\mu_{\approx}(x,y)$.

In a discrete form, using discretizations at integer values in the range $[0, 10]$, this fuzzy relation is described by:

$$\begin{aligned} R_{\approx} &= 1/(0,0) + \tfrac{1}{2}/(1,0) + \tfrac{1}{2}/(0,1) + 1/(1,1) \\ &\quad + \tfrac{1}{2}/(2,1) + \tfrac{1}{2}/(1,2) + 1/(2,2) + \tfrac{1}{2}/(3,2) \\ &\quad + \cdots + \tfrac{1}{2}/(10,9) + \tfrac{1}{2}/(9,10) + 1/(10,10) \end{aligned}$$

In the continuous case, assuming x and y are real numbers, the fuzzy relation from figure 2.13 would be denoted by:

$$R_{\approx} = \int_{X \times Y} \mu_{R_{\approx}}(x,y)/(x,y) \tag{2.29a}$$

with the membership function:

$$\mu_{R_{\approx}}(x,y) = \max(1 - \tfrac{1}{2}|x-y|, 0)/(x,y) \tag{2.29b}$$

Although an increasing fuzziness for increasing values of the variables x and y would be closer to the way humans tend to interpret *approximately equal*, it is not relevant right now; the important thing to remember is that fuzzy relations can be used to model linguistic associations, correlations, relations or, following Dubois and Prade (1980), correspondences. This includes statements like, for example, *smaller than*, *about twice as old*, *much cheaper than*, ...

The properties of fuzzy sets and operations on fuzzy sets are extendable to properties of fuzzy relations and operations on fuzzy relations (being regarded as multi-dimensional fuzzy sets). Properties like height and support can be determined for fuzzy relations in the same manner as for fuzzy sets. Operations like *α-cut*, T-norms and T-conorms can be applied to fuzzy relations only when the relations are fuzzy subsets of the same universe(s).

2.4.1 Projection and cylindrical extension

Zadeh (1975) defined the *projection* of a fuzzy relation R in $X^i = X_{i_1} \times \ldots \times X_{i_k}$, by:

$$\mathrm{proj}(R; X^i) = \int_{X^i} \sup_{x_{j_1},\ldots,x_{j_l}} \mu_R(x_{i_1}, \ldots, x_{i_k})/(x_{i_1}, \ldots, x_{i_k}) \tag{2.30}$$

where R is a fuzzy subset of $X^n = X_1 \times \ldots \times X_n$ and $X^i \times X^j = X^n$. The indices $j_1, \ldots, j_l$ are a complementary to $i_1, \ldots, i_k$ with respect to the indices $1, \ldots, n$. This definition might seem complicated, but it is actually very simple. In fact, the projection mechanism eliminates dimensions of the product space a fuzzy relation is a fuzzy subset of, by taking the supremum of the membership function for the dimension(s) to be eliminated. In figure 2.14 an example is given which shows the projection defined by (2.30) of a fuzzy relation on $X \times Y$ to a fuzzy set on Y:

$$B = \mathrm{proj}(R; Y) = \int_Y \sup_y \mu_R(x, y)/y \tag{2.31}$$

Besides the projection mechanism for fuzzy relations, Zadeh (1975) defined the *cylindrical extension* of a fuzzy relation (or set) by:

$$\mathrm{cext}(R; X^n) = \int_{X^n} \mu_R(x_{i_1}, \ldots, x_{i_k})/(x_1, \ldots, x_n) \tag{2.32}$$

where R is a fuzzy relation on X^i. This means that a fuzzy relation or set is extended over an enclosing Cartesian product space with the restriction that, if R is a fuzzy set on X^n and $X^n \subset X^m$:

$$R = \mathrm{proj}(\mathrm{cext}(R; X^m); X^n) \tag{2.33}$$

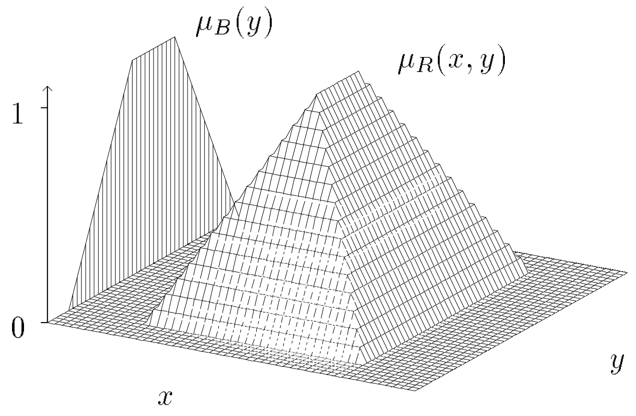


Figure 2.14: *Example of projection mechanism for fuzzy relations, where* $B = \mathrm{proj}(R; Y)$.

In figure 2.15, an example is given of a fuzzy set on X and its cylindrical extension on $X \times Y$. Using (2.33) the cylindrical extension shown in figure 2.15 is given by:

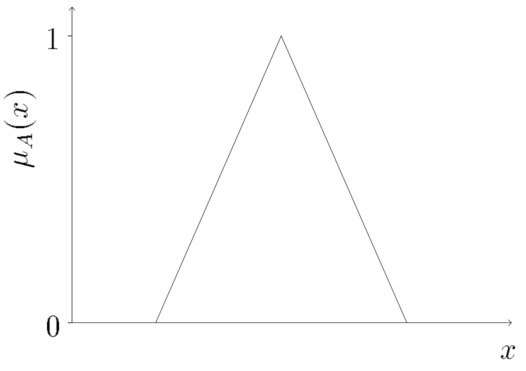
$$R = \mathrm{cext}(A; X \times Y) = \int_{X \times Y} \mu_A(x)/(x, y) \tag{2.34}$$

The cylindrical extension "extends" the product space of which a fuzzy set or relation is a subset of without any loss of information, which is shown by (2.33).

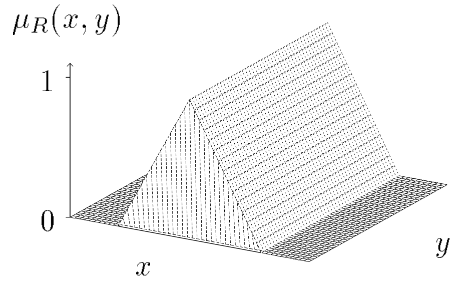
2.4.2 Composition of fuzzy relations

The *composition* is defined as follows (Zadeh, 1973): suppose there exists a fuzzy relation R in $X \times Y$ and A is a fuzzy set in X, then fuzzy subset B of Y can be induced by A, given the composition of R and A. This is denoted by:

$$B = A \circ R \tag{2.35a}$$



(a)
Fuzzy set A with membership function $\mu_A(x)$.



(b)
Cylindrical extension of A in $X \times Y$.

Figure 2.15: *Example of cylindrical extension R of fuzzy set A, where $R = \mathrm{cext}(A; X \times Y)$.*

and is defined by:

$$B = \mathrm{proj}(R \cap \mathrm{cext}(A; X \times Y); Y) \tag{2.35b}$$

Assuming the cylindrical extension as implicit the composition of relations can be regarded as consisting of two phases: *combination* and *projection*. Zadeh proposed to use *sup-min* composition, which leads to the following implementation of the composition where A is a fuzzy set with membership function $\mu_A(x)$ and R is a fuzzy relation with membership function $\mu_R(x, y)$:

$$\mu_B(y) = \sup_x \min(\mu_A(x), \mu_R(x, y)) \tag{2.36}$$

where the cylindrical extension of A is implicit and *sup* and *min* represent the projection and combination phase, respectively. Various others, among which Hellendoorn (1990), have pointed out that this could be generalized by taking a general T-norm and T-conorm for the *min* and *sup* operator, respectively. However, considering continuous domains, a general T-conorm cannot be taken, since its working is not defined. For example,

$a + b - ab$ cannot be determined for a continuous interval. Therefore the composition of fuzzy relations is normally represented by:

$$\mu_B(y) = \sup_x \mathrm{T}(\mu_A(x), \mu_R(x, y)) \tag{2.37}$$

where T is a general T-norm. Here, however, the *sup-min*-implementation proposed by Zadeh is mostly used. For example, if we use the examples given in previous sections, the fuzzy set *"about 5"*:

$$\mu_{\approx 5}(x) = \max(1 - \tfrac{1}{2}|x - 5|, 0)$$

and the fuzzy relation representing *approximately equal*:

$$\mu_{\approx}(x, y) = \max(1 - \tfrac{1}{2}|x - y|, 0)$$

then the composition operation provides a means to obtain a fuzzy set representing the value of y. Using the composition according to Zadeh, the membership function $\mu_B(y)$ of the resulting fuzzy set B is:

$$\begin{aligned}
\mu_B(y) &= \sup_x \min(\mu_{\approx 5}(x), \mu_{\approx}(x, y)) \\
&= \sup_x \min(\max(1 - \tfrac{1}{2}|x - 5|, 0), \max(1 - \tfrac{1}{2}|x - y|, 0)) \\
&= \max(1 - \tfrac{1}{4}|y - 5|, 0)
\end{aligned}$$

Note that the result given above cannot easily be obtained, although this is just a simple example. The different stages during the composition according to (2.36) of $\mu_{\approx 5}(x)$ and $\mu_{\approx}(x, y)$, as well as the result of the composition is shown in figure 2.16.

A discretized version of $R_{\approx}$ was given in section 2.4. If the compositional rule of inference is applied on discretized universes, with $x, y = 0, 1, 2, \ldots$, in the range $[0, 10]$,

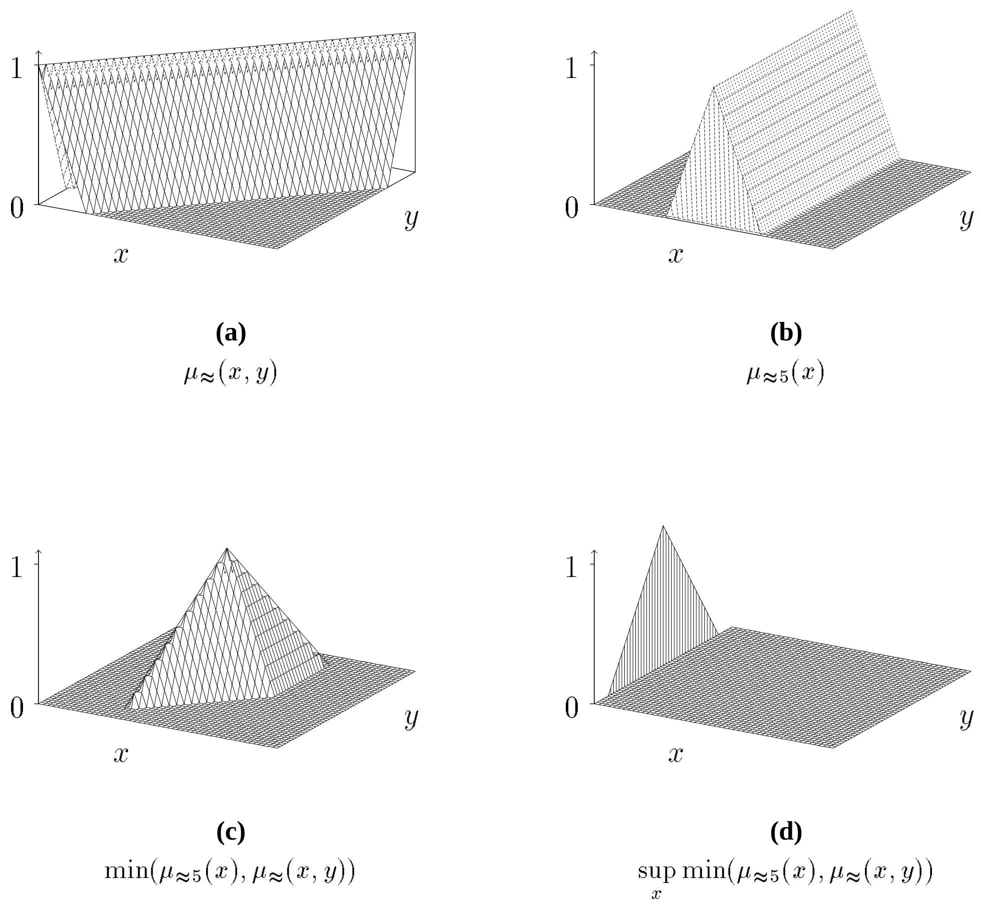


Figure 2.16: *Example of composition for which the fuzzy relation (a), the cylindrical extension of the data (b), the intersection (c) of the relation and the cylindrical extension of the data, and the projection on Y (d) are shown.*

then it results in:

$$\mu_B(y) = \overbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{2} \\ 1 \\ \frac{1}{2} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}^{\mu_{\approx 5}(x)} \circ \overbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}}^{\mu_{\approx}(x,y)}$$

$$= \overbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}}^{\min(\mu_{\approx 5}(x), \mu_{\approx}(x,y))}$$

$$= \overbrace{\begin{pmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \end{pmatrix}}^{\max\limits_{x} \min(\mu_{\approx 5}(x), \mu_{\approx}(x,y))}$$

When one looks at the result, also shown in figure 2.17, it is clear that this is not a discretization of the result when the composition was applied on continuous universes, as shown in the same figure. Therefore, one should take care when using discretizations of fuzzy sets and relations to implement a fuzzy system.

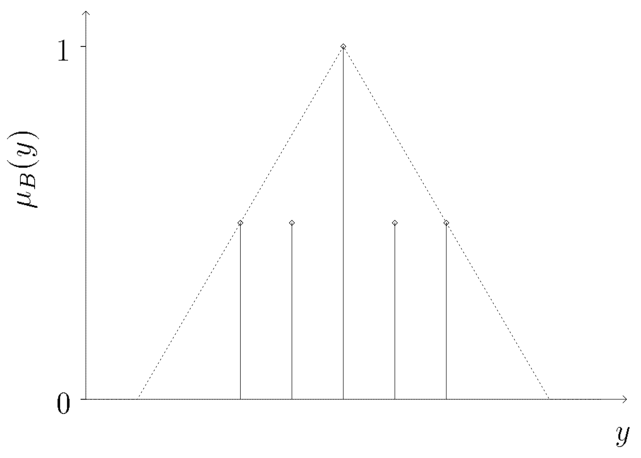


Figure 2.17: *Comparison of results obtained from composition on continuous domains and discretizations of those.*

2.5 Summary and remarks

Fuzzy sets are sets without sharp (crisp) boundaries: membership of a fuzzy set is a grade in the interval $[0, 1]$. Fuzzy sets can be used to model linguistic labels, where the "vagueness" of the label is modeled by the non-crisp boundaries of the fuzzy set used to represent the label. Linguistic modifiers (hedges) can be used to modify the linguistic meaning of fuzzy set. Different approaches to hedges are possible as discussed in section 2.2.

An important concept within fuzzy set theory is the *extension principle*, described in section 2.1.4. This allows the extension of classical (mathematical) concepts to fuzzy ones (Dubois and Prade, 1980). In practice, the extension principle cannot be applied easily in many cases. If the mathematical operation to be fuzzified operates on fuzzy sets which are fuzzy subsets of the same universe, a simplification based on level sets is possible.

The composition of relations, using projection and cylindrical extension (see section 2.4.1), is an important concept in the field of fuzzy logic and reasoning, which is addressed in the next chapter. As shown in section 2.4.2, the approximation of fuzzy relations using discretizations can lead to quite different results than the analytical results obtained by the composition of relations.

3 Fuzzy logic and reasoning

Fuzzy logic and reasoning is addressed in this chapter. As classical set theory serves as the basis for classical logic, fuzzy set theory serves as the basis for fuzzy logic. This means that theoretic operations on fuzzy sets are a base for logical operations. The operations defined for sets, like union, intersection and complement, have a corresponding logical meaning, like *or*, *and* and *not*, respectively. However, as described in 2.3, in the case of fuzzy sets, the set-theoretic operations have many representations instead of just one as in classical set theory. This, of course, holds also for the logical operators in fuzzy logic.

First the basic primitives of fuzzy logic are described: fuzzy propositions (section 3.1). A fuzzy proposition represents a statement "x is A", where x is a variable and A is a linguistic label, represented by a fuzzy set. In section 3.2 the modeling of fuzzy rules and fuzzy rule bases is addressed. The modeling of fuzzy rules is based on fuzzy implications and a fuzzy rule base is a set of parallel fuzzy rules which are aggregated. Section (3.3) describes how fuzzy rules and rule bases can be used for reasoning. This chapter focuses only on the basics of fuzzy reasoning. For a more detailed discussion of fuzzy reasoning, the reader is referred to chapter 6.

3.1 Fuzzy propositions

An important concept in fuzzy logic is a *fuzzy proposition*. Fuzzy propositions represent statements like "x is big", where "big" is a linguistic label, defined by a fuzzy set on the universe of discourse of variable x. *Fuzzy (linguistic) labels* are also referred to as *fuzzy constants, fuzzy terms or fuzzy notions*. Fuzzy propositions connect variables with linguistic labels defined for those variables.

These fuzzy propositions are the basis for fuzzy logic and reasoning. Fuzzy propositions can be combined by means of logical connectives like *and* and *or*. Linguistic modifiers can be used to modify the meaning of the linguistic label used in a fuzzy proposition. For example, the linguistic modifier *very* can be used to change "x is big" to "x is *very* big".

3.1.1 Logical connectives

As in classical logic, fuzzy propositions can be combined by using the logical connectives *and* and *or*. The *and* and *or* connectives are implemented by T-norms and T-conorms, respectively. As shown in 2.3 there are an infinite number of T-norms and T-conorms. There are no general guidelines as to which T-norm or T-conorm to choose in a specific situation. However, based on some properties of T-norms and T-conorms the use of specific T-norms and T-conorms can be intuitively justified in some cases. Examples of results using the T-norm and T-conorm according to Łukasiewicz compared to the T-norm and T-conorm according to Zadeh are shown in figure 3.1. Here the operations are performed on the same universe of discourse.

The operators originally proposed by Zadeh have the advantage that redundancy is ignored: the combination of two equal fuzzy propositions will after combination represent the same information:

$$\mu_{A\cap A}(x) = \min(\mu_A(x), \mu_A(x)) = \mu_A(x) \tag{3.1a}$$

$$\mu_{A\cup A}(x) = \max(\mu_A(x), \mu_A(x)) = \mu_A(x) \tag{3.1b}$$

which does not hold for any other T-norms and T-conorms. However, when fuzzy propositions are not equal but correlated or interactive it can be justified to use other operators than *min* and *max*. The correlation or interactivity of fuzzy propositions represent cases where dependencies exist between the fuzzy propositions. For example, it can be justified that using the T-norm and T-conorm according to Łukasiewicz is appropriate in the case

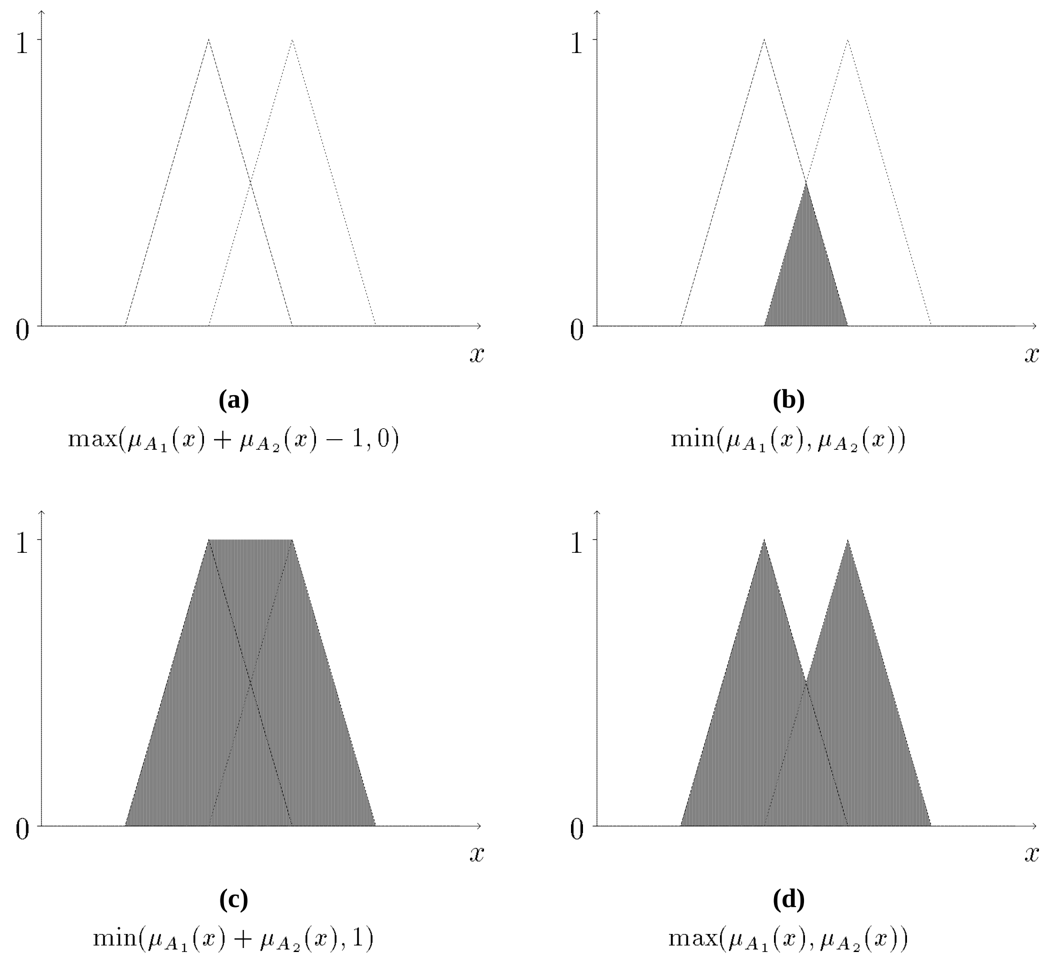


Figure 3.1: *Results of Łukasiewicz's type of T-norm (a) and T-conorm (c) and Zadeh's type of T-norm (b) and T-conorm (d) for logical connectives* and *and* or. *Note that the result in figure (a) is* 0.

of complete interactivity:

$$\mu_{A_1 \cap A_2}(x) = \max(\mu_{A_1}(x) + \mu_{A_2}(x) - 1, 0) \tag{3.2a}$$

$$\mu_{A_1 \cup A_2}(x) = \min(\mu_{A_1}(x) + \mu_{A_2}(x), 1) \tag{3.2b}$$

where the above-given T-norm is also known as the *bounded sum*. As shown in figure 3.1c, the *or* connective of two fuzzy propositions results in a fuzzy set which also has complete membership for values of the variable "between" the fuzzy sets (provided the membership functions sum up to 1 or higher). When only two fuzzy propositions are considered, *the noise level is high* and *the noise level is low*, intuitively the result when using the operators according to Łukasiewicz can be regarded as more appropriate than the operators according to Zadeh. When the fuzzy sets used in two previously mentioned propositions for a fuzzy partition, the combined proposition *the noise level is high or low* will cover the complete universe of discourse when the *or* connective according to Łukasiewicz is used. However, as stated before, only some indications can be given, since each T-norm and -conorm has advantages and disadvantages. The choice of T-norms and T-conorms for the logical connectives depends on the meaning and context of the propositions and the relations between them. The most frequently used operators in fuzzy logic are the ones given in table 3.1.

Table 3.1: *Frequently used operators for* and *and* or *connectives in fuzzy logic.*

and	or	remark
$\min(a, b)$	$\max(a, b)$	Zadeh
$\max(a + b - 1, 0)$	$\min(a + b, 1)$	Łukasiewicz
ab	$a + b - ab$	probability

Thus far, we have only considered cases where the propositions are related to the same universe of discourse. If the propositions are related to different universes, a logical connective will result in a fuzzy relation. For example, consider the following proposition:

p : x_1 is A_1 **and** x_2 is A_2

where A_1 and A_2 have membership functions $\mu_{A_1}(x_1)$ and $\mu_{A_2}(x_2)$. The proposition p can then be represented by the fuzzy relation P with membership function:

$$\mu_P(x_1, x_2) = \mathrm{T}(\mu_{A_1}(x_1), \mu_{A_2}(x_2)) \tag{3.3}$$

where T is a general T-norm that is used to model the *and*-connective. Such a combination of propositions, in fact a proposition itself, can be the premise of a fuzzy rule.

3.1.2 Negation in fuzzy propositions

Similarly, the logical connectives can be related to the intersection and union of fuzzy sets, the negation within a fuzzy proposition can be related to the complement of a fuzzy set. A simple example of a fuzzy proposition with a negation in it, is:

the noise level is **not** high

and using the standard complement $c(a) = 1 - a$ results in:

$$\mu_{\text{not high}}(x) = 1 - \mu_{\text{high}}(x)$$

More general, the negation in a fuzzy proposition x ***is not*** A results in:

$$\text{not}(A) = \int_X \text{c}(\mu_A(x))/x \tag{3.4}$$

where the complement complies with the criteria **c-1,2,3** given in section 2.3.2. Although several fuzzy complements are possible, generally the standard complement:

$$\mu_{\text{not}(A)}(x) = 1 - \mu_A(x)$$

is used. From (3.4) one can see the resemblance with hedges, linguistic modifiers for fuzzy sets as described in section 2.2.

3.2 Fuzzy rules

In order to reason with fuzzy logic, fuzzy rules have to be represented by an implication function. Such a fuzzy implication has the same function as the truth table of the classical implication in classical logic. In classical logic the implication is denoted by:

$$A \rightarrow B \tag{3.5}$$

which can be seen as a representation of the statement:

if A **then** B

In fuzzy logic these types of statements are often referred to as *fuzzy if-then* statements or *fuzzy rules*.

Table 3.2: *Truth table of classical implication.*

A	B	$A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

The truth table of the classical implication is given in table 3.2. The next section shows how a fuzzy rule can be represented by a fuzzy relation by means of a fuzzy implication function. The section thereafter describes fuzzy implications which are used in fuzzy logic. Besides fuzzy implications which comply with the classical implication as defined by table 3.2, fuzzy implications which are interpreted as (fuzzy) conjunctions are used. In section 3.2.4, a general classification of fuzzy implications is given. Section 3.2.3 describes how a set of parallel fuzzy rules is combined by means of an aggregation operator.

3.2.1 Representation of a fuzzy rule

A fuzzy rule is an *if-then* statement where the premise and the consequent consist of fuzzy propositions as described in section 3.1. The premise can contain a combination of propositions by means of the logical connectives *and* and *or* (section 3.1.1). It is also possible that a fuzzy proposition is based on a negation (section 3.1.2). For the sake of simplicity the following rule is considered:

if x_1 is A_1 **and** x_2 is A_2 **then** y is B

When fuzzy sets A_1, A_2 and B are identified by the membership functions $\mu_{A_1}(x_1)$, $\mu_{A_2}(x_2)$ and $\mu_B(y)$, the following fuzzy relation R representing the fuzzy rule can be constructed:

$$R = \mathrm{I}(\mathrm{T}(A_1, A_2), B) \tag{3.6}$$

where T is a conjunction based on a general T-norm and I is a fuzzy implication function. As the T-norm T represents (or models) the *and* connective, the fuzzy implication function I represents (or models) the implication: the *if-then* connective. Hence, a fuzzy rule can be represented by a fuzzy relation. The membership function of R of the above-given example is given by:

$$\mu_R(x_1, x_2, y) = \mathrm{I}(\mathrm{T}(\mu_{A_1}(x_1), \mu_{A_2}(x_2)), \mu_B(y)) \tag{3.7}$$

Fuzzy implication functions are described in the following two sections. The implication function I is mostly denoted by $I(a, b)$ where $a, b \in [0, 1]$.

3.2.2 Fuzzy implications

As any operator in classical set theory and classical logic has an infinite number of possible representations in fuzzy set theory and fuzzy logic, this is also the case for the implication. Besides fuzzy implications complying with the classical implication as defined by table 3.2, the implication is sometimes interpreted as a conjunction, in which case the "causality" relation dictated by the if-then statement is not preserved in the fuzzy relation representing the implication. Hence, if necessary, the direction of the if-then statement needs to be preserved as background knowledge, since the conjunction of two propositions looses the causality represented by the fuzzy rule. Dubois and Prade (1991) made a summary of the different types of fuzzy implications, and they distinguish the following types of fuzzy implications:

- Implications in fuzzy logic **based on the classical implication**; a fuzzification of material implication (where $a \rightarrow b$ is defined by $\neg a \vee b$):

  $$\mathrm{I}(a, b) = \mathrm{S}(\mathrm{c}(a), b) \tag{3.8}$$

  which are referred to as **S-implications** (Dubois and Prade, 1991).

- Fuzzy implication based on the **implication in quantum logic**:

  $$\mathrm{I}(a, b) = \mathrm{S}(\mathrm{c}(a), \mathrm{T}(a, b)) \tag{3.9}$$

  where T and S are c-dual (see (2.25) in section 2.3.1). These types of implications is known as **QL-implications** (Dubois and Prade, 1991), where 'QL' stands for *quantum logic*. Lee (1990a) referred to this type of implication as *propositional calculus* and also lists the *extended propositional calculus* :

  $$\mathrm{I}(a, b) = \mathrm{S}(\mathrm{T}(\mathrm{c}(a), \mathrm{c}(b)), b) \tag{3.10}$$

  which results from (3.9) when a and b are replaced by $1 - b$ and $1 - a$, respectively.

- Fuzzy implications **reflecting partial ordering on propositions**:

$$\mathrm{I}(a,b) = \begin{cases} 1, & \text{if } a \leq b \\ 0, & \text{if } a = 1 \wedge b = 0 \\ \in [0,1\rangle, & \text{otherwise} \end{cases} \tag{3.11a}$$

Most implications of this type belong to the **R-implications** (the 'R' stands for *residuated*; Dubois and Prade, 1991):

$$\mathrm{I}(a,b) = \sup\{\gamma \in [0,1] \mid \mathrm{T}(a,\gamma) \leq b\} \tag{3.11b}$$

where T is a general T-norm. Lee (1990b) refers to this type of fuzzy as *generalization of modus ponens* and also lists the *generalization of modus tollens*[*]:

$$\mathrm{I}(a,b) = 1 - \inf\{\gamma \in [0,1] \mid \mathrm{S}(b,\gamma) \geq a\} \tag{3.12}$$

which results from (3.11b) when a and b are replaced by $1-b$ and $1-a$, respectively.

- Interpretation of the implication **as a conjunction**:

$$\mathrm{I}(a,b) = \mathrm{T}(a,b) \tag{3.13}$$

where T is a T-norm. This type of implication is clearly not a generalization of the classical implication, but complies with the classical conjunction. Fuzzy implications which are represented by a conjunction are usually used in fuzzy control; this is described and analyzed in more detail in chapter 4.

[*]In the original paper Lee (1990b) denotes the *generalization of modus tollens* as:

$$\mathrm{I}(a,b) = \inf\{\gamma \in [0,1] \mid \mathrm{S}(b,\gamma) \leq a\}$$

which is not correct (probably due to a typing error), because there is no solution in case $b > a$. Lee (1990b) probably meant (3.12) which can easily be derived, using (3.11b) and replacing a by $1-b$ and b by $1-a$ (modus tollens):

$$\begin{aligned}\mathrm{I}(a,b) &= \sup\{\gamma \in [0,1] \mid \mathrm{T}(1-b,\gamma) \leq 1-a\} \\ &= 1 - \inf\{\gamma' \in [0,1] \mid \mathrm{T}(1-b,1-\gamma') \leq 1-a\} \\ &= 1 - \inf\{\gamma' \in [0,1] \mid 1-\mathrm{S}(b,\gamma') \leq 1-a\} \\ &= 1 - \inf\{\gamma' \in [0,1] \mid \mathrm{S}(b,\gamma') \geq a\}\end{aligned}$$

where $\gamma' = 1-\gamma$ and T and S are c-dual. Hence, the $\leq$ in Lee's version should be a $\geq$ and a negation should be added to obtain (3.12).

- As the implications reflecting partial ordering on propositions are related to the classical implication, a similar class of implications can be defined relating to the classical intersection:

  $$\mathrm{I}(a,b) = \begin{cases} 0, & \text{if } a+b \leq 1 \\ 1, & \text{if } a=1 \wedge b=1 \\ \in \langle 0,1], & \text{otherwise} \end{cases} \tag{3.14a}$$

  with the following subclass (compare R-implications):

  $$\mathrm{I}(a,b) = \inf\{\gamma \in [0,1] \mid \mathrm{S}(1-a,\gamma) \geq b\} \tag{3.14b}$$

  It is clear that this is type includes the classical intersection as a special case.

Other fuzzy implications, which cannot be categorized within the above-listed types of implications, have been proposed. An example of such an implication is the one proposed by Yager (1980a):

$$\mathrm{I}(a,b) = b^a \tag{3.15}$$

The S-implications represent straightforward fuzzy interpretations of the classical implication $a \to b$. An example of this type of fuzzy implication is the Kleene-Dienes implication:

$$\mathrm{I}(a,b) = \max(1-a,b) \tag{3.16}$$

and is shown in figure 3.2e. An example of an R-implication is the implication according to Goguen (1969), shown in figure 3.2f:

$$\mathrm{I}(a,b) = \begin{cases} 1, & \text{if } a=0 \\ \min(\frac{b}{a},1), & \text{otherwise} \end{cases} \tag{3.17}$$

where the T-norm in (3.11b) is the *product* operator. Another implication which reflects partial ordering on propositions is the implication according to Gaines (1976):

$$\mathrm{I}(a,b) = \begin{cases} 1, & \text{if } a \leq b \\ 0, & \text{otherwise} \end{cases} \tag{3.18}$$

which, however, is not an R-implication. Well-known examples of implications based on conjunctions (T-norms) are the implications used by Mamdani (1974), shown in figure 3.2c:

$$\mathrm{I}(a, b) = \min(a, b) \tag{3.19}$$

and Larsen (1980) (shown in figure 3.2d):

$$\mathrm{I}(a, b) = ab \tag{3.20}$$

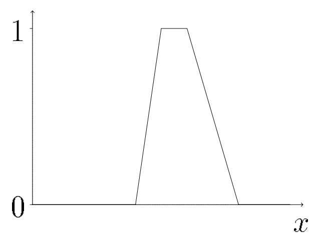
In figure 3.2, one can see the resulting fuzzy relation, representing a fuzzy rule, for different implication functions. In table A.3 on page 245, a summary of commonly known fuzzy implications is given.

3.2.3 Aggregation of fuzzy rules

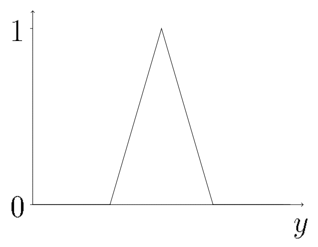
To here, it has been shown how to translate the premise of a fuzzy rule into a fuzzy relation and to translate an if-then statement into a fuzzy relation. Thus we are able to translate fuzzy rules into fuzzy relations. If there is more than one proposition in the consequences of fuzzy rules, the fuzzy rules are assumed to be separable with respect to the propositions in the consequent. The following step is to combine a set of fuzzy rules into one fuzzy relation. The fuzzy rules are considered as a set of N_r parallel rules which have a premise based on N_X variables:

r_1 : **if** x_1 is $A_{1,1}$ **and** . . . **and** x_{N_X} is $A_{N_X,1}$ **then** y is B_1
else
. . .
else
r_k : **if** x_1 is $A_{1,k}$ **and** . . . **and** x_{N_X} is $A_{N_X,k}$ **then** y is B_k
else
. . .
else
r_{N_r} : **if** x_1 is A_{1,N_r} **and** . . . **and** x_{N_X} is A_{N_X,N_r} **then** y is B_{N_r}

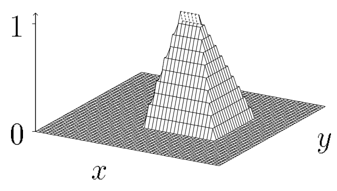
The translation of such a set of parallel fuzzy rules into a fuzzy relation is done by constructing the fuzzy relation R_k for each fuzzy rule r_k and combining these relations into a single fuzzy relation R. This combining of fuzzy rules into a fuzzy relation is called *aggregation*. The way this is done is different for different types of implication functions.



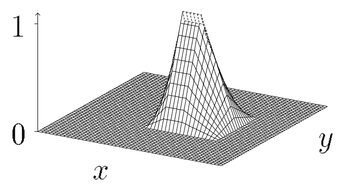
(a)
$\mu_A(x)$



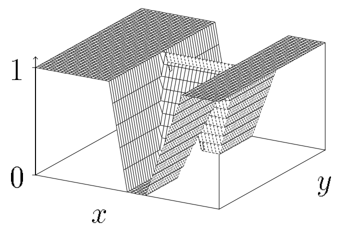
(b)
$\mu_B(y)$



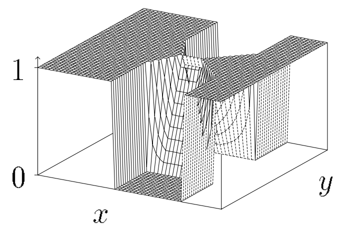
(c)
$\mathrm{I}(a,b)=\min(a,b)$



(d)
$\mathrm{I}(a,b)=ab$



(e)
$\mathrm{I}(a,b)=\max(1-a,b)$



(f)
$\mathrm{I}(a,b)=\min(\frac{b}{a},1)$

Figure 3.2: *Fuzzy relations $R=\mathrm{I}(A,B)$ as result of implication functions according to Mamdani* **(c)**, *Larsen* **(d)**, *Kleene-Dienes* **(e)** *and Goguen* **(f)**. *The two top figures show the fuzzy sets used in the premise* **(a)** *and consequent* **(b)** *of the rule:* **if** x is A **then** y is B.

For the implications which comply with the classical conjunction, this aggregation operator is a disjunction. For those complying with the classical implication, a conjunction is used for aggregation. If the N_r fuzzy rules r_k are represented by the fuzzy relations R_k, the resulting fuzzy relation R, which is the aggregation of relations R_k, is determined by:

$$R = \bigcup_k R_k \tag{3.21}$$

Hence the *else* connective is interpreted as a disjunction. A more general definition is possible, which uses an S-norm for the aggregation of the relations R_k. Figure 3.3f shows the aggregation of two fuzzy rules, using the *max* operator. The two rules can easily be recognized in this figure. The *min* operator is used as the implication function for the fuzzy rules in the figure.

The aggregation of fuzzy rules when the implication function complies with the classical implication, is performed by a conjunction:

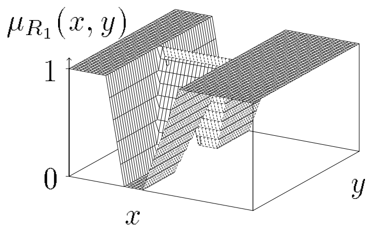
$$R = \bigcap_k R_k \tag{3.22}$$

Here also a more general definition is possible, using a T-norm for the aggregation of the rules. This means that the *else* connective is interpreted as a conjunction. Figure 3.3e shows the aggregation, using the *min* operator, of two fuzzy rules which are modeled by the Kleene-Dienes implication. From this figure we note that it is not easy to distinguish the two rules used. This is typical for fuzzy implications which comply with the classical implication.

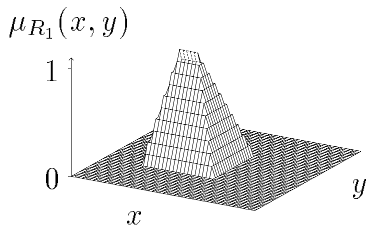
After reading this section it might well be possible that one still wonders about the different types of aggregation for different types of implications. In chapter 6, a justification based on possibility theory is discussed. For the sake of simplicity, no more details will be given at this stage; for the next two chapters it is enough to take note of the distinction between the two basic types of implications and note that for each type a different type of aggregation has to be used.

3.2.4 Classification of fuzzy implications

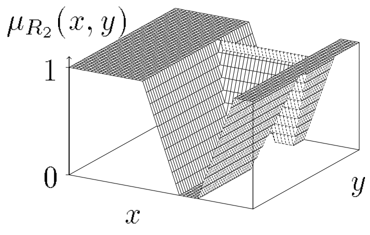
In the previous section different types of fuzzy implications were given, using the classification of fuzzy implications according to Dubois and Prade (1991). In this section, we will describe a more general classification of fuzzy implications. It is primarily based on distinguishing between two basic types of implications:



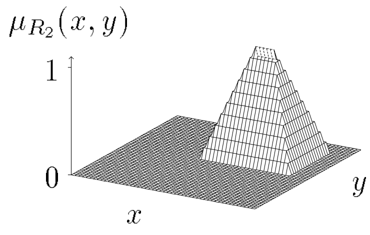
(a)
$\max(1-\mu_{A_1}(x),\mu_{B_1}(y))$



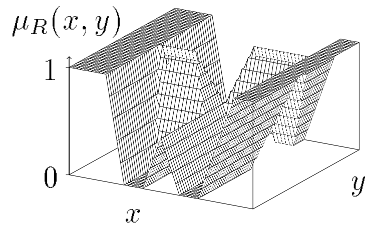
(b)
$\min(\mu_{A_1}(x),\mu_{B_1}(y))$



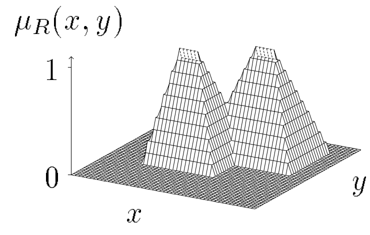
(c)
$\max(1-\mu_{A_2}(x),\mu_{B_2}(y))$



(d)
$\min(\mu_{A_2}(x),\mu_{B_2}(y))$



(e)
$\min(\mu_{R_1}(x,y),\mu_{R_2}(x,y))$



(f)
$\max(\mu_{R_1}(x,y),\mu_{R_2}(x,y))$

Figure 3.3: *Aggregation of fuzzy rules. The left column shows aggregation in the case of Kleene-Dienes' implication; the right column shows aggregation in the case of Mamdani's implication. The relation R is the aggregation of R_1 and R_2. See figure 3.2 for the construction of the relation for each individual rule.*

I. Fuzzy implications **complying with the classical implication**:

$$a \rightarrow b \;\equiv\; \neg a \vee b$$

which are aggregated by means of a conjunction as in (3.22).

II. Fuzzy implications **complying with the classical conjunction**:

$$a \rightarrow b \;\equiv\; a \wedge b$$

which are aggregated by means of a disjunction as in (3.21).

Using this basic distinction of two types of fuzzy implications a number of "compositions" can be defined. By composition we mean a combination of implications or nested implications: a variable is an implication itself. Possible compositions of the two basic types of implications I and II are:

1. Fuzzy implication based on implication from **quantum logic**:

   $$a \rightarrow b \;\equiv\; \neg a \vee (a \wedge b)$$

   which can be seen as $\neg a \vee b'$ (type I), where b' is an implication itself: $b' = a \wedge b$ (type II).

2. Fuzzy implications based on **modus tollens interpretation**:

   $$a \rightarrow b \;\equiv\; \neg b \rightarrow \neg a$$

   which can be any type of implication and replaces a and b by $\neg b$ and $\neg a$, respectively.

3. Fuzzy implications based on **symmetry between modus ponens and modus tollens**:

   $$a \rightarrow b \;\equiv\; (a \rightarrow b) \wedge (\neg b \rightarrow \neg a)$$

   which combines two implications by means of a conjunction.

The generalization of the two basic types of implication is not based on the use of T-norms and T-conorms for the generalization of the conjunction and the disjunction, respectively.

It allows the generalizations of the binary valued operators to be less restrictive than T-norms and T-conorms. Let us define a generalization of the classical conjunction, denoted by C:

$$\mathrm{C}(a,b) = \begin{cases} 0, & \text{if } a = 0 \vee b = 0 \\ 1, & \text{if } a = 1 \wedge b = 1 \\ \in [0,1\rangle, & \text{otherwise} \end{cases} \tag{3.23a}$$

and $C(a,b) \leq C(c,d)$, whenever $a \leq c, b \leq d$ (compare criterion T-2 on page 31). A generalization of the classical disjunction, denoted by D, is obtained in a similar way:

$$\mathrm{D}(a,b) = \begin{cases} 1, & \text{if } a = 1 \vee b = 1 \\ 0, & \text{if } a = 0 \wedge b = 0 \\ \in \langle 0,1], & \text{otherwise} \end{cases} \tag{3.23b}$$

These generalizations of the classical conjunction and disjunction do not have the restrictions that they have to be commutative and/or associative. T-norms and -conorms as defined in section 2.3.1 fall within these definitions as a subclass. A more general subclass of the D- and C-operators can be defined by:

$$\mathrm{C}^*(a,b) = \begin{cases} 0, & \text{if } a + b \leq 1 \\ 1, & \text{if } a = 1 \wedge b = 1 \\ \in [0,1\rangle, & \text{otherwise} \end{cases} \tag{3.24a}$$

$$\mathrm{D}^*(a,b) = \begin{cases} 1, & \text{if } a + b \geq 1 \\ 0, & \text{if } a = 0 \wedge b = 0 \\ \in \langle 0,1], & \text{otherwise} \end{cases} \tag{3.24b}$$

which are commutative. It can easily be shown that the class of implications representing partial ordering on propositions defined by (3.11), with R-implications, defined by (3.11b), as subclass, is defined by:

$$\mathrm{I}_{\mathrm{PO}}(a,b) = \mathrm{D}^*(1-a,b)$$

This class of implications falls within type I, as defined before. Also the implication of Yager (1980a) can be seen as a generalization of the classical implication using a generalization of the classical disjunction:

$$\mathrm{D}(a,b) = b^{1-a}$$

which results in $I(a,b) = D(1-a,b) = b^{1-(1-a)} = b^a$.

The following fuzzy implication (Zadeh, 1975):

$$\mathrm{I}(a,b) = \max(1-a, \min(a,b))$$

is a composition according to the implication in quantum logic, namely $a \rightarrow \equiv \neg a \vee (a \wedge b)$. This can be interpreted as nested implications: a classical-implication-based implication I_{CI} (type I) applied to the antecedent and the result of a classical-conjunction-based implication I_{CC} (type II) applied to the antecedent and the consequent:

$$\mathrm{I}_{\mathrm{QL}}(a,b) = \mathrm{I}_{\mathrm{CI}}(a, \mathrm{I}_{\mathrm{CC}}(a,b)) \tag{3.25}$$

Lee (1990b) referred to the implication from quantum logic as *propositional calculus* and also mentions *extended propositional calculus*:

$$\mathrm{I}(a,b) = \max(b, \min(1-a, 1-b))$$

which can be written as the implication from quantum logic based on modus tollens:

$$\begin{aligned}\mathrm{I}(a,b) &= \mathrm{I}_{\mathrm{QL}}(\neg b, \neg a)\\ &= \max(1-(1-b), \min(1-b, 1-a))\\ &= \max(b, \min(1-a, 1-b))\end{aligned}$$

When these two implications are combined to obtain symmetry for the variables, using $a \rightarrow b \equiv (a \rightarrow b) \wedge (\neg b \rightarrow \neg a)$, the result is the implication proposed by Willmott (1980):

$$\mathrm{I}_{\mathrm{Wm}}(a,b) = \min(\max(1-a,b), \max(a, 1-b, \min(b, 1-a))) \tag{3.26}$$

because this can be written as the combined implication:

$$\mathrm{I}_{\mathrm{Wm}}(a,b) = \min(\mathrm{I}_{\mathrm{QL}}(a,b), \mathrm{I}_{\mathrm{QL}}(1-b, 1-a)) \tag{3.27a}$$

where:

$$\mathrm{I}_{\mathrm{QL}}(a,b) = max(1-a, \min(a,b)) \tag{3.27b}$$

Table 3.3: *Simplification for different relations between a and b with implication proposed by Willmott (1980):* $\mathrm{I_{Wm}}(a,b) = \min(\max(1-a,b), \max(a, 1-b, \min(b, 1-a)))$.

a and b	$\mathrm{I_{Wm}}(a,b)$	$\min(\mathrm{I_{QL}}(a,b), \mathrm{I_{QL}}(1-b, 1-a))$
$1-b \le 1-a \le a \le b$	a	$\min(a,b)$
$1-b \le a \le 1-a \le b$	$1-a$	$\min(1-a,b)$
$b \le 1-a \le a \le 1-b$	$1-a$	$\min(1-a,1-a)$
$b \le a \le 1-a \le 1-b$	$1-a$	$\min(1-a,1-a)$
$1-a \le 1-b \le b \le a$	b	$\min(b,b)$
$1-a \le b \le 1-b \le a$	b	$\min(b,b)$
$a \le 1-b \le b \le 1-a$	b	$\min(1-a,b)$
$a \le b \le 1-b \le 1-a$	$1-b$	$\min(1-a,1-b)$

To prove the equality of those two, we consider eight different possible inequalities concerning a, b and their negations. The results for (3.26) are shown in table 3.3. It can easily be checked that the same results are obtained for (3.27).

Hence, a general classification of fuzzy implications can be made distinguishing two basic types and three types of composition/modification when allowing non-commutative and non-associative generalizations of the classical conjunction and disjunction. This distinction is used in the remainder of the thesis. For example, in section 3.2.3, the aggregation of a set of rules was described and the aggregation operator is different for the two basic types of implication. This property plays an important role in the inference of a set of fuzzy rules. See section 3.3.2 for more details on inference of a parallel set of fuzzy rules, also known as a fuzzy rule base.

3.2.5 Rule base properties

In the following section a number of properties of a fuzzy rule base are considered: *consistency*, *continuity* and *completeness* of a fuzy rule base. These properties provide classifications of rule bases which are used in the remainder of this thesis.

3.2.5.1 Continuity of a rule base

Continuity of a rule base requires that rules with “adjacent” premises have “adjacent” consequents. To clarify this, let us first explain the notion of adjacent fuzzy sets by means of an ordered set of fuzzy sets:

$$A_1 < A_2 < \cdots < A_{i-1} < A_i < A_{i+1} < \cdots \tag{3.28}$$

where A_{i-1} and A_i are adjacent fuzzy sets, as well as A_i and A_{i+1} are. Here, it is assumed that the adjacent fuzzy sets overlap. A good example is a fuzzy partition, where only adjacent fuzzy sets overlap. This is considered in most cases where continuity of a rule base is addressed in this thesis.

Here, a rule base is said to be continuous in case the premises of rules are “adjacent” as well as the consequents. Rule premises are considered adjacent when they contain the same conditions (fuzzy sets) except for one, in which case the fuzzy sets in these conditions have to be adjacent. To explain this, consider the following example. Suppose a rule base containing the following rules:

$$r_k : \textbf{if } x_1 \text{ is } A_{1,k} \textbf{ and } x_2 \text{ is } A_{2,k} \textbf{ then } y \text{ is } B_k$$

The premises of rules r_k and $r_{k'}$, with $k' \neq k$, are considered adjacent when one of the following holds:

1. $A_{1,k} = A_{1,k'} \wedge A_{2,k}$ and $A_{2,k'}$ are adjacent

2. $A_{2,k} = A_{2,k'} \wedge A_{1,k}$ and $A_{1,k'}$ are adjacent

The rule base is said to be continuous when B_k and $B_{k'}$ are adjacent for the above-given cases, where it is assumed that the rule base is complete (see section 3.2.5.3 on this topic). Note that the case where $A_{1,1}$ and $A_{1,2}$ are adjacent, and $A_{2,1}$ and $A_{2,2}$ are adjacent, does not imply adjacent premises in this definition. Hence, the definition of continuity of a rule base as given above is less strict than a definition purely based on overlapping of fuzzy sets used in the rule premises. As will be shown in section 4.2.1, the continuity of a rule base plays an important role when fuzzy implications which comply with the classical implication are used to model the fuzzy rules.

3.2.5.2 Consistency of a rule base

Consistency of a rule base addresses the consistency of the knowledge represented by that rule base. A well-known example of an inconsistent rule base is the following, where two fuzzy rules co-exist in the rule base used for the control of a robot and the data "obstacle in front" is considered:

if obstacle in front **then** go left
if obstacle in front **then** go right

One might state that "this is a badly designed rule base". That may be right, but these inconsistencies are hardly unavoidable in complex rule bases. Also the use of *or* connectives in rule premises can lead to this kind of problem. A simple example showing this is the following rule base:

if x_1 is A **or** x_2 is E **then** y is H
if x_1 is C **or** x_2 is F **then** y is I
if x_1 is B **and** x_2 is D **then** y is G

which leads to ambigious conclusions about y as shown in table 3.4. From this table one can see the ambigious conclusions in case x_1 is A and y is F, and in case x_1 is C and x_2 is E. In this simple example it is clear that the previous stated two rules will give ambigious conclusions for some situations, but in the case of larger, more complex rule bases this phenomena cannot be recognized easily. A similar problem can occur when using the *not* operator in rule premises.

Table 3.4: *Examples of possible inconsistencies in rule base.*

$x_1 \setminus x_2$	A	B	C
D	H	G	I
E	H	H	H, I
F	H, I	I	I

3.2.5.3 Completeness of a rule base

In the following we discuss the *completeness of a fuzzy rule base*, which can be used as a measure to denote the completeness of the knowledge represented by the rule base.

An incomplete rule base has so-called blank spots: for certain situations of the input space (on the semantic level) no output actions are defined. This does not mean that the result of inference of an incomplete rule base does not exist. The fuzzy sets used in the rule premises play a role in this. Using the fuzzy sets in combination with the rules, a measure of completeness of a fuzzy rule base can be defined:

$$\mathrm{CM}(\boldsymbol{x}) = \sum_{k=1}^{N_r} \{ \prod_{i=1}^{N_X} \mu_{A_{i,k}}(x_i) \} \tag{3.29}$$

where $\boldsymbol{x}$ is a numerical data vector. The completeness measure $CM(\boldsymbol{x})$ has a value greater than 0 in case one or more fuzzy rules have the data vector $\boldsymbol{x}$ in the support of their premise. Using the completeness measure as defined by (3.29), we can summarize the following about the completeness of the fuzzy rule base for a certain data vector:

$\mathrm{CM}(\boldsymbol{x}) = 0$	:	incomplete (blank spot)
$0 < \mathrm{CM}(\boldsymbol{x} < 1$	:	subcomplete
$\mathrm{CM}(\boldsymbol{x}) = 1$	:	strict complete
$\mathrm{CM}(\boldsymbol{x} > 1$	:	overcomplete (redundant)

It is obvious that a fuzzy rule base can have different properties for different regions in the input space. For example, certain regions can be overcomplete while other regions can be subcomplete. Subcompleteness, strict completeness and overcompleteness of a fuzzy controller are grades of completeness, as opposed to incompleteness.

From (3.29) it is clear that the completeness of a fuzzy rule base is strict in case the fuzzy sets on the universes of discourse of each universe used in the rule premises are fuzzy partitions (see section 2.1.2) and the rule base contains all possible rules which can be defined using the fuzzy sets defined on the universes. Incompleteness of the rule base is something which should be avoided, if possible, because it represents a lack of knowledge ("blank spot" on a semantic level). Kóczy and Hirota (1993) propose a method to perform inference with a sparse fuzzy rule base by means interpolative reasoning (see section 4.6.3.1). However, this method is based on T-implications. Research in the redundancy of fuzzy rule bases consisting of certainty rules and gradual rules was reported by Dubois and Prade (1994b). In section 4.6.3 it will be shown that incompleteness of a fuzzy rule base can lead to undesired control behavior in fuzzy control.

3.3 Fuzzy reasoning

This section discusses reasoning with fuzzy logic. As described in the previous section, there exists a large number of possible fuzzy implications to model a fuzzy rule. A fuzzy rule can be used to infer knowledge about the consequent of the rule using data which is a fuzzy subset of the same universe as the premise of the rule. First we focus on the inference of a single rule. After this, a set of parallel fuzzy rules is considered.

3.3.1 Inference of a fuzzy rule

The inference of a single fuzzy rule is a straightforward application of the composition of fuzzy relations. Zadeh (1973) introduced this in fuzzy logic as the *compositional rule of inference*, sometimes abbreviated to CRI in the remainder of this thesis. Also generalizations of the reasoning schemes from classical logic, *generalized modus ponens* and *generalized modus tollens*, are described in section 3.3.1.2.

3.3.1.1 Compositional rule of inference

The compositional rule of inference (CRI) was introduced by Zadeh (1973) and it assumes that a fuzzy rule:

if x is A **then** y is B

is represented by a fuzzy relation R. A result B' can then be inferred (from R by A') through the composition of A' and R:

$$B' = A' \circ R \tag{3.30}$$

The composition of relations was described in section 2.4.2. Hence, the CRI assumes that a fuzzy relation representing the rule exists. This fuzzy relation can be one of the fuzzy implications as described in section 3.2.2. When a suitable implication operator is chosen, the composition operators should be chosen. Normally the *sup-min* composition is used, but other combinations are possible. Next, it is shown how the CRI can be used within generalized modus ponens and tollens.

3.3.1.2 Generalized modus ponens and tollens

The *generalized modus ponens* was introduced by Zadeh (1973). It is a generalized version of the well-known inference rule from classical logic. It is based on an *if-then* relation:

$$\begin{array}{l} \textbf{if } x \text{ is } A \textbf{ then } y \text{ is } B \\ x \text{ is } A' \\ \hline y \text{ is } B' \end{array}$$

where A' represents the data and B' the inferred result. The truth table for the (classical) modus ponens is given in table 3.5. To solve this using the compositional rule of inference, a relation representing the if-then rule is necessary. There are many possible ways to represent an if-then rule by a relation; see section 3.2.2.

Table 3.5: *Truth table for modus ponens.*

A'	$A \to B$	B'
1	1	1
0	1	?

An example of the (generalized) modus ponens is the following: suppose the rule "if there is smoke then there is fire" and the data that "there is smoke" then "there is fire" can be inferred using the modus ponens. This inference is normally defined by the CRI:

$$B' = A' \circ R$$

where R is the fuzzy relation representing the fuzzy rule "if x is A then y is B".

The generalized modus ponens is not always the same as the compositional rule of inference. Many other inference schemes are possible which are based on the modus ponens, but not on the composition of fuzzy relations. Different types of reasoning are presented in chapter 6. One can view the generalized modus ponens as a reasoning scheme which includes the compositional rule of inference as a special case, namely, when the rule and data are represented by fuzzy relations.

Like the modus ponens, the *modus tollens* can be generalized: *generalized modus tollens*. The inference scheme is:

$$\begin{array}{l}\textbf{if } x \text{ is } A \textbf{ then } y \text{ is } B \\ y \text{ is } B' \\ \hline x \text{ is } A'\end{array}$$

The truth table for the (classical) modus tollens is given in table 3.6. Using the example presented for explaining the modus ponens, the modus tollens can be used to infer from “if there is smoke then there is fire” and “there is no fire” that “there is no smoke”.

Table 3.6: *Truth table for modus tollens.*

B'	$A \to B$	A'
1	1	?
0	1	0

For solving the modus tollens, the compositional rule of inference can be used, assuming a fuzzy relation for the if-then rule exists. This is normally denoted by:

$$A' = R \bullet B'$$

where R is the fuzzy relation representing the fuzzy rule “if x is A then y is B”.

3.3.1.3 Criteria for generalized modus ponens

In classical logic the criteria which the modus ponens has to meet are unique. However, in fuzzy logic there are more possibilities. Several criteria, proposed by a number of authors are discussed in this section. To date, much work has been done on the investigation of implication functions and the generalized modus ponens. Baldwin and Pilsworth (1980) proposed the following conditions to be met by the generalized modus ponens, where $\circ_m$ stands for the *sup-m* composition and m is a (pseudo-)conjunction (Dubois and Prade, 1991):

GMP-1 $B' \supseteq B$, which means that “nothing better” than B can be inferred from $A \to B$;

GMP-2 $B'_1 \subseteq B'_2$ if $A'_1 \subseteq A'_2$, requiring monotonicity;

GMP-3 $B' = Y$ if $A' = \overline{A}$, meaning that the negation of A results in B' is *unknown*;

GMP-4 if $B' = A' \circ_{\mathrm{m}} \mathrm{I}(A, B)$ then $\overline{A}' = \overline{B}' \circ_{\mathrm{m}} \mathrm{I}(A, B)$, requiring symmetry between modus ponens and modus tollens (see also section 3.3.1.2 for the generalized modus tollens);

GMP-5 $B' \supset B$ if $A' \not\subseteq A$, meaning that if A' is less restrictive that A, B' can never be more restrictive than B.

Other (partly overlapping) conditions to be met by the generalized modus ponens were proposed by Fukami, Mizumoto and Tanaka (1980):

GMP-6 $A \circ_{\mathrm{m}} \mathrm{I}(A, B) = B$, which demands that $B' = B$ in case $A' = A$. This is known as the *fundamental property*;

GMP-7a $\mathrm{very}(A) \circ_{\mathrm{m}} \mathrm{I}(A, B) = B$, where *very* is a powered hedge: $\mu_{\mathrm{very}(A)}(x) = \mu_A^2(x)$;

GMP-7b $\mathrm{very}(A) \circ_{\mathrm{m}} \mathrm{I}(A, B) = \mathrm{very}(B)$, which is even stronger that GMP-6;

GMP-8 $\text{more-or-less}(A) \circ_{\mathrm{m}} \mathrm{I}(A, B) = \text{more-}or\text{-less}(B)$, which is similar to GMP-6 and GMP-7b;

GMP-9a $\overline{A} \circ_{\mathrm{m}} \mathrm{I}(A, B) = Y$, which is the same as condition GMP-3;

GMP-9b $\overline{A} \circ_{\mathrm{m}} \mathrm{I}(A, B) = \overline{B}$, demanding that a negation of the premise results in a negation of the consequent (compare a conjunction).

Driankov (1987) also, besides conditions GMP-3, GMP-6, GMP-7a and GMP-8, proposed the following conditions to be met by the generalized modus ponens (Hellendoorn, 1990):

GMP-10 $B' = Y$ if $A' = X$, stating that if A' is *unknown*, then B' is *unknown*;

GMP-11 $B' = \int_Y c/y$ if $A' = \int_X c/x$, which represents that if A' is *undefined*, then B' should be undefined. Normally *undefined* is represented by $c = 0$.

It can be seen from the above-listed criteria that several of those criteria are conflicting. For each fuzzy implication it can be checked whether it fulfills a number of desired criteria or not. Hence, to meet certain criteria considered necessary for a specific application, appropriate implication functions can be chosen. Després (1989) proposed a tool for fuzzy rule acquisition which chooses appropriate implication functions to satisfy the expected behavior which is stated by the knowledge engineer.

Criterion GMP-1 has been considered quite often in research, since it is trivial in classical logic. Trillas and Valverde (1985) proposed to use *implication generating functions* for the generalized modus ponens. A modus ponens generating function, denoted by $m(a, b)$, has to meet the following conditions:

M-1 $\mathrm{m}(a, \mathrm{I}(a, b)) \leq b$

M-2 $\mathrm{m}(1, 1) = 1$

M-3 $\mathrm{m}(0, b) = b$

M-4 $\mathrm{m}(a, b) \leq \mathrm{m}(c, b)) \leq b$ if $a \leq c$

The generalized modus ponens is then the application of a *sup-m* composition:

$$\mu_{B'}(y) = \sup_x \mathrm{m}(\mu_{A'}(x), \mathrm{I}(\mu_A(x), \mu_B(y))) \tag{3.31}$$

Based on a *sup-m* composition condition, GMP-1 can be met by using an *m*-operator related to the used implication I. For example, Trillas and Valverde (1985) show that for an S-implication, where $I(a, b) = S(c(a), b)$, the *m*-operator defined by:

$$\mathrm{m}(a, b) = \inf\{\gamma \in [0, 1] \mid \mathrm{S}(\mathrm{c}(a), \gamma) \geq b\} \tag{3.32}$$

will result in satisfying condition GMP-1. This is in fact a multi-valued non-commutative conjunction; see also section 3.2.4 for use of this type of operator for implication modeling. When R-implications are considered, where:

$$\mathrm{I}(a, b) = \sup\{\gamma \in [0, 1] \mid \mathrm{T}(a, \gamma) \leq b\}$$

the following *m*-operator is to be used to satisfy condition GMP-1:

$$\mathrm{m}(a, b) = \mathrm{T}(a, b) \tag{3.33}$$

The use of different types of composition for different implications as Trillas and Valverde (1985) proposed indeed results in satisfying GMP-1, but this does not imply that it also results in intuitively correct behavior for other data. Trillas and Valverde (1985) chose the m-operator to satisfy criterion GMP-1 for a specific implication function I in the *sup-min* composition:

$$B = A \circ_{\mathrm{m}} \mathrm{I}(A, B)$$

This can also be reversed: Dubois and Prade (1984) proposed to meet criterion GMP-1 by choosing an implication function, which in combination with a known m-operator used in the *sup-m* composition, satisfies criterion GMP-1.

3.3.1.4 Inference of a rule modeled by a T-implication

When the implication function is a T-norm, the inference can be simplified in many cases. Assume the following rule:

r_k : **if** x_1 is $A_{1,k}$ **and** x_2 is $A_{2,k}$ **then** y is B_k

where the conjunction and implication are represented by a T-norm. In general, the following holds because of criterion T-4 on page 31:

$$\begin{aligned} B'_k &= \mathrm{T}(A'_1, A'_2) \circ_{\mathrm{T}} R_k && (3.34a)\\ &= \mathrm{T}(A'_1, A'_2) \circ_{\mathrm{T}} \mathrm{T}(\mathrm{T}(A_{1,k}, A_{2,k}), B_k) && (3.34b)\\ &= \mathrm{T}(\mathrm{T}(\mathrm{hgt}(\mathrm{T}(A'_1, A_{1,k})), \mathrm{hgt}(\mathrm{T}(A'_2, A_{2,k})), B_k) && (3.34c) \end{aligned}$$

where $\circ_{\mathrm{T}}$ stand for *sup-T* composition. In most cases the T-norm is chosen to be either a *min* operator or a *product* operator. This simplification is possible, provided that the conjunction, implication and composition are based on the same T-norm. If the conjunction is based on another T-norm, denoted by T_C, then the T-norm for the implication and composition, denoted by T_I, then the simplification is only partly possible:

$$\begin{aligned} B'_k &= \mathrm{T_C}(A'_1, A'_2) \circ_{\mathrm{T_I}} R_k\\ &= \mathrm{T_C}(A'_1, A'_2) \circ_{\mathrm{T_I}} \mathrm{T_I}(\mathrm{T_C}(A_{1,k}, A_{2,k}), B_k)\\ &= \mathrm{T_I}(\mathrm{hgt}(\mathrm{T_I}(\mathrm{T_C}(A'_1, A'_2), \mathrm{T_C}(A_{1,k}, A_{2,k}))), B_k) \end{aligned}$$

In the following, we focus on the cases where the T-norm is the *min* operator or the *product* operator.

When the *min* operator is used for the conjunction and implication, a major simplification of the inference of a fuzzy rule can be achieved. Then the CRI, when using a *min* operator for the *and* connective, results in:

$$\begin{aligned} \mu_{B'_k}(y) &= \sup_{x_1,x_2} \{(\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_2)) \wedge \mu_{R_k}(x_1, x_2, y)\} && (3.35a)\\ &= \sup_{x_1,x_2} \{(\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_2)) \wedge \\ &\qquad [(\mu_{A_{1,k}}(x_1) \wedge \mu_{A_{2,k}}(x_2)) \wedge \mu_{B_k}(y)]\} && (3.35b)\\ &= \{\sup_{x_1,x_2} [(\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_2)) \wedge \end{aligned}$$

$$(\mu_{A_{1,k}}(x_1) \wedge \mu_{A_{2,k}}(x_2))]\} \wedge \mu_{B_k}(y) \tag{3.35c}$$

$$= \{\sup_{x_1}(\mu_{A'_1}(x_1) \wedge \mu_{A_{1,k}}(x_1)) \wedge$$

$$\sup_{x_2}(\mu_{A'_2}(x_2) \wedge \mu_{A_{2,k}}(x_2))\} \wedge \mu_{B_k}(y)) \tag{3.35d}$$

$$= \mathrm{hgt}(A'_1 \cap A_{1,k}) \wedge \mathrm{hgt}(A'_2 \cap A_{2,k}) \wedge \mu_{B_k}(y) \tag{3.35e}$$

Hence, the inference reduces to "clipping" the fuzzy set B_k in the consequent of rule r_k by a numerical value β_k:

$$\beta_k = \bigwedge_{i=1}^{N_X} \sup_{x_i} \min(\mu_{A'_i}(x_i), \mu_{A_{i,k}}(x_i)) \tag{3.36a}$$

$$= \bigwedge_{i=1}^{N_X} \mathrm{hgt}(A'_i \cap A_{i,k}) \tag{3.36b}$$

The numerical values β_k are referred to as *support values*, *degree of fulfillment* or *degree of matching* between data and premise of the rule. And indeed, this eliminates the calculations on product spaces, since the composition of relations is eliminated. The inference is reduced to a simple calculation scheme. It is clear that there is no distinction between numerical (crisp) or fuzzy data. In the chapter about fuzzy control it is shown that the "max-min" inference method is often used in fuzzy control (section 4.2.3.1). The inference of a rule is shown schematically in figure 3.4.

The inference of a fuzzy rule can also be simplified when the implication function and the conjunction are chosen to be the *product* operator, and the *sup-product* composition is used. This is shown in figure 3.4. The fuzzy set B_k is multiplied (scaled) by a support value β_k, given by:

$$\beta_k = \bigwedge_{i=1}^{N_X} \sup_{x_i} \mu_{A'_i}(x_i)\mu_{A_{i,k}}(x_i) \tag{3.37a}$$

$$= \bigwedge_{i=1}^{N_X} \mathrm{hgt}(A'_i * A_{i,k}) \tag{3.37b}$$

where $*$ represents the conjunction of fuzzy sets implemented by the product operator. This provides a calculation scheme similar to the previously described method based on the *min* operator.

$$B'_k = A' \circ_* R_k \tag{3.38a}$$

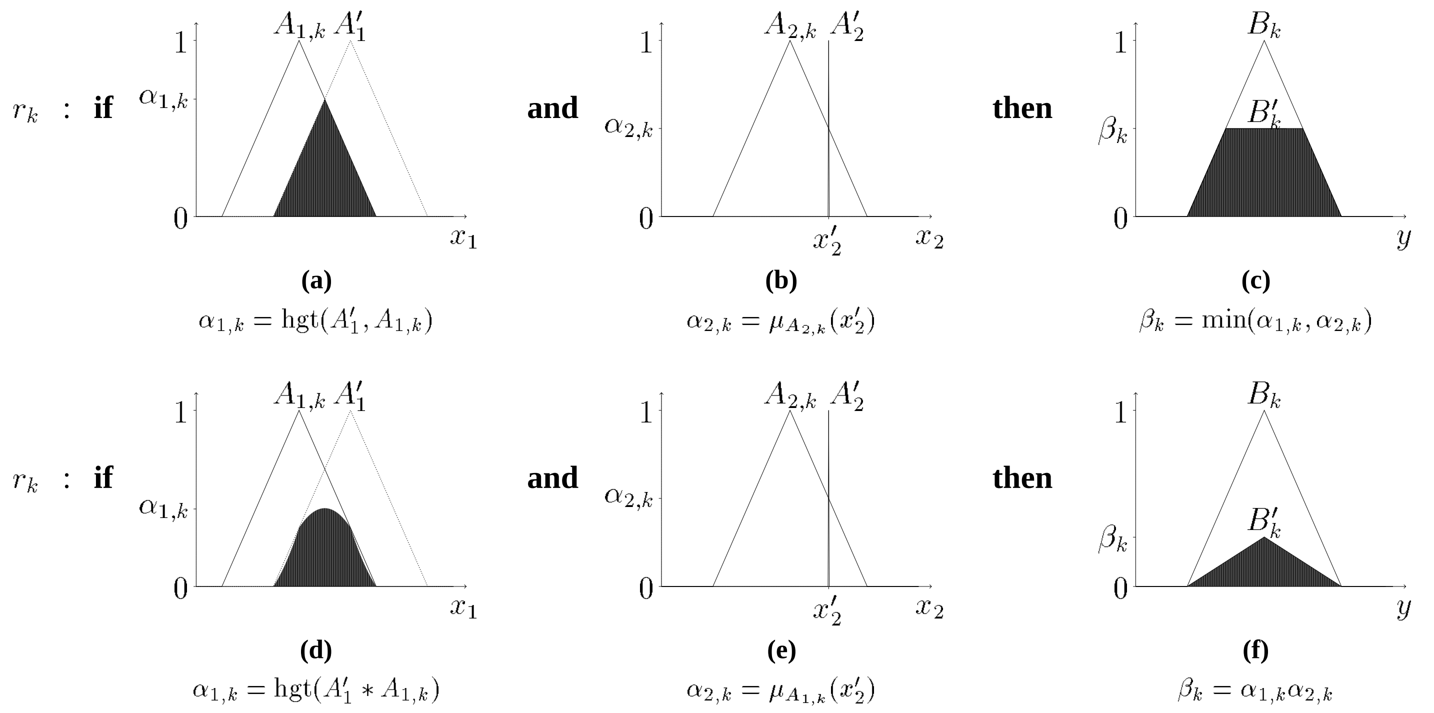

r_k : **if**
and
then
$A_{1,k}$ A'_1
$\alpha_{1,k}$
x_1
$A_{2,k}$ A'_2
$\alpha_{2,k}$
x'_2
x_2
B_k
B'_k
β_k
y
(a)
$\alpha_{1,k} = \mathrm{hgt}(A'_1, A_{1,k})$
(b)
$\alpha_{2,k} = \mu_{A_{2,k}}(x'_2)$
(c)
$\beta_k = \min(\alpha_{1,k}, \alpha_{2,k})$
(d)
$\alpha_{1,k} = \mathrm{hgt}(A'_1 * A_{1,k})$
(e)
$\alpha_{2,k} = \mu_{A_{1,k}}(x'_2)$
(f)
$\beta_k = \alpha_{1,k}\alpha_{2,k}$

Figure 3.4: *Inference of one rule when the conjunction, implication and composition are based on the* min *operator,* **(a)**, **(b)** *and* **(c)**, *or the* product *operator,* **(d)**, **(e)** *and* **(f)**. *The left column shows the case of fuzzy data* **(a)** *and* **(d)**. *The center column shows the case of a crisp (numerical) data. The rule r_k is:* **if** x_1 is $A_{1,k}$ **and** x_2 is $A_{2,k}$ **then** y is B_k.

$$= A' \circ_* (A_k * B_k) \tag{3.38b}$$
$$= \mathrm{hgt}(A' * A_k) * B_k \tag{3.38c}$$

where $\circ_*$ denotes the *sup-product* composition. In section 4.2.3.2 this method is addressed as a typical inference scheme in fuzzy control, known as the “max-product” inference method.

3.3.2 Inference of a fuzzy rule base

In section 3.3.1.1, it was explained how a fuzzy rule, represented by a fuzzy relation, could be used to obtain new data by applying composition of the relation describing the data and the relation describing the rule. Normally, however, the knowledge is represented by a set of parallel fuzzy rules: a *fuzzy rule base*. In section 3.2.3, the aggregation of fuzzy rules was addressed; different types of aggregation operators for different types of implications. Before giving a more detailed description of specific types of implications, different approaches to the inference of a set of parallel rules will be discussed.

3.3.2.1 Local versus global inference

When considering a set of (parallel) rules, it is possible to infer results from individual rules and combine those results into an overall result. This approach is normally used in conventional expert systems. Another approach is to combine all rules beforehand (aggregation, see section 3.2.3) and infer the overall result from this. Hence, a distinction can be made between two approaches to the inference of a set of parallel rules:

- the **local inference** approach performs inference with individual rules (using R_k) and aggregates the results afterwards;
- the **global inference** approach, which assumes a relation R to represent the rule base and R is the aggregation of the fuzzy relations R_k representing the individual rules.

With this nomenclature of the two approaches to the inference of a set of parallel rules we follow Dubois and Prade (1992). Other terminology is possible; for example, Driankov, Hellendoorn and Reinfrank (1993) use *individual-rule based inference* and *composition based inference* to address local and global inference, respectively.

In the following sections a more detailed description is given of specific types of implications. Using T-implications, there is no difference between the results of local and global inference. Using implications which comply to the classical implication will result in differences between local and global inference.

3.3.2.2 Rules modeled by classical-conjunction-based implications

When the implications are represented by T-norms (T-implications), a simplification of the CRI can be obtained without loss of final results. As will be shown in the following, when the aggregation is performed by means of a disjunction, the results of local and global inference are equal. This also holds for implications which comply with the classical conjunction but which are not T-norms. See sections 3.2.2 and 3.2.4 for more details on implications.

The relation R, representing the rule base, is the aggregation of the relations R_k:

$$R = \bigcup_k R_k \tag{3.39}$$

where R_k is the fuzzy relation representing rule r_k. The application of the CRI on the set of rules can be simplified:

$$\begin{aligned} B' &= A' \circ R && \text{(3.40a)} \\ &= A' \circ \left\{ \bigcup_k R_k \right\} && \text{(3.40b)} \\ &= \bigcup_k \{ A' \circ R_k \} && \text{(3.40c)} \\ &= \bigcup_k B'_k && \text{(3.40d)} \end{aligned}$$

Hence, the results from local inference and global inference are equal. This does not mean that the calculus on product spaces (fuzzy relations) is completely eliminated, since the inference of each individual rule still has to be performed by composition: $B'_k = A' \circ R_k$. However, it is possible to obtain analytical results for B'_k in some cases. In section 3.3.1.4 this was shown for some T-implications in combination with a specific choice for the composition.

3.3.2.3 Rules modeled by classical-implication-based implications

In this section, we consider the inference of a set of rules, where the rules are modeled by implications which comply with the classical implication. Hence, a conjunction is used for aggregation of the individual fuzzy relations R_k to obtain the overall fuzzy relation R. Assuming the *min* operator for aggregation, inducing a fuzzy set B' from relation R by data A' results in:

$$B' = A' \circ R \tag{3.41a}$$

$$= A' \circ \left\{ \bigcap_k R_k \right\} \tag{3.41b}$$

$$\subseteq \bigcap_k \{A' \circ R_k\} \tag{3.41c}$$

From this it is clear that the result obtained from global inference, as in (3.41a,b), and the aggregation of results obtained from local inferences, as in (3.41c), can differ. A simple example showing the inefficiency of the local inference approach is the following (Dubois and Prade, 1991). Suppose two rules, $A_1 \to B_1$ and $A_2 \to B_2$, and data $A' = A_1 \cup A_2$, where A_1, A_2, B_1 and B_2 are classical subsets of X and Y, respectively. Then the global inference results in:

$$A' \circ R = B_1 \cup B_2 \tag{3.42}$$

whereas the local inference approach results in:

$$\{A' \circ R_1\} \cap \{A' \circ R_2\} = Y \tag{3.43}$$

where the result Y stands for *unknown*, and indeed $B_1 \cup B_2 \subseteq Y$. However, using local inferences, the result obtained contains no information whatsoever: $B' = \textit{unknown}$. Using global inference, the result is the disjunction of B_1 and B_2, which is the correct result. This simple example shows that less restrictive results can be obtained when using local inference in combination with implications which comply to the classical implications, in the case of fuzzy inputs.

However, in the case of a numerical input a' (a fuzzy set with a singleton as its membership function) this problem is eliminated. When the data A' is represented by the singleton a', the following can be derived:

$$B' = a' \circ R \tag{3.44a}$$

$$= a' \circ \left\{ \bigcap_k R_k \right\} \tag{3.44b}$$

$$= \bigcap_k \{a' \circ R_k\} \tag{3.44c}$$

$$= \bigcap_k \{a' \circ \mathrm{I}(A_k, B_k)\} \tag{3.44d}$$

$$= \bigcap_k \mathrm{I}(\mathrm{hgt}(a' \cap A_k), B_k) \tag{3.44e}$$

In general the results obtained by local inference are less “restrictive” than the results obtained by global inference. The results of local inference are not wrong, only less informative than they could be, based on the available data and knowledge. See section 6.4.2 for a method to obtain analytical solutions for the inference of complex (higher dimensional) rule bases.

3.4 Summary and remarks

Fuzzy propositions are the basic elements for fuzzy logic and reasoning. Combinations of propositions related to different universes are represented by fuzzy relations. Logical connectives can be used to combine fuzzy propositions. Different operators can be used in different contexts or to represent correlation or interactivity between propositions.

Fuzzy rules are if-then statements with fuzzy propositions in the antecedent and consequent of the rules. This is often referred to as *fuzzy if-then* statements. The fuzzy relation representing a fuzzy rule is an implication function applied to a fuzzy set representing the premise of the fuzzy rule, and fuzzy set representing the consequences of the fuzzy rule. As seen in the previous sections, this can be done in several ways. The distinction can be made between implication functions which satisfy the truth table of the classical implication (table 3.2) and implication functions which satisfy the truth table of the classical conjunction.

Two approaches to the inference of a set of parallel fuzzy rules can be distinguished: local and global inference. When the data is numerical, there is no difference between the results of the two approaches. When the rules are modeled by implications which comply with the classical conjunction, the results of local inference and global inference are always equal, because of the disjunction used for aggregation. This is not the case for implications which comply with classical implication. Then the results of local inference are (possibly) less restrictive (informative) than the results obtained from global inference.

4 Fuzzy control

Fuzzy control is addressed in this chapter. The amount of literature on fuzzy control has grown enormously over the last few years. Today, fuzzy control applications can be found in numerous consumer products, and many software tools for the design and development of fuzzy controllers are available these days. In this chapter we do not focus on *applications* of fuzzy control, but describe and analyze the *working* of a fuzzy controller. It is shown in this chapter that fuzzy control can in many cases be simplified by "translation" to a situation where the use of fuzzy logic is not necessary anymore. Then fuzzy set theory is merely used for user interfacing during the *design and development* stages. This concept can be compared to that of high-level programming languages: a compilation phase translates the high-level coded "program" to a lower-level code. During execution of the resulting low-level code, the high-level code is no longer important.

Using the theory given in the previous chapter, a theoretic approach to fuzzy control is given in the first section (4.1). The practical approach to fuzzy control is derived from this theoretic approach and is described in section 4.2. The practical approach is based on local inference of fuzzy rules as described in section 3.3.2.2 of the previous chapter. Also other types of implications with their properties are addressed. Section 4.3 describes the two types of fuzzy rules which are currently in use. The use of linear model structures, where the model parameters are defined by fuzzy rule bases, and controllers based on these models, is described in section 4.4. In section 4.5, a fuzzy controller is described as an input-output mapping and it is shown that a fuzzy controller can "emulate" a linear controller. Section 4.6 analyzes the effect of different aspects of a fuzzy controller on its behavior, defined by the resulting control hypersurface. Some concluding remarks are made in the final section.

4.1 Theoretical approach to fuzzy control

Basically, fuzzy control is the application of the compositional rule of inference as described in section 3.3.1.1. Given a relation R, representing the controller, and a relation A', representing the controller input, a fuzzy output B' can be obtained by composition of A' and R:

$$B' = A' \circ R$$

However, the in- and outputs of a controller are normally numerical values, so a translation is necessary from the numerical inputs to a fuzzy input, and a translation from the fuzzy output to numerical outputs. The first translation is known as *fuzzification*, the latter as *defuzzification*. A schematic representation of a fuzzy controller is given in figure 4.1.

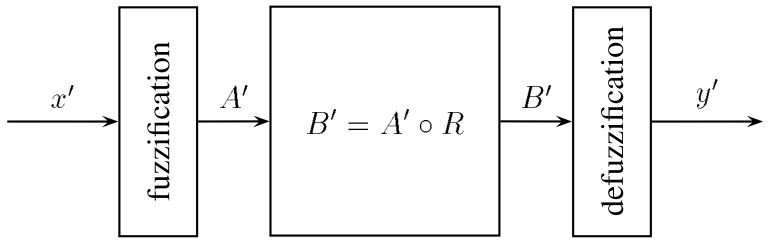


Figure 4.1: *Schematic representation of fuzzy controller. The numerical (crisp) measurement x' is "fuzzified" into A'. The fuzzy output B' is "defuzzified" into a numerical output y'.*

The control algorithm is represented by fuzzy rules. For example, rules have classifications of controller inputs in the premise of the rule, and classifications of an increment of the controller output as consequent. It was already shown in section 3.2.2 how *fuzzy rules* can be represented by *fuzzy relations* using *fuzzy implication functions* and how those relations possibly can be combined into one relation by aggregation (section 3.2.3). The resulting fuzzy relation R is used to represent the controller. The following sections describe the fuzzification and defuzzification, respectively.

4.1.1 Fuzzification of inputs

The fuzzification phase of the fuzzy controller as depicted in figure 4.1 is the construction of a fuzzy input relation. It is possible that the input is represented by a fuzzy relation, in which case fuzzification is not necessary. However, normally the fuzzy input relation A', which is used with the compositional rule of inference, is not available and thus fuzzification is necessary. Then the fuzzy input relation A' is the conjunction of the N_X fuzzy input sets A'_i, where N_X is the number of controller inputs. The fuzzy sets A'_i are fuzzy representations of the controller inputs x'_i:

$$A'_i = \mathrm{fuzz}(x'_i) \tag{4.1}$$

where *fuzz* is a *fuzzifier* function: an operator which translates a numerical value into a fuzzy set representation. When an input is just a numerical value (read: no uncertainty at all), which is normally the case in controllers, the fuzzy set A'_i is simply given by a singleton:

$$\mu_{A'_i}(x_i) = \begin{cases} 1, & \text{if } x_i = x'_i \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

Uncertainty, imprecision or inaccuracy in the inputs can be modeled by using *fuzzy numbers* to represent the inputs (see section 2.1.3 for definition of fuzzy numbers). The complete fuzzy input relation is determined by combining the fuzzy sets for each input:

$$A' = \int_{X_1 \times \cdots \times X_{N_X}} \left(\overset{N_x}{\underset{i=1}{\mathrm{T}}} \mu_{A'_i}(x_i) \right) / (x_1, \ldots, x_{N_X}) \tag{4.3}$$

where T is the T-norm used to perform the conjunction in the premise. This shows that the input data is in fact represented by: A'_1 **and** . . . **and** A'_i **and** . . . **and** A'_{N_X}.

4.1.2 Defuzzification of output

Defuzzification is needed to translate the fuzzy output of a fuzzy controller to a numerical representation. When we consider a fuzzy controller from the theoretical point of view, the fuzzy output can be a multi-dimensional fuzzy set (fuzzy relation). This assumes that the controller can have multiple outputs (SIMO[*] or MIMO system) which causes the fuzzy

[*]Single-Input Multiple-Outputs.

output of the controller to be a multi-dimensional fuzzy set. For the defuzzification of fuzzy relations two basic methods are available: *center-of-gravity* and *mean-of-maxima*. The center-of-gravity method is first described. Section 4.1.2.2 addresses a modifier for this defuzzification method. The mean-of-maxima method can be derived as a special case and will be discussed in section 4.1.2.3. Another defuzzification method is the center-of-area method, which is described in section 4.1.2.4.

4.1.2.1 Center-of-gravity defuzzification

Intuitively, defuzzification can be done using an averaging technique. The center-of-gravity method is nothing but the same method employed to calculate the center of gravity of a mass. The difference is that the (point) masses are replaced by the membership values. This method is often called the center-of-area defuzzification method in the case of 1-dimensional fuzzy sets. In section 4.1.2.4, however, another defuzzification method for fuzzy sets is described for which the name "center-of-area" suites better. The center-of-gravity defuzzification method is defined by:

$$\mathrm{cog}(B') = \frac{\displaystyle\int_Y \mu_{B'}(y)\, y\, dy}{\displaystyle\int_Y \mu_{B'}(y)\, dy} \tag{4.4a}$$

and the discrete form is defined by:

$$\mathrm{cog}(B') = \frac{\displaystyle\sum_{q=1}^{N_q} \mu_{B'}(y_q)\, y_q}{\displaystyle\sum_{q=1}^{N_q} \mu_{B'}(y_q)} \tag{4.4b}$$

where N_q is the number of quantizations used to discretize membership function $\mu_{B'}(y)$ of fuzzy output B'. In figure 4.2 examples are given of both the continuous (figure 4.2a) and the discrete (figure 4.2b) center-of-gravity defuzzification method.

The application of the center-of-gravity defuzzification method is not limited to 1-dimensional fuzzy sets. It is a defuzzification method of a n-ary fuzzy relation. To obtain a numerical value for the j^{th} dimension the following holds:

$$\mathrm{cog}_{y_j}(B') = \frac{\displaystyle\int_Y \mu_{B'}(\boldsymbol{y})\, y_j\, dy}{\displaystyle\int_Y \mu_{B'}(\boldsymbol{y})\, dy} \tag{4.5a}$$

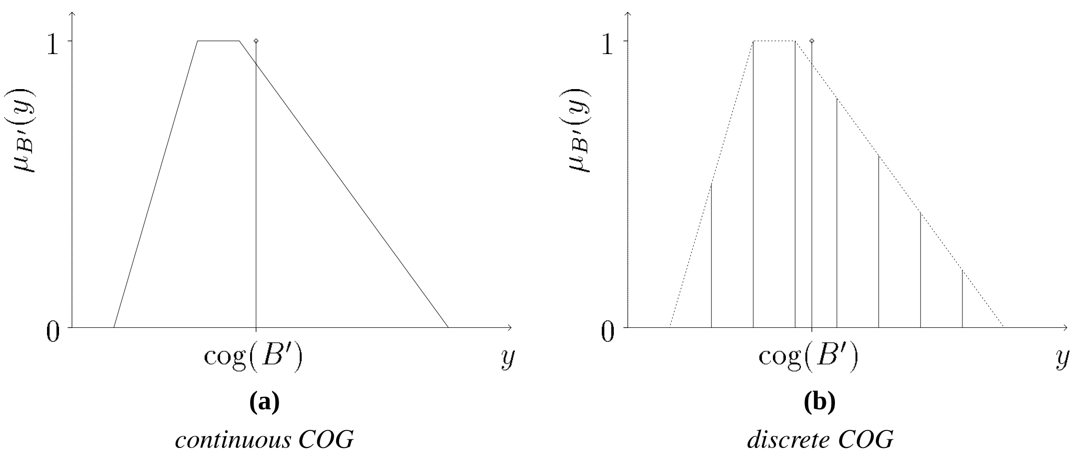


Figure 4.2: *Examples of continuous* **(a)** *and discrete* **(b)** *center-of-gravity (COG) defuzzification method. See also figure 4.4.*

where Y is the product space of the output universes and $\boldsymbol{y}$ represents the controller outputs as a vector. In discrete form, the center-of-gravity defuzzification method to obtain a numerical value for the j^{th} dimension is defined by:

$$\mathrm{cog}_{y_j}(B') = \frac{\sum_{q=1}^{N_q} \mu_{B'}(\boldsymbol{y}_q)\, y_{j,q}}{\sum_{q=1}^{N_q} \mu_{B'}(\boldsymbol{y}_q)} \tag{4.5b}$$

where N_q is the number of quantizations used for the discretization of membership function $\mu_{B'}(\boldsymbol{y})$ of fuzzy output relation B'. The point $\boldsymbol{y}_q$ is the q^{th} quantization of the product space Y of the output universes. Mostly, the defuzzification methods are used for 1-dimensional fuzzy sets, however.

4.1.2.2 Indexed defuzzification methods

Indexed (or threshold) defuzzification methods (here identified by *idfz*) are used to discriminate part of a fuzzy output of which the membership values are below a certain threshold

β_t:

$$\text{idfz}(B', \beta_t) = \text{dfz}(B' \cap \alpha\text{-cut}(B', \beta_t)), \text{ with } \alpha = \beta_t \tag{4.6a}$$
$$= \text{dfz}(B' \cap \beta_t\text{-cut}(B')) \tag{4.6b}$$

where *dfz* denotes a defuzzification method and *idfz* is the indexed version of that method. The defuzzification method *dfz* is only applied on the part(s) of the fuzzy output, which has a membership value greater than or equal to β_t. This discrimination provides a way to let the defuzzification of the fuzzy output tend towards the maxima of the fuzzy output. In figure 4.3a, an example is given of the indexed center-of-gravity with $\beta_t = \frac{1}{2}$. One can see indexed defuzzification methods as a sort of filter for the fuzzy output. The next section describes the mean-of-maxima defuzzification method which can be defined as an indexed center-of-gravity defuzzification method.

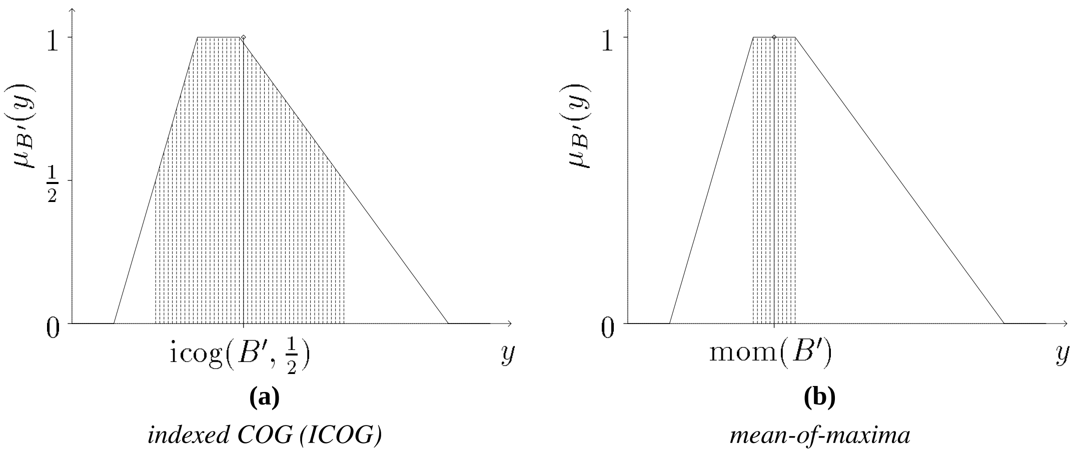
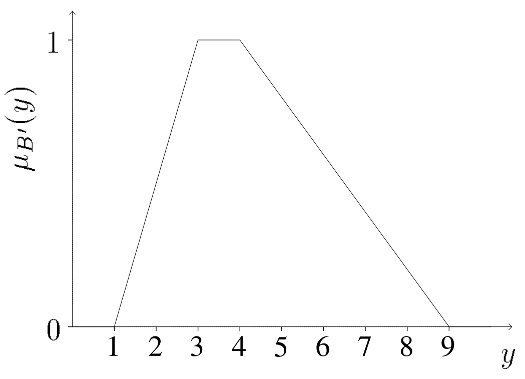


Figure 4.3: *Examples of indexed (threshold) COG* **(a)** *and mean-of-maxima (MOM) defuzzification method* **(b)**. *See also figure 4.4.*

4.1.2.3 Mean-of-maxima defuzzification

Besides the center-of-gravity method, another basic defuzzification method is the mean-of-maxima defuzzification method (MOM), which is defined by:

$$\text{mom}(B') = \text{icog}(B', \text{hgt}(B')) \tag{4.7a}$$



(a)
membership function

method	$\mathrm{dfz}(A)$	$\mathrm{idfz}(A, \frac{1}{2})$
cont. COG	$4\frac{4}{9}$	$4\frac{4}{29}$
discr. COG	$4\frac{4}{9}$	4
COA	$4\frac{1}{2}$	$4\frac{1}{16}$
MOM	$3\frac{1}{2}$	$3\frac{1}{2}$

(b)
numerical results

Figure 4.4: *The membership function of a fuzzy set B'* **(a)** *and the numerical results of different defuzzification methods* **(b)**. *The discrete COG uses discretizations at 0,1,2,.... See also figures 4.2, 4.3 and 4.5.*

$$= \mathrm{cog}(B' \cap \alpha\text{-}\mathrm{cut}(B', \mathrm{hgt}(B'))) \tag{4.7b}$$

where *icog* is an indexed- or threshold-version of the center-of-gravity defuzzification method as defined by (4.6). In figure 4.3b one can see an example of determining *mom*(B'). It is obvious that this defuzzification method ignores a great deal of the information provided by the fuzzy set due to the application of an α-*cut* with $\alpha =$ *hgt*(B').

4.1.2.4 Center-of-area defuzzification method

Probably the best-known defuzzification method (by name) is the *center-of-area* defuzzification method (COA). However, almost always another defuzzification method is meant, namely the center-of-gravity defuzzification method, described in section 4.1.2.1 and 4.2.4.1. We make this distinction because, although the names are interchanged frequently in literature, there are two different defuzzification methods with the same name. In this thesis the center-of-area is defined by:

$$\int_{\inf\limits_Y y}^{\mathrm{coa}(B')} \mu_{B'}(y)\,dy = \int_{\mathrm{coa}(B')}^{\sup\limits_Y y} \mu_{B'}(y)\,dy \tag{4.8}$$

resulting in a numerical value $y_{\mathrm{coa}(B')}$ which divides the area under the membership function into two equal parts. In figure 4.5, an example of the center-of-area defuzzification method is shown. It is clear that this defuzzification method is primarily ment for 1-

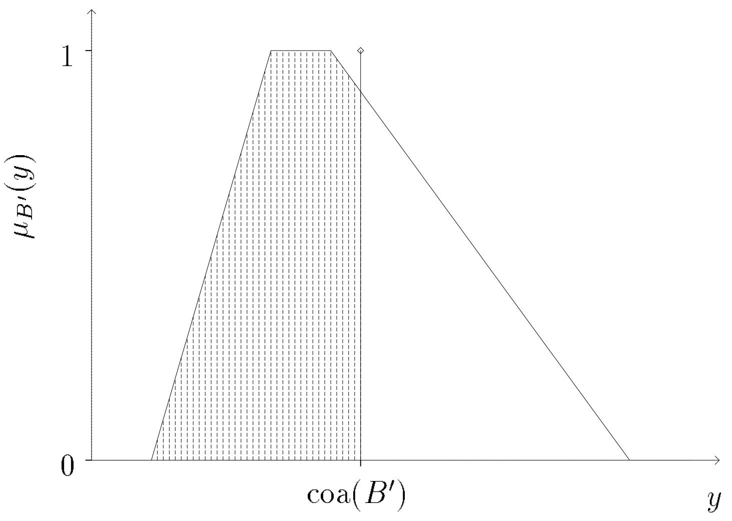


Figure 4.5: *Example of center-of-area defuzzification method; the dark and the light areas are equal in magnitude. See also figure 4.4.*

dimensional fuzzy sets (considering the word “area” in its name), although an extension to multi-dimensional fuzzy sets is possible. For example, a 2-dimensional fuzzy set with membership function $\mu_{B'}(y_1, y_2)$ can be defuzzified by finding a value y_1' and y_2' for which:

$$\int_{\inf\limits_{Y_2} y_2}^{\sup\limits_{Y_2} y_2} \int_{\inf\limits_{Y_1} y_1}^{y_1'} \mu_{B'}(y_1, y_2)\, dy_1\, dy_2 = \int_{\inf\limits_{Y_2} y_2}^{\sup\limits_{Y_2} y_2} \int_{y_1'}^{\sup\limits_{Y_1} y_1} \mu_{B'}(y_1, y_2)\, dy_1\, dy_2$$

$$\int_{\inf\limits_{Y_2} y_2}^{y_2'} \int_{\inf\limits_{Y_1} y_1}^{\sup\limits_{Y_1} y_1} \mu_{B'}(y_1, y_2)\, dy_1\, dy_2 = \int_{y_2'}^{\sup\limits_{Y_2} y_2} \int_{\inf\limits_{Y_1} y_1}^{\sup\limits_{Y_1} y_1} \mu_{B'}(y_1, y_2)\, dy_1\, dy_2$$

In practice, the application of this defuzzification method would normally be done by using discretizations of the fuzzy sets to defuzzify, although one should note that in practice usually 1-dimensional fuzzy outputs are considered. In section 4.2.4.3, a parameterized version of the COA method is described: the *extended center-of-area* defuzzification method (XCOA).

4.1.3 Example of theoretical approach

A simple example of a fuzzy controller is given in this section. The fuzzy controller is based on the theoretical approach to fuzzy control as described in the previous sections and shown schematically in figure 4.1. The fuzzy controller is a SISO[*] system and can be considered as a nonlinear P controller when the controller input is the difference between the desired and actual process output.

First the fuzzy sets for the controller input and output have to be chosen. In figure 4.6, the fuzzy sets for the input (figure 4.6a) and output (figure 4.6b) are shown. This choice is rather arbitrary, but 5 to 9 sets per universe is quite common in fuzzy control. The following step is to set up the fuzzy rule base. The rule base contains the following fuzzy rules:

r_1 : **if** x is *NB* **then** y is *NB*
r_2 : **if** x is *NS* **then** y is *NB*
r_3 : **if** x is *AZ* **then** y is *AZ*
r_4 : **if** x is *PS* **then** y is *PS*
r_5 : **if** x is *PB* **then** y is *PB*

where:

AZ	: about zero
NB	: negative big
NS	: negative small
PB	: positive big
PS	: positive small

For each rule a fuzzy relation R_k has to be constructed. To obtain the fuzzy controller relation R, the fuzzy relations R_k are aggregated. The fuzzy relation R is shown in figure 4.6c. What is now left is to choose a defuzzification method. The center-of-gravity defuzzification method is used in this example. At this stage the design of the fuzzy controller is complete when the knowledge represented by the rules is assumed to be correct.

To determine the behavior of the fuzzy controller a numerical controller input x' is considered as shown in figure 4.6a. In figure 4.6d, one can see the result of the composition of the input relation A' (a singleton in this case) and the controller relation R. This fuzzy result has to be defuzzified to obtain a numerical controller output.

[*] Single-Input Single-Output.

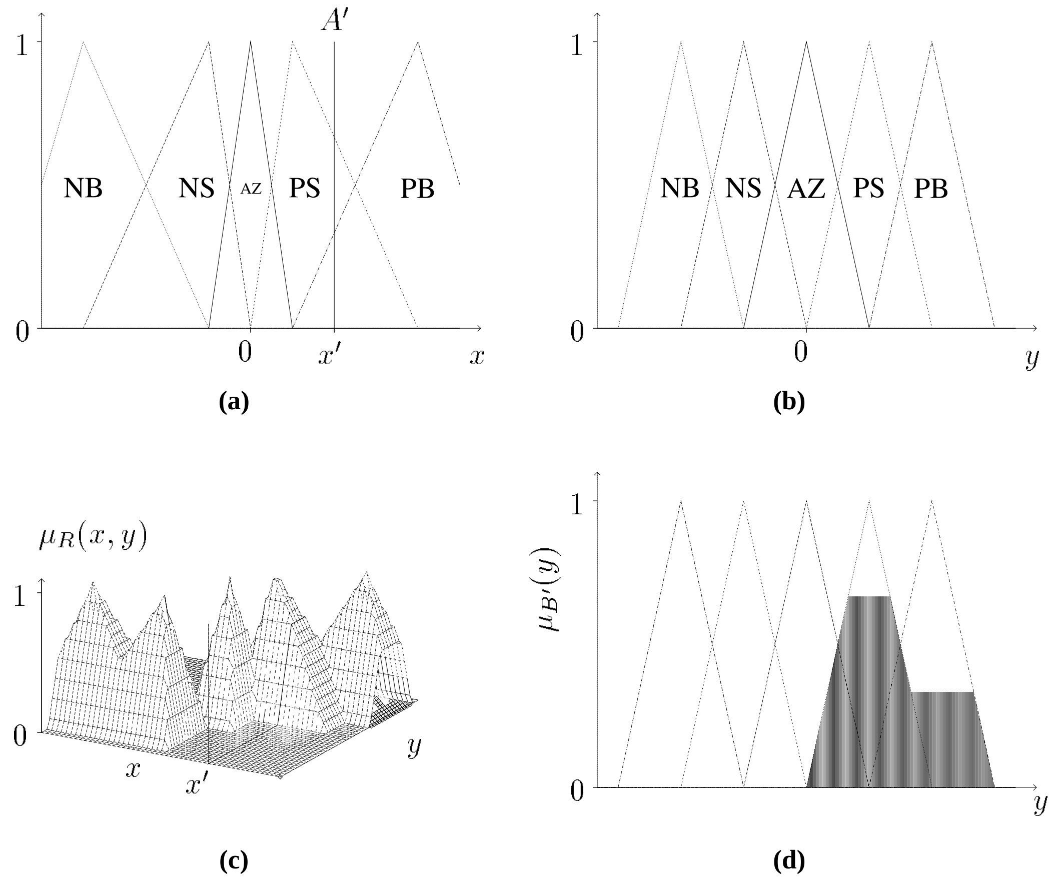


Figure 4.6: *Example of theoretical approach to fuzzy control. Upper figures shown fuzzy sets for input* **(a)** *and output* **(b)** *universe. The fuzzy controller relation R* **(c)** *and result after composition with fuzzy input relation A'* **(d)** *which is the singleton shown in* **(a)**.

It should be noted that the implementation of a fuzzy controller based on the theoretical (relation-based) approach requires the fuzzy relations to be discretized for storage in computer memory. To minimize errors due to the discretization (see also section 2.4.2) the discretization step chosen should be small enough.

In the above-given example, the result B' can also be obtained by:

$$\mu_{B'}(y) = \max(\min(\mu_{\mathrm{PS}_x}(x'), \mu_{\mathrm{PS}_y}(y)), \min(\mu_{\mathrm{PB}_x}(x'), \mu_{\mathrm{PB}_y}(y)))$$

This is in fact the method that is mostly used in fuzzy control. Here, this is referred to as the *practical approach* to fuzzy control and is addressed in more detail in the following section.

4.2 Practical approach to fuzzy control

In the description of the theoretical approach to fuzzy control, it was shown that the fuzzy control algorithm consists of three phases: *fuzzification*, *composition of fuzzy input and controller relation*, and *defuzzification*. This involves a quite complex calculus with multi-dimensional functions (fuzzy relations), which is undesirable in practice in terms of memory requirements and calculational load. Thus, in practice, fuzzy control is applied using local inferences; see section 3.3.2.1 for an explanation of local and global inference of fuzzy rules.

For the sake of simplicity, numerical inputs are considered for most cases described in this section. This is not a severe restriction, since in (low-level) control the controller inputs are normally numerical values, for example signals read from sensors. In addition to the practical approach to the inference method (section 4.2.1), practical approaches to the fuzzification (section 4.2.2 and defuzzification (section 4.2.4) are discussed.

4.2.1 Fuzzy inference in practice

In the practical approach to fuzzy control, the inference of a rule base is based on local instead of global inference. Using local inference, the inference of a rule base is broken down to inference of individual fuzzy rules and the results are aggregated afterwards. It has been shown in section 3.3.2.1 that the results of local inference equal the results of global inference if the controller inputs are assumed to be numerical. In the following sections a practical fuzzy inference scheme[*] is described and properties of different fuzzy implications with respect to control are addressed.

4.2.1.1 Practical fuzzy inference scheme

In section 4.1, the fuzzy inference scheme according to the "theoretical approach" was shown in figure 4.1. In the following we focus on a fuzzy inference scheme according to the "practical approach", which is primarily characterized by local inference. Basically the inference in fuzzy control is represented by the following steps:

1. Matching of fuzzy propositions x *is* $A_{i,k}$, used in the premises of fuzzy rules r_k, with the numerical data x_i' (controller inputs):

   $$\alpha_{i,k} = \mu_{A_{i,k}}(x_i')$$

   where $\alpha_{i,k}$ is a numerical value representing the matching. In the case of fuzzy inputs A_i' the matching is normally represented by: $\alpha_{i,k} = \mathrm{hgt}(A_i' \cap A_{i,k})$.

2. Determining the degrees of fulfillment (DOF) β_k for each rule r_k:

   $$\beta_k = \overset{N_X}{\underset{i=1}{\mathrm{T}}} \, \alpha_{i,k}$$

   where T is the T-norm representing the *and* connective in the premises of the rules. If the *or* connective is used, this T-norm has to be replaced by an S-norm. Of course, both the *and* and the *or* connective can be used in the same premise. In that case some values $\alpha_{i,k}$ have to be combined by means of a T-norm (*and* connective), others by means of an S-norm (*or* connective).

3. Determining the result B_k' of each individual rule r_k:

   $$\mu_{B_k'}(y) = \mathrm{I}(\beta_k, \mu_{B_k}(y))$$

   where I is the implication used to model the fuzzy rules. This can be classical conjunction-based fuzzy implications or classical implications-based fuzzy implications.

[*]Today, many commercial software packages, sometimes in combination with dedicated hardware, are available for developing fuzzy controllers. The fuzzy controllers developed with these tools are primarily based on the practical approach to fuzzy control as described in this section.

4. Aggregation of the overall result B' of the results B'_k of the individual fuzzy rules r_k:

$$\mu_{B'}(y) = \begin{cases} \bigcup\limits_k \mu_{B'_k}(y), & \text{for classical conjunction-based implications} \\ \bigcap\limits_k \mu_{B'_k}(y), & \text{for classical implication-based implications} \end{cases}$$

The different aggregation operators for different types of implications were discussed in section 3.2.2.

As can be observed from the above given fuzzy inference scheme, the fuzzification as depicted in figure 4.1 is more or less embedded in the practical inference scheme: there is no construction of a fuzzy input relation. In section 4.2.2 we discuss a way of matching fuzzy propositions with the available data which is sometimes referred to as "fuzzification" in the application of fuzzy control. A number of cases for specific types of fuzzy implications are discussed in the following sections.

4.2.1.2 Inference with T-implications

In section 3.3.2.2, it was shown that the inference can be simplified when T-norms are used for the implication function. First of all, the results of local and global inference are equal, even in the case of fuzzy inputs:

$$A' \circ \left\{ \bigcup_k R_k \right\} = \bigcup_k \{A' \circ R_k\} \tag{4.9}$$

Considering only numerical controller inputs (singletons), the fuzzy inference of each rule is reduced to matching the data with the premise of the rule:

$$\mu_{B'_k}(y) = \mathrm{T}(\beta_k, \mu_{B_k}(y)), \text{ with} \tag{4.10a}$$

$$\beta_k = \mathop{\mathrm{T}}_{i=1}^{N_X} \mu_{A_{i,k}}(x'_i) \tag{4.10b}$$

where x'_i are the (numerical) controller inputs. Hence, each consequent B_k is restricted by the value β_k by means of a T-norm, representing the implication. The fuzzy output is obtained by the aggregation of those subresults using the *max* operator:

$$\mu_{B'}(y) = \max_k \mu_{B'_k}(y) \tag{4.11}$$

This yields a simple and straightforward way to obtain analytical results for the fuzzy controller output without severe calculational load and memory requirements. This method is primarily used in the application of fuzzy control. A simple example is given by the following rule:

if error is small **and** error change is big **then** reduce control signal

which can be part of a PI-like fuzzy controller. Considering literature on fuzzy control, one might get the impression it is the only means of fuzzy inference. Indeed, in the field of fuzzy control mostly T-implications are used. In the next section we describe other implications and their effects in the application of fuzzy control.

4.2.1.3 Inference with S-implications

In the case of S-implications, complying with the classical implication, the results of local inference only equal the results of global inference when the inputs are not fuzzy. Because this is normally the case in control, we assume this also in this section. The fuzzy output of the controller is then determined by:

$$\mu_{B'}(y) = \min_k \mu_{B'_k}(y) \tag{4.12a}$$

$$= \min_k \mathrm{S}(1-\beta_k, \mu_{B_k}(y)), \text{ with } \beta_k = \mathop{\mathrm{T}}_{i=1}^{N_X} \mu_{A_{i,k}}(x'_i) \tag{4.12b}$$

where x'_i are the numerical inputs. This result might seem as simple and straightforward as in the case of T-implications, but applying S-implications in fuzzy control can lead to some results which are undesired in control. These problems occur because after the fuzzy output of the controller is obtained, this fuzzy output has to be defuzzified to obtain a numerical output. Fuzzy results which are "correct" from a logical point of view can result in an *undefined controller output* since defuzzification is meaningless in some cases. In the following we focus on characteristic properties of fuzzy controllers based on S-implications.

A problem when using S-implications in fuzzy control is the possible *indetermination* of the fuzzy output. This is because the following holds for S-implications:

$$\mathrm{I}(\beta_k, 0) = \mathrm{S}(1-\beta_k, 0) = 1-\beta_k$$

This implies that in case the consequents B_k of the rules (at least two rules with different consequents), for which $\beta_k = \beta$, have an empty intersection, and $\beta_k = 0$ for all other rules, the fuzzy output $\mu_{B'}(y) = 1 - \beta$ is obtained because the aggregation is performed by means of the *min* operator. Hence, defuzzification is useless to obtain a numerical controller output. Although such a situation is not likely to occur often, it can occur and thus using S-implications in the application of fuzzy control can lead to an undetermined controller output.

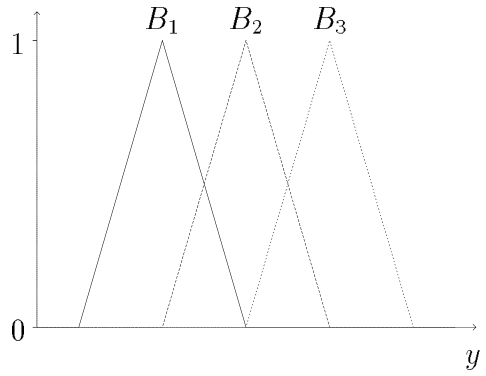
To avoid empty intersections of the consequents of the rules, continuity of the rule base is desired (see also section 3.3.2.3 and 3.2.5). For specific S-implications the indetermination can occur under less severe conditions. For example, in the case of the Kleene-Dienes implications, $I(a, b) = \mathit{max}(1 - a, b)$, the output can even be undeterminated when the rule base is continuous and complete (see section 3.2.5.3) and $\beta \le \frac{1}{2}$.

When we consider more than one input, continuity of the rule base is not sufficient. For example, consider the S-implication according to Łukasiewicz, $I(a, b) = \mathit{min}(1-a+b, 1)$. In the case of two inputs it is possible that the situation occurs that $\beta_k = \beta$ for a number of rules and $\beta_k = 0$ for others. The result is undetermined when there are more than two output fuzzy sets for which $\beta_k = \beta$ and the fuzzy sets for the output are a fuzzy partition. An example showing this possible indetermination is given in figure 4.7.

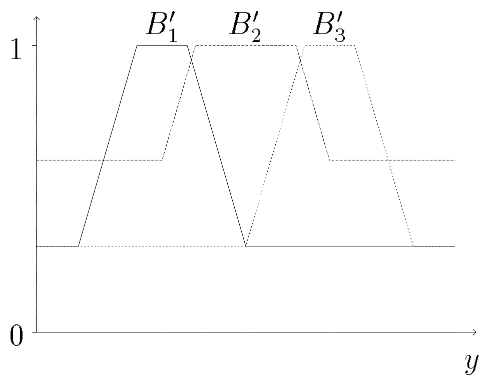
Another problem related to the required continuity of the rule base is that the inference, when using S-implications, ignores results of individual rules in favor of the result of the rule with the highest value for β_k (DOF) when the rule base is not continuous. In figure 4.8 an example is shown. It can be seen that one subresults influences the height of the other. From a logical point of view, this can be explained as having a contradiction in the rule base and the inference causes a choice for the conclusion with the highest degree of fulfillment β_k: the inference forces a decision in favor of the rule with the highest DOF.

It is clear that averaging defuzzification methods are not suitable in the case of S-implications, since these will take the complete universe into account and thus produce a bias. The mean-of-maxima method does not have this drawback, but stresses the decision already forced by the inference when the rule base lacks continuity. In fuzzy control, this will result in a crisp transition from one consequent to another due to the defuzzification, which is necessary to obtain numerical controller outputs. This even holds for continuous rule bases when the Kleene-Dienes implication is used. In case the rule base is continuous, the inference performs “interpolation” when using the Łukasiewicz in combination with the mean-of-maxima defuzzification method. However, indetermination of the output in the case of more than one input is possible as discussed above.

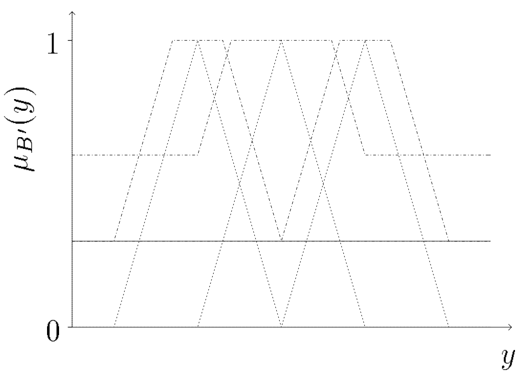
To summarize, S-implications are not well suited for the application of fuzzy control since they can lead to undetermined (fuzzy) outputs. Another important property is that the inference makes a decision in favor of the rule with the highest degree of fulfillment. This decision making can cause discontinuous transitions from one numerical controller output to another.



(a)
fuzzy sets for output

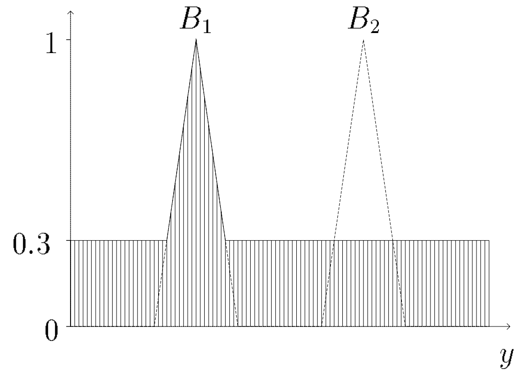


(b)
inferred subresults

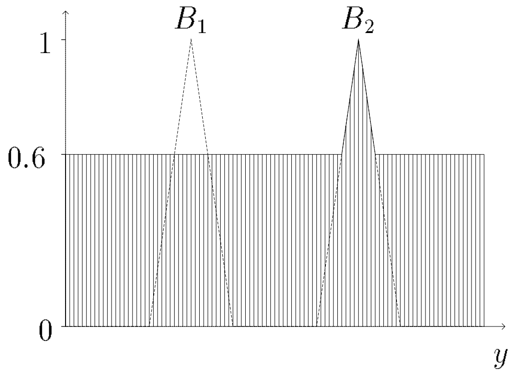


(c)
overall result (solid)

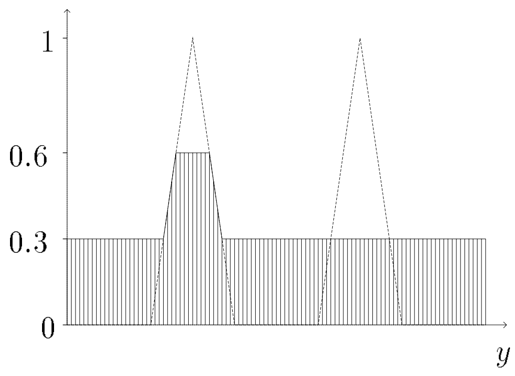
Figure 4.7: *Example showing possible indetermination in the case of S-implications; Łukasiewicz in this case,* $\mathrm{I}(a, b) = \min(1 - a + b, 1)$.



(a)
$\mu_{B_1'}(y) = \max(1 - \beta_1, \mu_{B_1}(y))$,
with $\beta_1 = 0.7$



(b)
$\mu_{B_2'}(y) = \max(1 - \beta_2, \mu_{B_2}(y))$,
with $\beta_2 = 0.4$



(c)
$\mu_{B'}(y) = \min(\mu_{B_1'}(y), \mu_{B_2'}(y))$

Figure 4.8: *Example showing a possible problem when using S-implications (where* $\mathrm{I}(a, b) = \max(1 - a, b)$ *in this case) in fuzzy control: the inference forces a decision in favor of the consequent with the highest degree of fulfillment.*

4.2.1.4 Inference with other implications

The use of implications that comply with the classical implication, and are not S-implications, are discussed in this section. The results of local inference equal the results of global inference when the inputs are not fuzzy and, as in the previous section, for this reason only numerical controller inputs are considered. The fuzzy output of the controller is determined by:

$$\mu_{B'}(y) = \min_k \mathrm{I}(\beta_k, \mu_{B_k}(y)), \text{ with } \beta_k = \mathop{\mathrm{T}}_{i=1}^{N_X} \mu_{A_{i,k}}(x_i') \tag{4.13}$$

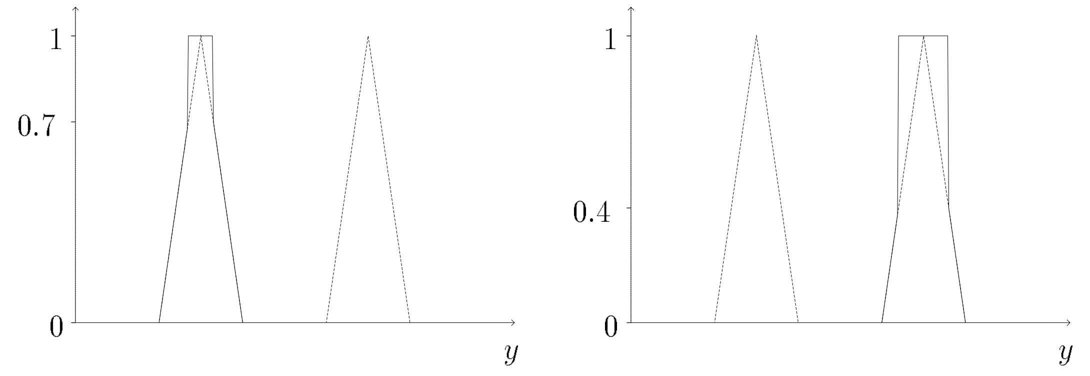
As in the case of S-implications, applying other types of classical-implication-based implications in fuzzy control can result in “undesired” outcomes.

A major problem which might occur is that the fuzzy output is undetermined. If we consider implications which are restricted by:

$$\mathrm{I}(\beta_k, \mu_{B_k}(y)) = 0, \text{ if } \mu_{B_k}(y) = 0 \tag{4.14}$$

and suppose that the intersection of the consequents B_k of the rules for which $\beta_k > 0$ is empty, then $\mu_{B'}(y) = 0$. Many R-implications (except those R-implications which are also S-implications, like Łukasiewicz’s implication) suffer from this. This problem is because the aggregation is based on the *min* operator. In figure 4.9 an example is shown using Gödel’s implication. Using this type of implication requires continuity of the rule base: “adjacent” rules should have “adjacent” consequents. This continuity of rule bases is addressed in section 3.2.5. In the previous section it was shown that continuity of the rule base is not sufficient in the case of multiple inputs in combination with S-implications. This also holds for the implications meeting the condition given by (4.14). When, for example, the fuzzy sets for the output are a fuzzy partition and the rule base is continuous and complete, indetermination of the output occurs when at least three rules have a DOF $\beta_k > 0$. This situation is very well possible when the controller has more than one input, which is normally the case in control problems.

The restriction that the continuity of the rule base is required is something which is probably not met in the practical application of fuzzy control. Moreover, requiring continuity of the rule base (as described in section 3.2.5.1) is not sufficient in the case of multiple controller inputs, which is normally the case in practice. This makes implications based on partial ordering, among which R-implications (see section 3.2.2), not really suitable for fuzzy control based on the inference scheme given in section 4.2.1.1. In the previous section, it was shown that S-implications also had a number of disadvantages with respect to application in fuzzy control. Hence, in general it can be stated that implications which



(a)
$\mu_{B'_{1,k}}(y) = \mathrm{I}(\beta_{1,k}, \mu_{B_{1,k}}(y),$
with $\beta_k = 0.7$

(b)
$\mu_{B'_{2,k}}(y) = \mathrm{I}(\beta_{2,k}, \mu_{B_{2,k}}(y),$
with $\beta_k = 0.4$

Figure 4.9: *Possible problem when using R-implications in fuzzy control (Gödel's implication in this case, given by (3.11b) with* $\mathrm{T} \equiv \min$*): the resulting fuzzy output is* 0*, yielding indetermined controller output.*

comply with the classical implication are not suitable for application in fuzzy control based on the inference scheme given in section 4.2.1.1, since indetermination of the controller output is possible, even when the rule base is continuous and complete.

4.2.2 Input fuzzification

In the practical approach to fuzzy control, the fuzzification is not the construction of a fuzzy input relation. In section 4.2.1.1 the fact that the fuzzification is more or less embedded in the inference mechanism was discussed. With this in mind, it can be stated that the "fuzzification" phase consists of determining the matching between the controller inputs and the fuzzy sets that represent the linguistic labels for the inputs in rule premises. This is normally not a preprocessing phase like the construction of a fuzzy input relation in the theoretical approach. When evaluating the premise of a rule, the necessary matching values can be calculated on the fly. This prevents the unnecessary calculation of matching values, since they are only calculated when needed. In section 3.3.2.1 it was shown that the results of local inference and global inference can be different when fuzzy inputs are considered. In section 4.2.1 it was shown that the use of implications which comply with the classical implication have the disadvantages that indetermination of the output can occur. Therefore, in the following, only the fuzzification is considered for fuzzy control where T-implications are used to model the rules.

Since fuzzification in the practical approach in fact consists of determining the matching between inputs and linguistic labels used in the premises of rules, the "fuzzification" in the case of T-implications is normally determined by (compare with (3.36) on page 69):

$$\alpha_{i,j} = \mathrm{hgt}(A'_i \cap A_{i,j})$$

where $\alpha_{i,j}$ represents the matching between the data A'_i for input x_i and the j^{th} fuzzy set $A_{i,j}$ on the universe of discourse of x_i. Using this type of matching in combination with certain types of modifications of the consequent (step 3 in the enumerated steps on page 1) do not conform to the theoretical approach based on the composition of relations. In section 4.2.3 a number of commonly used practical inference methods are described. For some types of inference it is shown that certain combinations of matching (step 1 and 2) and modification (step 3) do not conform to the composition of fuzzy relations.

In figure 4.10, examples are shown of the matching of (fuzzy) inputs and fuzzy sets defined for a specific input. The matching of numerical inputs with the rule premises is a special case of the matching of a fuzzy input with the rule premises.

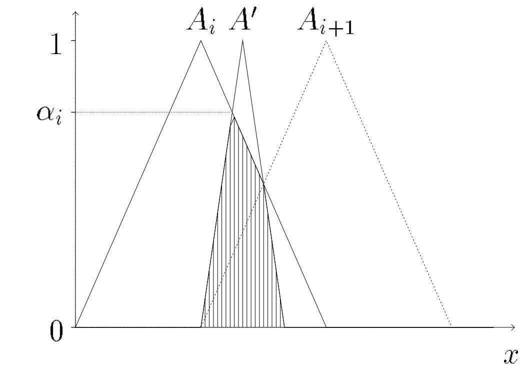
Sometimes fuzzification in fuzzy control is presented as a transformation of an input to a vector containing degrees of matching for every linguistic label available on the universe of discourse of that input. When the example given in figure 4.10 is considered, the following vector with degrees of matching for inputs x can be constructed:

$$\boldsymbol{\alpha} = [0 \ \ \ldots \ \ 0 \ \ \alpha_i \ \ \alpha_{i+1} \ \ 0 \ \ \ldots \ \ 0]$$

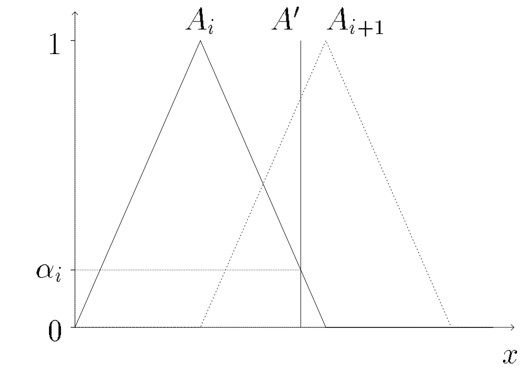
where α_i represents the matching between the data A' and the fuzzy set A_i on the universe of discourse of x:

$$\alpha_i = \begin{cases} \mathrm{hgt}(A'_i \cap A_i), & \text{in general} \\ \mu_{A_i}(x'), & \text{for numerical input} \end{cases}$$

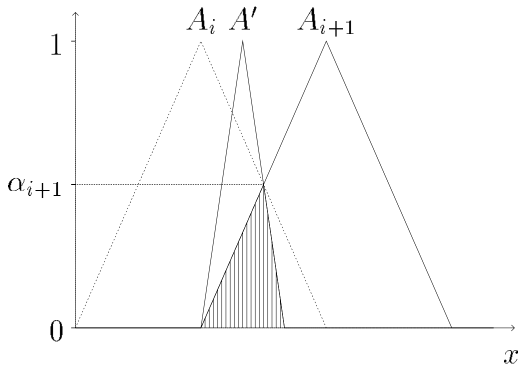
However, if shifted hedges are allowed to be used within the premise of fuzzy rules (see section 2.2.2 for shifted hedges and section 2.2.3 for scaled hedges as a special case of shifted hedges), this method is not applicable, because, in that case, there is no fixed number of fuzzy sets A_i on the universes of discourse of the input that can be used in the premises of fuzzy rules. This is because shifted hedges do not operate on the membership function, but on its domain.



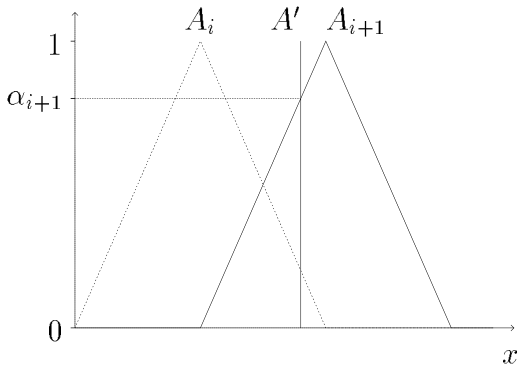
(a)
fuzzy input



(b)
numerical input



(c)
fuzzy input



(d)
numerical input

Figure 4.10: *"Fuzzification" in practice when using T-implications for fuzzy rules: matching between (fuzzy) inputs and fuzzy sets representing linguistic labels used in the premise of fuzzy rules. Matching for both fuzzy input,* **(a)** *and* **(c)***, and numerical input,* **(b)** *and* **(d)***, is shown.*

If powered hedges (see section 2.2.1) are to be used, this “fuzzification” method is indeed applicable, provided that the controller inputs are numerical, since powered hedges operate on the membership functions and not on their domains. Hence, the value α_{i,m_p} representing the matching between the numerical input x' and the proposition *“x is $m_p(A_i)$”* is determined by:

$$\begin{aligned}\alpha_{i,m_p} &= \mu_{\mathrm{m}_p(A_i)}(x') \\ &= \mu^p_{A_i}(x') \\ &= \alpha^p_i\end{aligned}$$

where m_p is the linguistic modifier according to the powered hedges approach. This leaves a simple and easy-to-implement algorithm to determine the matching values of numerical data and fuzzy proposition used in the rules in the fuzzy controller’s rule base if powered hedges are used in the fuzzy propositions.

4.2.3 Common “inference” methods

In literature, one can find statements such as “we used *sum-product* inference” or “*max-dot* inference method was used”. The most common inference methods applied in fuzzy control are described in the following sections. Some of these methods are based on the practical approach to fuzzy inference as described in section 4.2.1.1, although some of these methods can also be represented in the theoretical approach and are, in fact, based on the analytical solution of fuzzy inference which is based on the composition of fuzzy relations.

Before discussing various inference methods, it should be noted that some people confuse the different aspects of fuzzy inference. What some people refer to as “composition” is in fact aggregation. Composition consist of two phases: a combination and a projection phase. See section 2.4.2 on the composition of fuzzy relations, and section 3.3.1.1 on the compositional rule of inference. Aggregation is the combining of fuzzy relations, representing fuzzy rules, into a single fuzzy relation. In the discussion on common fuzzy inference methods, which operation is represented by an operator or function is denoted.

These inference methods are, however, not always what they seem to be. The inference is in most cases the same as that in the compositional rule of inference, namely by means of *sup-min* composition. The implication function that is used to represent the fuzzy rules, is what is really addressed by the second part of an “inference method”. The first part represents the aggregation operator in most cases. The *sup* (*inf*) and *max* (*min*) operators are interchanged frequently in the nomenclature of inference methods. For example, the *max-min* method is sometimes refered to as *sup-min* method.

4.2.3.1 Max-min method

The fuzzy controller introduced by Assilian (1974) and Mamdani (1974) used what is known as the *max-min* method. Choosing a *min* operator for the conjunction in the premise of the rule as well as for the implication function and a *max* operator for the aggregation, the application of the compositional rule of inference results in:

$$\mu_{B'}(y) = \overbrace{\max_k}^{\text{aggregation}} \underbrace{\min}_{\text{implication}}(\beta_k, \mu_{B_k}(y)) \tag{4.15a}$$

with:

$$\beta_k = \overbrace{\min_i}^{\text{conjunction in premise}} \alpha_{i,k} \tag{4.15b}$$

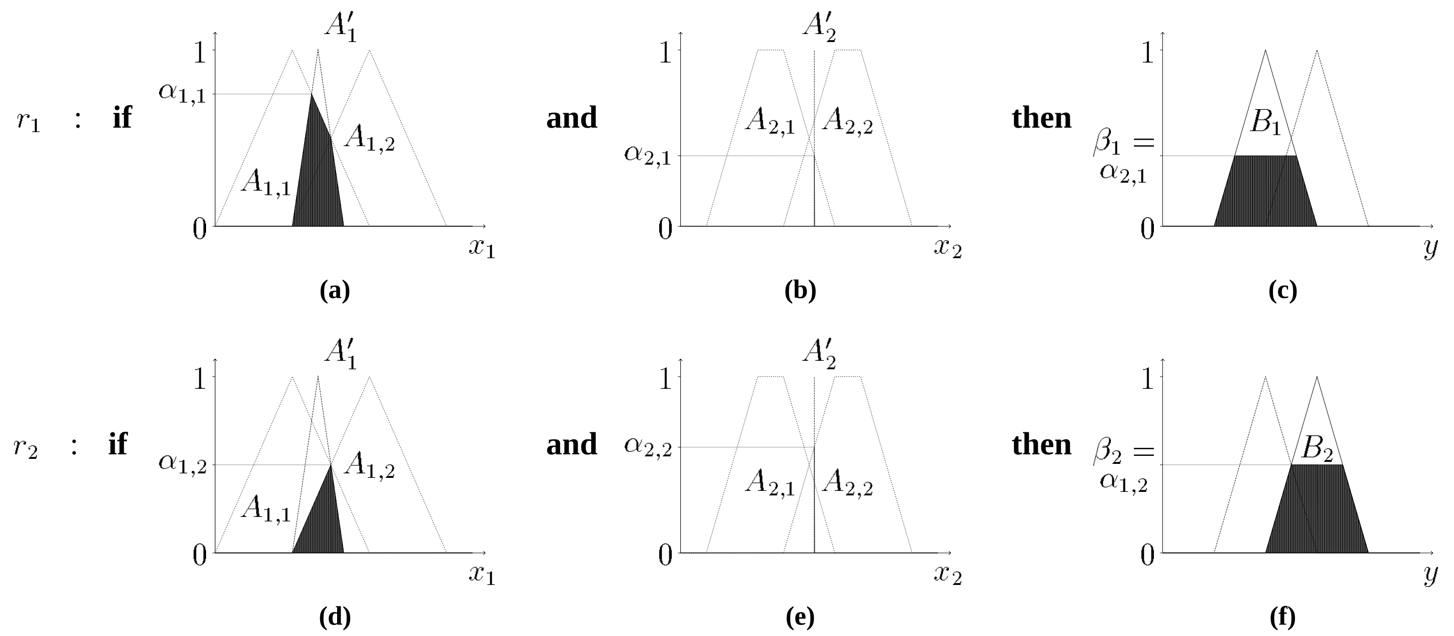
$$\alpha_{i,k} = \underbrace{\overbrace{\sup_x}^{\text{projection}} \underbrace{\min}_{\text{combination}}}_{\text{composition}}(\mu_{A_i'}(x_i), \mu_{A_{i,k}}(x_i)) \tag{4.15c}$$

This is exactly the inference described by (3.35) in section 3.3.1.4. Although, in principle, another operator could be used for the conjunction in the premise of the rule, in literature one normally finds the *min* operator in combination with the *max-min* method. As shown by (3.34), the conjunction, implication and composition are required to be based on the same T-norm in order to obtain a simple analytical solution for the fuzzy inference. In figure 4.11 a schematic representation of fuzzy inference in practice using the *max-min* method is shown. Of the two controller inputs, one is fuzzy (A_1') and one is numerical (A_2'). The fuzzy output is the aggregation (*max*) of the two "clipped" fuzzy sets.

4.2.3.2 Max-prod method

The *max-product* inference method is another commonly applied inference method in fuzzy control. The *max-product* method is also known as *max-dot* method. This inference method is characterized by scaling (product) the consequent B_k of a fuzzy rule r_k with the degree of fulfillment β_k of that rule and aggregating these result B_k' to obtain the fuzzy controller output by means of a *max* operator:

$$\mu_{B'}(y) = \overbrace{\max_k}^{\text{aggregation}} \beta_k \underset{\text{implication}}{\underset{|}{*}} \mu_{B_k}(y) \tag{4.16}$$



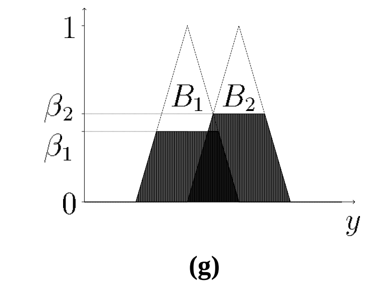


Figure 4.11: *Practical fuzzy reasoning according to max-min method; using min operator for both and-connective and implication. Sup-min composition is used for the compositional rule of inference. The matching of the data,* A_1' *(fuzzy) and* A_2' *(numerical), and the premise,* **(a)**, **(b)** *and* **(d)**, **(e)**, *determine the fuzzy result of each rule,* **(c)** *and* **(f)**, *respectively. The subresults are aggregated to obtain a fuzzy controller output* **(g)**. *The two rules are:* **if** x_1 is $A_{1,k}$ **and** x_2 is $A_{2,k}$ **then** y is B_k, for $k = 1, 2$.

where '$*$' represents multiplication. Note that the aggregation is the same as in the *max-min* method. The implication represented by the *product* operator is known as Larsen's implication (Larsen, 1980).

There are two distinguishable variations of the *max-product* method with respect to the determination of the support value β_k. Either the *min* or the *product* operator is used for the combination of the matching values $\alpha_{i,k}$ and thus representing the conjunction in the rule premises:

$$\beta_k = \begin{cases} \min\limits_i \alpha_{i,k} \\ \prod\limits_i \alpha_{i,k} \end{cases} \tag{4.17}$$

This operation represents the conjunction in the premise of the fuzzy rules. From a theoretical point of view, based on the compositional rule of inference as described in section 3.3.1.1, we can identify which operations are used for the conjunction in the premise, the implication and the composition. In table 4.1, a number of combinations of different operators in the *max-product* inference method is shown and their theoretical counterpart.

Table 4.1: *Different combinations of operators in the* max-product *inference method and their counterparts in the CRI-based point of view. The CRI-based counterparts are identified by three operators: conjunction in premise (*$\mathrm{T_C}$*), implication (*$\mathrm{T_I}$*) and* max-T *the* sup-m *composition (*$\circ_\mathrm{m}$*). The operator* $\circ$ *denotes "standard"* max-min *composition. A question mark represents unknown.*

$\mathrm{T_C},\mathrm{T_I},\circ_\mathrm{m}$	$\beta_k = \min\limits_i \alpha_{i,k}$	$\beta_k = \prod\limits_i \alpha_{i,k}$
$\alpha_{i,k} = \mathrm{hgt}(A'_i \cap A_{i,k})$	$\min, *, \circ_?$	$*, *, \circ_?$
$\alpha_{i,k} = \mathrm{hgt}(A'_i * A_{i,k})$	$\min, *, \circ_*$	$*, *, \circ_*$

From table 4.1, it can be seen that some combinations of operators, marked by a question mark, by do not have a straightforward counterpart in the theoretical approach. In table 4.1, it is assumed that the implication is a product, and the conjunction in the premise is represented by the same operator used to combine the matching values $\alpha_{i,k}$ to obtain the support value β_k. If the controller inputs are numerical, there is no problem since in that case:

$$\mathrm{hgt}(a'_i \cap A_{i,k}) = \mathrm{hgt}(a'_i * A_{i,k}) \tag{4.18}$$

where a'_i is the singleton (fuzzy set) representing the numerical input x'_i. However, in the case of fuzzy inputs, the operator m in the *sup-m* composition has to be determined from:

$$\sup_x \mathrm{m}(\mu_{A'}(x), \mu_{A_k}(x)\, \mu_{B_k}(y)) = \left\{ \sup_x \min(\mu_{A'}(x), \mu_{A_k}(x)) \right\} \mu_{B_k}(y)$$

There is no solution for the operator m if only the fuzzy relations A'_k and $R_k = A_k * B_k$ are considered. When A_k and B_k are separated a solution is possible, but then m is no longer a binary operator and thus not a (pseudo-)conjunction.

However, as stated previously, it is justifiable to only consider numerical inputs, since this is normally the case in control. Hence, the problems described are solved because of (4.18). This simplifies the determination of the support values β_k, since the matching values are given by $\alpha_{i,k} = \mu_{A_{i,k}}(x'_i)$, where x'_i are the numerical inputs. In figure 4.12, an example is given of the fuzzy output using the *max-product* inference method.

4.2.3.3 Sum-prod method

The *sum-product* or *sum-dot* method uses the *product* operator in the same way as the *max-product* method as discussed in the previous section. In literature, one can find examples of using a product operator, as well as examples of using a minimum operator to represent the conjunction in the premises. This is also similar to the way in which the degrees of fulfillment are determined in case of the *max-product* inference method. Therefore we focus on the type of aggregation that is used in the *sum-product* inference method. The aggregation in the *sum-product* method is mainly a *sum* operator, but there exist several variations which are described and discussed in the following.

- The aggregation of the (sub)results B'_k of all individual fuzzy rules r_k is done by means of summation:

  $$\mu_{B'}(y) = \sum_k \beta_k \mu_{B_k}(y) \tag{4.19}$$

  This could result in values $\beta_j > 1$ for a certain fuzzy set B_j, defined on the output universe, resulting in a supernormal fuzzy set, which does, in fact, not conform to

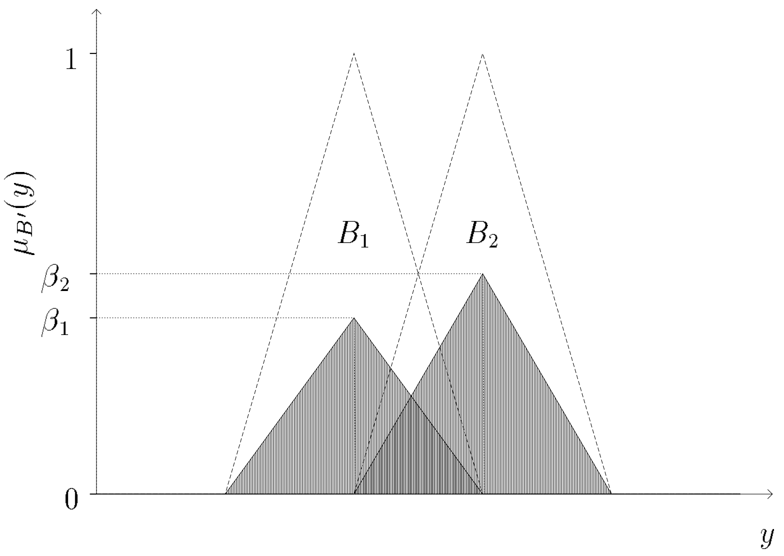


Figure 4.12: *Schematic representation of the result of practical fuzzy reasoning with* max-product *inference method: the consequents B_1 and B_2 are scaled by β_1 and β_2, respectively, and aggregation is performed by means of the* max *operator. The values β_1 and β_2 are determined by (4.17). Compare also with figure 4.11g.*

fuzzy set theory. Because of the defuzzification, necessary in fuzzy control, this is, however, a minor problem from the practical point of view. The use of a bounded sum is also possible, resulting in:

$$\mu_{B'}(y) = \min(\sum_k \beta_k \mu_{B_k}(y), 1) \tag{4.20}$$

which eliminates the possibility of supernormal fuzzy sets and thus conforms to fuzzy set theory.

- Only the fuzzy results of rules with equal consequent are aggregated by summation, after which everything is aggregated by a *max* operator:

$$\mu_{B'}(y) = \max_j \left\{ \sum_{k_j} \beta_{k_j} \right\} \mu_{B_j}(y) \tag{4.21}$$

where k_j is used to identify the rules which have a consequent $B_k = B_j$. Here also a bounded summation can be used:

$$\mu_{B'}(y) = \max_j \left\{ \min(\sum_{k_j} \beta_{k_j}, 1) \right\} \mu_{B_j}(y) \tag{4.22}$$

which will avoid $\beta_j > 1$ and thus preserve correctness with respect to fuzzy set theory.

In the above list of possible use of aggregation by means of a summation, it was stated that in some cases the support values for a specific fuzzy set on the output universe can be greater than 1. This is, in fact, not according to fuzzy set theory since its implies supernormal fuzzy sets. However, when the fuzzy sets for the input universes are considered fuzzy partitions, only numerical inputs are considered and the *and* connective in the premise is represented by a product, then the following holds:

$$\begin{aligned}\sum_k \beta_k &= \sum_k \left\{ \prod_i \alpha_{i,k} \right\} \\ &= \sum_k \left\{ \prod_i \mu_{A_{i,k}}(x'_i) \right\} \\ &\leq 1\end{aligned} \tag{4.23}$$

where the equality holds when the rule base is complete (see section 3.2.5.3). This means that the in case of fuzzy partitions for the input universes and numerical inputs, the problem of supernormal fuzzy sets does not exist, because, in that case, aggregation by means of a summation equals aggregation by means of a bounded summation. The same property of a fuzzy system meeting the above-mentioned criteria is used in section 4.5.2, where it is used to "emulate" linear controllers by fuzzy controllers. A proof for this can be found in appendix B.

Usually this method is used in combination with the *fuzzy-mean* defuzzification which eliminates the aggregation phase if all rules are considered individually, neglecting whether the consequents are "equal" or not. The fuzzy-mean defuzzification is, in fact, a weighted sum and can be considered a special case of the discrete center-of-gravity method. This defuzzification method is described in the next section.

4.2.4 Defuzzification in practice

In section 4.1.2, the center-of-gravity, center-of-area and the mean-of-maxima defuzzification methods were described. Some of these are discussed again in this section, but from a more practical point of view. In this section, we consider only 1-dimensional fuzzy output sets. First the center-of-gravity related defuzzification methods are addressed. Section 4.2.4.2 addresses defuzzification methods which focus on the height of the fuzzy output set. Lastly, related defuzzification methods which are a merge of basic defuzzification methods are described.

4.2.4.1 Averaging defuzzification methods

In a practical set-up, simplifications of the center-of-gravity defuzzification method are often used. The discrete version of the method was given in section 4.1.2.1 and this method is mostly used in fuzzy control. In this section related and similar averaging defuzzification methods, like the *fuzzy-mean* defuzzification method, are described. However, first we focus on the nonlinearity which is introduced by the centre-of-gravity method in combination with commonly used operators for aggregation.

When applying the center-of-gravity method, the result of defuzzification is a nonlinear function of the controller inputs in most cases. This is normally the case for fuzzy controllers, but the introduced nonlinearity is not trivial: it is inherent in the combination of operators and defuzzification method. The nonlinearity is mainly due to the *max* aggregation in combination with the center-of-gravity method. To show this, two examples are given in figure 4.13, where the numerical input x' goes from a_1 to a_2. Because of the numerical input and the fuzzy sets shown in figure 4.13a, $\beta_1 = 1 - \beta_2$.

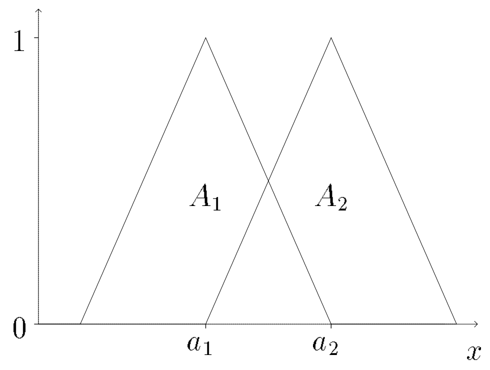
When it is required that the controller output is a linear function of the controller input for the given examples in figure 4.13, the choice for the T-norm and S-norm are *product* and *summation* operator, respectively, since in that case:

$$y' = \frac{\int_Y \{\beta_1 \mu_{B_1}(y) + \beta_2 \mu_{B_2}(y)\}\, y\, dy}{\int_Y \{\beta_1 \mu_{B_1}(y) + \beta_2 \mu_{B_2}(y)\}\, dy} \tag{4.24a}$$

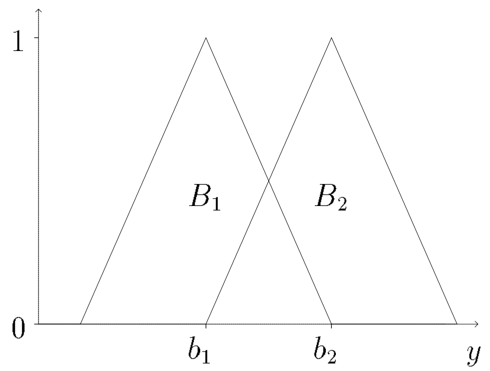
$$= \frac{\beta_1 \int_Y \mu_{B_1}(y)\, y\, dy + \beta_2 \int_Y \mu_{B_2}(y)\, y\, dy}{\beta_1 \int_Y \mu_{B_1}(y)\, dy + \beta_2 \int_Y \mu_{B_2}(y)\, dy}, \tag{4.24b}$$

$$\text{with } \beta_1 = 1 - \beta_2 \text{ and } \int_Y \mu_{B_1}(y)\, dy = \int_Y \mu_{B_2}(y)\, dy$$

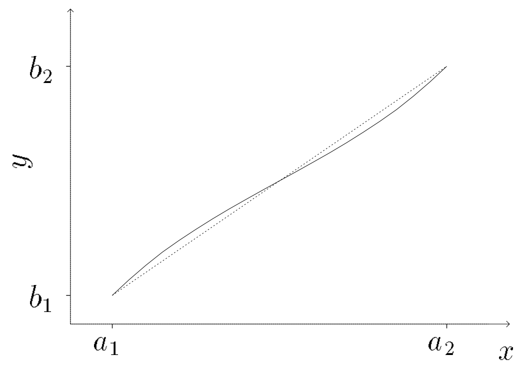
$$= \frac{\beta_1 \int_Y \mu_{B_1}(y)\, y\, dy}{\int_Y \mu_{B_1}(y)\, dy} + \frac{\beta_2 \int_Y \mu_{B_2}(y)\, y\, dy}{\int_Y \mu_{B_2}(y)\, dy} \tag{4.24c}$$



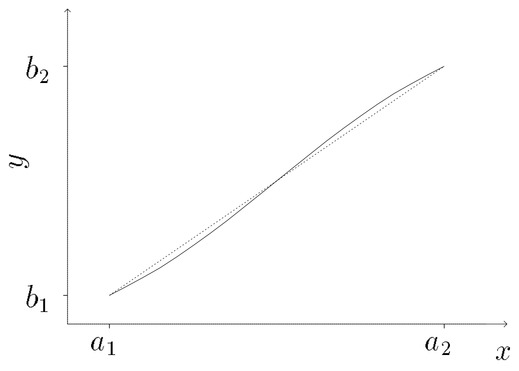
(a)
input universe



(b)
output universe



(c)
result for min *implication*



(d)
result for product *implication*

Figure 4.13: *Nonlinearities in controller output as function of controller input when center-of-gravity defuzzification method is used in combination with* max *aggregation. The dotted lines represent linear interpolation.*

$$= \beta_1 b_1 + \beta_2 b_2 \tag{4.24d}$$

Hence, center-of-gravity defuzzification in combination with aggregation operators other that the *summation* operator can introduce nontrivial nonlinearities in the controller function. In the following, defuzzification methods are described which are related to the COG method, but avoid the previously discussed problem.

The *fuzzy-mean* (FM) defuzzification method is close to the discrete form of the center-of-gravity defuzzification method as defined by (4.4b): it performs a weighted sum. The difference between the discrete COG method and the FM method is that the COG discretizes the fuzzy output of the controller, and the FM method uses numerical representations of the fuzzy sets on the output universe. Hence, the "discretization" in the case of the FM does not have to be equidistant, which is normally the case with the discrete COG. The *fuzzy-mean* defuzzification method is defined by:

$$\mathrm{fm}(B') = \frac{\sum\limits_{j=1}^{N_B} \beta_j\, b_j}{\sum\limits_{j=1}^{N_B} \beta_j} \tag{4.25}$$

where N_B is the number of fuzzy sets defined on the universe of discourse of the controller output, β_j denote the degrees of fulfillment for membership function B_j as the result of the inference and b_j represent the numerical representations of B_j. Here it is assumed that the the support values β_j for each fuzzy output set B_j are determined. It is also possible that each rule is treated individually, which yields:

$$\mathrm{fm}(B') = \frac{\sum\limits_{k=1}^{N_r} \beta_k\, b_k}{\sum\limits_{k=1}^{N_r} \beta_k} \tag{4.26}$$

where N_r is the number of fuzzy rules. The difference between (4.26) and (4.25) is the possibly multiple occurrences of the same consequent B_j, represented by the numerical value b_j. Hence, the effect of multiple rule with the same consequent is accumulated. Using the FM method this way in fact *combines the aggregation and defuzzification phase into one operation*, and it is thus more than a defuzzification method. From a theoretical viewpoint, this can lead to supernormal fuzzy sets, which is not in accordance with fuzzy set theory. See also section 4.2.3.3 on this topic. Hence, the results from the inference are not considered a set, but are considered a bag (Yager, 1994).

Since the numerical representations b_j of the fuzzy sets B_j can be precalculated, the fuzzy-mean defuzzification method is very suitable when the calculation load of the fuzzy controller should be small. The numerical representation b of a fuzzy set B is normally chosen to be the "center" of that set:

$$b_j = \mathrm{dfz}(B_j) \tag{4.27}$$

where *dfz* is a defuzzification method, in most cases the mean-of-maxima method.

A weighted version of the fuzzy-mean method, *weighted fuzzy-mean* (WFM), is defined by (Jager et al., 1992):

$$\mathrm{wfm}(B') = \frac{\sum_{j=1}^{N_B} w_j\, \beta_j\, b_j}{\sum_{j=1}^{N_B} w_j\, \beta_j} \tag{4.28}$$

in which w_j are weights assigned to the membership functions on the universe of discourse of the controller output to emphasize certain membership functions. When the weights are chosen as:

$$w_j = \mathrm{area}(B_j) = \int_Y \mu_{B_j}(y)\, dy \tag{4.29}$$

then the FM method more or less mimics the COG method. In which case, each fuzzy set B_j on the output universe is interpreted as a crisp set B_j' with support equal to w_j, and thus:

$$\mathrm{area}(B_j') = \mathrm{area}(B_j) = w_j \tag{4.30}$$

Obviously, the weights w_j can be normalized, since they are used in a weighted sum. The main characteristic of this defuzzification method is the fact that "relative importance" of fuzzy sets defined for the output can be embedded. In the COG method this is implicit, since the whole output universe is taken into account and not just numerical representations of each fuzzy set. When the WFM method is compared with the COG method, the WFM method avoids the nonlinearity which is introduced when using the COG method in combination with the *max* aggregation in the same way as shown by (4.24): the aggregation is done by summation.

4.2.4.2 Height-related methods

In literature, the height-related defuzzification method generally used is the mean-of-maxima method as described in section 4.1.2.3. Other height-related defuzzification methods which are sometimes used are the *first-height* (FHGT), or *first-maximum*, and *last-height* (LHGT), or *last-maximum*, defuzzification method. These defuzzification methods take the most left- or right-hand side value of the part of the domain where the membership function equals the height of the fuzzy set:

$$\mathrm{fhgt}(B') = y', \text{with } \mu_{B'}(y') = \mathrm{hgt}(B') \text{ and } \forall y < y', \mu_{B'}(y) < \mu_{B'}(y') \tag{4.31}$$

$$\mathrm{lhgt}(B') = y', \text{with } \mu_{B'}(y') = \mathrm{hgt}(B') \text{ and } \forall y > y', \mu_{B'}(y) < \mu_{B'}(y') \tag{4.32}$$

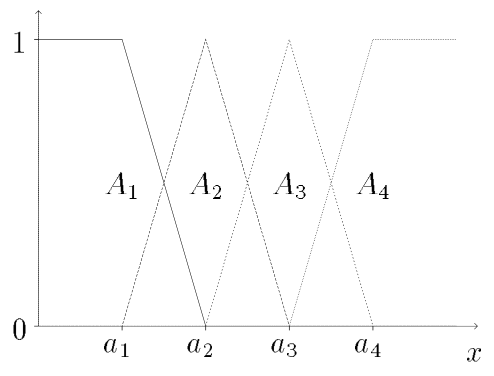
The usability of these methods is very application dependent and they are not further considered here. In the following, we take a closer look at the mean-of-maxima defuzzification method.

Although the *mean-of-maxima* method is an obvious method for defuzzification (Dubois and Prade, 1980), applying it in fuzzy control if T-implications are used, reduces the "fuzziness" of the controller completely in many cases. If the mean-of-maxima method is used for defuzzification of the fuzzy output, the membership functions describing the inputs can even be chosen to be crisp (classical) sets, without having any effect on the numerical output (Jager et al., 1992). This can best be shown by a simple example. In figure 4.14, an example is given which shows that the fuzzy controller acts as a multi-level relay because of the mean-of maxima defuzzification (Kickert and Mamdani, 1978). Note that there is an intermedial value for the output at the crossover points of the input sets. It is clear that a similar multi-level relay function can be obtained by using classical sets for the input classification and, hence, eliminating the need for fuzzy sets.

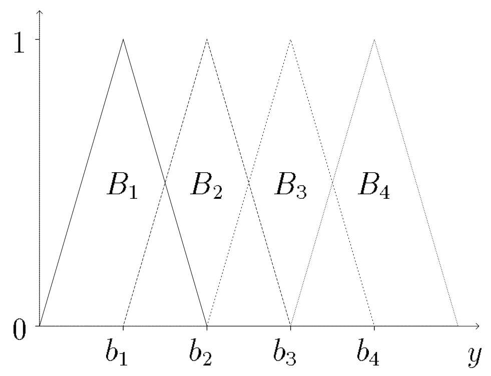
Another disadvantage of the mean-of-maxima method is the fact that non-symmetrical fuzzy sets defined for the output can result in undesired behavior. In figure 4.15, two situations are depicted which show that a shift of the fuzzy output can possibly result in a shift of the numerical output in the opposite direction.

4.2.4.3 Extended defuzzification methods

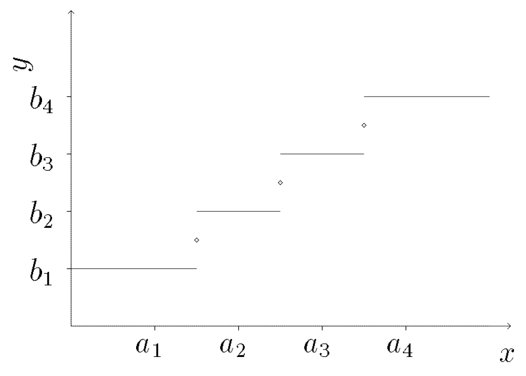
Several extensions and adaptations of defuzzification methods can be found in literature. In this section, we briefly describe these extended defuzzification methods.



(a)
input universe



(b)
output universe



(c)
controller function

Figure 4.14: *Example showing controller output as function of controller input* **(c)** *in the case of mean-of-maxima defuzzification. The fuzzy sets for input and output are shown in figures* **(a)** *and* **(b)**, *respectively. The controller is based on four rules:* ***if*** *x is A_k* ***then*** *y is B_k, for $k = 1, \ldots, 4$. The resulting controller functions shows that the controller output "jumps" from one level to another. Note the intermedial values between two levels.*

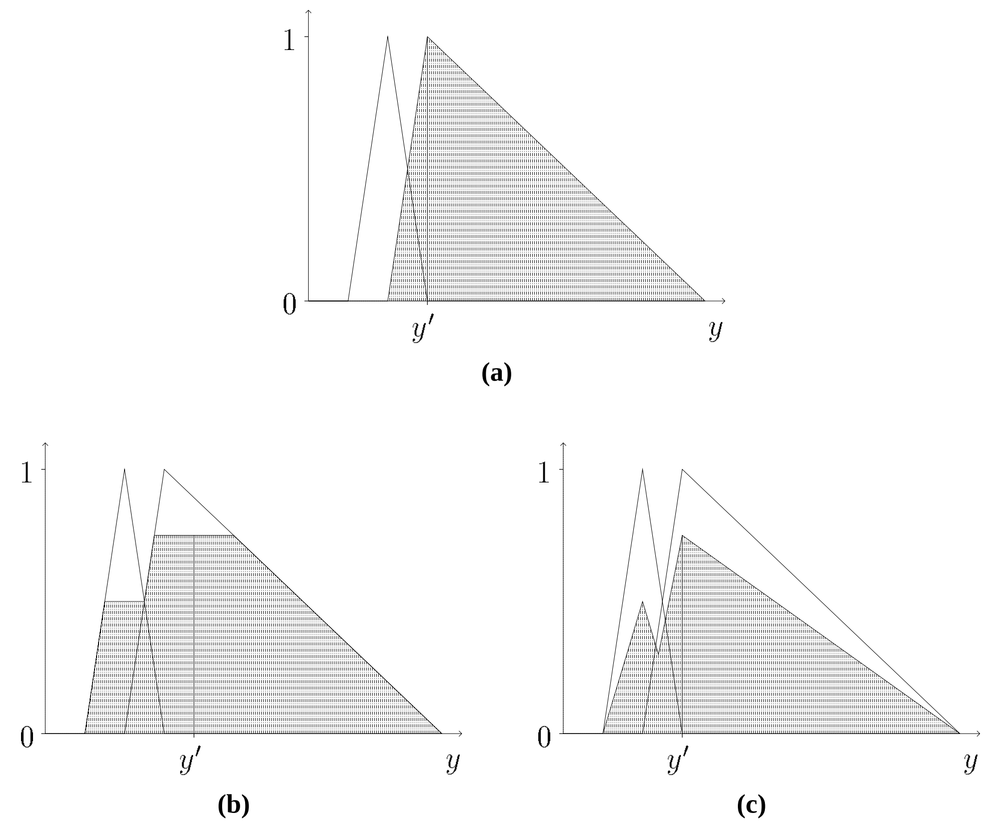


Figure 4.15: *Undesired result of mean-of-maxima method in the case of non-symmetrical fuzzy sets: possible shift of numerical output opposite to shift in fuzzy output. The upper figure* **(a)** *shows the fuzzy output for a certain input. The left figure* **(b)** *shows another situation as result of the max-min inference method. The "symbolic" meaning is "shifted" to the left, but the defuzzified value shifts to the right! The right figure* **(c)** *shows the result obtained by the max-product inference method. Here the numerical output is the same as in figure* **(a)***.*

Filev and Yager (1991) introduced the BADD defuzzification method, where BADD stands for *BAsic Defuzzification Distributions*. It is a modification of the center-of-gravity defuzzification method and defined by[*]:

$$\mathrm{badd}(B', \delta) = \frac{\displaystyle\int_Y \mu_{B'}^{\delta}(y)\, y\, dy}{\displaystyle\int_Y \mu_{B'}^{\delta}(y)\, dy}, \text{ with } \delta \geq 0 \tag{4.33}$$

where δ is the parameter which is used to "tune" the defuzzification method, with the following special cases:

$$\begin{aligned} \mathrm{badd}(B', \delta = 1) &= \mathrm{cog}(B') \\ \mathrm{badd}(B', \delta \to \infty) &= \mathrm{mom}(B') \end{aligned}$$

Another extended defuzzification method proposed by Yager and Filev (1993) is the SLIDE method, where SLIDE stands for *Semi-LInear DEfuzzification* and which is defined by:

$$\mathrm{slide}(B', \alpha, \beta) = \frac{(1-\beta)\displaystyle\int_{Y_L} \mu_{B'}(y)\, y\, dy \; + \; \int_{Y_H} \mu_{B'}(y)\, y\, dy}{(1-\beta)\displaystyle\int_{Y_L} \mu_{B'}(y)\, dy \; + \; \int_{Y_H} \mu_{B'}(y)\, dy} \tag{4.34a}$$

with $\alpha \in [0, \mathit{hgt}(B')]$ and $\beta \in [0, 1]$ and:

$$Y_L = \{y \in Y \mid \mu_{B'}(y) < \alpha\} \tag{4.34b}$$

$$Y_H = \{y \in Y \mid \mu_{B'}(y) \geq \alpha\} \tag{4.34c}$$

and with the following special cases:

$$\begin{aligned} \mathrm{slide}(B', \alpha = 0, \beta) &= \mathrm{cog}(B') \\ \mathrm{slide}(B', \alpha > 0, \beta = 0) &= \mathrm{cog}(B') \\ \mathrm{slide}(B', \alpha = \mathrm{hgt}(B'), \beta = 1) &= \mathrm{mom}(B') \end{aligned}$$

[*]Note that the modification consists of taking the membership function to the power. This same principle could be noticed in the implication Yager (1980a) proposed, defined by (3.15), as well as the negation proposed by Yager given in table A.3.

This defuzzification method is characterized by two parameters, a *confidence level* α and a *rejection parameter* β. For $\beta = 1$ all membership values $\mu_{B'}(y) < \alpha$ are rejected. In case $\alpha = \mathit{hgt}(B')$, the parameter β can be used to continuously go from the COA ($\beta = 0$) to the MOM ($\beta = 1$).

Runkler and Glesner (1993) proposed an extension of the center-of-area defuzzification method, denoted by XCOA:

$$\int_{\inf\limits_Y y}^{\mathrm{xcoa}(B',\gamma)} \mu_{B'}^{\gamma}(y)\, dy = \int_{\mathrm{xcoa}(B',\gamma)}^{\sup\limits_Y y} \mu_{B'}^{\gamma}(y)\, dy \tag{4.35}$$

which includes the COA and the MOM defuzzification methods as special cases:

$$\begin{aligned} \mathrm{xcoa}(B', \gamma = 1) &= \mathrm{coa}(B') \\ \mathrm{xcoa}(B', \gamma \to \infty) &= \mathrm{mom}(B') \end{aligned}$$

It is claimed by Runkler and Glesner (1993) that the computational effort of the XCOA method is much smaller than of the BADD method. However, in fuzzy control the COG defuzzification is in many cases reduced to the FM method which has a much smaller calculational effort than the COG method; see section 4.2.4.1.

The main characteristic of these extended defuzzification methods is the possibility to “filter” out parts of the fuzzy output. This filtering should be understood as giving preference to the higher membership values. It gives the operator or designer of a fuzzy controller the opportunity to tune its defuzzification controller to met some constraints. This filtering can also be noticed in the case of index defuzzification methods as described in section 4.1.2.2. Yager and Filev (1993) proposed an adaptation method for the SLIDE method to tune the defuzzification part automatically.

However, one could question whether it is necessary to have parameterized defuzzification methods. In section 4.5, a fuzzy controller (system) is considered as an input-output mapping, and it is shown that all combinations of logical operators and defuzzification methods, except some specific combinations, will result in nontrivial nonlinearities in the input-output mapping performed by the fuzzy system. It can be disputed whether the behavior of a fuzzy controller should be altered by means of adaptation of the defuzzification method used. In our opinion, *the behavior should be altered by modifying the rule base* since the fuzzy rules represent the knowledge to control the process in question.

4.3 Fuzzy control rules

In fuzzy control, we can distinguish two types of fuzzy rules: Mamdani rules and Sugeno rules. The Mamdani rules are the rules we have considered thus far in this thesis. The Sugeno rules are based on a different principle: the consequents of those rules are (linear) functions of the controller inputs. These two types of fuzzy rules are described in the following sections.

4.3.1 Mamdani fuzzy rules

This type of fuzzy rule was used in the first reported applications of fuzzy control (Mamdani, 1974; Assilian, 1974; Mamdani and Assilian, 1975) and has the following general form:

$$\begin{array}{lll} r_k & : & \textbf{if } x_1 \text{ is } A_1^k \textbf{ and} \ldots \textbf{and } x_{N_X} \text{ is } A_{N_x}^k \\ & & \textbf{then } y_1 \text{ is } B_1^k, \ldots, y_{N_Y} \text{ is } B_{N_Y}^k \end{array}$$

This is the same type of fuzzy rule considered thus far in this thesis. An example of such a fuzzy control rule is:

if error is big **and** error change is small **then** control signal is big

which can be compared to a discrete PD controller algorithm: the controller output is based on the error (difference between the process output and the desired process output) and its first difference. The use of other variables are possible. For example, using the process output change instead of the error change, or using an additional classification of the reference signal to be able to "capture" nonlinearities of the process. However, this is merely a design issue and is not further discussed here.

Normally the *min* operator is used for conjunction and implication, the *max* operator for aggregation and *max-min* composition (Mamdani and Assilian, 1975). This is known as the *max-min* inference method (section 4.2.3.1). Mamdani and co-workers used this type of fuzzy rule and were the first ones to report in literature on the application of fuzzy logic in control, and therefore this type of rule is referred to as the *Mamdani rule* in this thesis. With this nomenclature we do not refer to the operators used by Mamdani and co-workers, but to the fact that the fuzzy rules have fuzzy propositions as consequences, and the implication is represented by a conjunction (T-implication).

It can be stated that using a limited number of membership functions for the output restricts the control hypersurface if the fuzzy sets for the input are normal and form a fuzzy partition, and numerical inputs are assumed. Considering a fuzzy controller with single input and single output, the resulting controller function is limited to an interpolation between the characteristic points as shown in figure 4.16. This limitation occurs because the consequents of the fuzzy rules use only one of the linguistic labels defined for the output of the system. When the numerical input x' is at the center of a fuzzy set A_i, the membership value $\mu_{A_i}(x') = 1$ and only one rule is active.

The limitation of the control hypersurface can be reduced by defining more fuzzy sets for the input universe(s). This results in an increasing number of possible rules and increases the complexity of the system, making the rule base less understandable. Another possibility is to allow fuzzy rules to have multiple weighted consequents (Jager et al., 1991), for example:

if error is big **and** error change is medium
then control change is medium (80%), control change is big (20%)

This makes it possible to obtain any control hypersurface desired. For example, the solid curve shown in figure 4.16 can also be placed between points on a vertical line as is shown by the dotted curve. Allowing multiple weighted consequents is, in fact, a trick, and the same result can be obtained by using constant numerical consequents or consequents with fuzzy numbers, like *about 3.5*. As an advantage of the weighted fuzzy consequents, it can be stated that its representation is still on the linguistic level, e.g., it is still based on linguistic labels like “big”, “medium”, etc., and thus can be related to the knowledge of the designed or user (operator).

4.3.2 Sugeno fuzzy rules

Another fuzzy rule type is here referred to as Sugeno rules, because of the introduction of this type of rule by Takagi and Sugeno (1983) which was further exploited by Sugeno and co-workers. This type of rule is also referred to as Takagi-Sugeno rules, or TS-rules, for short. The general form is as follows:

$$\begin{array}{ll} r_k: & \textbf{if } x_1 \text{ is } A_{1,k} \textbf{ and} \ldots \textbf{and } x_{N_X} \text{ is } A_{N_X,k} \\ & \textbf{then } y_1 = f_{1,k}(x_1, \cdots, x_{N_X}), \ldots, y_{N_Y} = f_{N_Y,k}(x_1, \cdots, x_{N_X}) \end{array}$$

which shows that the consequents of these fuzzy rules are functions ($f_{j,k}$) of the controller inputs x_i. In the remainder of this section only one output is considered, without loss of generality. Sugeno and co-workers used linear functions in the consequents of the fuzzy rules:

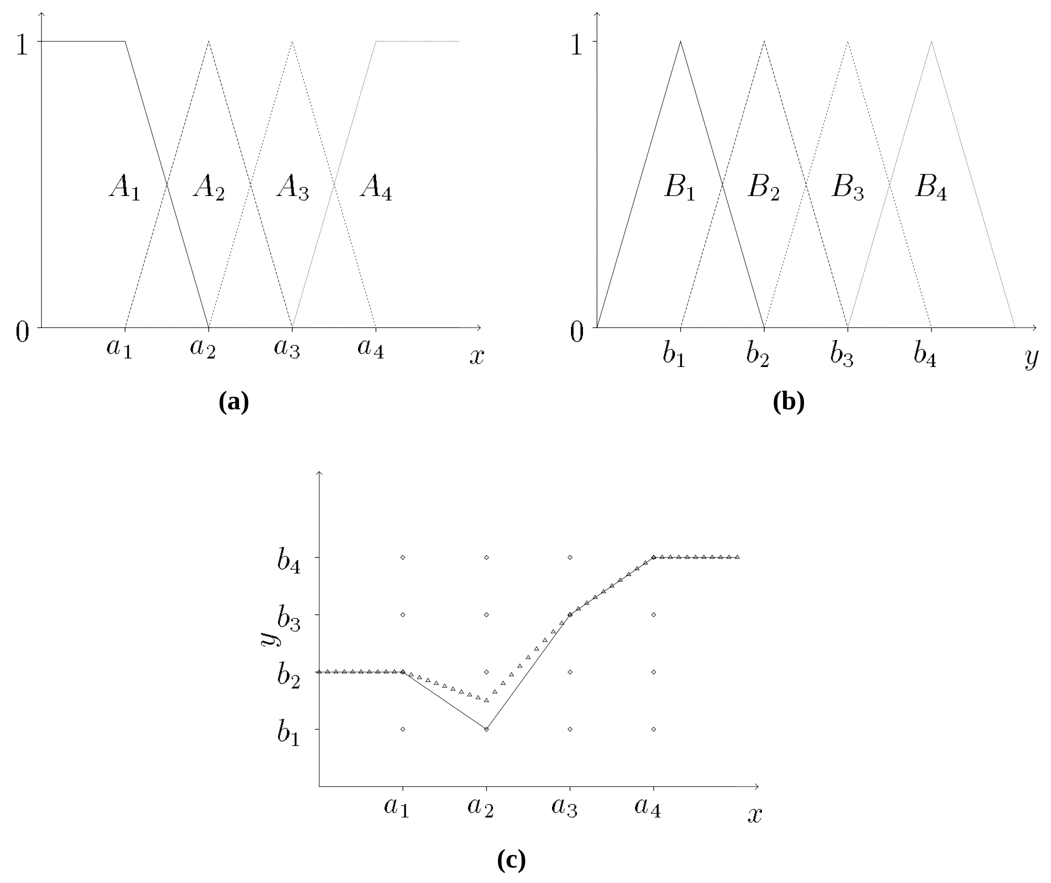


Figure 4.16: *Restriction of controller function* **(c)** *using Mamdani rules. The controller output as function of the controller input is restricted to a function which connects points from left to right in a monotonic way between two points. In figures* **(a)** *and* **(b)** *the fuzzy sets for the input and output are given. Compare also figure 4.14. The solid curve shown in figure* **(c)** *represents a possible result when using the* sum-product *inference method and the* fuzzy-mean *defuzzification method. The controller in this case is based on four rules:* **if** x is $A_1/A_2/A_3/A_4$ **then** y is $B_2/B_1/B_3/B_4$, *for* $k = 1, \ldots, 4$. *It is important to note that the controller function is restricted to interpolation between a fixed number of tuples* (a_i, b_j). *The dotted curve is produced by using multiple weighted consequents; the rule "***if** x is A_2 **then** y is B_1*" is replaced by "***if** x is A_2 **then** y is B_1 (50%), y is B_2 (50%)*".*

$$r_k : \textbf{if } x_1 \text{ is } X_1^k \textbf{ and} \cdots \textbf{and } x_{N_x} \text{ is } X_{N_x}^k \textbf{ then } y = b_{0,k} + \sum_{i=1}^{N_X} b_{i,k} x_i$$

with constant parameters $b_{i,k}$ and $b_{0,k}$. An example of such a fuzzy rule is:

$$\textbf{if } e \text{ is big } \textbf{and } \Delta e \text{ is small } \textbf{then } u = 4e + 2\Delta e$$

where u is a control signal and e and Δe are the error and its first difference. Successful use of this type of fuzzy rule in the control of a model car was reported by Sugeno and Murakami (1984), Sugeno and Nishida (1985) and Sugeno and Murakami (1985). The Sugeno rules can be even more simplified for $b_{i,k} = 0, i = 1, \ldots, N_X$, resulting in a constant numerical consequent $b_{0,k}$. A set of Sugeno rules can be seen as a set of local controllers. A simple weighted sum as defuzzification is used to obtain the final controller output and is an interpolation between the outputs of the local controllers. Compare the fuzzy-mean defuzzification method described in section 4.2.4.1. In the case of linear consequent functions this results in:

$$\begin{aligned}
y' &= \frac{\sum_{k=1}^{N_r} \beta_k \{b_{0,k} + \sum_{i=1}^{N_X} b_{i,k} x_i\}}{\sum_{k=1}^{N_r} \beta_k} \\
&= \frac{\sum_{k=1}^{N_r} \beta_k b_{0,k} + \sum_{i=1}^{N_X} \{\sum_{k=1}^{N_r} \beta_k b_{i,k} x_i\}}{\sum_{k=1}^{N_r} \beta_k} \\
&= b'_0 + \sum_{i=1}^{N_X} b'_i x_i
\end{aligned}$$

where β_k is the support value for rule r_k and:

$$b'_0 = \frac{\sum_{k=1}^{N_r} \beta_k b_{0,k}}{\sum_{k=1}^{N_r} \beta_k}$$

$$b'_i = \frac{\sum_{k=1}^{N_r} \beta_k b_{i,k}}{\sum_{k=1}^{N_r} \beta_k}, \text{for } i = 1, \ldots, N_X$$

The above shows that application of a fuzzy controller based on Sugeno rules can be seen as a fuzzy "supervisor" which changes the parameters of a linear controller. An example of this is a fuzzy controller consisting of Sugeno rules with a PID algorithm as consequents. A system like this is equal to a system consisting of a fuzzy supervisor based on Sugeno rules and a conventional PID controller, where the supervisor changes the parameters of the conventional PID controller (see also section 5.3.1 on fuzzy supervisory control).

From another point of view, a rule base with Sugeno rules can be seen as a set of local controllers each with its own set of controller parameters. This mechanism is in fact the same as *gain scheduling*, although it is not known as "fuzzy" technique: different controller parameters for different input combinations/situations are defined. The use of fuzzy sets and inference and defuzzification result in "fuzzy gain scheduling", where the transition from one set of controller parameters to another is smooth. However, the problem of discontinuous transitions from one set of controller parameters to another was recognized a long time ago, and the solution is known as "bumpless" transfers (Åström and Wittenmark, 1984).

When using Sugeno rules, the resulting controller output is, in fact, a weighted sum of functions of the controller inputs. Because of this weighted sum, an interpolation between the functions (consequents) is performed. Interpolation between (linear) functions, however, has the disadvantage that the result is possibly not what was intuitively expected, especially when the Sugeno rules are designed in a graphical way. A typical example is the one given by the following rules:

r_1 : **if** x is A_1 **then** $l_1 : \ y = c_1 x + d_1$
r_2 : **if** x is A_2 **then** $l_2 : \ y = c_2 x + d_2$

where the parameters c_1, c_2, d_1 and d_2 are chosen in such a way that the lines l_1 and l_2 intersect between the centers of the membership functions representing the linguistic labels A_1 and A_2 for controller input x. Assuming that the fuzzy sets defined for x form a fuzzy partition (see section 2.1.2), the controller output has values between the functions l_1 and l_2. An example of the resulting controller output as a function of the input is shown in figure 4.17. Clearly, this may not be the control function that is desired: the linear consequents do not describe the local approximations of the resulting controller function with respect to the derivative of the desired controller function! These problems occur because of the bias caused by the constant parameters $b_{0,k} \neq 0$. A more detailed analysis is described by Babuška, Jager and Verbruggen (1994).

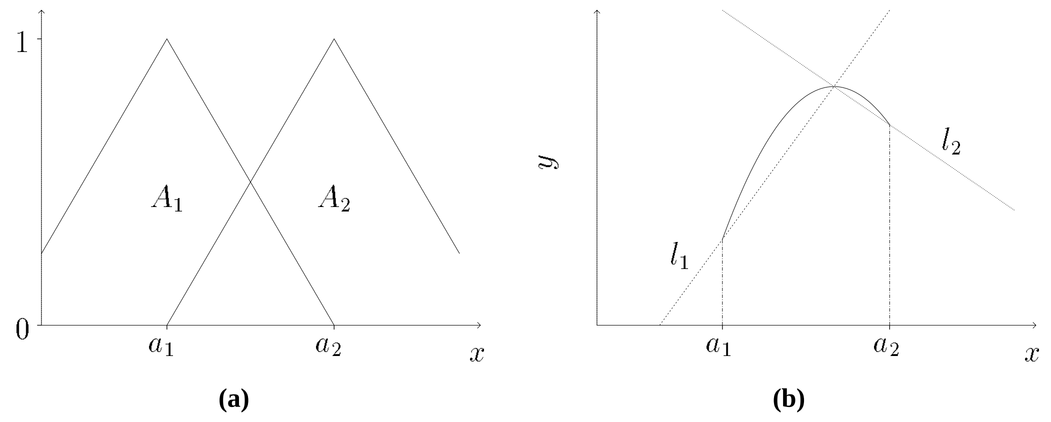


Figure 4.17: *Possible controller output* **(b)** *as function of inputs using Sugeno-rules with linear consequent functions; a_1 and a_2 represent the centers of fuzzy sets A_1 and A_2* **(a)**.

4.3.3 Differences and similarities

Although one can distinguish the Mamdani rules from the Sugeno rules, in practical set-ups the Mamdani rules are quite often similar to Sugeno rules with constant consequents. For example, see Batur and Kasparian (1991), Harris and Moore (1989) and Matsuoka (1991) for cases where there is, in fact, no difference when Mamdani or Sugeno rules are used. This is because of the defuzzification method used to obtain a numerical output of the controller, namely the fuzzy-mean defuzzification method, and the fact that rules are not aggregated but all contribute individually to the output. This is the same as applying a summation as aggregation operator, but note that it is, in principle, not a bounded-sum operator and thus is not a set-theoretic operator like a T-norm or T-conorm. See section 4.2.4 for details of defuzzification methods, and section 2.3.1 for set-theoretic operators like T-norms and T-conorms.

When the fuzzy-mean defuzzification method is applied the Mamdani rules simplify to:

$$r_k : \textbf{if } x_1 \text{ is } A_1^k \textbf{ and} \ldots \textbf{and } x_{N_X} \text{ is } A_{N_X}^k \textbf{ then } y_1 \text{ is } b_1^k, \ldots, y_{N_Y} \text{ is } b_{N_Y}^k$$

where b_j^k are numerical values. This is precisely the simplest case of a Sugeno rule as described in section 4.3.2, where the only difference is the '=' instead of a symbolic 'is'.

4.4 Fuzzy linear control

By fuzzy linear control, we mean the control of (nonlinear) processes based on linear model structures with parameters that are determined by a fuzzy system. Such a fuzzy system can be regarded as a supervisor, but the time scale on which such a supervisor operates is the same as the time scale on which the process is controlled (see also section 5.3). In the following the parameters of the controller, determined by such a fuzzy system, are refered to as "fuzzy parameters". The models and the controllers based on these fuzzy linear models have a linear structure and one or more fuzzy parameters. The fuzzy parameters of the linear models depend on signals related to the process, such as process input or output signals, or external signals, and thus the complete model is (possibly) a nonlinear process model. The use of fuzzy parameters in a linear model or controller can be seen as an extension of the purely linear case.

Another approach to make a controller based on linear controller structures is to design a set of fuzzy control rules, each based on a local model, where the fuzzy rules are according to Takagi and Sugeno (1983). First, fuzzy linear models are described (section 4.4.1). Fuzzy linear control is described in section 4.4.2. In the remaineder of this section, a pole-placement controller structure is used to explain the concept. Some simple experiments given in section 4.4.3 show the possible use of these controllers based on fuzzy linear models.

4.4.1 Fuzzy linear models

In this section, we describe the idea of a fuzzy linear model. The model that is described is used to design a pole-placement controller (Jager et al., 1993a). The following nonlinear process is considered:

$$\tau(y)\dot{y}(t) + y(t) = K(y)u(t - T_d) \tag{4.36a}$$

with:

$$\tau(y) \ = \alpha_\tau + \beta_\tau |y| \tag{4.36b}$$

$$K(y) = \alpha_K e^{-\beta_K |y|} \tag{4.36c}$$

The fuzzy linear model for this process is described by the following ARX model with "variable" parameters:

$$y[k+1] = \tilde{a}_m[k]y[k] + \tilde{b}_m[k]u[k+d] \tag{4.37}$$

where parameters $\tilde{a}_m$ and $\tilde{b}_m$ are described by fuzzy rules or, for example, a look-up table based on a fuzzy rule base in combination with an interpolation algorithm. Any (fuzzy) modeling technique method could be used to obtain the fuzzy relations between the model parameters and the signals they depend on. The model can also be described by a number of Sugeno rules:

r_j : **if** absolute value of $y[k]$ is about y_j **then** $y[k+1] = a_j y[k] + b_j u[k+d]$

where the linear consequent is modeled using conventional modeling techniques. Each rule represents a submodel and the fuzzy premises of the rules (about y_j) in combination with the compositional rule of inference provides interpolation between the submodels. To obtain an interpolation without nonlinearities which are not trivial (see section 4.6), the *sum-prod* “inference” method is used.

To obtain a fuzzy linear model, standard modeling techniques can be used to determine “local” models. When, for example, it is known that the DC gain of the process is related to the process output, this relation can be determined by probing the process (van Graafeiland, 1992). In the following we do not go into further detail on (fuzzy) modeling techniques, but focus on controllers based on fuzzy models (which are assumed to be present).

4.4.2 Fuzzy linear controllers

In figure 4.18, the scheme for applying the fuzzy pole-placement controller is shown. The polynomials R, S and T are the controller polynomials. The polynomials A and B represent the model of the process transfer function.

The controller contains a feedback and a feedforward part:

$$R(q)u[k] = T(q)v[k] - S(q)w[k] \tag{4.38}$$

where $R(q)$, $S(q)$ and $T(q)$ are the discrete controller polynomials and where:

$u[k]$	: control signal fed to process
$v[k]$	: reference signal
$w[k]$	: process output

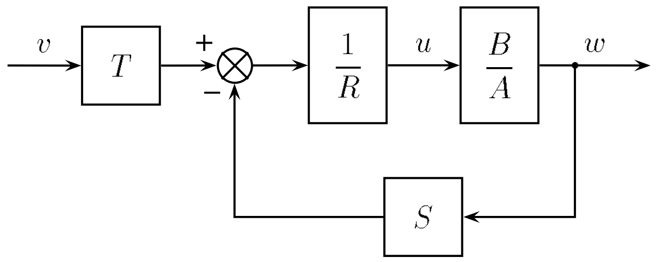


Figure 4.18: *Fuzzy pole-placement controller scheme.*

The process model $H(z)$ is:

$$H(z) = \frac{B(z)}{A(z)} = \frac{\tilde{b}}{z - \tilde{a}} \tag{4.39}$$

where $\tilde{a}$ and $\tilde{b}$ represent the "fuzzy" parameters of $A(z)$ and $B(z)$, respectively. These parameters are determined by a fuzzy rule base each sampling time. In section 4.4.3 this is described in more detail.

The desired closed-loop transfer function $H_m(z)$ is chosen to be a second-order transfer function:

$$H_m(z) = \frac{B_m(z)}{A_m(z)} = \frac{(1 + p_1 + p_2)z^{-1}}{1 + p_1 z^{-1} + p_2 z^{-2}} \tag{4.40}$$

where:

$$\begin{aligned} p_1 &= -2\exp(-\zeta\omega T_s)\cos(\omega T_s\sqrt{1-\zeta^2}) \\ p_2 &= \exp(-2\zeta\omega T_s) \end{aligned}$$

with:

ζ	: desired damping ratio
ω_n	: desired natural frequency (bandwidth)
T_s	: sampling time

The controller parameters ζ and ω_n can be chosen to define a desired process response. This results in the following settling time and overshoot for step responses:

$$t_{\text{set}} = \frac{4}{\zeta\omega} \text{ seconds} \tag{4.41}$$

$$\text{os} = \exp\left(-\frac{\pi\zeta\omega}{\omega\sqrt{1-\zeta^2}}\right) \% \tag{4.42}$$

The controller polynomials are derived from:

$$\frac{BT}{AR+BS} = \frac{B_m}{A_m} \tag{4.43}$$

When polynomial $R(z)$ is chosen to minimize the steady-state error e_{ss}, the following controller polynomials can be derived:

$$R(z) = 1 - z^{-1} \tag{4.44}$$

$$S(z) = \frac{p_1 + \tilde{a} + 1}{\tilde{b}} + \frac{p_2 - \tilde{a}}{\tilde{b}} z^{-1} \tag{4.45}$$

$$T(z) = \frac{1 + p_1 + p_2}{\tilde{b}} \tag{4.46}$$

The next section shows the results of some experiments made using the fuzzy pole-placement controller described in this section.

4.4.3 Experiments with fuzzy pole-placement controller

The fuzzy relations between $K(y)$ and $\tau(y)$, and $|y|$ is determined by five simple fuzzy rules each, and five membership functions on the universe of $K(y)$, $\tau(y)$ and $|y|$. Table 4.2 show the centers of the membership functions for the five fuzzy rules. In section 4.4.1 it was described how such a fuzzy linear model can be determined. The reference model parameters where chosen as:

$$\zeta = 0.95$$

$$\omega_n = 3.5$$

$$T_s = 0.1$$

$$T_d = 0.1$$

Table 4.2: *Centers of membership functions used in fuzzy pole-placement control experiment.*

$\lvert y\rvert$	0	0.25	0.5	0.75	1
$K(y)$	2	1.75	1.2	0.95	0.75
$\tau(y)$	0.5	0.75	1	1.25	1.5

The nonlinear proces used in the experiments is given by:

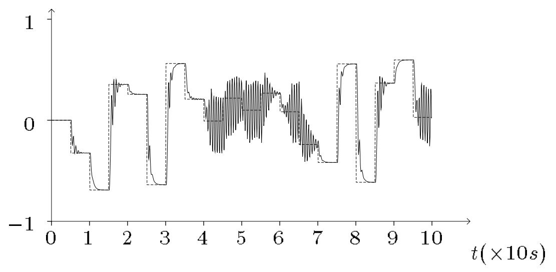
$$\underbrace{(\tfrac{1}{2} + |y|)}_{\tau(y)}\dot{y}(t) + y(t) = \underbrace{2e^{-|y|}}_{K(y)}u(t - T_d) \tag{4.47}$$

where $T_d = 0.1$. The used sampling time is $T_s = 0.1$. The experimental results are shown in figure 4.19.

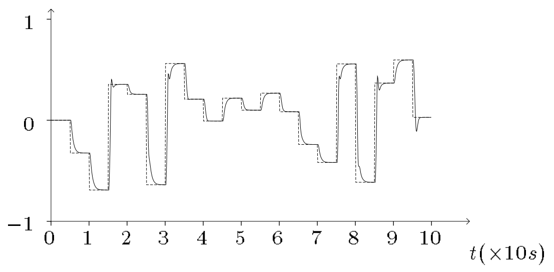
Figure 4.19 shows the performance improvement when the pole-placement controller is based on the fuzzy linear model. In this example this is equal to using a Takagi-Sugeno model (Sugeno-rules). The first approach performs interpolation between parameters of the continuous model, the second approach performs interpolation between different outputs of the discrete controller. This means that less fuzzy rules are needed in the case of fuzzy linear control if the nonlinearities of the process can be “decoupled”.

4.4.4 Remarks and considerations

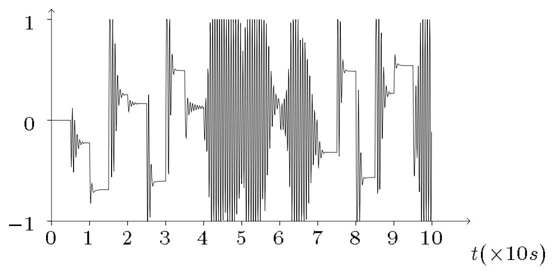
The modeling of processes using linear model structures and fuzzy parameters provides a way to model nonlinear processes. These models with “fuzzy” parameters, depending on a priori chosen signals, can compensate for non-linearities of the process to be modeled. Controllers based on (linear) process models can be extended to use the (local) parameters of the fuzzy linear model of the process. Experiments with a pole-placement controller show how a non-linear process gain can be compensated for by using a model with fuzzy parameters, or by using a fuzzy model consisting of Sugeno rules. Experiments have shown that in some cases local models can be obtained which are instable. However, the control signals resulting from these instable local models force the process into regions where the local models are stable and, hence, do not result in a “global” instability (van Graafeiland, 1992; Jager et al., 1993a).



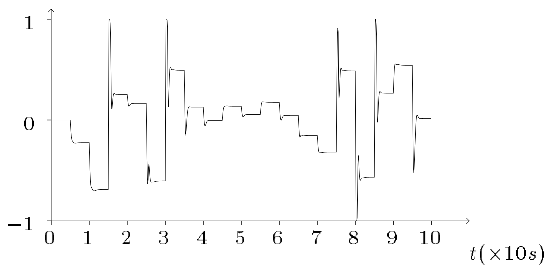
(a)
linear PPC - set-points and process



(b)
fuzzy PPC - set-points and process



(c)
linear PPC - control signal



(d)
fuzzy PPC - control signal

Figure 4.19: *Results of fuzzy pole-placement controller (PPC) experiments. The left column shows the results in case a linear process model is used ($\tau = 1, K = 1$): linear pole-placement controller. The right column shows the results of the fuzzy pole-placement controller.*

The preliminary result given in the previous section show no significant differences between the results obtained from using a linear model with fuzzy parameters and the results obtained from using a fuzzy model according to Takagi and Sugeno. Whether the interpolation should be performed on the continuous model parameter level (linear model with fuzzy parameters) or on the discrete controller output (Takagi-Sugeno-model) is not very clear, and more research has yet to be done on this topic. The number of fuzzy rules in a Takagi-Sugeno model depends exponentially on the number of signals the model uses to model the nonlinearities of the process, since for every input combination a rule should exist. Such a model has a local model for every input combination. In the case of a linear model structure with fuzzy parameters, this dependency is a linear one when the nonlinearity of a parameter depends on only one signal. Hence, a decoupling of the nonlinearities of the process parameters is performed.

The next interesting step is to perform experiments with the described method of fuzzy linear modeling and control in the field of predictive control. Using fuzzy linear models will still provide the possibility of using linear model structures. Another interesting topic for further research is the on-line adaptation of the fuzzy parameters of the model. This should be done by some supervisory level, for stability reasons. Fuzzy linear control could, for example, provide a way of "learning" a time-varying nonlinearity caused by "wearing out" or changes in the environment of the process to be controlled. The experiments have so far focussed on the simulation of nonlinear processes which were time-invariant (van Graafeiland, 1992; Jager et al., 1993a; Jager et al., 1993b).

4.5 Fuzzy controller as input-output mapping

This section describes a fuzzy controller (system) as an input-output mapping. When we look at a fuzzy controller as a (static) transformation of controller inputs to control outputs, then a fuzzy controller is actually a (non)linear function. Therefore, a fuzzy controller maps controller inputs to controller outputs and this mapping is in most cases a nonlinear one. A general notation for this mapping is:

$$\boldsymbol{y} = f(\boldsymbol{x}) \tag{4.48}$$

The numerical output $\boldsymbol{y}'$ is the defuzzification of the fuzzy output B', which in turn is the result of applying the compositional rule of inference:

$$\boldsymbol{y}' = f(\boldsymbol{x}') = \text{dfz}(B') = \text{dfz}(A' \circ R) \tag{4.49}$$

Clearly, this is not a function that can be easily analyzed. However, this mapping can be regarded as the result of interpolation between points in a multi-dimensional space, when some conditions are met. These conditions are given in the next section. In section 4.5.2, it is shown that under certain conditions, a fuzzy controller can emulate a linear controller.

4.5.1 Fuzzy system as universal approximator

In the previous introduction, it was stated that a fuzzy controller (or model) can be regarded as an input-output mapping: $\boldsymbol{y} = f(\boldsymbol{x})$ The fuzzy rules are supposed to define the characteristic of the mapping $f(\boldsymbol{x})$ since they represent the knowledge on which the fuzzy controller or model is based. Among others, Kosko (1992) proved that fuzzy systems are *universal approximators*. With universal approximators, systems are addressed which can approximate any mapping (function). This implies that:

$$\forall\, \boldsymbol{x} \in X,\ |F(\boldsymbol{x}) - f(\boldsymbol{x})| < \varepsilon \tag{4.50}$$

where $F(\boldsymbol{x})$ is the function to be approximated and ε can be chosen to be arbitrarily small. When we regard the fuzzy system as a discretization of $F(\boldsymbol{x})$, and know that between these discretizations interpolation is performed by means of fuzzy inference, it is clear that increasing the number of rules can provide a better approximation. This is similar to the approximation of a continuous function by a number of points: the higher the number of points is the better the approximation of the continuous function can be.

The fuzzy system can be regarded as an interpolation between a number of points, each defined by a fuzzy rule. It should be noted that the interpolation is only done between at most 2^{N_X} characteristic points, each represented by one of the 2^{N_X} fuzzy rules, if the following conditions are met:

- the fuzzy sets defined for the inputs and the outputs form fuzzy partitions as defined by (2.9);
- the membership functions are convex and normal; this results in case of fuzzy partitions in no more than two overlapping fuzzy sets (see section 2.1.2);
- the rule base is complete (see section 3.2.5 for completeness of rule bases).

When these conditions are met, there are at most 2^{N_X} rules active, resulting in an interpolation between the consequents of those active fuzzy rules in a N_X-dimensional space. In the following section this is described in more detail, because it is used to obtain a linear mapping. Besides the fact that the mapping can be nonlinear as a result of the fuzzy rules, the mapping can also be nonlinear because of the chosen operators, membership functions, etc. The analysis of the effects these parameters have on the nonlinearity of the mapping are described and discussed in section 4.6.

4.5.2 Linear controller $\subset$ fuzzy controller

Below we discuss the relation between fuzzy and linear control. It is shown that any linear controller can be described as a fuzzy controller: fuzzy control can be seen as a superset of linear control or linear control as a subset of fuzzy control. When designing a fuzzy controller and applying specific choices for the shape of membership functions, logical operators and the scaling of in- and outputs, the fuzzy controller can emulate a linear controller. From this point of view, linear control can be seen as a subset of fuzzy control. As in previous sections, only numerical inputs are considered. In the case of a linear controller the mapping is considered as a linear algebraic equation:

$$y = \boldsymbol{c}^T \boldsymbol{x} + d = \sum_{i=1}^{N_X} c_i x_i + d \tag{4.51}$$

where d is an offset. The fuzzy controller function $y = f(\boldsymbol{x})$ can emulate the linear controller (4.51) when meeting the following criteria:

- **L-1** the membership functions of the fuzzy sets on the universe of discourse of the inputs are triangularly shaped and normal;
- **L-2** the fuzzy sets for each input form a fuzzy partition: the sum of the membership functions equals 1;
- **L-3** the fuzzy rule base is complete;
- **L-4** a T-norm is used for the implication function (T-implication);
- **L-5** the operator for the conjunction in the premises of the fuzzy rules is the *product* operator;
- **L-6** the *(bounded) sum* operator (union according to Łukasiewicz) is used for the aggregation and for the *or* connective if it is used;
- **L-7** the defuzzified consequents (constant numerical representations) of the individual fuzzy rules are chosen according to equation (4.51);
- **L-8** the fuzzy-mean defuzzification method is used; this implies the choice for the aggregation operator in L-6.

In many fuzzy controllers found in literature, criteria L-1 to L-4 are met. The criteria L-1 to L-4 imply that there exist a fuzzy rule for every input combination. Although not always explicitly stated, many fuzzy controllers normally also meet criterion L-8 (see also section

4.2.4). This leaves criteria L-5 to L-7 to be the main differences between "standard" fuzzy controllers and linear controllers. Using the summation and product operators instead of the *max* and *min* operators, respectively, is necessary because the emulation of linear controllers requires operators that result in linear interpolation.

The most important criterion to be met is L-7, because, due to the rest of the criteria, the output of a fuzzy controller with N_X inputs results in a weighted sum of 2^{N_X} points in a N_X-dimensional space. The numerical consequents of the at most 2^{N_X} contributing fuzzy rules determine whether or not a linear relation by interpolating between these points (hyperplane), exists. When this hyperplane exists, the input-output mapping of the fuzzy system equals (4.51).

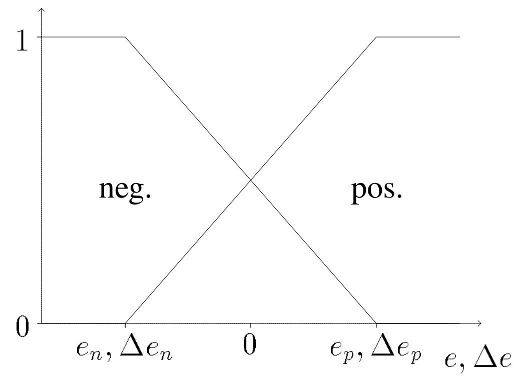
Summarizing the above, one can state that **a fuzzy system (controller or model) can emulate any linear system, if the linear system is represented by a static functional description**. The minimum number of necessary rules is 2^{N_X} in the case of N_X inputs. The proof is given in appendix B. It is shown in figure 4.20 how a linear PD controller can be emulated by a fuzzy controller based on four fuzzy rules. When changing the control signal u to the control signal change Δu in the consequents of the rules, a linear PI controller can be emulated (see also the example given in the introduction, section 1.3.1). In the application of fuzzy control, this emulation of a linear controller can provide an initial fuzzy controller based on a linear controller, which can already exist, and which can be tuned by adding and modifying fuzzy rules.

In the case of Sugeno rules with (non-constant) functions as consequents (see section 4.3.2), there is also another way to emulate a linear controller by a fuzzy controller. The method is very simple: all rules should have (4.51) as consequent. Because of the weighted sum of the subresults of the individual rules, this results in a controller output which equals (4.51):

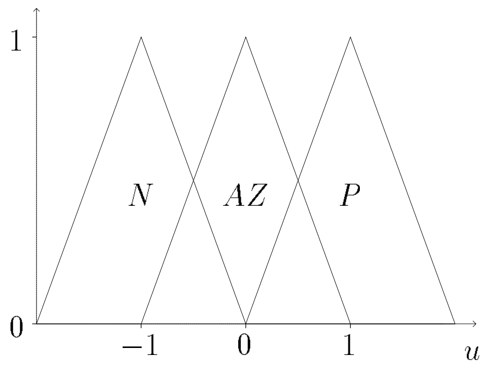
$$y = \frac{\sum\limits_{k=1}^{N_r} \beta_k (\boldsymbol{c}_k^T \boldsymbol{x} + d)}{\sum\limits_{k=1}^{N_r} \beta_k} \tag{4.52a}$$

$$= \frac{(\boldsymbol{c}^T \boldsymbol{x} + d) \sum\limits_{k=1}^{N_r} \beta_k}{\sum\limits_{k=1}^{N_r} \beta_k} \tag{4.52b}$$

$$= \boldsymbol{c}^T \boldsymbol{x} + d \tag{4.52c}$$



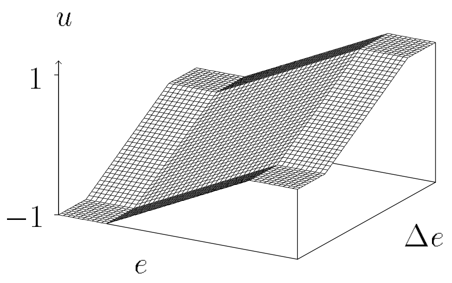
(a)
input universes



(b)
output universe

$\Delta e \setminus e$	neg.	pos.
pos.	AZ	P
neg.	N	AZ

(c)
rule base



(d)
control surface

Figure 4.20: *Fuzzy controller emulating PD controller. The error* e *and first difference* Δe *are the controller inputs* **(a)***; the control signal* u *is the output* **(b)***. The rule base is shown in* **(c)** *and the resulting control surface is given by* **(d)***. Note that if* $e \in [e_n, e_p]$ *and* $\Delta e \in [\Delta e_n, \Delta e_p]$*, the control surface is linear.*

As (4.52) shows, there are no restrictions on the membership functions or the T-norm used in the rule of inference for the *and* connective. The only restriction is that there should at least be one active fuzzy rule. This is because in Sugeno rules, each consequent can be a (local) linear controller.

Although this approach to emulating a linear controller seems trivial and rather useless, it can be used as a starting point for a nonlinear controller. Because a fuzzy controller with Sugeno rules can be seen as a collection of local linear controllers in between which is interpolated, the local linear controllers can initially be equal (an overall linear controller) and tuned separately, giving a "more optimal" non-linear controller. However, as shown in section 4.3.2, the use of functional consequents can lead to interpolation, which causes counter-intuitive results when not properly applied.

4.6 Fuzzy controller analysis

The influence of specific parts of a fuzzy controller are analyzed in this section. The analysis is done on the practical approach to fuzzy controllers as described in the previous sections. It was shown in the previous section that a fuzzy controller can be regarded as a nonlinear mapping from inputs to outputs. The following subsections use this fact to analyze the influence of specific parts of a fuzzy controller. In section 4.6.1 the role of the fuzzy sets is addressed. Additionally, the role of different types of operators is analyzed in section 4.6.2. Aspects of the rule base and its influence on the controller function are discussed in section 4.6.3.

4.6.1 Role of fuzzy sets

In the following subsections, a number of aspects concerning the influence of membership functions in the application of fuzzy control are addressed. It is shown that the shape of the membership functions can introduce nonlinearities and that more that two overlapping membership functions on a universe of discourse leads to a "filtering" of the controller function.

4.6.1.1 Number of fuzzy sets

When building a fuzzy controller, one of the first questions which arises, after having chosen the inputs, is that of how many fuzzy sets are needed and how the fuzzy sets should

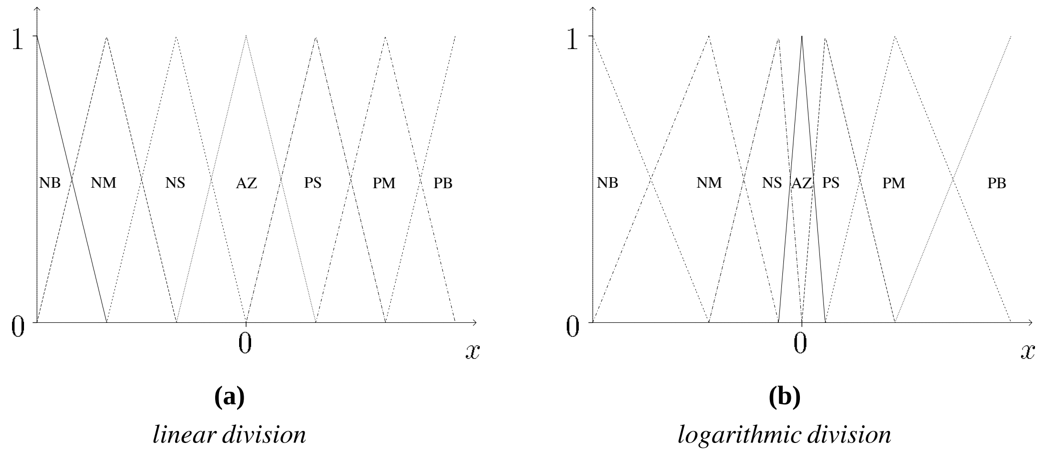


Figure 4.21: *Commonly used divisions of fuzzy sets on a universe of discourse. Abbreviations: N(egative), P(ositive), A(bout), B(ig), M(edium), S(mall) and Z(ero).*

be divided on the universes of discourse of the inputs. More or less standard types of fuzzy sets on a universes of discourse of controller inputs are shown in figure 4.21.

Whether to use a linear (figure 4.21a), a non-linear, for example the one in figure 4.21b, or another set of fuzzy sets on the universe of discourse of an input, depends on the problem to be dealt with. It is not very unlikely that one can find a paper where the fuzzy sets for the input are chosen as those shown in figure 4.21b, and the fuzzy sets for the outputs are chosen as those shown in figure 4.21a, and an averaging defuzzification method is used. The resulting controller function $y = f(x)$ is just a "bad" approximation of the following function if the rule base is continuous (see section 3.2.5):

$$f(x) = \operatorname{sgn}(x)\, k\, \sqrt{|x|}$$

This principle also holds in the case of more inputs. What we want to show with this example, is that a number of fuzzy controllers represent a controller function which approximates a (non)linear function, which, in fact, can be implemented more easily by a mathematical description of that (non)linear function. One can imagine that, after a fuzzy controller is designed, an analysis of the resulting controller function shows that a simpler (and faster) implementation is possible by using a "similar" mathematical expression.

The higher the density of the fuzzy sets on a certain part of the universe of discourse, the more complex the controller output as function of the controller inputs can be defined. Actually, the choice of the number of fuzzy sets and how those fuzzy sets are divided over the universes of discourse requires knowledge of how the controller output should be related to the controller inputs. For example, designing a fuzzy controller for controlling a non-linear process requires knowledge of the non-linearity of the process. There is no standard design scheme that can be employed to choose the number and positions of the fuzzy sets, and too few people realize that this is a problem.

The number and positions of the fuzzy sets define the control hypersurface, and proper design of this control hypersurface requires process knowledge. When this process knowledge exists in the form of operator expertise, then one has to deal with the knowledge acquisition problem known from the field of expert systems. When a process engineer has knowledge of nonlinearities of the process to be controlled, then this knowledge can be used to choose the number and positions of the fuzzy sets of the fuzzy sets. For example, if the process exhibits a dead zone, a fuzzy set can be used to “cover” this dead zone. Then this fuzzy set can be used in a rule to “compensate” for the dead zone. This is similar to compensating for (simple) nonlinearities like it is done in the case of conventional controllers in practice.

4.6.1.2 Overlapping fuzzy sets

The influence of overlapping fuzzy sets is discussed in this section. The overlapping of fuzzy sets together with fuzzy inference and defuzzification result in interpolation. If the membership functions are convex and normal and the sets are a fuzzy partition, then the interpolation depends only on the “nearest” surrounding characteristic points and each characteristic point is uniquely defined by a fuzzy rule. This is because there are no more than two overlapping membership functions on the universes of discourse. See also section 4.5.2, where this property was used to emulate a linear controller by a fuzzy controller.

When there are more than two overlapping fuzzy sets the interpolation does not only depend on the “nearest” surrounding characteristic points. More characteristic points influence the output value and, for example, in the case of Gaussian membership functions all characteristic points influence the output which means that all fuzzy rules are fired. It can be shown that having more than two overlapping fuzzy sets can lead to a sort of “smoothening” of the control hypersurface that would result from having only two overlapping fuzzy sets. The latter represents the case where each rule really represents a point of the controller function. To show this phenomenon we consider the following three rules:

r_1 : **if** x is A_1 **then** y is B_1
r_2 : **if** x is A_2 **then** y is B_2
r_3 : **if** x is A_3 **then** y is B_3

where A_1, A_2 and A_3 all overlap. For the sake of simplicity, the fuzzy-mean defuzzification is assumed, using only numerical representations b_1, b_2 and b_3 of B_1, B_2 and B_3, respectively. The resulting output y' based on input x' is determined by:

$$y' = \frac{\mu_{A_1}(x')\,b_1 + \mu_{A_2}(x')\,b_2 + \mu_{A_3}(x')\,b_3}{\mu_{A_1}(x') + \mu_{A_2}(x') + \mu_{A_3}(x')}$$

When, for example, $b_1, b_3 < b_2$, then the output $\forall\, x'$, $y' < b_2$, since A_1, A_2 and A_3 all overlap. This means that extreme characteristic points of the controller function become less extreme due to the overlapping of the fuzzy sets, and a fuzzy rule is never the only one that is "active".

In figure 4.22, the smoothening effect on the output y as a function of the input x is shown, which occurs when more than two overlapping fuzzy sets on a universe of discourse exist. This interference, due to more than two overlapping fuzzy sets leads to effects on the controller hypersurface which are not easy to see a priori. **Changing the consequences of fuzzy rules will not have a trivial change in the controller mapping**, because the change is "filtered" by other rules which are active at the same moment.

4.6.1.3 Shape of fuzzy sets

The influence of the shape of the membership functions of fuzzy sets is analyzed in this section. As discussed in the previous subsection, more than two overlapping fuzzy sets will result in some nontrivial characteristics of the controller hypersurface. Therefore, we restrict ourselves in this section to fuzzy sets that form a fuzzy partition for a universe of discourse. Zadeh defined a bell-shaped membership function, known as the π-function, which is symmetrical. A more general representation is given by:

$$\mathrm{bell}(x, a, b, c, d) = \min(\mathrm{sigma}(x, a, b), 1 - \mathrm{sigma}(x, c, d)) \tag{4.53}$$

with:

$$\mathrm{sigma}(x, a, b) = \begin{cases} 0, & \text{if } x < a \\ 2\frac{(x-a)^2}{(b-a)^2}, & \text{if } a \le x \le \frac{1}{2}(a+b) \\ 1 - 2\frac{(x-b)^2}{(a-b)^2}, & \text{if } \frac{1}{2}(a+b) \le x \le b \\ 1, & \text{if } x > b \end{cases} \tag{4.54}$$

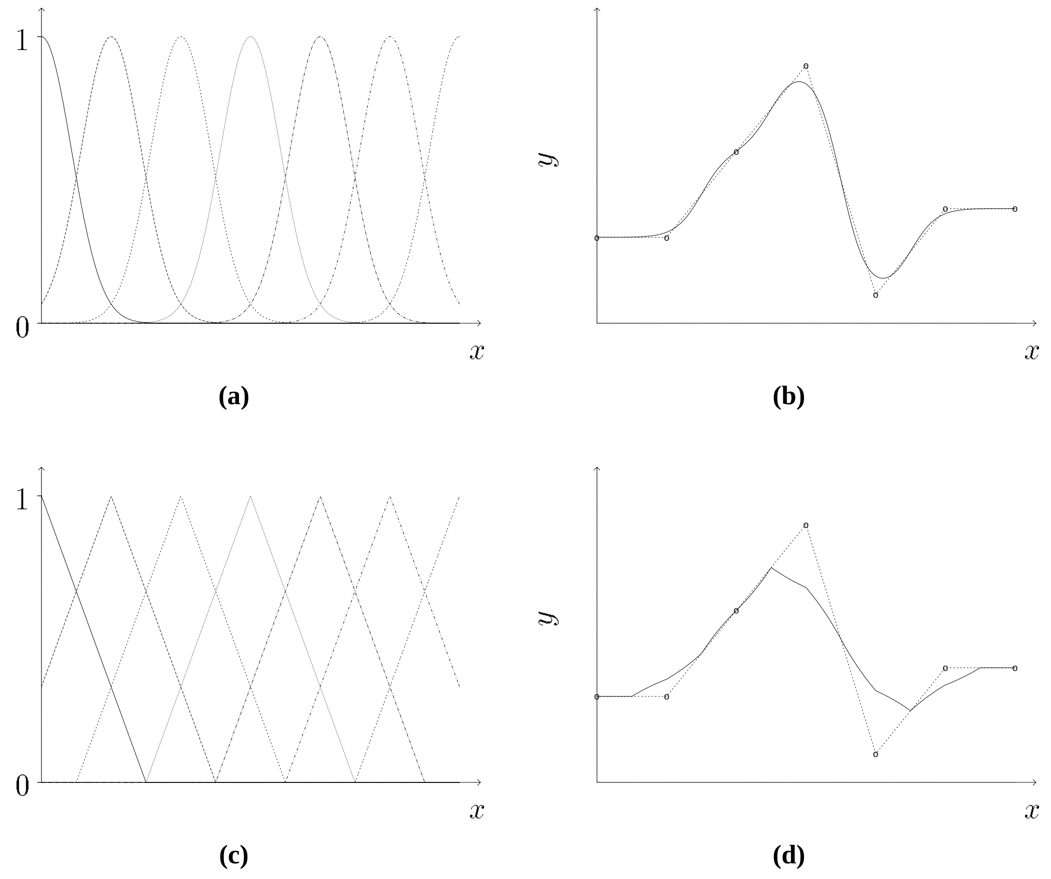


Figure 4.22: *Membership functions, Gaussian* **(a)** *and triangular* **(c)**, *and resulting (solid) controller functions* **(b)** *and* **(d)**, *respectively. The piece-wise linear curves (dotted) represent the case of a fuzzy partition with triangularly-shaped membership functions (linear interpolation). The points in* **(b)** *and* **(d)** *are the characteristic points of the controller function, each representing a rule, between which is interpolated.*

A general membership function using straight lines is given by:

$$\text{trap}(x,a,b,c,d) = \begin{cases} 1, & \text{if } b \leq x \leq c \\ \frac{(x-a)}{(b-a)}, & \text{if } a < x < b \\ \frac{(x-d)}{(c-d)}, & \text{if } c < x < d \\ 0, & \text{otherwise} \end{cases}$$

In figure 4.23, the two defined types of membership functions are shown: bell-shaped (figure 4.23a) and trapezoidally shaped (figure 4.23b). The "straight" membership functions, like trapezoidally shaped membership functions, represent the linear approach, the continuous membership functions, like the bell-shaped membership function, represent the non-linear approach.

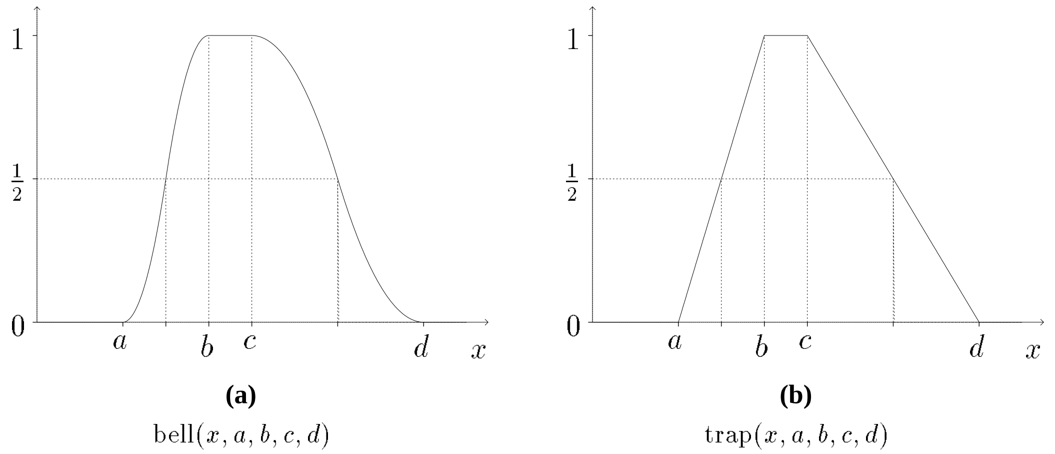


Figure 4.23: *Different types of membership functions.*

When the output of the fuzzy controller is regarded as a function of the inputs, which is determined by the interpolation between the numerical results of the individual fuzzy rules, then this interpolation is non-linear in the case of membership functions other than trapezoidally shaped. By using non-linear membership functions a non-linear characteristic is introduced in the fuzzy controller, which is not trivial. In figure 4.24, bell-shaped membership functions, with a core of only one point and forming a fuzzy partition, and the corresponding controller function are shown. In the same figure, the results are shown of when the corresponding straight membership functions are used (triangularly shaped). Clearly this nonlinear interpolation can provide the control surface desired, but this is, in our opinion, not according to the basic idea of a fuzzy controller: **the nonlinearity of the fuzzy controller should be defined by the fuzzy rules**, including the number and positions of the fuzzy sets, because these rules are the representation of the operator or process knowledge.

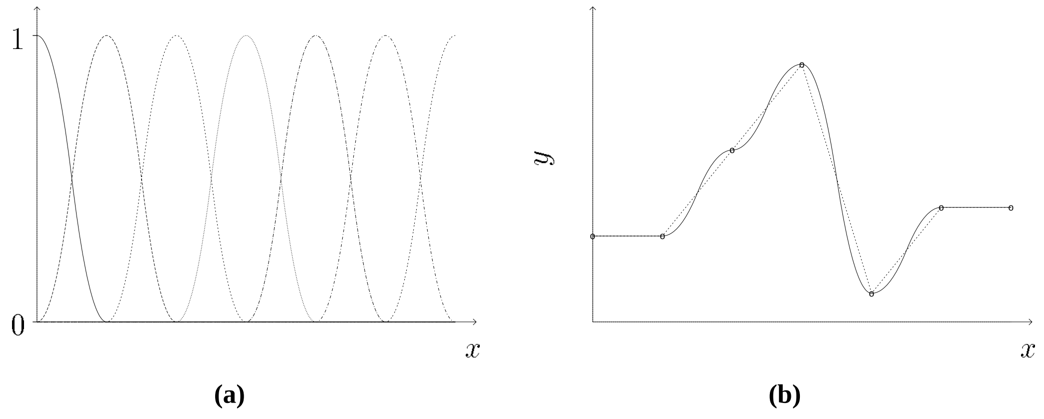


Figure 4.24: *Membership functions* **(a)** *and the resulting (solid curve) controller function* **(b)**. *The piece-wise linear curves (dotted) represent the case of a fuzzy partition with triangularly-shaped membership functions (linear interpolation). The points in* **(b)** *are the characteristic points of the controller function, each representing a rule, between which is interpolated.*

4.6.1.4 Fuzzy sets for the output

In many cases, fuzzy sets are defined on the universe of discourse of the output of a fuzzy controller. This, however, is not always the case as shows in section 4.3. If fuzzy sets are defined for the controller output, this is normally done using a set of equidistant membership functions for the fuzzy sets. It is hard to make any general remark concerning the "fuzzification" of the output, because its effect depends for a great deal on the fuzzy rules.

Because of the interpolative character of a fuzzy controller, a chosen set of membership functions does not limit the resolution of the controller output, assuming an averaging defuzzification method (see also sections 4.1.2 and 4.2.4). The number and positions of the membership functions on the universe of discourse of the output, in combination with the fuzzy rules and the universes of discourse of the controller inputs, should be chosen according to the control hypersurface desired; and this is exactly the problem with nonlinear controller design: how should the control hypersurface be shaped.

Supposing the rules describe the shape of the control hypersurface, any number of sets for the output will restrict this hypersurface. In section 4.3.1, it was discussed that a static set of fuzzy sets for the controller output restricts the resulting controller function. Hence, **it seems reasonable to use (constant) numerical consequents for the the fuzzy rules instead of fuzzy consequents chosen from a predefined number of fuzzy sets** on the universe of discourse of the output. This means that during design, the rules are based on an input classification and a (numerical) consequent which is not predefined, but determined “on the fly”. This allows fast and easy adjustment of the rule consequents and thus easy tuning of the fuzzy controller in the case of optimization based on trial-and-error. This more or less implies that Sugeno rules and fuzzy-mean defuzzification method are used.

4.6.2 Role of operators

The mapping which is obtained by a fuzzy controller depends on the fuzzy sets as shown in the previous section, but also on the operators used to represent the logical connectives. The implication is not considered since numerical inputs, T-norms for implications, and fuzzy-mean defuzzification are assumed. Also the aggregation is not considered in this section, since fuzzy-mean defuzzification is assumed. Further, fuzzy partitions for the inputs and numerical consequents are assumed. These assumptions are made because the analyses described in previous section(s) imply the following:

- numerical inputs are normally the case in (fuzzy) control;
- other implications that T-implications have severe drawbacks (section 4.2.1);
- numerical consequents do not limit the control function (see section 4.6.1.4) as opposed to fuzzy sets chosen from a predefined number of fuzzy sets;
- fuzzy-mean defuzzification is very suitable in the case of numerical consequents which are considered independently;
- aggregation is “embedded” in the fuzzy-mean defuzzification when all consequents are considered independently (see section 4.2.4.1 and 4.2.3.3).

In the following subsections, the influence of the logical connectives *and* and *or*, as well as the negation *not*, is considered.

4.6.2.1 Negation in rule premises

For the sake of convenience the *not* operator is sometimes used in the premises of fuzzy rules. This can lead to inconsistency, which is demonstrated by the following. Suppose that on the universe of discourse of variable x and y the fuzzy sets A_1, A_2, A_3 and B are defined. Now suppose the following rules:

r_1 : **if** x is **not** A_1 **then** y is B
r_2 : **if** x is A_2 **or** A_3 **then** y is B

In the case A_1, A_2 and A_3 are the only possible linguistic labels for x, then these two rules are logically equivalent. However, the application of fuzzy reasoning normally results in different outcomes. In figure 4.25, the result for the two cases is shown in the case when the *or* connective is implemented by the *max* operator.

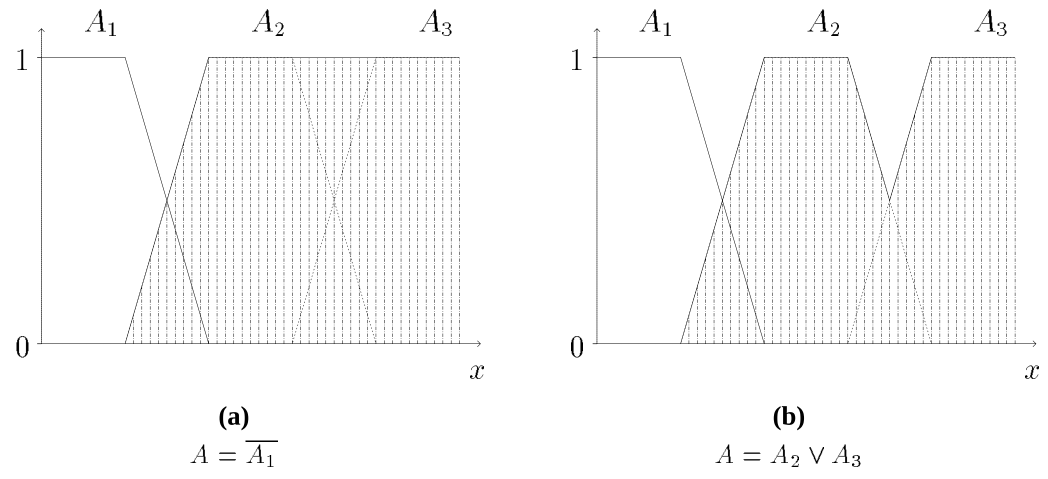


Figure 4.25: *Resulting fuzzy set (shaded) when using negation in a fuzzy rule premise* **(a)** *or disjunction of other sets* **(b)**.

When the fuzzy sets form a fuzzy partition (see section 2.1.2), it is possible to obviate these inconsistencies by meeting the following criteria:

- the bounded sum (*bsum* operator) is used for the *or* connective;

- standard complement is used for negation (*not*).

It can be suggested that the use of the bounded sum for the *or* connective is only suitable when the fuzzy propositions it operates on are "highly correlated", which is indeed the case when the *or* connective operates on two fuzzy sets on the same universe of discourse. In section 4.6.2.3 we go into more detail about the *or* connective.

4.6.2.2 Logical *and* connective

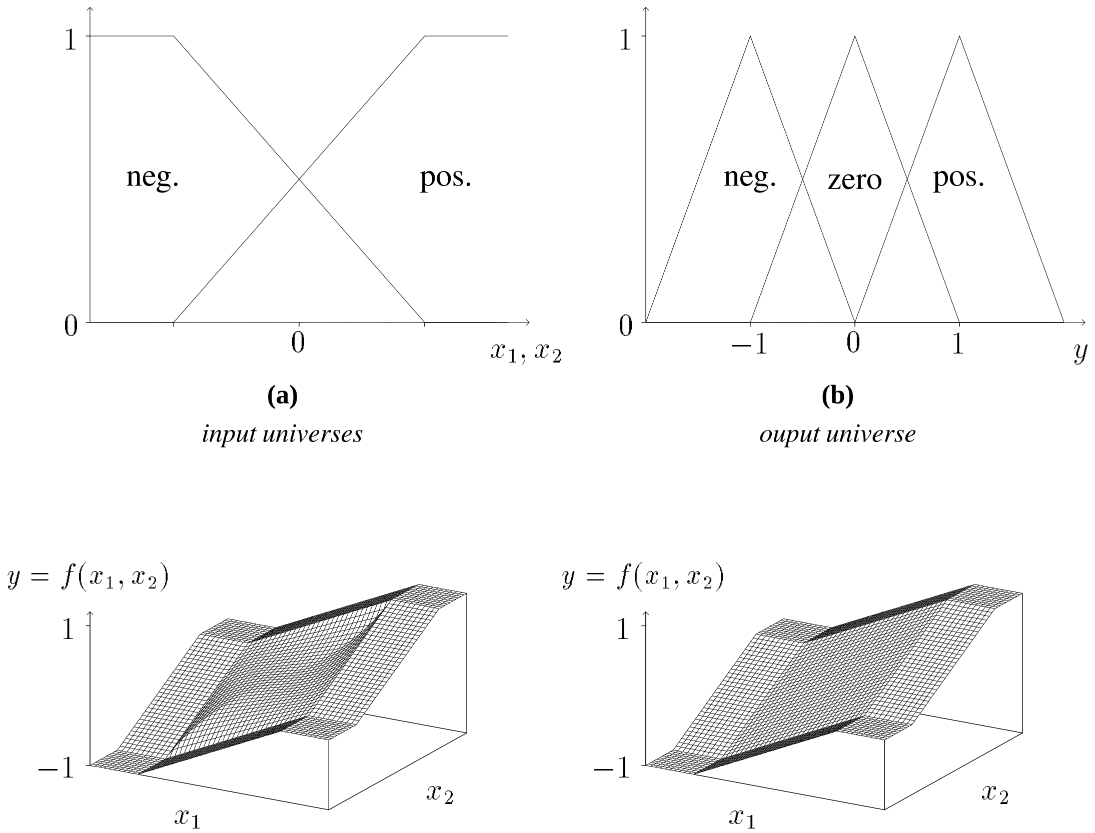
The *and* connective is essential to fuzzy control when there are two or more inputs. It is used to combine fuzzy propositions in the premises of the fuzzy rules. The underlying operator can be any T-norm, which poses the question of which T-norm to use. Many fuzzy controllers described in literature use the *min* operator for the *and* connective. However, is this a justified choice? Is the *min* operator used because it was first proposed by Zadeh, or because it was found to be the best choice for a certain problem? In many cases the use of the *min* operator for the *and* connective in the application of fuzzy control can be disputed, because it introduces non-linearities which are not trivial. To demonstrate this, let us give an example of a rule base with the following rules:

r_1 : **if** x_1 is negative **and** x_2 is negative **then** y is negative
r_2 : **if** x_1 is negative **and** x_2 is positive **then** y is zero
r_3 : **if** x_1 is positive **and** x_2 is negative **then** y is zero
r_4 : **if** x_1 is positive **and** x_2 is positive **then** y is positive

In figures 4.26 the fuzzy sets are shown for x_1, x_2 and y. Using the *min* operator for the *and* connective, the resulting controller output as function of the input is shown in figure 4.26c. Note that the type of implication function, assuming it is a T-implication function, does not have any influence, because the fuzzy-mean defuzzification method is used. Figure 4.26d shows the result when the *product* operator is used for the *and* connective. As can be seen, this is a linear function, which is obvious, because of the obtained results reported in section 4.5.2.

From these results it can be concluded that using the *min* operator for the *and* connective is questionable since it results in nonlinearities in the controller function which cannot be influenced.[*] In fact, any other T-norm besides the *product* operator will have similar

[*]Gupta and Qi (1991a) described studies which are based on a simple (4 rules) PI-like controller as shown in figure 4.26. They showed that different operators yield different control behavior for this simple fuzzy controller. This is obvious since different T-norms result in different types of interpolations, and because only 4 rules are considered, the interpolation (solely) determines the control surface. Hence, the results from these studies cannot be generalized to fuzzy controllers in general.



(a) *input universes*

(b) *ouput universe*

(c) *min operator*

(d) *product operator*

Figure 4.26: *Results obtained using* min *operator* **(c)** *and* product *operator* **(d)** *for the* and *connective, and fuzzy sets for x_1, x_2* **(a)** *and y* **(b)***. The controller output is normalized to $[-1, 1]$. Note the nonlinear interpolation in case of the* min *operator* **(c)***. See also figure 4.20.*

results. This can easily be shown by looking at the sensitivity to changes in the inputs for a general T-norm. If the sensitivity for all inputs has to be constant, but not equal zero, for the whole range, then the sensitivity for the inputs has to be independent of those same inputs. In other words, the partial derivative should be constant and unequal zero:

$$\frac{\partial \mathrm{T}(x_i, \ldots, x_{N_X})}{\partial x_i \cdots \partial X_{N_X}} = c, \text{ with } c > 0 \tag{4.55}$$

The only T-norm which satisfies this condition is the *product* operator ($c = 1$). Any **other operator will introduce nonlinearities which cannot be influenced in a trivial way**. Hence, these nonlinearities are not adjustable like the fuzzy rules themselves.

4.6.2.3 Logical *or* connective

As stated in 4.6.2.1, it is convenient to use the *not* operator in fuzzy rule bases. Also the use of the *or* connective provide a means for convenient rule base design. The number of rules can be decreased by use of the *not* operator and *or* connective. Besides the problem occuring when using the *not* operator (see section 4.6.2.1), the *or* connective can lead to similar problems. To solve the problem of the *not* operator it was suggested to use a *bounded sum* for the *or* connective, provided that it operates on the same universes. Extending this to fuzzy rules, it can be suggested to combine rules with the same consequent by means of the *or* connective. For example, the following rules:

r_1 : **if** x_1 is $A_{1,1}$ **and** x_2 is $A_{2,1}$ **then** y is B
r_2 : **if** x_1 is $A_{1,2}$ **and** x_2 is $A_{2,2}$ **then** y is B
r_3 : **if** x_1 is $A_{1,3}$ **and** x_2 is $A_{2,3}$ **then** y is C

can, on a semantic level, be transformed in:

$r_{1,2}$: **if** (x_1 is $A_{1,1}$ **and** x_2 is $A_{2,1}$) **or** (x_1 is $A_{1,2}$ **and** x_2 is $A_{2,2}$) **then** y is B
r_3 : **if** x_1 is $A_{1,3}$ **and** x_2 is $A_{2,3}$ **then** y is C

Assuming the *max* operator for the *or* connective and aggregation, and the *min* operator for the the *and* connective and implication, this results in:

$$\begin{aligned}
R_k &= \{A_{1,k} \cap A_{2,k}\} \times B, \text{ for } k = 1, 2 \\
R_{1,2} &= \{\{A_{1,1} \cap A_{2,1}\} \cup \{A_{1,2} \cap A_{2,2}\}\} \times B \\
&= \{\{\{A_{1,1} \cap A_{2,1}\} \times B\} \cup \{\{A_{1,2} \cap A_{2,2}\} \times B\}\} \\
&= R_1 \cup R_2
\end{aligned}$$

As can be seen in the above equations, the use of the *or* connective can be used to combine rules with equal consequents. This leads to a minimal number of rules within a fuzzy rule base: if N_B different consequents exist, then the minimal number of fuzzy rules is N_B. However, this combining of rules with equal consequents results in different outcomes of the inference in most cases. In fact, it is only possible when the used operators for implication and logical connectives meet certain conditions. The example given above is an example of this, but when, for example, the *or* connective is represented by the *bounded sum* operator, the transformation does not hold. The transformation does hold when the *or* connective is represented by the *bounded sum* operator, the fuzzy sets used in the premises form fuzzy partitions, and the implication is represented by the *product* operator. Hence, the usage of the *or* connective and its effect on the controller function is strongly related to the operators chosen for other connectives and implication.

4.6.3 Role of the rule base

The fuzzy rule base contains the rules of a fuzzy controller and represents the knowledge which is used to control the process. A number of problems can be detected with respect to the design of a rule base. In section 3.2.5 a number of properties of rule bases were addressed. In this section we will address incompleteness of a rule base and its effect on the controller function. Additionally, some aspects of exception handling and rule precedence are discussed.

4.6.3.1 Incompleteness and interpolation

Incompleteness of a rule base can result in undesired controller behavior. When the *blank spots* result in keeping the outputs of the fuzzy controller constant, the control hypersurface can exhibit discontinuous behavior. In figure 4.27 an example is given which shows this. As can be seen, keeping the output constant when no rule can be fired can lead to effects like hysteresis. In case the controller output is a change of the control signal fed to the process, similar effects occur if no control change is made when none of the rules are fired.

A solution to the problem described above, is interpolation between the fuzzy rules. Interpolation in sparse rule bases is described by Kóczy and Hirota (1993). The proposed mechanism is based on distances between α-*cuts* of the fuzzy sets. When the following two rules and data are considered:

if x is A_k **then** y is B_k, for $k = 1, 2$
x is A'

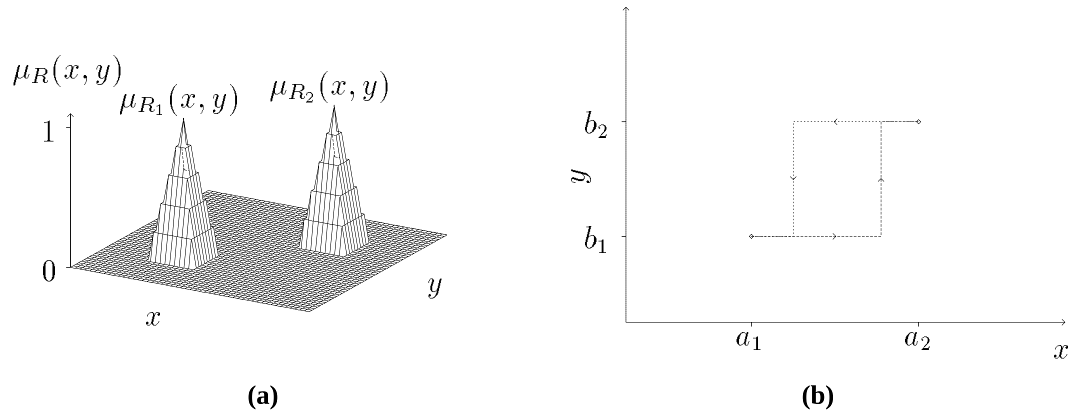


Figure 4.27: *Example of controller function* **(b)** *as result of incompleteness of the rule base. The fuzzy relation representing the rule base is shown in* **(a)**. *The value of x goes from a_1 to a_2 (dashed) and back (dotted). The (x,y)-tuples (a_1,b_1) and (a_2,b_2) represent the centers of the membership functions $\mu_{R_1}(x,y)$ and $\mu_{R_2}(x,y)$, respectively. The controller output is kept constant when non rules are fired.*

then the interpolation is determined by:

$$\frac{d_{\alpha,l}(A_1,A')}{d_{\alpha,l}(A_2,A')} = \frac{d_{\alpha,l}(B_1,B')}{d_{\alpha,l}(B_2,B')} \quad \text{and} \quad \frac{d_{\alpha,r}(A_1,A')}{d_{\alpha,r}(A_2,A')} = \frac{d_{\alpha,r}(B_1,B')}{d_{\alpha,r}(B_2,B')} \tag{4.56}$$

where:

$d_{\alpha,l}(A_1,A_2)$:	distance between the most left values within the supports of the level sets (α-*cuts*) of A_1 and A_2
$d_{\alpha,r}(A_1,A_2)$:	similar as above, but now for the most right values

An example of this interpolation technique is shown in figure 4.28. Kóczy and Hirota (1993) also show extensions of this mechanism for the cases of multiple rules and multiple propositions in the rule premises.

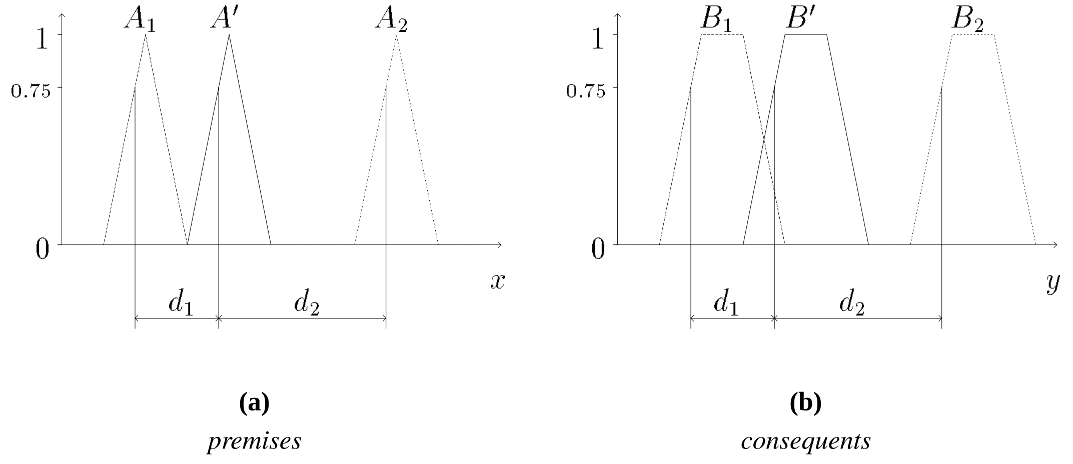


Figure 4.28: *Example of interpolation in a sparse rule base according to Kóczy and Hirota (1993), with* $d_1 = d_{\alpha=0.75,l}(A_1, A')$ *and* $d_2 = d_{\alpha=0.75,l}(A_2, A')$.

4.6.3.2 Exceptions and rule precedence

The rule bases that are used in fuzzy control are normally "flat". This means that there is no chaining of rules: rules "connect" the controller inputs directly to the controller outputs. Therefore, only flat rule bases are considered in the following. When we consider a set of rules, a number of problems can be detected. We briefly address the problem of rule exceptions and precedence which are closely related. To show the problem, let us consider the following two rule in a rule base:

r_1 : **if** x is A **then** y is B_1
r_2 : **if** x is *very*(A) **then** y is B_2

where *very*$(A) \subseteq A$. An example of this is shown in figure 4.29. Interpretation of these two rules and the meaning of the hedge *very* favors the idea that rule r_2 should overrule r_1. However, when x has full membership in *very*(A), it also has full membership in A and thus both rule will contribute to the output. In that case the fuzzy sets *very*(A) and A can be considered as full overlapping, resulting in a "filtering" of the control function (see also section 4.6.1.2 on this). A solution to this problem can be the transformation of the rules r_1 and r_2 to the following rules:

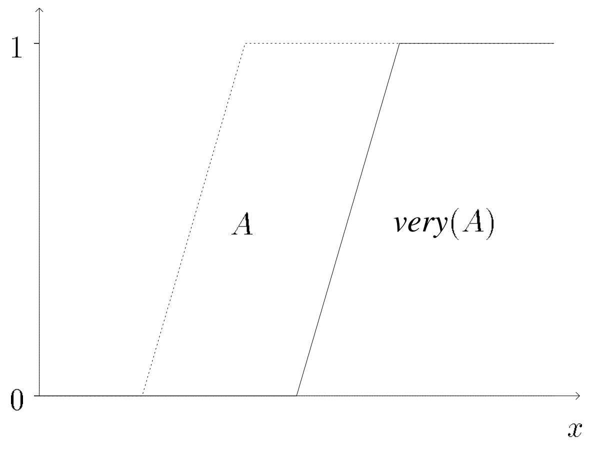


Figure 4.29: *Fuzzy sets* A *(dotted) and* very(A) *(solid) where* very$(A) \subseteq A$.

r'_1 : **if** x is $A \cap \overline{very(A)}$ **then** y is B_1
r'_2 : **if** x is $very(A)$ **then** y is B_2

This mechanism can also be applied for the cases of other types of linguistic modifiers. In general it can be stated that the rule with more restrictive (more specific) premises have to take precedence over the rules with less restrictive (less specific) premises. Similar problems occur when we consider a rule base which contains general rules and rules which express exception. The inference algorithm should fire the general rules when the exception rules cannot be fired. The solution is in fact based on the same principle that is used in the above-given example with the two rules. It should be noted that rule exceptions and rule precedence are normally not considered in fuzzy control.

4.7 Conclusions and remarks

A number of fuzzy control aspects have been addressed in this chapter. This section summarizes the main topics and give a number of conclusions based on the analysis of fuzzy controllers as described in the foregoing.

A distinction has been made between the *theoretical and practical approach to fuzzy control*. The theoretical or relation-based approach represents fuzzy control based on fuzzy relations and the composition of fuzzy relations. The practical or rule-based approach is based on a "matching" phase and a "modification" phase. The matching phase matches the fuzzy rule premises with the available data, yielding degrees of fulfillment for the fuzzy rules. The modification phase uses these support values to modify the consequents of the fuzzy rules to obtain the result of a specific fuzzy rule. Those results of the individual fuzzy rules are aggregated to obtain the overall result of the fuzzy inference. Some practical inference schemes (rule-based, local inference) are analytical solutions of the relation-based approach (relation-based, global inference; see section 4.2.3). Others are practical inference schemes which do not have a counterpart in theory; an example was given in section 4.2.3.2.

A number of *defuzzification methods* were discussed in sections 4.1.2 and 4.2.4. The basic defuzzification methods are center-of-gravity, mean-of-maxima and center-of-area. Extended defuzzification methods, described in section 4.2.4.3, have these basic defuzzification methods as special cases. The defuzzification, necessary in control to obtain numerical controller outputs, is one of the aspects which provide the interpolative behavior of fuzzy controllers. The mean-of-maxima defuzzification reduces a fuzzy controller to a multi-level relay. The centre-of-gravity defuzzification introduces nonlinearities in combination with aggregation operators other than summation. The fuzzy-mean defuzzification in combination with summation as aggregation operator provides an interpolation which does not introduce nontrivial nonlinearities.

In the practical approach to fuzzy control *two types of fuzzy rules* can be distinguished: rules with symbolic consequents (Mamdani rules) and rules with functions of the controller inputs as consequents (Sugeno rules). The Sugeno rules are based on a purely practical approach to fuzzy control (and modeling). Under certain restrictions these types of rules are equal (section 4.3.3). In section 4.2, different *types of implications* have been considered. It has been shown that implications based on the classical implication can lead to a number of problems which makes them less suitable for fuzzy control based on local inference (section 4.2.1.3 and 4.2.1.4). Using a conjunction to represent the implication (T-implications) has several advantages over an implication based on the classical implication and it leads to simple and straightforward inference schemes (section 4.2.3). The classical-conjunction-based implications are the types of implications which are normally used in fuzzy control. The type of conjunction does not play a rule in the case of Sugeno rules because of the weighted sum (fuzzy-mean) to obtain the controller output.

Fuzzy linear control is based on linear controllers of which the parameters are determined in-line by a fuzzy system (section 4.4). This is close to a controller based on Sugeno rules. The difference with such a controller is that in the case of Sugeno rules, the controller output is the weighted mean of the results of "local" controller outputs, and in the case of a fuzzy linear controller, the parameters of the controller are determined by a fuzzy system. Hence, fuzzy linear control focuses on the controller parameters. This entails that the number of required rules is at most equal to the number of rules required by a controller based on Sugeno-rules.

A fuzzy controller can be regarded as an *input-output mapping*. This mapping can be made equal to the mapping performed by a linear controller. When viewing a fuzzy controller as an input-output mapping, several aspects concerning the operators and types of fuzzy sets to be used can be analyzed. In section 4.6, we showed the influence of logical operators and types of membership functions, and stressed that many nontrivial nonlinearities can be introduced. In our opinion, **nonlinearities of the control hypersurface should be determined by the fuzzy rules**, since these rules represent the knowledge the controller is based on. When using the product operator for conjunction in combination with the fuzzy-mean defuzzification method, yielding summation for aggregation, a interpolation is obtained which does not introduce nontrivial nonlinearities in the control hypersurface.

5 Adaptive fuzzy control

Adaptivity of a controller can provide robustness for time-varying behavior or nonlinearities of the process to be controlled. Adaptivity can also provide auto-tuning of a controller of which the design is based on a model of the process. In literature, adaptivity of a controller has been addressed many times for all kinds of controllers. Also, adaptive fuzzy controllers have been reported many times. In the seventies the *self-organizing controller* was introduced by Procyk and Mamdani (1979) and since then many authors have reported research based on this type of controller, or on modifications and improvements of this controller. In recent years, numerous contributions could be found in literature which deal with *fuzzy neural networks*, also referred to as *neuro-fuzzy systems*. These fuzzy systems (controller or model) use a gradient-descent adaptation algorithm to adapt parameters of the fuzzy system.

First, the self-organizing controller and related methods are addressed in section 5.1. In section 5.2, an adaptation method for fuzzy controllers is described which uses a fuzzy relation as an associative memory: the same fuzzy relation is used for both modeling and control. These fuzzy systems adapt the control algorithm by identifying or enhancing a model of the process to be controlled. Section 5.3 addresses the adaptation of controllers by means of supervisors. Adaptive fuzzy systems based on gradient-descent adaptation rules are discussed in section 5.4. In section 5.5, comparisons are made between fuzzy systems and "comparable" learning algorithms, like the *radial-basis function network*

(RBFN) and the (generalized) *cerebellar model articulation controller* (CMAC). The final section summarizes and concludes this chapter.

5.1 Self-organizing fuzzy control

The *self-organizing controller*[*] (SOC) was introduced by Procyk and Mamdani (1979) and is a well known method of adaptation for fuzzy controllers. Applications and studies into self-organizing control can be found in literature (Brown et al., 1991; de Neyer et al., 1990; Shao, 1988; Sugiyama, 1988). Examples of applications of self-organizing fuzzy control are robot control (Wakileh and Gill, 1990) and application to muscle relaxant anaesthesia (Linkens and Hasnain, 1991). The SOC consists of two parts: a fuzzy controller and an adaptation mechanism. The adaptation mechanism acts directly on the parameters (read: fuzzy rules) of the fuzzy controller.

5.1.1 Self-organizing controller scheme

The schematic representation of the self-organizing controller is given in figure 5.1. The fuzzy controller in this scheme is based on Mamdani rules as described in chapter 4. The adaptation mechanism consists of a module which determines a performance measure and a module which modifies the rules of the fuzzy controller based on the performance measure, using a minimal process model. The performance measure is given by a fuzzy rule-based system that is based on the same inputs as the fuzzy controller, the error and error change, except for the fact that the conclusions of the rules represent a performance measure instead of a control action. This performance measure $p[kT]$ is a numerical value obtained from what Procyk and Mamdani (1979) refer to as a decision table, usually based on the error and its first difference:

$$p[kT] = f(e[kT], \Delta e[kT]) \tag{5.1}$$

where f is a look-up table. The decision table represents, in fact, a reference model: the more the process deviates from the implicit reference model, the worse the performance is classified. The main idea is that the better the performance is, the more the performance measure equals zero and that the sign and magnitude of the performance measure denote

[*]Today this type of control is usually referred to as *self-organizing fuzzy control* (SOFC) or *self-organizing fuzzy logic control* (SOFLC) to stress the application of fuzzy (logic) control. Here, we use the abbreviation SOC to address the controller proposed by Procyk and Mamdani (1979) and SOFC to address this type of controller in general.

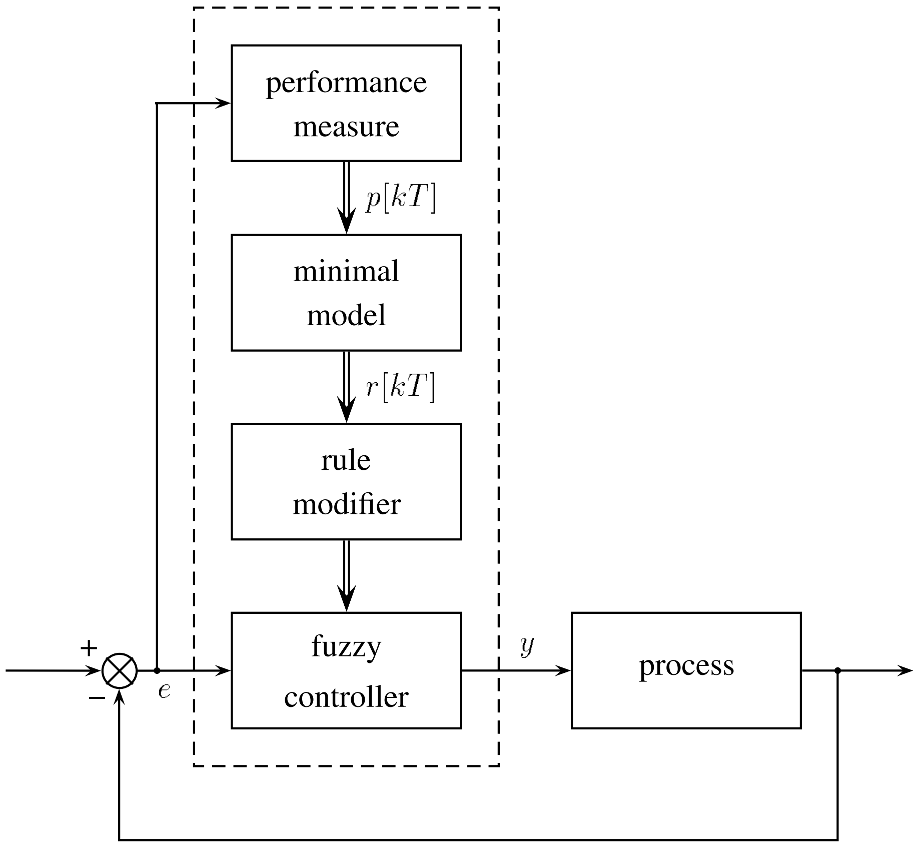


Figure 5.1: *Self-organizing fuzzy controller scheme.*

a direction and relative magnitude for improvement of the systems performance. To summarize, the working of the self-organizing controller according to Procyk and Mamdani (1979):

> The *rule base modifier* modifies the fuzzy rule base of the fuzzy controller according to the result of the *performance measure*, using a *minimal model* of the process.

Since the performance measure represents a measure of how to change the controller output(s), a process model is needed which can relate these changes to control signal changes. Hence, an incremental model is needed. The inverse of this incremental process model is used to determine the changes that should be made to the inputs of the process if certain output changes of the process are desired. The incremental model $\boldsymbol{M}$ is determined by:

$$\boldsymbol{M} = T\boldsymbol{J} \tag{5.2}$$

where $\boldsymbol{J}$ is the Jacobian of the process model and T is the sampling time. Therefore, if the process to be controlled is modeled by a linear first-order system, the most simple incremental model is the sign of the DC gain of the process, assuming the magnitude of the DC gain is compensated for by scaling factors of the inputs and outputs of the SOC. However, this is not valid for higher-order processes and in that case the partial derivatives have to be determined. Although an incremental model is probably easier to obtain than a transfer function, determining the partial derivatives needed for the Jacobian $\boldsymbol{J}$ can still be a hazardous task. Procyk and Mamdani therefore state that it is easier to keep the incremental model $\boldsymbol{M}$ constant and hope that by means of subsequent adaptations the inaccuracies caused by this approximation will be overcome. Hence, each adaptation can be seen as a step in the direction of a better performing system. It is clear that the self-organizing controller is based on local optimization, which hopefully results in global optimization.

When the minimal process model is determined, the changes (reinforcements) to the controller outputs, denoted by $\boldsymbol{r}[kT]$, have to be determined. As stated before, Procyk and Mamdani used numerical performance measures obtained from a look-up table. Each sampling instant a performance measure $p[kT]$ is determined. The reinforcement values $\boldsymbol{r}[kT]$ are determined by means of the numerical performance measure $p[kT]$ in combination with the minimal process model $\boldsymbol{M}$:

$$\boldsymbol{r}[kT] = \boldsymbol{M}^{-1} p[kT] \tag{5.3}$$

When $\boldsymbol{M}$ is simplified to be constant and a SISO process is considered, (5.3) can be reduced to:

$$r[kT] = k\, p[kT] \tag{5.4}$$

where k is a scalar. The user has to have (some) knowledge of the dynamics and delay times of the process to be able to determine which tuple of the inputs and outputs was responsible for the current performance. The “most” responsible tuple of the inputs and outputs will be used to update the fuzzy controller. Distribution over a number of responsible tuples is possible and Procyk and Mamdani (1979) state that distribution over 2 or 3 samples improves the initial response and the convergence rate. On convergence in general they state the following:

It should be borne in mind that in general convergence, when it takes place, is not global but only local to the set-point changes being applied. Nevertheless, this serves as a useful indication of the convergence properties of the controller.

At this point, two approaches can be distinguished: the *relation-based* approach and the *rule-based* approach. Both methods have their advantages and drawbacks and they are discussed in the following subsections. For the sake of simplicity, a controller with only one output is considered in the rest of this section.

5.1.2 Relation-based approach

In the *relation-based* approach, the modification of the rule base takes place using fuzzy relations. This is done by applying the update rule (Procyk and Mamdani, 1979; Harris and Moore, 1989; Pedrycz, 1989):

$$R[kT+T] = (R[kT] \ \cap \ \overline{R_{\text{bad}}[kT]}) \ \cup \ R_{\text{new}}[kT] \tag{5.5}$$

where $R[kT]$ is the fuzzy relation representing the fuzzy controller at time kT and R_{bad} and R_{new} are the relations describing the "bad" rule to be eliminated and the new "better" rule, respectively. The updating of R by (5.5) can be explained as extracting the "bad rule" R_{bad} from the controller, represented by R, and inserting the "new rule" R_{new} (which hopefully improves performance). However, since relations are involved, there is no clear distinction between rules within relation R after subsequent adaptations. The rules to be extracted and inserted are created instantly and are based on the in- and output history of the controller.

The relation R_{bad} is based on the controller inputs and outputs (x_i and y, respectively) which are responsible for the current process state and is constructed by:

$$\begin{aligned} R_{\text{bad}}[kT] = \text{fuzz}(x_1[kT-mT]) \times \cdots \\ \cdots \times \text{fuzz}(x_{N_X}[kT-mT]) \times \text{fuzz}(y[kT-mT]) \end{aligned} \tag{5.6}$$

where *fuzz* is a fuzzification operator which translates a numerical value to a fuzzy set (see section 4.1.1) and m is the number of samples it takes before a controller output change effects the process output. Hence, m includes the delay time of the process as well as delays due to zero-order hold functions. It can also include system dynamics which are to be neglected in order to obtain model reduction. The new relation R_{new} is constructed in a similar way by:

$$\begin{aligned} R_{new} = fuzz(x_1[kT-mT]) \times \cdots \\ \cdots \times fuzz(x_{N_X}[kT-mT]) \times fuzz(y[kT-mT]+r[kT]) \end{aligned} \tag{5.7}$$

where the only difference with (5.6) is the term $r[kT]$, which represents the reinforcement of the controller output, determined by the performance measure according to (5.3). Be aware that y denotes the output of the fuzzy controller, not the process output!

The fuzzification operator *fuzz* could simply result in a singleton membership function, but conversion to a fuzzy number is also possible. Converting the in- and outputs to fuzzy numbers will result in a generalization of the effects obtained by applying (5.5). The main disadvantage of the relation-based approach is the fact that using relations to apply self-organizing control requires a substantial amount of memory to store the fuzzy relation(s) and can cause the calculational loads necessary to perform the calculations according to (5.6), (5.7) and (5.5) for higher order systems to be severe. Another disadvantage is the fact that individual rules cannot be distinguished in the relational approach, which makes an interpretation after adaptation difficult. The advantage of this method is, however, the fact that memory requirements are constant and calculational efforts can be determined a priori. Shao (1988) proposed a self-organizing control algorithm which eliminates the application of (5.5) by directly modifying the fuzzy controller relation R. This modification decreases the amount of necessary computer memory and the calculational effort.

Next a numerical example showing the adaptation of relation R is given. The rule-based approach to self-organizing control is discussed in section 5.1.3.

5.1.2.1 Numerical example of the relation-based approach

An example to show the working of the relation-based approach of self-organizing control (Procyk and Mamdani, 1979) follows. Consider a fuzzy rule base with the following rules:

r_1 : **if** x is A_1 **then** y is B_1
r_2 : **if** x is A_2 **then** y is B_2

where the fuzzy sets are defined on discrete universes:

$$A_1 = 1/1 + 0.5/2 + 0/3$$
$$A_2 = 0/1 + 0.5/2 + 1/3$$

For the sake of simplicity, the fuzzy sets B_1 and B_2 are chosen equal to A_1 and A_2, respectively. The relation R is given by:

$$R = R_1 \vee R_2$$

$$
\begin{aligned}
&= (A_1 \times B_1) \vee (A_2 \times B_2) \\
&= \begin{bmatrix} 1 & 0.5 & 0 \\ 0.5 & 0.5 & 0 \\ 0 & 0 & 0 \end{bmatrix} \vee \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0.5 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0.5 & 0 \\ 0.5 & 0.5 & 0.5 \\ 0 & 0.5 & 1 \end{bmatrix}
\end{aligned}
$$

When a numerical input $x' = 2.4$ is considered, the value is rounded off to the nearest point on the discrete universe of discourse, resulting in the fuzzy set:

$$A' = 0/1 + 1/2 + 0/3$$

Applying the CRI yields:

$$B' = A' \circ R = 0.5/1 + 0.5/2 + 0.5/3$$

which results in the numerical output $y' = 2$ by means of the MOM defuzzification method. Now suppose the performance table in combination with the inverse minimal model, according to (5.3), results in the reinforcement value $r = 1$, then the “bad” and “new” relations are determined by:

$$
\begin{aligned}
R_{\text{bad}} &= \text{fuzz}(x'_q) \times \text{fuzz}(y') \\
R_{\text{new}} &= \text{fuzz}(x'_q) \times \text{fuzz}(y' + r)
\end{aligned}
$$

where x'_q represents the value of x' rounded off to the nearest point on the discrete universe of discourse. When the fuzzification operator is chosen to translate the numerical values in singleton fuzzy sets, the adaptation of relation R results in:

$$
\begin{aligned}
R' &= (R \wedge \overline{R_{\text{bad}}}) \vee R_{\text{new}} \\
&= \left(\begin{bmatrix} 1 & 0.5 & 0 \\ 0.5 & 0.5 & 0.5 \\ 0 & 0.5 & 1 \end{bmatrix} \wedge \overline{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}} \right) \vee \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0.5 & 0 \\ 0.5 & 0 & 1 \\ 0 & 0.5 & 1 \end{bmatrix}
\end{aligned}
$$

When the same input is provided as before ($x' = 2.4$), then the fuzzy output by applying the CRI, is:

$$B' = A' \circ R' = 0.5/1 + 0/2 + 1/3$$

which gives the numerical output $y' = 3$ by means of the MOM defuzzification method.

The above-given example is very simple, but is sufficient to show the different steps involved in the relation-based approach to self-organizing fuzzy control. The use of the MOM defuzzification method is one of the reasons why the result after one adaptation step is the desired result. When the COG defuzzification method is considered, the initial output is also 2, but the output after adaptation would be $2\frac{1}{3}$ and not 3 as obtained by the MOM method. When another adaptation step is considered with the same value for x', the output remains $2\frac{1}{3}$ because rounding off this value yields 2, causing R_{bad} to be the same as before. Hence, after the first adaptation step, the fuzzy controller relation R remains constant and the adaptation does not “improve” by subsequent adaptations:

$$\begin{aligned} R' &= (R \wedge \overline{R_{\text{bad}}}) \vee R_{\text{new}} \\ &= \left(\begin{bmatrix} 1 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 1 \end{bmatrix} \wedge \overline{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}} \right) \vee \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0.5 & 0 \\ 0.5 & 0 & 1 \\ 0 & 0.5 & 1 \end{bmatrix} \\ &= R \end{aligned}$$

By applying the CRI in combination with the data $x' = 2.4$, this results in a fuzzy output:

$$B' = A' \circ R' = 0.5/1 + 0/2 + 1/3$$

which is the same result that was obtained after the first adaptation. When using COG defuzzification, again the result $y' = 2\frac{1}{3}$ is obtained.

Note that the fuzzification of numerical values is chosen to produce singleton fuzzy sets in the above-given example. A fuzzification operator resulting in a fuzzy set, instead of a fuzzy singleton as in the above-given example, is possible. Then the adaptation is generalized over more elements of the (discrete) relations.

5.1.3 Rule-based approach

Another approach to self-organizing control is the rule-based approach . Procyk and Mamdani (1979) propose this method to obviate severe memory requirement and calculational loads. This approach uses the fact that the fuzzy controller relation is the disjunction of the fuzzy relations representing the fuzzy rules:

$$R = \bigcup_{k=1}^{N_r} R_k \tag{5.8}$$

where the fuzzy relation R_k, representing rule r_k, is the Cartesian product of the rule premise and consequent:

$$R_k = A_{1,k} \times \cdots \times A_{N_X,k} \times B_k \tag{5.9}$$

This yields for (5.5):

$$\begin{aligned} R = &\left(\bigcup_{k=1}^{N_r} R_k \cap \overline{A_1^{\text{bad}} \times \cdots \times A_{N_X}^{\text{bad}} \times B^{\text{bad}}} \right) \\ &\cup \left(A_1^{\text{new}} \times \cdots \times A_{N_X}^{\text{new}} \times B^{\text{new}} \right) \end{aligned} \tag{5.10a}$$

which, by applying the De Morgan's laws, results in:

$$\begin{aligned} R = &\left(\bigcup_{k=1}^{N_r} \left(A_{1,k} \cap \overline{A}_1^{\text{bad}} \right) \times \cdots \times A_{N_X,k} \times B_k \right) \cup \cdots \\ &\cdots \cup \left(\bigcup_{k=1}^{N_r} A_{1,k} \times \cdots \times \left(A_{N_X,k} \cap \overline{A}_{N_X}^{\text{bad}} \right) \times B_k \right) \\ &\cup \left(\bigcup_{k=1}^{N_r} A_{1,k} \times \cdots \times A_{N_X,k} \times \left(B_k \cap \overline{B}^{\text{bad}} \right) \right) \\ &\cup \left(A_1^{\text{new}} \times \cdots \times A_{N_X}^{\text{new}} \times B^{\text{new}} \right) \end{aligned} \tag{5.10b}$$

This means that each rule in the rule base is replaced by at most $(N_X + 1)$ fuzzy rules; this shows immediately the disadvantage of this approach, namely a constantly growing number of fuzzy rules in the rule base. Only rules which overlap with the bad rule have to be considered when applying (5.10). Further note that, when a fuzzification operator is used which translates a numerical value to a fuzzy set, consequents of new fuzzy rules can be subnormal and non-convex, because of the term $(B_k \cap \overline{B}^{\text{bad}})$ in (5.10b). The growing number of rules can be limited by using some sort of garbage-collection mechanism. This garbage-collection mechanism should remove rules which have similar premises as the new added rule, and also the rules which are "equal" should be reduced to only one rule. However, mechanisms like these are time consuming and time is a critical issue in direct control.

Procyk and Mamdani (1979) also propose another solution, namely replacing the consequent of rules which overlap the most with the rule to be extracted (the bad rule). If such a rule cannot be found, then a new rule should be inserted right away. In the next section, an even simpler method is described, based on the view that a fuzzy controller can be regarded as a mapping. When a certain mapping is implemented by a look-up table and an interpolation algorithm, the self-organization can be simplified and the need for large amounts of memory or garbage collection is eliminated; see also section 4.5.

5.1.4 Simplified rule-based approach

When a fuzzy controller is regarded as a combination of a look-up table and an interpolation algorithm, a major simplification of the self-organizing controller is possible. This simplification of the fuzzy controller and its adaptation mechanism also provides a means for speeding up the self-organizing controller scheme. The modification consists of changing the consequents of responsible rules, using a history of fired rules. Just as in the original scheme, there must be some knowledge of the time delays and time constants of the process. The simplified adaptation mechanism only changes the consequents of rules and, hence, (5.6) and (5.7) are not applied. The resulting controller can be interpreted as being based on Sugeno rules with constant consequents and a weighted sum (fuzzy mean) for defuzzification (considering rule consequents independently). If the adaptation delay is m samples, rules r_k that fired m samples ago, are described by:

$$r_k : \textbf{if } x_1 \text{ is } A_{1,k} \textbf{ and} \ldots \textbf{and } x_{N_X} \text{ is } A_{N_X,k} \textbf{ then } y = y_k[kT - mT]$$

and are changed to:

$$r'_k : \textbf{if } x_1 \text{ is } A_{1,k} \textbf{ and} \ldots \textbf{and } x_{N_X} \text{ is } A_{N_X k} \textbf{ then } y = y_k[kT - mT] + r[kT]$$

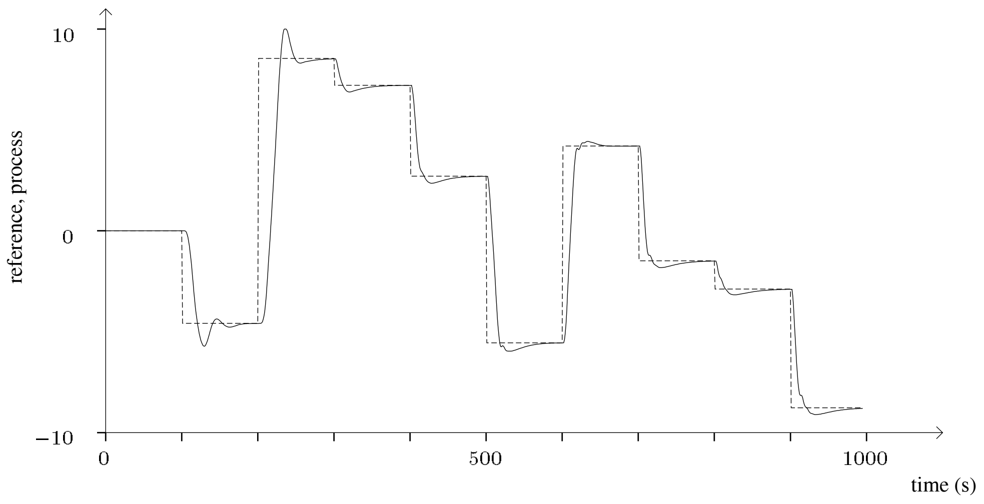
by the adaptation mechanism and where $r[kT]$ is given by (5.3). It is possible to update all rules that fired m samples ago, but the rules to be updated could also be selected by using a certain threshold above which the support value of a rule should be: adapting the "most" responsible rules. A simple example of the adaptation is show in figure 5.2. As can be seen from the results, the performance improves in time because of the adaptation of the elements of the look-up table and hence the rule consequents. In figure 5.3, the controller function is shown after 1000 samples. The controller function was initially completely "empty" (0 everywhere).

A similar adaptation scheme can be achieved from another point of view: extend Mamdani rules to have multiple weighted consequents (see the example on page 113) and let the adaptation mechanism change the weights of the rule consequents. Hence, similar results can be obtained as when using the method described earlier in this section. This method was described by Jager et al. (1991), van Kesteren (1991) and Jager (1992). A translation of the reinforcement value $r[kT]$ to changes of the consequent weights is necessary. The mechanism used is quite simple and an example of the adaptation of weighted rule consequents is shown in figure 5.4. Because a fuzzy-mean defuzzification method is used, the use of two weighted consequents is functionally equivalent to a constant numerical consequent which equals the defuzzification of those two weighted consequents. Hence, the result is the same as that obtained when using Sugeno rules, except that the user has a limited set of symbolic consequents to choose from, although a combination of (weighted) symbolic consequents is allowed. As stated in section 4.3.1, this can be seen as a trick, but it can provide a way to (still) express rules linguistically. This also works the other way around: after adaptation the numerical values in the look-up table can be translated into a combination of two weighted fuzzy sets on the output universe, thus providing a linguistic interpretation of the results after adaptation. Therefore, the multiple weighted consequents are not necessary on the level of implementation, but can be used for "user-interfacing".

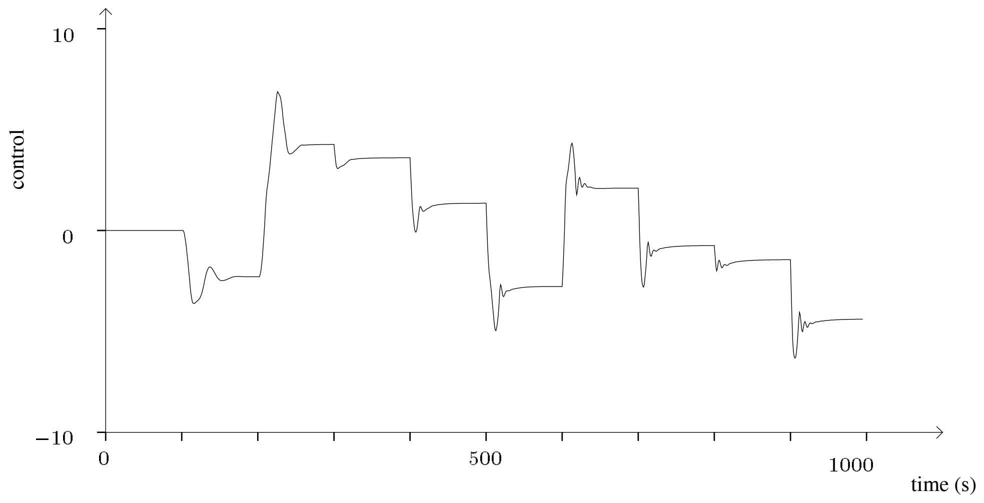
5.2 Fuzzy relations as associative memories

A fuzzy relation can be seen as "describing" a relation between variables and it can be used (i.e. the fuzzy relation) as associative memories (Harris et al., 1993). In this method a fuzzy "system" is used both as a model and a controller at the same time. Suppose the fuzzy model to be identified is based on rules like:

$$r_k : \textbf{if } x_1 \text{ is } A_{1_k} \textbf{ and } x_2 \text{ is } A_{2_k} \textbf{ then } y \text{ is } B_k$$



(a)



(b)

Figure 5.2: *Example of simplified self-organizing control. The simulated process has the transfer function $H(s) = 2e^{-s}/(10s + 1)$. The sampling rate T is 1 second and the adaptation is done by updating all rules that fired two samples ago. The rules had initially 0 as their consequent.*

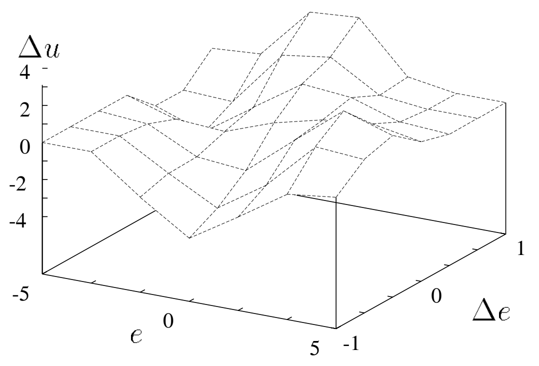


Figure 5.3: *Controller function after adaptation (*1000 *samples). It can be seen that situations which have not yet been encountered, for example (*$e = 5, \Delta e = 1$*), still have* $\Delta u = 0$ *as their corresponding table element (numerical rule consequent).*

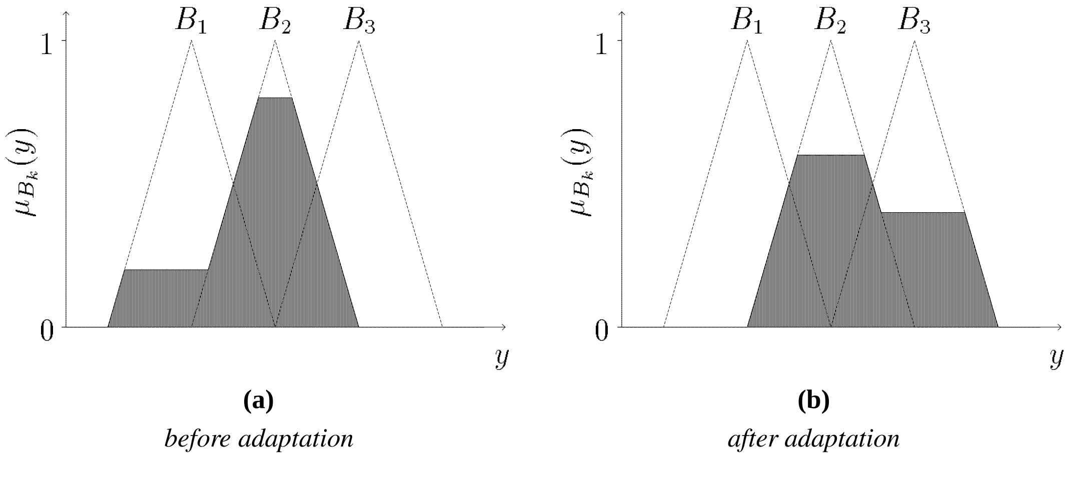


Figure 5.4: *Adaptation of consequent weights of rule with multiple weighted consequents. The consequent of the rule is changed from "*y *is* B_1 *(20%),* B_2 *(80%)"* **(a)** *to "*y *is* B_2 *(60%),* B_3 *(40%)"* **(b)**.

This implies that the model is based on two inputs and one output. The membership function of the fuzzy relation representing a set of these rules is represented by:

$$\mu_R(x_1, x_2, y) = \bigvee_{k=1}^{N_r} \mu_{R_k}(x_1, x_2, y) \tag{5.11a}$$

$$= \bigvee_{k=1}^{N_r} \mu_{A_{1,k}}(x_1) \ \wedge \ \mu_{A_{2,k}}(x_2) \ \wedge \ \mu_{B_k}(y) \tag{5.11b}$$

where other operators than the *min* operator for conjunction and implication are possible. When a simple first-order process is considered, the following signals can be chosen:

x_1 : control signal of m samples ago
x_2 : process output at previous sample
y : current change in process output

where m represents an appropriate approximation of delay times of the system, including delay times caused by zero-order hold circuits. New data, represented by tuples (x_1, x_2, y), can be "added" to the fuzzy model by updating the fuzzy relation R. Assume D is a fuzzy relation, 3-dimensional in this case, constructed with new measured data:

$$D[kT] = \text{fuzz}(x_1) \ \times \ \text{fuzz}(x_2) \ \times \ \text{fuzz}(y) \tag{5.12}$$

Only the elements of R which are "affected" by the new data D have to be considered. Thus, for all non-zero elements $\mu_D(x_1', x_2', y')$ in relation D, the corresponding elements $\mu_R(x_1', x_2', y')$ of fuzzy relation R are updated according to:

$$\mu_R(x_1', x_2', y') = \max(\lambda \, \mu_R(x_1', x_2', y'), \mu_D(x_1', x_2', y')) \tag{5.13}$$

where $\lambda \in [0, 1]$ is a forgetting factor. Now, the fuzzy relation R is constantly updated and can be used as an associative memory. This method considers the fuzzy relation R as a description of the "correlation" between x_1, x_2 and y. In other words, constructing a fuzzy relation R based on y as a function of x_1 and x_2, but using the resulting fuzzy relation R to infer a fuzzy representation of x_1 when x_2 and y are given. In other words, using R as a fuzzy relation which is based on rules like:

r_k : **if** $\hat{x}_2$ is A_{2_k} **and** $\tilde{y}$ is B_k **then** x_1 is A_{1_k}

where $\hat{x}_2$ is the current process output, assuming $m = 1$, or an estimation of the process output over $(m - 1)$ samples when a delay time is assumed. A more concrete example of such a rule is the following:

if the process output is *high*
and the desired process output change is *positive large*
then make the control signal *positive medium*

although the process identification was based on rules like:

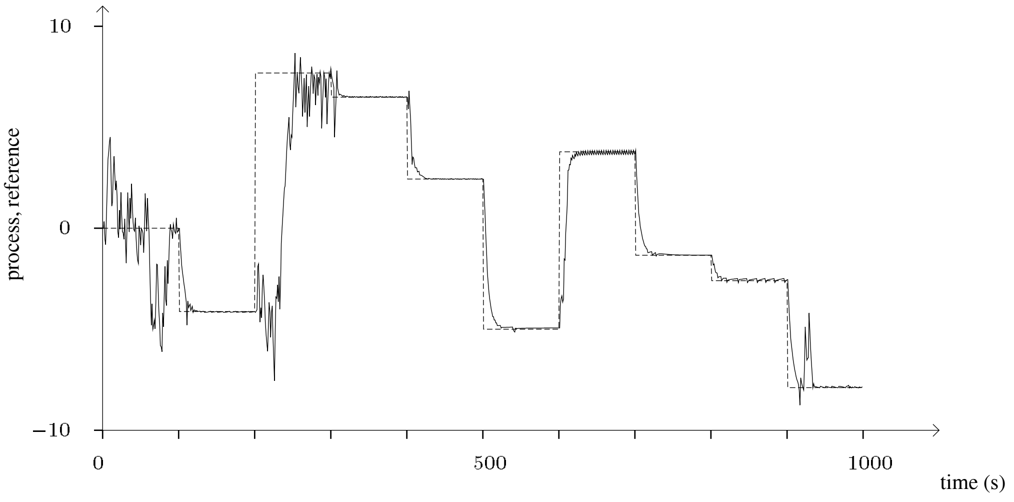
if the control signal was *positive medium*
and the process output was *high*
then the process output change is *positive large*

This is known as *causality inversion*; sometimes referred to as *linguistic inversion* (Harris et al., 1993). To apply this method, some type of reference model is needed, because deriving a control action needs the process output and a desired process output change $\tilde{y}$. When a delay time is considered, an estimation of future process outputs are necessary to derive a control action.

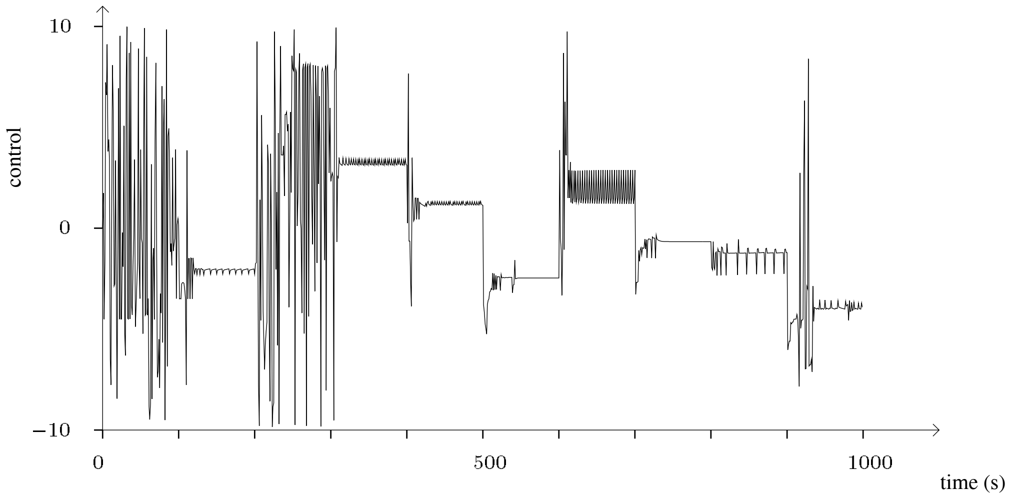
In figure 5.5 a run is shown of this method applied on a first-order process. A second-order reference model is used and no time delays are assumed. From figure 5.5, it can be seen that, although the system starts with a relation $\mu_R(x_1, x_2, y) = 0$, the derived control actions are able to control the process. Parts of the relation R which are not yet filled in appropriately (for example, the glitches after the last step change) can cause high frequencies in the control signal, which are undesired.

Now let us summarize some properties of this method. Besides providing an adaptive controller, a process model which can be used for analyzing the process is also obtained. It is clear that increasing the order of the model and thus of the controller results in the exponential growth of fuzzy relation R since it is based on the relational approach. Suppose this method is implemented by using discretized fuzzy relations where 11 discretizations of each domain are used, then for a simple first-order process model to begin with 1331 $(= 11^3)$ elements are necessary to store relation R. Using a second-order process model, which would probably be more appropriate for real problems, would require 14641 $(= 11^4)$ elements to store R. Note that 11 discretizations for a universe is not so many and thus the number of required memory elements given above is a rather optimistic estimation.

Another problem with this method, and probably the biggest objection to it, is the fact that a *causality inversion* takes place. The application in the case of processes with either no delay time or very small delay times can be done without much difficulties by using



(a)



(b)

Figure 5.5: *Experimental results using a fuzzy relation as associative memory: process output* **(a)** *and controller output* **(b)**. *The reference model is a second-order system.*

a suitable reference trajectory (see example in figure 5.5). However, when the process has a considerable time delay (or system dynamics which can be modeled by a large time delay), the control rules require a prediction of the process output over multiple samples. This poses the need for a process model. Note that this principle is also used in predictive control. If there is a "good" process model, adaptation methods which result in less severe calculational effort and memory requirements can be used. The fuzzy relation R can also be used to predict the process output, but this requires that relation R is already a good representation of the process. Of course, different possibilities can be combined: first using an existing model to predict future process outputs and then using the relation R when it is "reliable" enough. The advantage of using the relation R is that this relation can constantly be adapted to process parameter changes.

5.3 Adaptation by fuzzy supervisors

Besides the adaptation schemes described in the previous sections, another approach to adaptation can be distinguished: adaptation of a controller by a supervisor. In the field of fuzzy control different approaches to this type of adaptation are known. One possibility is to have a fuzzy supervisor which adapts a conventional controller. In the next section we will describe this method, where the conventional controller is a PID controller. More generally, hierarchical fuzzy control can be considered, where the controller consist of different modules: modules for direct control and modules to adapt other modules. An example of such a hierarchical fuzzy controller is briefly described in section 5.3.2.

5.3.1 Fuzzy supervised PID-control

In this section we describe fuzzy supervisory PID control, where a fuzzy system is used to supervise a conventional PID controller. Several examples of fuzzy supervisory PID control can be found in literature and in the following we address the methods described by Tzafestas and Papanikolopoulos (1990), van Nauta Lemke and De-Zhao (1985), van Nauta Lemke and Krijgsman (1991) and Li, Bruijn and Verbruggen (1994).

Tzafestas and Papanikolopoulos (1990) describe an approach which they refer to as "incremental fuzzy expert PID control". The system consists of a conventional discrete PID controller of which the proportional, integral and derivative gains, K_P, K_I and K_D respectively, are changed by a fuzzy supervisor each sampling time. The method has been developed to improve the characteristics of step responses with the idea that it should be supplemental to existing industrial PID controllers. The controller parameters are at each sampling instant defined by:

$$\begin{aligned}
K'_P[k] &= K_P + k_1\,\mathrm{CV}\{e[k], \Delta e[k]\}\\
K'_I[k] &= K_I + k_2\,\mathrm{CV}\{e[k], \Delta e[k]\}\\
K'_D[k] &= K_D + k_3\,\mathrm{CV}\{e[k], \Delta e[k]\}
\end{aligned}$$

where $CV\{e[k], \Delta e[k]\}$ is the value obtained from a look-up table; e and Δe are discretized to be used as indices for the look-up table. The scaling parameters k_1, k_2 and k_3 are used to tune the adjustments to K_p, K_i and K_d, respectively. The look-up table contains the knowledge used for adaptation and can be regarded as a nonlinear function of e and Δe. Hence, a similar controller function can be obtained by using Sugeno rules where the consequents of the rules are "local" PID controllers. Since the look-up table is constant, this method is in fact not adaptive in the sence that the control performance can be improved over time.

In the article by Tzafestas and Papanikolopoulos (1990), several improvements obtained when using this adaptation method are shown. An interesting point is that for the adaptations of the parameters of the PID controller the same look-up table is used. This entails that changes of the PID gains have the same sign (direction) when the scaling parameters k_1, k_2 and k_3 all have the same sign; in the experiments presented in their paper, k_1, k_2 and k_3 are positive. However, when supervising a PID controller, it is very well possible that in a certain situation the proportional gain should be increased and the integral gain should be decreased. In the following, a fuzzy PID supervisor is described which allows these adaptations.

The fuzzy PID supervisor described by van Nauta Lemke and De-Zhao (1985) and van Nauta Lemke and Krijgsman (1991) is also based on a fuzzy supervisor and a PID controller, but in their scheme the fuzzy supervisor has a rule base in which the fuzzy rules have consequents which address the gains of the PID controller. Hence, the fuzzy supervisor has three "outputs": ΔK_p, ΔK_i and ΔK_d which result in different parameters of the PID controller each sampling instant:

$$\begin{aligned}
K'_P[kT] &= K_P + \Delta K_P[kT]\\
K'_I[kT] &= K_I + \Delta K_I[kT]\\
K'_D[kT] &= K_D + \Delta K_D[kT]
\end{aligned}$$

The inputs of the fuzzy supervisor are the error $e[kT]$ and its first difference $\Delta e[kT]$. The controller scheme is shown in figure 5.6. In addition to the basic scheme, consisting of a PID controller and a fuzzy supervisor, an auto-tuning module is used to optimize the fuzzy supervisor. This auto-tuning module uses rules which classify the behavior of, for example, a step response to optimize the fuzzy supervisor. This classification is expressed as a performance measure and optimization is used to improve the performance measure. The optimization employs simulations. Hence, these adaptations of the fuzzy supervisor are performed on a larger time scale than the adaptation of the PID controller. When we consider the functionality of the controller scheme, without the auto-tuning module, a similar controller function can be implemented by Sugeno rules, of which the consequent is a “local” PID controller. However, this is no longer the case when different scaling of the input universes are considered which is possible when for each parameter of the PID controller a seperate rule base if used.

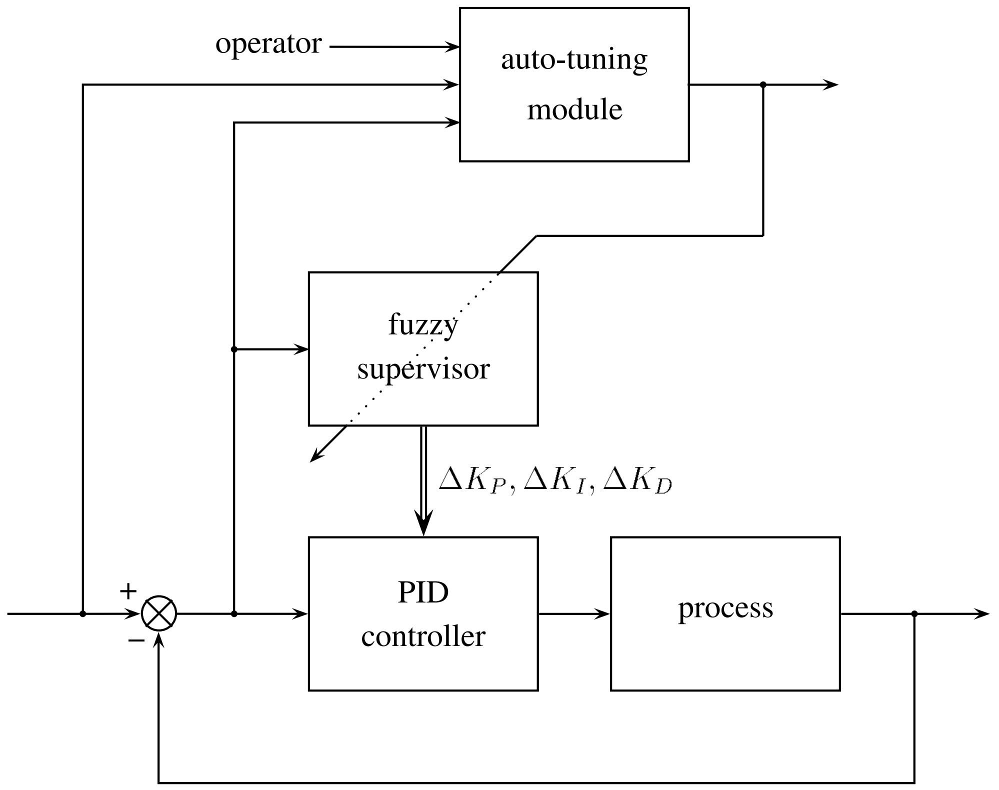


Figure 5.6: *Fuzzy PID supervisor as described by van Nauta Lemke and Krijgsman (1991).*

Adaptation schemes, where adaptation of a conventional controller is done each sampling instant, can be approximated by Sugeno rules. The advantage of using a supervisor is the fact that adaptations are expressed as changes of parameters of the conventional controller to be supervised, and these are directly derivable from the knowledge of an operator or process engineer. This also implies easier maintainability of the rule base of the supervisor. In case of Sugeno rules, local PID controllers have to be designed, which is less attractive than identifying an "overall" PID controller with changes to the controller parameters for certain situations.

Another interesting example of fuzzy supervisory control is the "expert supervisory control system for cascade PID control" reported by Li, Bruijn and Verbruggen (1994). The proposed control scheme is shown in figure 5.7. A pattern recognition module is used to extract performance indices from reponses of the closed-loop system to set-point changes and load disturbances. Additionally, test signals are used when the outputs of the master and slave loops are in steady state. The supervisory module is a small expert system with a fuzzy rule base, implemented by means of the RICE* software library (see appendix E for details on RICE). The rule base contains rules for tuning the master PID controller, rules for tuning the slave controller and rules to supervise the cooperation of the master and slave loop tuning mechanisms.

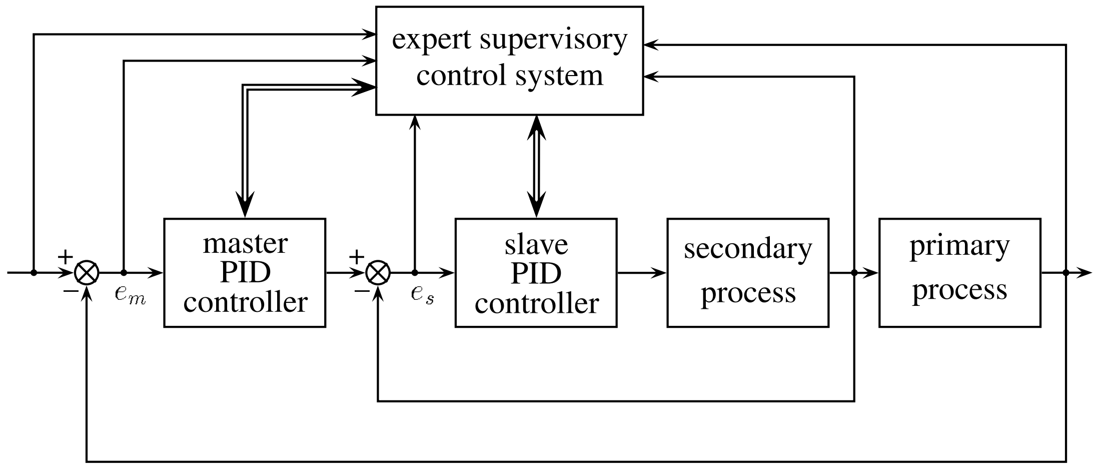


Figure 5.7: *Fuzzy expert suspervisory system for cascade PID control according to Li et al. (1994).*

*Routines for Implementing C Expert systems.

5.3.2 Adaptive fuzzy expert controller

A hierarchical adaptive fuzzy control system is described in this section. The system has modules for direct control, a module for in-line adaptations and a module for on-line adaptations by a supervisory module. The original system was based on classical sets (Jager et al., 1991) and a "fuzzy" version was described by Jager, Verbruggen, Bruijn and Krijgsman (1991) and Krijgsman (1993). In the following this hierarchical fuzzy control system is described in detail.

The knowledge in the control system is divided in several knowledge layers, each having its own specific task. Using these knowledge layers is primarily useful when applying the progressive reasoning principle (Lattimer Wright et al., 1986; Broeders et al., 1989). See also section 6.1.2 on this topic. In table 5.1 the tasks of the knowledge layers are given. The inference session of a certain knowledge layer is started when the inference of the "lower" knowledge layer is finished.

Table 5.1: *Knowledge layers and their tasks in fuzzy expert controller.*

layer	task
1	perform classifications of the measurements
2	reach setpoint using error and error difference
3	reach setpoint according to reference behaviour
4	perform in-line adaptations of controller
5	perform on-line adaptations of controller

The hierarchical control system consist of a module to perform preprocessing (layer 1), two direct control modules (layer 2 and 3) and two modules to perform adaptations (layer 4 and 5). The preprocessing module performs input filtering, pattern recognition to extract characteristics of set-point responses, and (fuzzy) classifications of these data, which are used in the higher layers. The relation between the control and adaptation modules are shown in figure 5.8. In the following paragraphs the working of the layers 2 to 4 are described.

Knowledge layer 2 is a basic fuzzy controller as described before in this thesis. The result is a new control signal (or control signal change) based on the error and error change. This is, in fact, a PI-like or PD-like fuzzy controller.

The behavior reference control module (layer 3) determines a control signal (change) based on a predicted behavior and a reference behavior. This predicted behavior is determined by a line through the current state $\boldsymbol{e}[kT]$ and the previous state $\boldsymbol{e}[kT-T]$. The reference

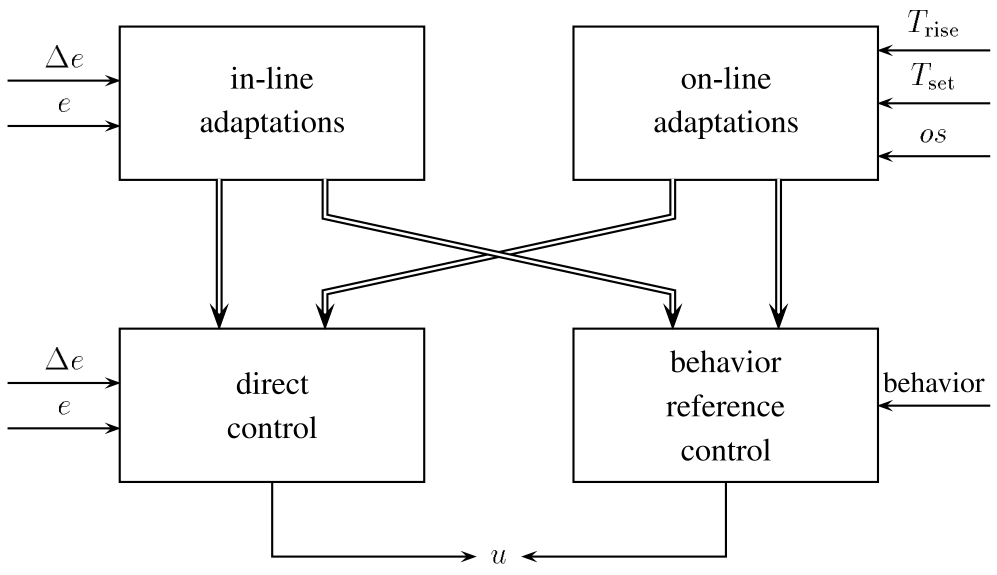


Figure 5.8: *Fuzzy expert control system (Jager et al., 1990; Jager et al., 1991; Krijgsman, 1993) with variables T_{rise} (rise time), T_{set} (settling time), os (overshoot), e (error), Δe (error change, first difference of error) and u (control signal).*

behavior is given by a line through the current state $\boldsymbol{e}[kT]$ and the origin (steady state) in the phase plane. This line represents a first-order reference model, but note that this model is different each sampling instant. This is the reason why this approach is referred to as "behavior reference control" instead of "model reference control". How the classification takes place is shown in figure 5.9. Compare this control strategy to taking a bend with a car at considerable speed: each moment the "ideal curve" is adjusted with respect to the current/predicted behavior.

In-line adaptations (layer 4) are (possibly) performed each sampling time. It consists of a zoommechanism, used to improve steady state control, of mechanisms to adapt the margins of the behavior reference control module, and of a mechanism to achieve a constant steady state value for the control signal in case of limit cycles. In addition to this, the margines (fuzzy sets) of layer 2 and 3 are adapted, for example, by means of scaling.

Knowledge layer 5 is used for on-line adaptation and is, in fact, the only supervisory module. Decisions of the knowledge base made in this layer are based upon the overall behaviour of the system and therefore works on a time scale which is several times the slowest time constant of the system. The on-line adaptations are performed to improve

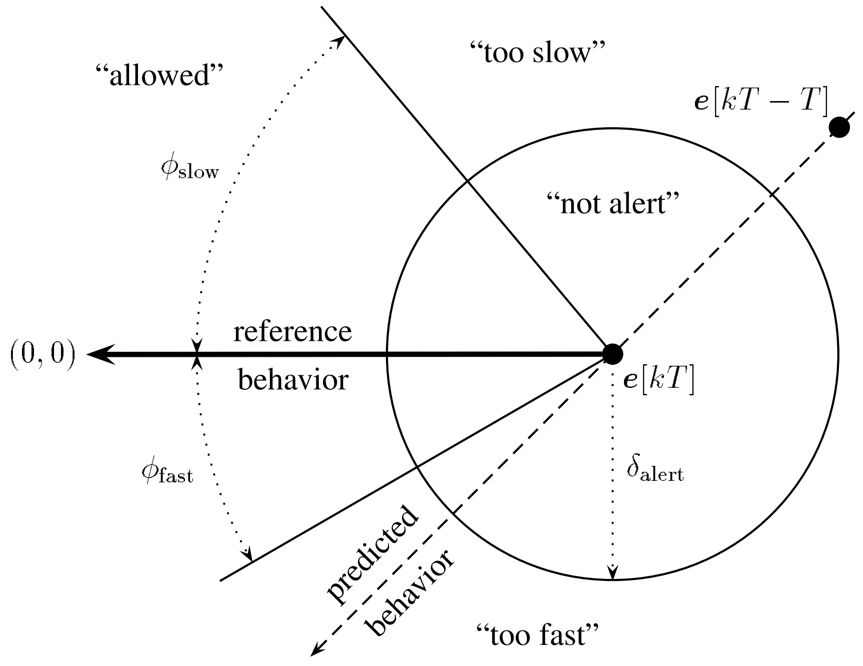


Figure 5.9: *Classification of "behaviour" in phase plane according to Jager et al. (1990).*

the overall performance of the controller. To achieve this, detections and classifications are made of the following characteristics of the step responses of the process: overshoot os, rise time (10-90%) T_{rise} and settling time T_{set}. Additionally, the delay time T_d is used.

The above-described hierarchical adaptive fuzzy control system is able to control various processes although a very simple (almost none) process model is required. Nonlinear processes as well as non-minimum phase processes can be controlled satisfactory after adaptation (Jager et al., 1990). Experimental results have shown that fuzzification of the expert system improves the performance of the control system (Jager et al., 1991). Implementations have been done in DICE[*] (Krijgsman et al., 1990; Krijgsman et al., 1991; Krijgsman and Jager, 1993a) and RICE (see appendix E).

[*]Delft Intelligent Control Environment.

5.4 Gradient-descent adaptation

More and more references to "fuzzy neural networks" or "fuzzy-neuro systems" can be found in literature. Only a few really use neural networks which are initialized by a fuzzy rule base. One example is the work of Horikawa, Furuhashi and Uchikawa (1993) which translates a fuzzy system into a neural network and use this neural network to learn a model. Most of the publications on fuzzy neural networks address adaptation of fuzzy systems based on a gradient-descent adaptation method for optimization. However, gradient-descent adaptation (optimization) is not specific to neural networks. Several authors have applied gradient-descent adaptation methods to fuzzy systems. What those methods have in common is that they minimize a similar objective function E as is done in case of the learning rules in neural networks, as well as in many other gradient-descent optimization methods:

$$E = \tfrac{1}{2}(y - \tilde{y})^2 \tag{5.14}$$

where $\tilde{y}$ is the reference for the fuzzy system output y. In the following subsections, these methods are described and discussed. It should be noted here that only one-layered systems are considered; no chaining of rules is assumed, just as in the previous chapter on fuzzy controllers. This is usually not the case in the field of neural networks, where in many cases multi-layered systems are used. An example of a multi-layered fuzzy system which is adapted by means of back propagation of errors is described by Uehara and Fujise (1993). This approach is not described in this section since the method is based on *fuzzy truth values* which is beyond the scope of this chapter. Fuzzy truth values are addressed in the next chapter. In this section we focus on fuzzy systems which are comparable to the fuzzy controllers as described in the previous chapter. It should be noted that the application of "fuzzy neural networks" is primarily in the field of modeling. However, this modeling is not restricted to the modeling of a process, but can also include the "learning" of the control behavior of a working controller, for example, a human operator.

5.4.1 The basic adaptation scheme

The first methods for adaptation (often called learning in this field) were rather basic. The fuzzy systems were in fact more regarded as simple neural networks, than as fuzzy systems reflecting human knowledge. A well known reference from literature is the work of Nomura, Hayashi and Wakami (1991). They consider triangularly-shaped membership functions for the inputs, Sugeno rules with constant consequent and the *product* operator for conjunction. The implication function is an arbitrary T-norm, because of the use of Sugeno rules and fuzzy-mean defuzzification. The rules have the following form:

r_k : **if** x_1 is $A_{1,k}$ **and** ... **and** x_i is $A_{i,k}$ **and** ... **and** x_{N_X} is $A_{N_X,k}$ **then** $y = b_k$

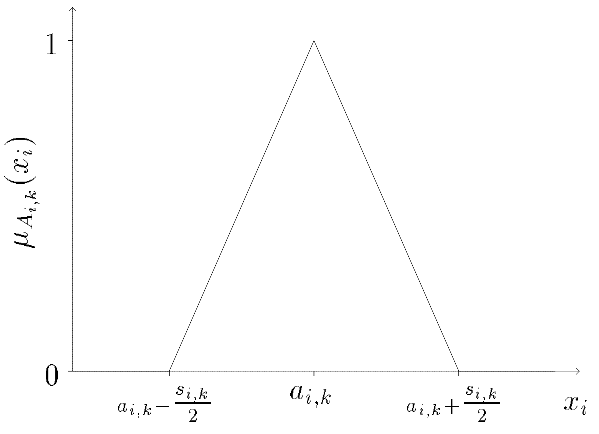


Figure 5.10: *Membership function used by Nomura et al. (1991) in gradient-descent adaptation of fuzzy system. Note that the parameters $a_{i,k}$ and $s_{i,k}$ are related to rule r_k.*

The membership functions are symmetrical and triangularly-shaped and thus characterized by their center $a_{i,k}$ and support $s_{i,k}$. Such a membership function is depicted in 5.10. Minimizing error function (5.14) leads to the following adaptation rules:

$$\Delta a_{i,k} = \frac{2K_a\beta_k(\tilde{y} - y)(b_k - y)\mathrm{sgn}(x_i - a_{i,k})}{\sum\limits_{k'=1}^{N_r} \beta_{k'} s_{i,k} \mu_{A_{i,k}}(x_i)} \tag{5.15a}$$

$$\Delta s_{i,k} = \frac{K_s\beta_k(\tilde{y} - y)(b_k - y)(1 - \mu_{A_{i,k}}(x_i))}{\sum\limits_{k'=1}^{N_r} \beta_{k'} s_{i,k} \mu_{A_{i,k}}(x_i)} \tag{5.15b}$$

$$\Delta b_k = \frac{K_b\beta_k(\tilde{y} - y)}{\sum\limits_{k'=1}^{N_r} \beta_{k'}} \tag{5.15c}$$

where K_a, K_s and K_b adaptation (learning) factors for $a_{i,k}$, $s_{i,k}$ and b_k respectively, and $\tilde{y}$ is the reference for output y. The adaptation according to (5.15) has been extended by Guély and Siarry (1993) for a number of cases:

- asymmetric triangular membership functions;
- *min* operator for *and* connective instead of *product* operator;
- non-constant consequents: standard Sugeno rules with linear functions of the inputs as consequents.

The adaptation of the membership functions of the fuzzy sets is based on a local error for each fuzzy rule. Initially equal fuzzy sets in different rules are adapted independently as if they were different fuzzy sets. However, the fact that the relations between fuzzy sets on the same universe of discourse are not accounted for in the update rules enables shifting of fuzzy sets along the universe of discourse, regardless of their linguistic meaning. Summarizing, the following can be concluded about this method:

- Initially equal membership functions can differ after adaptation, thus membership functions lose there linguistic meaning. For example, "small" has different membership functions in different rules, although defined on the same domain. The "swapping" of membership functions is possible, so logical interpretation can be lost after adaptation. For example, "small" in one rule can become bigger than "big" in another rule.
- The adaptation can cause a complete rule base on the linguistic level (see also section 3.2.5) which is incomplete on the numerical level (compare also section 4.6.3). In that case the adaptation results in fuzzy sets for an input which do not cover the complete universe. Typical for gradient-descent adaptation according to the above-given method is that discontinuities of the function to learn result in non-overlapping fuzzy sets (Guély and Siarry, 1993).

Considering these conclusions, it is clear that this method can lose linguistic interpretation of the rules after adaptation. Hence, if it is necessary to obtain a linguistically interpretable rule base after adaptation, this method is less attractive. Note that this does not mean that this method does not work properly; in literature, successful use of this learning method or similar methods has been reported by many authors, among which, Ishibuchi, Nozaki and Tanaka (1993), Chien and C.C.Teng (1993) and Jang (1993). In the following section the learning rules are modified to meet certain restrictions.

5.4.2 Restrictions on adaptation

A modification of the adaptation as described in the previous section has been reported by Bersini, Nordvik and Bonarini (1993). This FUNNY[*] system relates membership

[*]FUNNY is an acronym of *fuzzy* and *neural network*.

functions to the universe on which they are defined. In other words, fuzzy sets keep the same linguistic meaning in all rules: for example, "small" in one rule is the same as "small" in another rule when they are defined on the same universe of discourse. However, the other disadvantage noticed about the method according to Nomura et al. (1991) is not eliminated: after adaptation blank spots on a numerical level can occur. In figure 5.11, a membership function is depicted as used by Bersini et al. (1993). Note that the parameters defining these membership functions are not related to the k^{th} rule, but to the j^{th} fuzzy set on the universe of discourse of variable x_i.

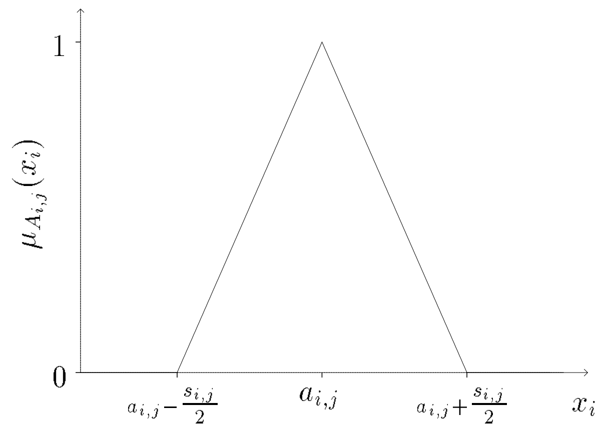


Figure 5.11: *Membership function used by Bersini et al. (1993) in gradient descent adaptation of fuzzy system. Note that the paremeters $a_{i,j}$ and $s_{i,j}$ are related to the j^{th} fuzzy set on the i^{th} input universe, not to rule r_k as in figure 5.10.*

Because the membership functions are not "related" to the rules but to the universe on which they are defined, minimizing error function (5.14) leads to slightly different adaptation rules than (5.15):

$$\Delta a_{i,j} = \frac{2K_a(\tilde{y} - y)\left(\sum_{k=1}^{N_{A_{i,j}}} \beta_k b_k - y \sum_{k=1}^{N_{A_{i,j}}} \beta_k\right) sgn(x_i - a_{i,j})}{\sum_{k=1}^{N_r} \beta_k s_{i,j} \mu_{A_{i,j}}(x_i)} \tag{5.16a}$$

$$\Delta s_{i,j} = \frac{K_s(\tilde{y} - y)\left(\sum_{k=1}^{N_{A_{i,j}}} \beta_k b_k - y \sum_{k=1}^{N_{A_{i,j}}} \beta_k\right)(1 - \mu_{A_{i,k}}(x_i))}{\sum_{k=1}^{N_r} \beta_k s_{i,j} \mu_{A_{i,j}}(x_i)} \tag{5.16b}$$

$$\Delta b_j = \frac{K_b \sum_{k=1}^{N_{b_j}} \beta_k(\tilde{y} - y)}{\sum_{k=1}^{N_r} \beta_k} \tag{5.16c}$$

where K_a, K_s and K_b adaptation (learning) factors for $a_{i,j}$, $s_{i,j}$ and b_j, and $\tilde{y}$ is the reference for output y. Note that $k = 1, \ldots, N_{b_j}$ refer to all rules that have b_j as consequent and $k = 1, \ldots, N_{A_{i,j}}$ refer to all rules which have $A_{i,j}$ in their premise. This method optimizes the fuzzy sets related to a universe, not the fuzzy sets related to a rule. Although this method yields results that are more related to a linguistic representation than the method according to Nomura et al. (1991), still the problem of possible blank spots on the numerical level due to nonoverlapping fuzzy sets remains. In the next section, a method is presented which eliminates this problem by maintaining fuzzy partitions on the input universes.

5.4.3 Maintaining fuzzy partitions

In the previous section we discussed the learning rules according to Bersini, Nordvik and Bonarini (1993), which obviates the ambiguous meaning of fuzzy labels, for example, after adaptation, "small" has the same meaning in all rules. However, the possibly resulting blank spots in the input-output mapping was not eliminated by their method. In this section, an adaptation method is given which maintains fuzzy partitions on the input universe(s). To achieve this, triangular-shaped fuzzy sets are used and, hence, the support of a fuzzy set is determined by the centers of adjacent fuzzy sets (see figure 5.12). This ensures that the fuzzy sets on a universe of discourse always form a fuzzy partition, keeping the sum of the membership functions equal to 1.

The learning algorithm could be seen as the gradient-descent adaptation of a look-up table, of which not only the elements are adapted, representing the rule consequents b_k, but also a (nonlinear) mapping of the index vectors of the look-up table. The index vectors contain

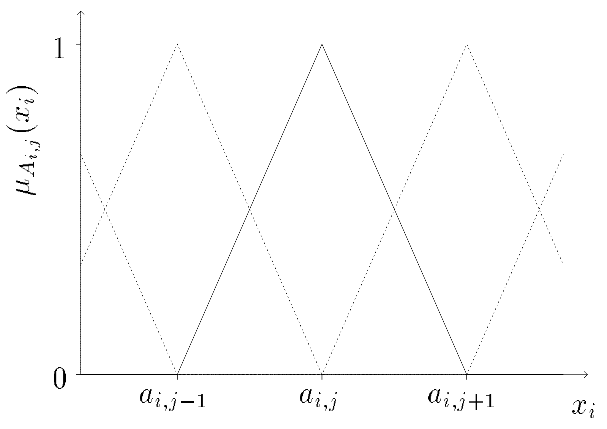


Figure 5.12: *Membership function used by modified gradient-descent adaptation of fuzzy system: maintaining fuzzy partitions on input universe(s).*

the centers $a_{i,j}$ of the membership functions. The learning rules are:

$$\Delta a_{i,j} = \begin{cases} \dfrac{K_a(y-\tilde{y})}{(a_{i,j}-a_{i,j-1})}\left[\dfrac{\mu_{A_{i,j}}(x_i)}{\mu_{A_{i,j-1}}(x_i)}\sum_{k'=1}^{N_{A_{i,j-1}}}\beta_{k'}(b_{k'}-y) - \sum_{k'=1}^{N_{A_{i,j}}}\beta_{k'}(b_{k'}-y)\right], & \text{if } a_{i,j-1} < x_i < a_{i,j} \\ \dfrac{K_a(y-\tilde{y})}{(a_{i,j}-a_{i,j+1})}\left[\dfrac{\mu_{A_{i,j}}(x_i)}{\mu_{A_{i,j+1}}(x_i)}\sum_{k'=1}^{N_{A_{i,j+1}}}\beta_{k'}(b_{k'}-y) - \sum_{k'=1}^{N_{A_{i,j}}}\beta_{k'}(b_{k'}-y)\right], & \text{if } a_{i,j} < x_i < a_{i,j+1} \\ 0, & \text{otherwise} \end{cases} \tag{5.17a}$$

$$\Delta b_k = K_b \beta_k (y-\tilde{y}) \tag{5.17b}$$

where K_a and K_b are adaptation (learning) factors for $a_{i,j}$ and b_k, and $\tilde{y}$ is the reference for the output y. Note that $k = 1, \ldots, N_{A_{i,j}}$ refer to all rules which have $A_{i,j}$ in their premises. Like the previously described methods, additional restrictions are necessary to prevent the swapping of fuzzy sets. In this case the centers of fuzzy sets can be swapped and adaptation should be restricted by maintaining:

$$a_{i,1} \; < \; a_{i,2} \; < \; \ldots \; < \; a_{i,N_{A_i}} \tag{5.18}$$

where N_{A_i} is the number of fuzzy sets for the i^{th} input universe. This is important when the fuzzy system is represented by a (look-up) table and the "inference" is performed by interpolation. In appendix C, the complete derivation of this adaptation method is given. The main characteristic of this method is that it optimizes the fuzzy partitions on the input universe(s), not the fuzzy rules. This entails that blank spots, for example due to discontinuities in the function to learn, are not possible.

5.5 Comparison with other "learning" systems

In literature, there are a number of publications which discuss the resemblance between adaptive fuzzy systems and other adaptive (or learning) single-layer feedforward systems. Examples of these systems are the *radial basis function network* (RBFN) and the *cerebellar model articulation controller* (CMAC). The similarities and differences between these systems and fuzzy systems are discussed in the following subsections. Here we do not focus on the adaptation or learning method, but on the functional equivalence between the different systems.

5.5.1 Relation to radial-basis function networks

This section contains a comparison of a fuzzy system (controller or model) with a radial basis function network (RBFN). A *radial basis function network* performs a mapping from inputs $\boldsymbol{x}$ to output y by means of radial basis functions Φ_k:

$$y = f(\boldsymbol{x}) = \sum_{k=1}^{N_r} w_k \Phi_k(\| \boldsymbol{x} - \boldsymbol{c}_k \|) \tag{5.19a}$$

$$= \sum_{k=1}^{N_r} w_k \Phi_k(r_k) \tag{5.19b}$$

where $\boldsymbol{c}_k$ is the center of radial basis function Φ_k and $r_k = \| \boldsymbol{x} - \boldsymbol{c}_k \|$ is the distance of input data $\boldsymbol{x}$ from center $\boldsymbol{c}_k$. For a detailed description of RBFNs see the work of Brouwn (1993). To make the comparison of a one-layered (without chaining of rules) fuzzy system with a RBFN is trivial when it is noticed that both have a one-layered structure and use a weighted sum to obtain the output(s) (Krijgsman, 1993). When we consider a fuzzy system with Sugeno rules (see also section 4.3.2) with constant consequents:

r_k : **if** x_1 is $A_{1,k}$ **and** . . . **and** x_{N_X} is $A_{N_X,k}$ **then** y is b_k

then the output of the system, obtained by a weighted sum (like fuzzy-mean defuzzification), can be written as:

$$y = f(\boldsymbol{x}) = \frac{\sum_{k=1}^{N_r} \left\{ \prod_{i=1}^{N_X} \mu_{A_{i,k}}(x_i) \right\} b_k}{\sum_{k=1}^{N_r} \prod_{i=1}^{N_X} \mu_{A_{i,k}}(x_i)} \tag{5.20}$$

To make the two systems equal, there are some restrictions to be met for both methods. When identifying these restrictions it should be noted that the radial basis functions operate on $\| \boldsymbol{x} - \boldsymbol{c}_k \|$, while the membership functions operate on one of the N_X input universes. When the premise of a rule is constructed as a relation, a Cartesian product $A_{1,k} \times \cdots \times A_{N_X,k}$ based on the product operator, the resulting relation should be a function of $\| \boldsymbol{x} - \boldsymbol{c}_k \|$. Hence, the relation between the membership functions and the radial basis functions can be given by:

$$\prod_{i=1}^{N_X} \mu_{A_{i,k}}(x_i) = \Phi_k(\| \boldsymbol{x} - \boldsymbol{c}_k \|) \tag{5.21}$$

A trivial solution for (5.21) is to choose the membership functions as:

$$\mu_{A_{i,k}}(x_i) = a_k^{(x_i - c_{i,k})^2 / s_k} \tag{5.22}$$

since the product of these functions will give:

$$\prod_{i=1}^{N_X} \mu_{A_{i,k}}(x_i) = \prod_{i=1}^{N_X} a_k^{(x_i - c_{i,k})^2 / s_k} \tag{5.23}$$

$$= a_k^{\sum_{i=1}^{N_X} (x_i - c_{i,k})^2 / s_k} \tag{5.24}$$

$$= a_k^{\| \boldsymbol{x} - \boldsymbol{c}_k \|^2 / s_k} \tag{5.25}$$

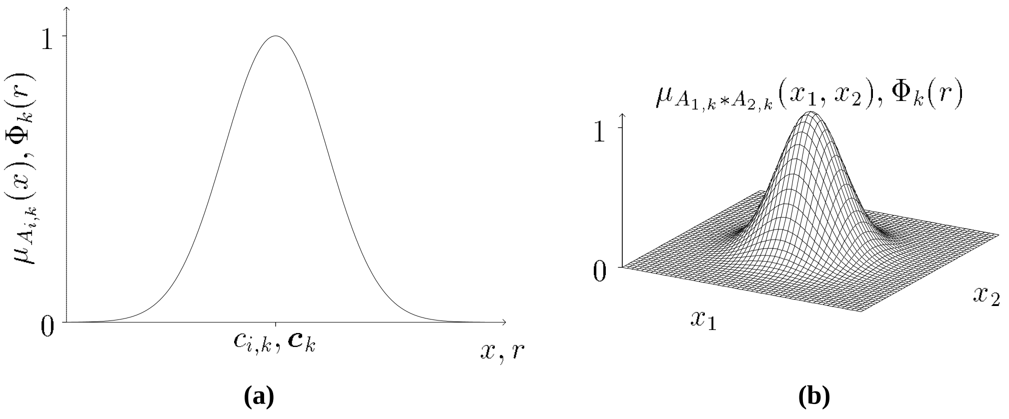


Figure 5.13: *Gaussian radial basis function $\Phi_k(r)$ with center $\boldsymbol{c}_k$ or fuzzy set $A_{i,k}$ with center $c_{i,k}$ is shown in* **(a)**. *A 3-dimensional plot is given in* **(b)**: $r = \| \boldsymbol{x} - \boldsymbol{c}_k \|^2$.

which can be regarded as a radial basis function that, with $r_k = \| \boldsymbol{x} - \boldsymbol{c}_k \|$, is defined by:

$$\Phi_k(r) = a_k^{r_k^2/s_k} \tag{5.26}$$

A well-known example is the Gaussian radial basis function, shown in figure 5.13:

$$\Phi_k(r) = \exp(-r_k^2/\rho^2) \tag{5.27}$$

Hence, a_k and s_k are the same for all radial basis functions: $a_k = e$ and $s_k = -\rho^{-2}$. The resemblance between a fuzzy system based on Gaussian membership functions and an RBFN based on Gaussian basis functions has already been reported by Jang and Sun (1993), but other functions are possible as long as (5.21) is satisfied. More specifically, considering (5.21), it can be noted that the membership functions are required to have unbounded support. The membership functions defined by (5.22) have unbounded support, but are not defined to be convex or not. The membership functions will be convex when $s_k < 0$. With respect to the corresponding RBF, this means that its value tends to zero outside the region where the function is centered. This property is normally desired when using RBFNs, because RBFs without this property may grow to infinity and cause stability problems (Brouwn, 1993).

Thus far, only the "weights" of the weighted sum, which is performed by both, have been considered. When considering the output, it follows from (5.19), (5.20) and (5.21) that:

$$w_k = b_k \left[\sum_{k'=1}^{N_r} \prod_{i=1}^{N_X} \mu_{A_{i,k'}}(x_i) \right]^{-1} \tag{5.28a}$$

$$= b_k \left[\sum_{k'=1}^{N_r} \Phi_{k'}(r_{k'}) \right]^{-1} \tag{5.28b}$$

$$= \nu_k b_k \tag{5.28c}$$

and thus the weights w_k in an RBFN correspond to weighted consequents $\nu_k b_k$ of the rules in the fuzzy system. The weights ν_k are not constant, but depend on $\boldsymbol{x}$. Hence, the translation of a fuzzy system based on Sugeno rules, given by:

$$y = f(\boldsymbol{x}) = \frac{\sum_{k=1}^{N_r} w_k \Phi_k(r_k)}{\sum_{k=1}^{N_r} \Phi_k(r_k)} \tag{5.29}$$

differs from the original RBFN algorithm (5.19). This means that w_k are functions of $\boldsymbol{x}$ and thus the learning rules for RBFNs are no longer valid. To choose $w_k = b_k$ will cause the membership functions problems, since $w_k = b_k$ would result in the restriction:

$$\prod_{i=1}^{N_X} \mu_{A_{i,k}}(x_i) = \frac{\Phi_k(r_k)}{\sum_{k'=1}^{N_r} \Phi_{k'}(r_{k'})} \tag{5.30}$$

which does not have a straightforward solution for $\mu_{A_{i,k}}(x_i)$. In fact, two contradicting criteria can be defined. On the one hand the fuzzy sets on each domain should be a fuzzy partition, resulting in the denominator in (5.29) being equal to 1, resulting in $w_k = b_k$. On the other hand, the fuzzy sets should have unbounded support to fulfill (5.21). Concluding the comparison of RBFNs and fuzzy systems, we can state that fuzzy systems are not equal to RBFNs, although the similarities between the two approaches are obvious. The main difference between a RBFN and a comparable fuzzy system is the normalization in the fuzzy system by means of a weighted sum. Kecman and Pfeiffer (1994) refer to a fuzzy system as a "Soft RBFN" as opposed to a (classical) RBFN in order to stress the normalization by means of a weighted sum. When the normalization is also part of the RBFN, the functional equivalence is obvious (Jang and Sun, 1993).

Because of the similarities, one can think of a fuzzy system which is used to initialize an RBFN. The RBFN can then be adapted by using the adaptation methods available for RBFNs. This provides a mechanism to initialize the RBFN with already available knowledge. Now the question arises as to whether it is possible to translate a RBFN into a fuzzy system? In this case each radial basis function has to be translated into a fuzzy rule, where the projection of the radial basis function on the input universes can be used to obtain the fuzzy sets for each rule. The problem, however, is how to obtain the consequent of the fuzzy rule. Simply taking w_k from the RBFN is not sufficient since these values are obtained by the learning rules based on (5.19), while the fuzzy system is based on (5.29). Hence, a direct translation is not possible, but the radial RBFN can be used to extract a rule base. In that case, the consequents of the rules have to be computed from the values w_k of the RBFN and an additional value to compensate for the influence of overlapping RBFs. This assumes that the centers of the RBFs represent characteristic points of the nonlinear input-output mapping. If the RBFN also incorporates the normalization as discussed before, a translation to and from a fuzzy system is trivial, since in that case the systems are functionally equal.

5.5.2 Comparison with generalized CMAC

In this section, a comparison is made between a fuzzy system (controller or model) and the *cerebellar model articulation controller* (CMAC), which was introduced by Albus (1975a,b) and used by many others for modeling or control (Krijgsman, 1993; Brown and Harris, 1991). A brief description of the CMAC algorithm and a generalized version of it can be found in appendix D; for those not familiar with CMAC, it is advised to read this before continuing. In literature relations or similarities between and combinations of CMAC and fuzzy systems have been described several times, for example, by Nie and Linkens (1993) and Pedrycz (1993). In this section it is investigated whether these similarities also induce (partial) equivalence.

Albus used a binary generalization function (also known as a *kernel function*) to generalize the input vector of the CMAC algorithm. This generalization function can be seen as a classical set. Hence, the generalization function can also be implemented by a fuzzy set, as described, among others, by Brown (1990), Krijgsman (1993), Krijgsman and Jager (1993b). The generalized version of CMAC allows generalization functions which are not two-valued $(f(\boldsymbol{x}) \in \{0,1\})$, but are real-valued in the interval $[0,1]$ (compare fuzzy sets). We refer to this modified CMAC algorithm as the *generalized cerebellar model articulation controller* (GCMAC). The rest of this section will assume GCMAC, because this enables us to compare fuzzy systems and the (G)CMAC algorithm. The memory cells in the table are updated by:

$$c_g[k+1] = c_g[k] + \lambda\,\Phi(\boldsymbol{x}[k], \boldsymbol{x}_g)\,(\tilde{y}[k] - y[k]) \tag{5.31a}$$

$$y[k] = \frac{\sum_{g=1}^{N_g} \Phi(\boldsymbol{x}[k], \boldsymbol{x}_g[k]) c_g[k]}{\sum_{g=1}^{N_g} \Phi(\boldsymbol{x}[k], \boldsymbol{x}_g[k])} \tag{5.31b}$$

where $g = 1, \ldots, N_g$ is used to denote the input vectors within the generalization of $\boldsymbol{x}[k]$ and:

$c_g[k]$	: g^{th} memory cell in table at time k addressed by generalization
$\boldsymbol{x}[k]$	: input vector at time k
$\boldsymbol{x}_g$	: input vector within generalization of $\boldsymbol{x}$
$y[k]$	: output at time k
$\tilde{y}[k]$	: desired output at time k
λ	: learning factor
$\Phi(\cdot)$	: kernel function for generalization

When we regard the GCMAC table with memory cells as being a rule base, we can notice that for every element in the table a rule exists, and due to the generalization of the inputs the rule base has a great amount of redundancy: an input combination is covered by many rules due to the great number of overlapping membership functions on the input universe. As shown in section 4.6.1.2, the overlapping of more than two membership functions on a universe will result in smoothening/filtering of the controller hypersurface. In GCMAC this phenomena explicitly appears when the GCMAC system has learned one data point with a learning factor equal to one. It seems reasonable that if the GCMAC is given that same input, the learned output would be produced. This, however, is not the case because of the “fuzzy” generalization of the inputs. In the following this will be shown.

For the learning of one data vector $\boldsymbol{x}$ with learning factor equal to 1 and assuming $c_j[0] = 0$ (and thus $y[0] = 0$), this will result in:

$$c_g[1] = \Phi(\boldsymbol{x}, \boldsymbol{x}_g)\tilde{y} \tag{5.32}$$

and the recall using the same input vector will produce:

$$y[1] = \frac{\sum_{g=1}^{N_g} \Phi(\boldsymbol{x}, \boldsymbol{x}_g) c_g[1]}{\sum_{g=1}^{N_g} \Phi(\boldsymbol{x}, \boldsymbol{x}_g)} \tag{5.33a}$$

$$= \frac{\sum_{g=1}^{N_g} \Phi^2(\boldsymbol{x}, \boldsymbol{x}_g) \tilde{y}}{\sum_{g=1}^{N_g} \Phi(\boldsymbol{x}, \boldsymbol{x}_g)} \tag{5.33b}$$

$$= \tilde{y} \frac{\sum_{g=1}^{N_g} \Phi^2(\boldsymbol{x}, \boldsymbol{x}_g)}{\sum_{g=1}^{N_g} \Phi(\boldsymbol{x}, \boldsymbol{x}_g)} \tag{5.33c}$$

Hence, the recall will reproduce the learned output when the kernel function is a binary one ($\Phi(\cdot) \in \{0, 1\}$), since this entails $\Phi^2(\cdot) = \Phi(\cdot)$. In the case of "fuzzy" kernel functions the reproduction $y[1]$ of $\tilde{y}$ does not equal $\tilde{y}$, as can be seen from (5.33), since $\Phi^2(\cdot) \leq \Phi(\cdot)$. When the GCMAC table is sufficiently filled with learned data points, this can be seen as a set of fuzzy rules with a lot of overlapping fuzzy sets and thus the consequents of the fuzzy rules (data points) do not correspond with the actual output of the system. This system which is highly interactive because of the (many) overlapping membership functions is therefore closer to neural networks than a fuzzy system. A straightforward linguistic interpretation is not possible.

Note that the above shown "differences" between a GCMAC system and a fuzzy system does not lead to conclusions about the use of "fuzzy" kernel functions. Experiments showed that the use of fuzzy kernel functions provide better learning capabilities than binary kernel functions in many cases (Krijgsman and Jager, 1993b; van Kats, 1993). However, this topic is beyond the scope of this thesis.

5.6 Conclusions and remarks

In this chapter, we have focussed on adaptive fuzzy control. Four main types of fuzzy adaptive systems were considered. In the following we briefly summarize these types of adaptive fuzzy control and give conclusions.

Adaptive fuzzy controllers based on the *self-organizing controller* was proposed by Procyk and Mamdani (1979) and is described in section 5.1. These controllers adapt the fuzzy relation or rule base based on a "reinforcement" value derived from a performance measure. The performance measure, mostly stored in a performance table, can be regarded as an implicit reference model. The adaptation is local and hopefully results in global optimization. A simplified method is based on the adaptation of (the elements of) a look-up table and described in section 5.1.4. Inference is performed by means of interpolation between the elements of this look-up table.

Adaptive fuzzy control according to the *associative memory* approach, described in section 5.2. In this approach a fuzzy relation is used to model the process and the same fuzzy relation is used to derive control actions (causality inversion). The advantage of this approach is that control can be improved and a process model is obtained at the same time. It is important to note that in the case of delay times a process model is needed to estimate future process outputs in order to derive the current control action.

Adaptation of controllers by *fuzzy supervisors*, for example, fuzzy supervised PID control was described in section 5.3.1. Many systems consisting of a fuzzy supervisor and a conventional control algorithm can be expressed as a fuzzy controller which is based on Sugeno rules. However, this is less "user-friendly" that a conventional control algorithm and a supervisor which supervises and adapts this controller. In section 5.3.2 a hierachical system was described where several fuzzy systems performed in-line and on-line adaptations of direct and indirect controller parts.

Adaptation of fuzzy systems by means of *gradient-descent optimization* algortihms was described in section 5.4. This type of adaptation is often used in the field of neural networks and has led to the term "fuzzy-neuro systems" and "fuzzy neural networks". Many proposed learning rules will decrease the linguistic interpretability of the fuzzy system after learning. Simplification of the learning rules which maintain fuzzy partitions can prevent this (section 5.4.3).

In addition to describing and analysing the different approaches to adaptive fuzzy control, comparisons were made with other learning systems: *radial basis function network* and *CMAC*. It was shown that these systems are different from fuzzy systems in section 5.5, although they are similar and in some cases even functionally equivalent. The main difference between a fuzzy system and an RBFN is the fact that normally no normalization is performed in an RBFN. A GCMAC memory/table can be regarded as a rule base with many rules which have overlapping premises (section 5.5.2). This overlapping of premises yields a "filtering" of the control hypersurface as was shown in section 4.6.1.2.

Despite the different approaches and numerous publication on this topic, there are almost no applications reported as far as we know. Applications of fuzzy controllers are numerous, see the previous chapter, but adaptive fuzzy control is probably not yet “accepted” like fuzzy control is.

6

Fuzzy logic in knowledge-based systems

The use of fuzzy logic in knowledge-based systems is addressed in this chapter. Considering the numerous applications of fuzzy control, one can state that it is a more or less accepted type of control. However, applications of approximate reasoning or derived reasoning schemes can barely be found in literature. In our opinion approximate reasoning can provide a user-friendly knowledge representation and a reasoning method which can model human reasoning in higher level expert systems. In the field of control, this entails planning, scheduling and plant-wide supervision. To apply approximate reasoning in these fields, (more) software tools should be developed and become available for industry. For these reasons this chapter is part of this thesis.

Various approaches to reasoning based on fuzzy logic are described and discussed in this chapter. Other uncertainty management systems, such as, for example *belief functions*[*] (Shafer, 1976), *Bayesian networks* (Pearl, 1990) and the *certainty factor model* (Buchanan and Shortliffe, 1984), are not addressed in this chapter. For comparisons of those methods

[*]For those familiar with belief functions: possibility distributions can be regarded as *consonant plausability functions* (Shafer, 1987).

with possibility theory, the reader is referred to the work of Dubois and Prade (1990), who have made extensive studies on the subject. Uncertainty management based on possibility and necessity measures (see section 6.2.2) is not considered in detail. Also, fuzzy inference schemes in which only discrete universes are considered, are not addressed in this chapter.

In this chapter, we focus on knowledge-based systems where reasoning is used for retrieving (new) information from knowledge, represented by production rules, objects, frames, etc., and data. An overview is given of different approaches to the application of fuzzy logic in knowledge-based systems (for control). Besides describing proposed methods from literature, problems and disadvantages are discussed, and possible solutions or improvements are suggested. Section 6.1 addresses knowledge-based systems and various knowledge representations including some additional requirements in the case of application of knowledge-based systems in real-time control. A summarizing section with conclusions ends this chapter.

6.1 Knowledge-based systems for control

Today, the use of knowledge-based systems for control is growing gradually. In the increasing complexity of high level control, for example plant-wide control, the need is emerging for knowledge-based techniques (Verbruggen and Åström, 1989). Also in low-level control, the proliferation of knowledge-based systems is increasing. Examples are the applications of fuzzy control in consumer electronics and process control. Knowledge-based techniques in low-level control do not eliminate the need for conventional control methods, but add an alternative or supplementary technique which can provide a way to be economically more “efficient”; in other words: a working control system for less money. Knowledge-based techniques applied on higher levels, like supervisory and plant-wide control, provide techniques to embed human knowledge and reasoning, and thus assist or even (partly) replace human supervision.

In the following sections some characteristic knowledge-based techniques are discussed. Also requirements of knowledge-based systems related to real-time control are addressed.

6.1.1 Knowledge representation

Knowledge-based systems, or more specifically, expert systems, have a knowledge base which contains the knowledge of some specific domain. Roughly speaking, an expert system consist of an inference mechanism, a knowledge base, containing the domain knowledge, and a data base, containing the inferred data. Several types of knowledge representations are possible. In the following list we follow Mylopoulos and Levesque (1984) who distinguish the following four types of knowledge representations (Luger and Stubblefield, 1989) to which, if considered necessary, comments are added.

1. **Logical knowledge representations** represent knowledge by expressions in formal logic. The best known logical representation scheme is first-order predicate logic. The logical programming language PROLOG[*] can be used to implement logical representations schemes. Fuzzy extensions of PROLOG are FPROLOG[†] by Martin, Baldwin and Pilsworth (1987) and FRIL[‡] (Baldwin and Zhou, 1984).

2. **Procedural knowledge representations** represent knowledge as a set of instructions for solving a specific problem. Production systems, based on *if-then* rules, are examples of a procedural knowledge representation, since a rule can be interpreted as a procedure to achieve a goal. The rule describes the procedure to perform the following: solve the premise in order to solve the goal (consequent). Within this class fall also the fuzzy controllers, as discussed in chapter 4. A well-known rule-based system shell is CLIPS[§] and a "fuzzy version" is nowadays available as FUZZYCLIPS (NRC/KSL, 1994). An extension of LISP[¶] based on fuzzy sets and logic is FLISP[||] (Sosnowski, 1990).

3. **Network knowledge representations** represent knowledge by means of graphs in which nodes represent object or concepts, and relations or associations between those objects and nodes are embodied by arcs connecting the nodes in the graph. Examples of network representations are: *semantic networks* and *conceptual graphs*.

   Semantic networks use nodes to represent concepts and arcs to represent relations between concepts. The arcs are labeled to denote the relation between the nodes which are connected by arcs. The first program implementing semantic networks was reported by Quillian (1968).

   Conceptual graphs are graph representations in which the nodes are either concepts or conceptual relations (Sowa, 1984). The arcs in a conceptual graph are not labeled and arcs can only connect concept nodes to conceptual relation nodes and vice versa. This type of graph is close to semantic networks.

4. **Structured knowledge representation schemes** are extensions of the above described network representation schemes. Nodes are extended to complex data struc-

[*]PROLOG stands for PROgramming in LOGic and was developed by Alain Colmeraurer and associates at the University of Marseilles in the early 1970s.

[†]FPROLOG stands for Fuzzy PROLOG.

[‡]FRIL stands for Fuzzy Relational Inference Language.

[§]CLIPS is an acronym for *C Logical Inference Production System.*

[¶]LISP stands for LISt Processing and was developed by John McCarthy in the late 1950s.

[||]FLISP stands for Fuzzy LISP.

tures with identifiable slots with values, which can be numerical, symbolic, other structures, procedures, etc. Examples of these types of knowledge representation schemes are summarized in the following list.

- *Frames*, which are a further development (and improvement) on semantic networks (van de Ree, 1994). A frame describes an object or concept by slots which consist of facets, and which on their turn have values. The concept of a frame is close to or, better, included by, the concept of an object, which is described further on.
- *Scripts* (Schank and Abelson, 1977) describe a (standardized) sequence of events for a particular domain. In natural language understanding, scripts are used to organize a knowledge base in terms of a situation which is understandable by the system (Luger and Stubblefield, 1989).
- *Objects*, as used in object-oriented programming languages, are data structures like frames, of which the slots can represents variables or procedures (encapsulation). Relations between objects are represented through class hierarchies (inheritance) and message passing between objects. Examples of object-oriented languages are SMALLTALK (Goldberg and Robson, 1983), OBJECTIVE C[*], C++ or object-oriented environments built upon LISP. Also the programming language ADA is extended to provide object-oriented programming techniques. Commercially available systems shells for real-time control are, for example, G2 (Gensym Corporation) and COGSYS (COGSYS Ltd.). In the field of fuzzy logic, only a few authors have reported work in this field, for example, Leung and Wong (1992) and Rine (1991).

Although many approaches to knowledge representation can be distinguished, in this thesis we mainly consider rule based knowledge representation. To restrict ourselves to rules only does not limit the derived results from it, since many properties of the object-oriented knowledge representation can be described by means of rules. In general, the methods employed for inference of rules and object-oriented systems are similar with respect to the underlying knowledge.

6.1.2 Real-time control requirements

The application of knowledge-based systems in a real-time environment requires extra facilities. A real-time (expert) systems is subject to requirements concerning correct results, but also concerning the time it takes to achieve these results. O'Reilly and Cromarty (1986) state the following:

[*]Objective C is an object-oriented extension to the C programming language with many features from SMALLTALK.

A hard real-time system is a system in which correctness of the solution not only depends on the logical results of a computation or reasoning process, but also depends on the time in which these results are produced.

If the system is an algorithm which has to be run on a computer, then the application of faster hard- and software can solve many of the real-time requirements. However, when the control system can be interrupted and real-time reaction is required, in addition to the continuous performance of tasks, then the system should be able to solve conflicts in a well-defined manner. In many systems, this is accomplished by giving priorities to tasks.

Another important aspect of real-time systems is the fact that information can change over time. For example, measurement are updated every sampling time in a control system. This requires that data structures include temporal information, and that the inference mechanism is capable of reasoning about past, present and future of data, using the temporal information of data structures. This is known as *temporal reasoning* (Krijgsman, 1993).

Nonmonotonic reasoning is a type of reasoning which deals with data which loses its validity due to some external event. This requires a so-called *truth-maintenance system* (TMS), which updates the data base when necessary. The TMS keeps track of the reasoning and dependencies between derived data, and is responsible for keeping the data base consistent while the growth of the data base is nonmonotonic.

A reasoning mechanism which is specific to real-time expert systems is *progressive reasoning* (Lattimer Wright et al., 1986). This reasoning mechanism is based on a layered knowledge base. The inference proceeds layer by layer, and the knowledge should be structured in such a way that the inference "higher" layer is based on the results obtained by inference from a "lower" layer. Thus, the more knowledge layers that are inferred, the more refined the results become. How many knowledge layers can be inferred depends on the time available and the initial conditions. Hence, when there is little time, the obtained results will not be "optimal", but progressive reasoning is based on the assumption that *some result is better than no result at all*. The application of this type of reasoning is described by Jager et al. (1990, 1991) and an adaptive fuzzy control system based progressive reasoning was described in section 5.3.2.

6.2 Possibility theory

An introduction to *possibility theory* is given in this section. This theory is based on representing (un)certainty by means of *possibility distributions*. Possibility theory is based on fuzzy set theory and was introduced by Zadeh (1978). Major research on this topic has been done by a number of researchers, among which, Dubois and Prade (1988) and Yager (1983).

6.2.1 Possibility distributions

Zadeh (1978) introduced the concept of *possibility distributions*. Possibility distributions are part of the theory of approximate reasoning (see section 6.3) and Zadeh (1981b) states that *the imprecision that is intrinsic in natural languages is, in the main, possibilistic rather than probabilistic in nature*. In the following subsections, the concept of a possibility distribution, the relations to fuzzy sets and different interpretations of propositions and their influence on possibility distributions are addressed.

6.2.1.1 The concept of a possibility distribution

In this section, the concept of a possibility distribution is explained. Possibility distributions associate a degree of possibility with each element of the domain the possibility distribution is defined on. Consider the proposition *the ship is large*. This proposition can imply that we are talking about the length of the ship and is denoted by:

$$\mathrm{R}(\mathrm{A}(\mathrm{ship})) = \textit{large}$$

where:

$$\mathrm{A}(\mathrm{ship}) \equiv \textit{the length of the ship}$$

and R denotes the (fuzzy) restriction which is placed on the length of the ship; R should not be confused with a fuzzy relation denoted by R in previous chapters. Thus, $A(\cdot)$ is used to denote the implied property *length* of the ship and $R(\cdot)$ denotes the restriction which is posed on the length of the ship. To state this more formally, let us consider the following proposition:

$$U \text{ is } A$$

which induces a possibility distribution Π_U that, by definition, is equal to the fuzzy set A, denoted by:

$$\Pi_U = A \tag{6.1}$$

This is known as the *possibility assignment equation* (Zadeh, 1981b). This possibility distribution implies that the possibility of $U = x$ is given by the membership x has in A:

$$\pi_U(x) \triangleq \text{poss}\{U = x\} = \mu_A(x),\ x \in X \tag{6.2}$$

where $\mu_A(x)$ is a membership function for linguistic label A and $\pi_U(x)$ is the possibility distribution function of possibility distribution Π_U. This means that the possibility that $U = x$, is equal to the membership x has in A. Further on in this section, we give Dubois and Prade's view (1991), which includes different interpretations of the possibility assignment equation. When we consider the proposition *the ship is large*, from before, then the following can be derived:

$$\pi_{\text{ship}}(x) = \mu_{\text{large}}(x)$$

where $\pi_{\text{ship}}(x)$ is the possibility distribution function and $\mu_{\text{large}}(x)$ is the membership function of the fuzzy set *large*.

6.2.1.2 Fuzzy sets and possibility distributions

After the previous section, the reader is probably wondering what the difference between a fuzzy set and a possibility distribution is. To clarify this consider the following (classical) set V containing a number of valves:

$$V = \{\text{valve}_1, \text{valve}_2\}$$

When the valves connected to a certain pipe are considered, there is a difference between the following cases:

$$V_C = V, \text{ valves connected to the pipe}$$
$$\Pi_C = V, \text{ valves } \textbf{possibly} \text{ connected to the pipe}$$

Hence, in the case of V_C, we know for certain that both valves are connected to the pipe, and in the case of Π_C, we know that the valves are possibly connected to the pipe. The latter can even be interpreted in a way that only one of the two valves is connected to the pipe (exclusive-or operator). This simple example shows that possibility distributions are used to represent uncertainty.

Whether a fuzzy set can be interpreted as a possibility distribution is usually clear from the context (Zadeh, 1981b). However, it should be stated that fuzzy sets can be used in the following ways (Dubois and Prade, 1994a):

- Fuzzy sets that **model the gradual nature of properties**. This is how fuzzy sets discussed in the previous chapters were used. The higher the membership x has in a fuzzy set A, the more true it is that *x is A*. Here, the fuzzy set models the linguistic uncertainty. Consider the fuzzy proposition *the water level is high*. Since there is no unique definition of the linguistic label *high*, the fuzzy set used to model *high* represent the uncertainty, or, better, ambiguity, of this linguistic label. When the level is precisely known, the "truth" of *the water level is high* can be determined.

- Fuzzy sets that **represents incomplete states of knowledge**, where the fuzzy set is in fact a possibility distribution; see the possibility assignment equation (6.1) on page 190. When considering that it is only known that *x is A* and thus x is not precisely known, then the fuzzy set A can be considered a possibility distribution: the higher the membership $\mu_A(x')$, the higher the possibility that $x = x'$. Taking the same example that was used previously, and only knowing that *the water level is high*, the fuzzy set *high* represents a possibility distribution, since the higher the membership $\mu_{\text{high}}(x')$, the higher the possibility that the water level is indeed x'.

Thus, on the one hand, fuzzy sets can be used to model linguistic uncertainty, on the other hand fuzzy sets can be used to model an incomplete state of knowledge, in which case the fuzzy sets plays the role of a possibility distribution (Dubois and Prade, 1994a).

6.2.1.3 Different interpretations of propositions

Recall the possibility assignment equation (6.1) given before. According to Dubois and Prade (1991), one can distinguish two ways of specifying a possibility distribution, which results in two derived versions of (6.1). They state that the proposition *U is A* can be interpreted as:

- U is A is *possible*
  This means that all values x which are in A are completely possible:

  $$\forall x \in A,\ \pi_U(x) = 1$$

  and the possibility of values x outside A is unspecified. This lead to the inequality:

  $$\forall x \in X,\ \mu_A(x) \leq \pi_U(x) \tag{6.3}$$

  This is less restrictive than the possibility assignment equation (6.1) on page 190. Here the membership function $\mu_A(x)$ is a lower bound of the possibility distribution function $\pi_U(x)$. A possibility qualification which is not complete can be represented by:

U is A is *α-possible*

When interpreted as *it is at least α-possible that U is A*, this leads to the following restriction for the possibility distribution function:

$$\forall x \in X,\ \min(\mu_A(x), \alpha) \leq \pi_U(x) \tag{6.4}$$

Note that the possibility qualification is numerical, for example, $\frac{3}{4}$-possible. If $\alpha = 1$, the statement *U is A is possible* is obtained.

- U is A is *certain*
  The proposition implies that values for x outside A are impossible and thus:

  $$\forall x \notin A,\ \pi_U(x) = 0$$

  The possibility of values x within A are unspecified and this leads to the inequality:

  $$\forall x \in X,\ \mu_A(x) \geq \pi_U(x) \tag{6.5}$$

  An incomplete certainty qualification is represented by:

  U is A is *α-certain*

  resulting in the following restriction for the possibility distribution function when it is interpreted as *it is at least α-certain that U is A*:

  $$\forall x \in X,\ \max(\mu_A(x), 1 - \alpha) \geq \pi_U(x) \tag{6.6}$$

  The proposition *U is A is certain* is obtained if $\alpha = 1$.

Hence, the possibility assignment equation as in (6.1) can be interpreted as a special case, representing a proposition *x is A is possible and certain*. This proposition restricts the possibility distribution $\pi_U(x)$ from below and above and thus:

$$\overbrace{\mu_A(x)}^{\text{see (6.3)}} \leq \pi_U(x) \overbrace{\leq \mu_A(x)}^{\text{see (6.5)}} \ \Rightarrow\ \pi_U(x) = \mu_A(x) \tag{6.7}$$

These different interpretations of a proposition are used in section 6.4.1, where different interpretations, in terms of (conditional) possibility distributions, of if-then rules are described. In the next section, the principles of minimum and maximum specificity are explained. The two principles are opposite and are closely connected to the different interpretations of a proposition.

6.2.2 Possibility and necessity measures

In possibility theory, a measure of possibility $\Pi(\cdot)$ and a measure of necessity (certainty) $\mathrm{N}(\cdot)$ are defined. Assuming a Boolean proposition, x *is* A, where A is thus a classical set, and an actual state of knowledge represented by the possibility distribution $\pi_u(x)$, these measures are defined by:

$$\Pi(A) = \sup_{x \in A} \pi_u(x) \tag{6.8a}$$

$$\mathrm{N}(A) = \inf_{x \in A} (1 - \pi_u(x)) = 1 - \Pi(\overline{A}) \tag{6.8b}$$

If we consider conjunctions or disjunctions of (Boolean) propositions, the following can be derived:

$$\Pi(\bigcup_i A_i) = \max_i \Pi(A_i) \tag{6.9a}$$

$$\mathrm{N}(\bigcap_i A_i) = \min_i \mathrm{N}(A_i) \tag{6.9b}$$

and also, although less restrictive:

$$\Pi(\bigcap_i A_i) \leq \min_i \Pi(A_i) \tag{6.9c}$$

$$\mathrm{N}(\bigcup_i A_i) \leq \max_i \mathrm{N}(A_i) \tag{6.9d}$$

Only if A_i are independent (subsets of different universes) the equalities in the previous equations are valid. In figure 6.1, a simple example is given which shows that $\Pi(A_1 \cap A_2) < \Pi(A_1) \cap \Pi(A_2)$ in the case where A_1 and A_2 are dependent.

To here, only Boolean propositions have been considered. When this is extended to the fuzzy proposition x *is* A, where A is a fuzzy set, the possibility and necessity measures from (6.8) become:

$$\Pi(A) = \sup_x \min(\mu_A(x), \pi_u(x)) \tag{6.10a}$$

$$\begin{aligned} \mathrm{N}(A) &= \inf_x \max(\mu_A(x), 1 - \pi_u(x)) \\ &= 1 - \sup_x \min(1 - \mu_A(x), \pi_u(x)) \\ &= 1 - \Pi(\overline{A}) \end{aligned} \tag{6.10b}$$

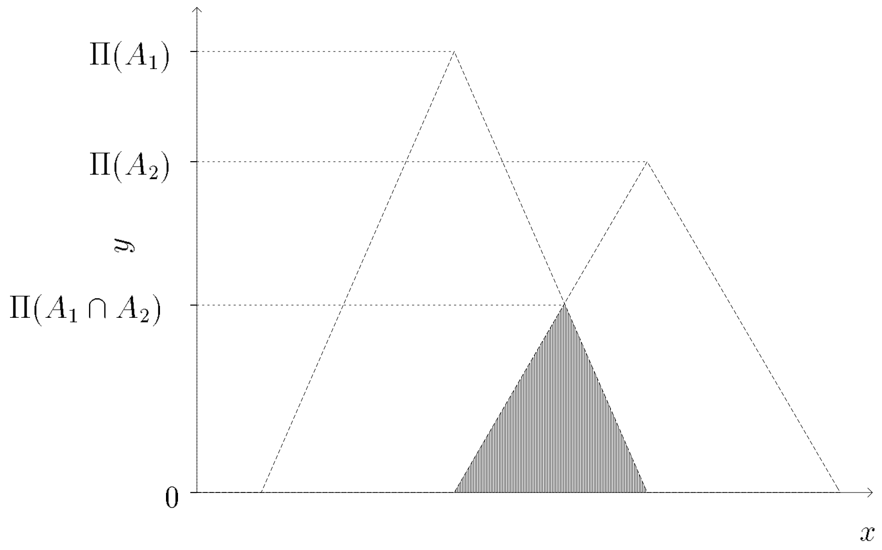


Figure 6.1: *Example showing that* $\Pi(A_1 \cap A_2) < \Pi(A_1) \cap \Pi(A_2)$ *if* A_1 *and* A_2 *are dependent.*

Now, suppose a fuzzy proposition *x is A* and some data *x is A'* with membership function $\mu_{A'}(x)$. Using (6.1) and (6.10) we can obtain:

$$\Pi(A; A') = \sup_x \min(\mu_A(x), \mu_{A'}(x)) \tag{6.11a}$$

$$\begin{aligned} N(A; A') &= \inf_x \max(\mu_A(x), 1 - \mu_{A'}(x)) \\ &= 1 - \sup_x \min(1 - \mu_A(x), \mu_{A'}(x)) \\ &= 1 - \Pi(\overline{A}; A') \end{aligned} \tag{6.11b}$$

where $(A; A')$ is used to denote x *is* A, given x *is* A'. To summarize, note the following. The possibility measure represents the matching between a proposition and the available information (data), the necessity measure represents the certainty of the proposition considering the available information. These measures can be used to represent the matching between propositions and data. As shown in section 6.4.1, these measures play an important role in different types of rules and can be used in performing (local) inference.

6.2.3 Principles of minimum and maximum specificity

In this section, the *principle of minimum specificity* and the *principle of maximum specificity* are described. In the previous section, two different interpretations of a proposition *x is A* were discussed. The principles of minimum and maximum specificity are each meant for one of the interpretations and are addressed in the following subsections.

6.2.3.1 Principle of minimum specificity

The *principle of minimum specificity* (Yager, 1983) states that the possibility distribution that represents a combination of pieces of information from different sources, is given by the least specific possibility distribution that satisfies the set of constraints induced by those pieces of information (Dubois and Prade, 1991).

When we consider the proposition U *is* A, where A has the membership function $\mu_A(x)$, then any possibility distribution $\pi_u(x) \leq \mu_A(x)$ is allowed, but the least specific possibility distribution is given by $\pi_u(x) = \mu_A(x)$. In section 6.2.1.3 this type of qualification was denoted as x *is* A *is certain*. Applying the principle of minimum specificity to a proposition like x *is* A *is certain and* y *is* B *is certain* leads to the assumption that the joint possibility distribution is given by:

$$\pi_{u,v}(x,y) \leq \min(\pi_u(x), \pi_v(y)) \tag{6.12}$$

since $\forall y \in Y,\ \pi_{u,v}(x,y) \leq \pi_u(x)$ and $\forall x \in X,\ \pi_{u,v}(x,y) \leq \pi_v(y)$. When the equalities are valid, the variables x and y are *noninteractive*[*] (Zadeh, 1975), resulting in $\pi_{u,v}(x,y) = \min(\pi_u(x), \pi_v(y))$.

When the variables x and y are interactive, then in fact $\pi_{u,v}(x,y) < \min(\pi_u(x), \pi_v(y))$ for some values (u,v), which does not mean that (6.12) is not correct, but that it gives an upper bound of the possibility distribution $\pi_{u,v}(x,y)$ and thus it might be less informative than the "actual" joint possibility distribution.

6.2.3.2 Principle of maximum specificity

In the previous section, the principle of minimum specificity was explained for certainty qualifications: *x is A is certain* (see section 6.2.1.3). If possibility qualifying propositions

[*]The noninteractivity in possibility theory plays the same role as independency in probability theory. However, stochastic independency does not lead to bounding properties as noninteractivity does, since stochastic independency assumes an actual absence of correlation while noninteractivity expresses a lack of knowledge about the correlation (Dubois and Prade, 1991).

are considered, then the principle of minimum specificity is to be reversed and leads to what Dubois and Prade (1991) refer to as the *principle of maximum specificity*. This entails that the most specific possibility distribution which represent the proposition *x is A is possible* is given by $\pi_u(x) = \mu_A(x)$, although any possibility distribution which satisfies $\pi_u(x) \geq \mu_A(x)$ is correct. Considering the statement *U is A is possible and V is B is possible* will lead to a joint possibility distribution given by:

$$\pi_{u,v}(x, y) \geq \max(\pi_u(x), \pi_v(y)) \tag{6.13}$$

since $\forall y \in Y,\ \pi_{u,v}(x, y) \geq \pi_u(x)$ and $\forall x \in X,\ \pi_{u,v}(x, y) \geq \pi_v(y)$. Hence, the combination is performed in a union-like manner (Dubois and Prade, 1991). When the variables x and y are noninteractive $\pi_{u,v}(x, y) = \max(\pi_u(x), \pi_v(y))$. When x and y are interactive, this equality represents the upper bound for $\pi_{u,v}(x, y)$, but is less informative than the "actual" joint possibility distribution.

6.2.4 Rules and conditional possibility distribution

In possibility theory, an implication is considered to be a *conditional possibility distribution*. Consider the following rule:

if U is A **then** V is B

From this, a conditional possibility distribution $\Pi_{v|u}$ can be induced with conditional possibility distribution function:

$$\pi_{v|u}(x, y) = \mathrm{I}(\mu_A(x), \mu_B(y)) \tag{6.14}$$

where the fuzzy sets A and B are interpreted as possibility distributions. The implication *I* is a fuzzy implication as discussed in section 3.2.2.

In (6.14), an assignment is used and hence, the conditional possibility distribution $\pi_{v|u}(x, y)$ equals the fuzzy relations representing fuzzy rules as described in section 3.2.2. This assumes the possibility assignment equation to be at work. However, in section 6.2.1.3 we discussed how the possibility assignment equation can be loosened, resulting in different interpretations of a proposition. Applying this in the case of rules will result in different interpretations of rules and, hence, result in different restrictions for the conditional possibility distributions representing the rules. Different interpretations of rules are addressed in section 6.4.

6.3 Approximate reasoning

The concept of possibility distributions was explained in section 6.2.1. Approximate reasoning was introduced by Zadeh (1979) and can be seen as an application of possibility theory: vague and imprecise knowledge is represented by possibility distributions. New knowledge can be inferred by applying the rules of possibility theory in combination with existing knowledge and data. This section describes the main aspects of approximate reasoning as well as practical considerations.

The methodology of approximate reasoning can be regarded as the propagation of fuzzy restrictions. Zadeh (1981b) defines a *fuzzy restriction* as follows:

> *a fuzzy restriction is a fuzzy set which serves as an elastic constraint on the values that may be assigned to a variable. A variable which is associated with a fuzzy restriction or, equivalently, with a possibility distribution, is a fuzzy variable.*

Various aspects of approximate reasoning are described in the following sections: different types of reasoning, referred to as *reasoning modes* and different representations of knowledge, referred to as *translation rules*.

6.3.1 Reasoning modes

Within the framework of approximate reasoning, Zadeh (1992) distinguishes different *modes of reasoning*:

1. **Categorial reasoning** represents a reasoning mode in which the premises are fuzzy propositions, but contain no fuzzy quantifiers like, for example, “most”, or fuzzy probabilities like, for example, “likely”. An example is the following:

   the level of tank 1 is high
   the level of tank 2 is much higher

   From this, a possibility distribution for the level of tank 2 can be derived by:

   $$\pi_{\text{level(tank 2)}}(y) = \sup_x \min(\pi_{\text{much higher}}(x, y), \pi_{\text{level(tank 1)}}(x)) \tag{6.15}$$

   where x and y denote the levels of tank 1 and 2, respectively. This is, in fact, the composition of relations as described in section 2.4.2. There, an example is given, which uses a relation *approximately equal* in combination with a fuzzy number.

2. **Syllogistic reasoning** enables inference in the case premises contain fuzzy quantifiers. A well known example is (Zadeh, 1992):

   most Swedes are blond
   most blond Swedes are tall

   from which it can be derived that:

   most^2 *Swedes are blond and tall*

   where *most*^{2} represents an intensification of *most*, hence, a smaller proportion. This intensification is not based on the type of intensification as used in the case of hedges, but is based on fuzzy arithmetic. Considering *most* a fuzzy number on the unit interval $[0, 1]$, *most*^{2} is the square of *most*.

3. **Dispositional reasoning** underlies nonmonotonic and default reasoning. An example is given by:

   (almost all) birds fly

   By means of dispositional reasoning it can be derived that, when it is known that an animal is a bird, that this animal can fly to a high degree of certainty. In the field of control, one could think of assumptions about the states of a process which are considered true as long as they are not "proven" otherwise.

4. **Qualitative reasoning**[*] uses knowledge represented by if-then rules. Many examples of this type of reasoning can be found in literature, especially in the field of fuzzy control (see chapter 4).

In expert systems, these different types of reasoning are normally not used, except the qualitative reasoning mode. In systems based on classical logic, only based on true and false, it is not possible to apply the various types of reasoning as listed above, since they do not provide sufficient mechanisms to incorporate them. For example, the case of dispositional reasoning requires the embedding of all exceptions of a default rule in the knowledge base of the system, since there are only two possibilities: a bird does fly or it does not. To allow exceptions to this, all exceptions should be listed. In approximate reasoning, these different types of reasoning provide a better modeling of human-like reasoning; this includes, for a great deal, natural language understanding, since reasoning by humans is for a great deal based on imprecise and uncertain knowledge and data.

[*]The terminology *qualitative reasoning* should not be confused with the pure symbolic reasoning with the same name proposed by Kuipers (1986).

6.3.2 Translation rules

To translate statements in natural language into a system able to perform approximate reasoning, Zadeh (1981b) distinguishes the following basic types of *translation rules*:

1. **Modification rules**, which modify the meaning of a fuzzy proposition. This is basically the application of hedges as described in section 2.2 and is thus not further addressed.

2. **Composition rules**, which are used to combine fuzzy propositions by means of *and* and *or*. Also *if-then* statements fall in this category. This is what was described in chapter 2 and 3 and is not further addressed.

3. **Quantification rules**, which are used to interpret statements Q *U's are A*, where Q is a fuzzy quantifier. An example is *most chemical processes are hard to model*, where *most* is the fuzzy quantifier. This type of translation rule is discussed in section 6.3.2.1.

4. **Qualification rules**, which are used for fuzzy propositions with an additional linguistic qualification. Three types of qualifications are considered: *truth qualifications* (. . . is true), *probability qualifications* (. . . is likely) and *possibility qualifications* (. . . is possible). This type of translation rule(s) is described in section 6.3.2.2.

In the following two sections, the latter two types are discussed in more detail, since they have not previously been addressed before in this thesis. This is because only the first two, and mainly the second, are found in fuzzy control applications. The reason why the latter two types of translation rules are addressed in this thesis is that they can be usefull in "higher" levels of knowledge-based control systems, for example, plant-wide supervisory control systems. In low-level direct control, where controller inputs and outputs are numerical and no or little interfacing with humans is necessary, these two types of translation rules are less usefull.

6.3.2.1 Quantification rules

Quantification rules are rules in which a *fuzzy quantifier* is used. The general form is as follows:

Q U's are A

where Q is a fuzzy quantifier, for example, *most*, *few*, etc. The representation of this type of knowledge is based on the notion of the *power* of a fuzzy set, also known as the *cardinality* a fuzzy set:

$$|A| = \sum_i \mu_A(x_i) \tag{6.16}$$

provided the support of fuzzy set A is finite. The cardinality of a fuzzy set is used to determine the *proportion* of A in U (Zadeh, 1981b):

$$\text{prop}(A/U) = \frac{|A \cap U|}{|U|} \tag{6.17}$$

When the support of the fuzzy sets is not finite, a density function is needed to determine the proportion:

$$\text{prop}(A) = \int_X \rho(x)\mu_A(x)dx \tag{6.18}$$

where $\rho(x)$ is a density function defined on X. The proposition Q *U's are* A induces a possibility distribution on the density function $\rho(x)$, resulting in (Zadeh, 1981b):

$$\pi(\rho) = \mu_Q(\int_X \rho(x)\,\mu_A(x)\,dx) \tag{6.19}$$

where $\rho(x)dx$ is the proportions of U's whose value lies in the interval $[x, x + dx]$.

Suppose the proposition *most chemical processes are hard to model*. Here the following can be retrieved:

Q :	*most*
U :	chemical process
A :	hard to model

When a proposition like this is contained by the knowledge base, Q can be seen as the proportion of chemical processes that are hard to model:

$$\text{prop}(A/U) = \textit{most}$$

However, when the knowledge base contains the proportion of chemical process with respect to processes in general, and the proportion of process that are hard to model with respect to process in general, then a query “are chemical processes hard to model?” can be evaluated by the expert system using (6.17). It is clear that this type of knowledge is not the type that is used in direct control. Its place is in the higher levels of knowledge within an expert system, for example, management and decision support systems.

6.3.2.2 Qualification rules

In the following, *qualification rules* are described. Qualification rules are statements or propositions which contain *linguistic qualifiers*. The following types can be distinguished: *truth-qualifying*, *probability-qualifying* and *possibility-qualifying* statements, and they are described hereunder.

Truth-qualifying statements

Truth-qualifying statements are propositions like x *is* A *is* τ, where τ is a linguistic truth value. Here τ can be seen as a fuzzy truth value. Fuzzy truth values (also referred to as fuzzy truth qualifiers), are fuzzy sets defined in the truth value space (Zadeh, 1981b):

$$\mu_\tau(t) \in [0,1], \text{ with } t \in [0,1] \tag{6.20}$$

Examples are *fairly true*, *absolutely false*, *very true*, etc. The fuzzy truth value *true* is, by definition, given by:

$$\mu_{\text{true}}(t) = t \tag{6.21}$$

When a statement like x *is* A is interpreted as x *is* A *is true*, then *true* is defined by (6.21). The examples given in the paragraph above (like *fairly true*) can be represented by $m(true)$, where m is a linguistic modifier or hedge (see section 2.2 for hedges) or the negation. The membership function of *false* is given by the negation of *true*:

$$\begin{aligned}\mu_{\text{false}}(t) &= \mu_{\text{not(true)}}(t)\\ &= 1 - \mu_{\text{true}}(t)\\ &= 1 - t\end{aligned} \tag{6.22}$$

Hence, *true* and *false* are each others complement, which is intuitively correct. A statement x *is not* A *is* τ, where τ is a fuzzy truth value, can be converted to x *is* A *is ant(*τ*)*, where *ant(*τ*)* is the antonym of τ and is defined by (Zadeh, 1981b):

$$\mu_{\text{ant}(\tau)}(t) = \mu_\tau(1-t) \tag{6.23}$$

This means that a statement like *x is not A is fairly true* is not translated to x *is* A *is not fairly true*, but is translated to x *is* A *is ant(fairly true)* resulting in x *is* A *is fairly false*. Note that when τ is *true* (*false*) in (6.23), the antonym equals the complement, resulting in *false* (*true*).

When it is known that x *is* A' then a fuzzy truth value τ_A can be constructed so that x *is* A *is* τ_A "equals" x *is* A'. The fuzzy truth value τ_A is determined by (Zadeh, 1981b):

$$\mu_{\tau_A}(t) = \begin{cases} 0, & \text{if } \mu_A^{-1}(t) = \emptyset \\ \sup\limits_x \{\mu_{A'}(x) \mid \mu_A(x) = t\}, & \text{otherwise} \end{cases} \tag{6.24}$$

A common notation for $\mu_{\tau_A}(t)$ is $\mu_A(A')$. However, when we know that x *is* A *is* τ_A, then the principle of minimum specificity (see section 6.2.3.1) leads to (Zadeh, 1981b):

$$\mu_{A'}(x) = \mu_{\tau_A}(\mu_A(x)) \tag{6.25}$$

This provides a method to express a proposition x *is* A *is* τ_A by a simpler proposition x *is* A'.

Probability-qualifying statements

As fuzzy truth values can be used to perform truth qualifications of propositions, fuzzy probability values can be used for probability qualifying statements. For example, it is possible to use statements like x *is* A *is likely* or, equivalently, x *is* A *is probable* within a knowledge base. Probability-qualifying statements (propositions) result in a possibility distribution, acting as a fuzzy restriction, on the probability density function for a specific event. Considering a general probability qualifying statement, x *is* A *is* λ, this possibility distribution is defined by:

$$\pi(p) = \mu_\lambda\left(\int_X \mu_A(x)p(x)dx\right) \tag{6.26}$$

where $p(x)$ is a probability distribution (density function) on X and $\mu_\lambda : [0,1] \to [0,1]$ is the membership function of the probability qualification λ. An example is the statement *it is unlikely that the pressure is high*, which will induce a possibility distribution on the probability distribution of the proposition *pressure is high*:

$$\pi(p(\cdot)) = \mu_{\text{unlikely}}\left(\int_X \mu_{\text{high}}(x)p(x)dx\right) \tag{6.27}$$

where the variable x represents the pressure. In section 6.3.2.1 fuzzy quantifiers were described. These fuzzy quantifiers are closely related to fuzzy probability qualifiers. For example, considering the fuzzy quantifier *most*, we can assume $\mu_{\text{most}} \equiv \mu_{\text{likely}}$. This makes it possible to interpret the statement ***most*** *chemical processes are hard to model* as a probability qualifying statement *it is* ***likely*** *that the process is hard to model* ***if*** *it is a* *chemical process*, or more formally: ***if*** *the process is a chemical process* **then** *it is hard to model is* ***likely***.

When several probability qualifying statements are considered, it is possible to derive the linguistic probability of another statement. For example, consider the following two statements:

x is A_1 is λ_1
x is A_2 is λ_2

and suppose we want to obtain the probability qualification in the statement x *is* A *is* λ. Then the solution is obtained by solving the following variational problem, which is obtained by means of the extension principle (Zadeh, 1981a):

$$\mu_\lambda(\gamma) = \max_p \left\{ \mu_{\lambda_1}\left(\int_X \mu_{A_1}(x)p(x)dx\right) \wedge \mu_{\lambda_2}\left(\int_X \mu_{A_2}(x)p(x)dx\right)\right\} \tag{6.28a}$$

subject to:

$$\gamma = \int_X \mu_A(x)p(x)dx \tag{6.28b}$$

where γ is the numerical probability for the statement x *is* A *is* λ. It is obvious that, when considering a number of probability qualifying propositions, the reasoning with this type of knowledge is "difficult" to implement by means of a computer program in the case of continuous universes. Hence, approximations by means of discretized universes are to be used.

Possibility-qualifying statements

Besides the fuzzy truth- and probability-qualifying propositions, possibility-qualifying propositions can be considered. Analogous to the use of fuzzy probability values, fuzzy possibility values in a possibility-qualifying proposition induce a possibility distribution on the possibility distribution for a variable:

$$\pi(P(\cdot)) = \mu_\rho\left(\sup_x \min(\mu_A(x), P(x))\right) \tag{6.29}$$

which corresponds to a proposition x *is* A *is* ρ and where ρ is a fuzzy possibility value with membership function $\mu_\rho(\cdot)$ and $P(x)$ is a possibility distribution on X. An example of such a proposition is *it is almost impossible that the tank is full*, resulting in a possibility distribution:

$$\pi(P(\cdot)) = \mu_{\text{almost impossible}}\left(\sup_x \min(\mu_{\text{full}}(x), P(x))\right) \tag{6.30}$$

where the variable x represents the level of the tank and $P(x)$ is the possibility distribution representing the knowledge about x (the tank volume or level in this example).

Another approach to possibility-qualifying statements is based on numerical possibility qualifications; see section 6.2.1.3.

6.3.3 Practical considerations

The basics of approximate reasoning have been previously described. Next we focus on the practical applicability of approximate reasoning. More than a decade ago, Zadeh (1981b) stated that:

> *In a decade or so from now - when the performance of natural language understanding and question-answering will certainly be much more impressive than it is today - it may well be very hard to comprehend why linguistics, philosophers, logicians and cognitive scientists have been so reluctant to come to grips with the reality of the pervasive imprecision of natural languages and have persisted so long in trying to fit their theories of syntax, semantics and knowledge representation into the rigid conceptual mold of two-valued logic.*

However, although much research has been done since, there are barely any commercial products which provide a tool for applying approximate reasoning in knowledge-based systems. It is true that there are many software tools (and additional hardware like "fuzzy" chips) for fuzzy control. However, those tools are merely based on a very limited part of fuzzy set theory and approximate reasoning.[*] The question is: Why are there no commercial expert system tools for applying approximate reasoning? When we consider the first academic application of approximate reasoning, the PRUF[†] system (Zadeh, 1981b), then examples show that, in these systems (intuitively) better modeling of

[*]The tools available for fuzzy control are based on what in chapter 4 was referred to as the "practical approach" to fuzzy control. Most tools only provide means to use one of the standard inference schemes as described in section 4.2.3.

[†]PRUF stands for "Possibilistic Relational Universal Fuzzy" and was developed by Zadeh and co-workers at the University of Berkeley.

human reasoning and natural language is possible than in systems based on classical logic. A good example given by Zadeh (1992) is the evaluation of a statement like “over the past few years Naomi earned far more than most of her friends”. Evaluating a statement like this using two-valued logic will result in either true or false. Using the concept of approximate reasoning with different types of reasoning (section 6.3.1) and translation rules (section 6.3.2) a statement like the example previously given can be evaluated more intuitively and can, for example, result in “fairly true” or “absolutely true”, which are probably more meaningful to humans than just “true” in either case.

However, a major problem in practice is the calculational load and memory requirements when applying approximate reasoning. Many parts of the inference involve calculations on product spaces with multi-dimensional functions (possibility distributions and membership functions). In cases where the universes are discrete, the calculational load and memory requirements can be severe, but implementation is straightforward. In the case of continuous universes one has to discretize the universes, resulting in loss of information, or obtain analytical solutions, which is only possible for a limited number of restricted cases (see, for example, section 6.4.2). This may be the reason that most types of fuzzy reasoning are based on local inference and analogical reasoning, using conjunctions for implications (see section 3.3.2), or reasoning based on similarity measures as described in section 6.5.2.

6.4 Reasoning with possibility distributions

Reasoning with possibility distributions is described in this section. In section 6.2.1, the concept of a possibility distribution was explained, and in section 6.2.4, how a rule can be represented by a conditional possibility distribution was described. In the following sections first different interpretations of rules are discussed and then the resulting modeling by conditional possibility distributions. In section 6.4.2, a method is described which “breaks up” the inference of a rule base under certain criteria.

6.4.1 Interpretation of rules

When we consider an if-then rule, it is obvious that this rule can be interpreted in different ways. For example, a rule stating that *if a ship is big, then its turning speed is slow* is normally understood to imply that *the bigger the ship, the slower its turning speed.* However, a rule *if a house is big then it is wanted* does not automatically imply that *the bigger the house, the more it is wanted*, because many people do not want a house that is too big for their budget. From this, it can be concluded that rules represent more knowledge than just a straightforward implication expressed by an if-then rule. Hence, the interpretation of rules can be different, although they have the same if-then structure. This implies that the modeling of rules with different meanings should be different in approximate reasoning. Dubois and Prade (1991) distinguish three types of fuzzy rules:

- **Truth-qualifying** or **gradual rules**. The *if A then B* rule in fact described a *the more A' is A, the more B' is B* relation between the premise and the consequent of the rule. An example of this is the well-known "tomato"-rule ***if*** *a tomato is red* ***then*** *a tomato is ripe*, which actually represents the knowledge *the* ***more*** *a tomato is red, the* ***more*** *a tomato is ripe*.
- **Certainty-qualifying rules** represent knowledge which can be formulated by: *the* ***more*** *A' is A, the* ***more certain*** *B' is B*.
- **Possibility-qualifying rules** are the type of rules that are normally used in fuzzy control. The rule ***if*** *A* ***then*** *B* represents *the* ***more*** *A is A', the* ***more possible*** *B' is B*. In section 6.4.1.4, the relation between this type of rule and fuzzy control is explained.

In the following sections we go into more detail on the different types of rules.

6.4.1.1 Possibility-qualifying rules

Possibility-qualifying rules (or possibility rules for short) are modeled by a conjunction as implication. The aggregation of a set of possibility-rules is done by using a *max* operation. As shown in section 3.3.2.1, using a conjunction for the implication leads to a simplified inference which eliminates the need for global inference, since the results from local inference equal the results from global inference. Therefore, possibility rules are well applicable in fuzzy expert systems, provided that they represent the meaning of the rules they model.

Considering the general rule ***if*** *x is A* ***then*** *y is B*, the inferred B' is simply given by:

$$\mu_{B'}(y) = \sup_x \min(\mu_{A'}(x), \mathrm{T}(\mu_A(x), \mu_B(y))) \tag{6.31}$$

where T is the T-norm chosen to model the conjunctive implication. Since the implication is modeled by a conjunction, $A' = \overline{A}$ does not result in $B' =$ *unknown*.

Dubois and Prade (1992) state that the inferred possibility distribution for B is restricted by:

$$\pi_{v|u}(x, y) \geq \min(\mu_A(x), \mu_B(y)) \tag{6.32}$$

Hence, this leaves the *min* operation the only possibility for the T-norm used in (6.31) when the principle of minimum specificity has to be fulfilled. However, this restriction can be loosened for the fuzzy result from the inference, although the general idea of this type of rule should be preserved: the inferred result is a restriction of the consequent of the rule. Using another T-norm than the *min* operator in (6.31) is not in contradiction with restriction (6.32), if we interpret the inferred result B' as B' *is possible for* y, which restricts the possibility distribution $\pi_v(y)$ by:

$$\pi_{v'}(y) \geq \mu_{B'}(y) \tag{6.33}$$

This restriction is always valid since the result:

$$\mu_{B'}(y) = \mathrm{T}(\beta, \mu_B(y)) \leq \min(\beta, \mu_B(y)) \leq \pi_{v'}(y) \tag{6.34}$$

In other words, the inferred membership function $\mu_{B'}(y)$ contains extra (subjective) information by choice of the T-norm, other than the *min* operator. The possibility rules are the fuzzy rules which are normally used in fuzzy control. In chapter 4, fuzzy controllers were discussed in detail. It was shown there that common choices for the conjunctive implication are the *min* and *product* operation. This shows that fuzzy controllers can be considered special cases of approximate reasoning: reasoning with possibility qualifying rules. See also section 6.4.1.4 on this topic.

6.4.1.2 Certainty-qualifying rules

Certainty-qualifying rules (or certainty rules, for short) are modeled by S-implications in approximate reasoning based on possibility distributions. The use of this type of rule requires operations on Cartesian product spaces. The inference results in:

$$\mu_{B'}(y) = \sup_x \min(\mu_{A'}(x), \mathrm{S}(1 - \mu_A(x), \mu_B(y))) \tag{6.35}$$

The possibility distribution function $\pi_v(y)$ is restricted by (Dubois and Prade, 1992):

$$\pi_{v'}(y) \leq \max(1 - \mathrm{N}(A; A'), \mu_B(y)) \tag{6.36}$$

where $\mathrm{N}(A; A')$ is the necessity measure as defined by (6.10) and $1-\mathrm{N}(A; A')$ is interpreted as the uncertainty of U *is* A, knowing U *is* A'. This restriction on $\pi_{v'}(y)$ is the result of the restriction Dubois and Prade (1991) place on the conditional possibility distribution representing the rule:

$$\pi_{v|u}(x,y) \leq \max(1-\mu_A(x), \mu_B(y)) \tag{6.37}$$

However, this does not restrict the inferred membership function for B', since:

$$\mu_{B'}(y) \geq \pi_{v'}(y) \tag{6.38}$$

when the inferred result is interpreted as *B' is certain for y*. This provides more or less a justification to use any S-implication for modeling certainty rules, but the problem of calculus on Cartesian product spaces still exists when a set of parallel rules is considered. The possibly extensive calculational load and memory requirements of this type of rule, is a disadvantage for its use in fuzzy knowledge-based systems in the case global inference is to be applied. To overcome the disadvantages of calculus on Cartesian product spaces, a practical method could be devised which has a similar property: the less the data and the premise of the rule match, the more uncertain the conclusion will be. However, this approach is based on local inference with the disadvantage that the results are less restrictive than they could be, based on the available knowledge; see also section 3.3.2.1. In section 6.4.2, a solution to avoid the calculus on product spaces (under certain criteria) is described.

Next, some approaches based on local inference are described. Prade (1983) proposed the following general model for the modus ponens based on local inference:

$$\mu_{B'}(y) = \begin{cases} 1, & \text{if } \mu_B(y)=1 \\ \in [\max(1-\beta, \mu_B(y)), 1], & \text{if } 0<\mu_B(y)<1 \\ 1-\beta, & \text{if } \mu_B(y)=0 \end{cases} \tag{6.39}$$

where β represents a comparison between A and A', namely a necessity measure. Magrez and Smets (1989) proposed a special case of the above-given model by Prade (1983). They state that the *shape of indetermination* criterion should be fulfilled by the modus ponens in fuzzy inference*:

$$A' \supseteq A \;\Rightarrow\; \exists\, \beta \in [0,1],\ \forall\, y \in Y:\ \mu_{B'}(y) = \mathrm{S}(\beta, \mu_B(y)) \tag{6.40}$$

*In their article Magrez and Smets (1989) state that none of the implications they consider, fulfill criterion (6.40). Among the implications they checked is also the implication $(1-a) \vee b$. Using this implication and *max-min* composition will result in:

$$\mu_{B'}(y) = \sup_x \min(\mu_{A'}(x), \max(1-\mu_A(x), \mu_B(y)))$$

where β represent the uncertainty due to the matching of the data and the premise of the rule, which is a necessity measure (see section 6.2.2). This means that an indetermination β (a level of uncertainty) is added to the whole universe Y by means of a T-conorm (S-norm) and that the shape of A' should not influence the shape of B'. In other words: if an indetermination should appear on the consequent, then that indetermination is allowed for all elements of Y. Magrez and Smets (1989) state:

Indeed, there is no information in a material implication which could allow us to assign different degrees of indetermination to different elements of the conclusion domain. The implication rule is unable to constrain more some subset of the domain than other for attributing ignorance.

This is correct when certainty rules are considered. However, in many cases *if-then* rules represent more than an implication in the classical sense. Rules can, for example, in

---

$$\begin{aligned}
&= \sup_x \max(\min(\mu_{A'}(x), 1-\mu_A(x)), \min(\mu_{A'}(x), \mu_B(y))) \\
&= \max(\sup_x \min(\mu_{A'}(x), 1-\mu_A(x)), \sup_x \min(\mu_{A'}(x), \mu_B(y))) \\
&= \max(\sup_x \min(\mu_{A'}(x), 1-\mu_A(x)), \mu_B(y))
\end{aligned}$$

which meets criteria (6.40), since it equals $S(\beta, \mu_B(y))$ choosing *S* to be the *max* operation and:

$$\beta = \sup_x \min(\mu_{A'}(x), \mu_A(x))$$

In addition, they state that using *max-*$T_Ł$ composition instead of *max-min* composition, will result in most S-implications to meet the criterion which states that B should be inferred from $A \to B$ and A. This is true, but using *max-*$T_Ł$ composition in combination with Łukasiewicz-implication will also meet criterion (6.40), since:

$$\begin{aligned}
\mu_{B'}(y) &= \sup_x \mathrm{T}_Ł(\mu_{A'}(x), \min(1-\mu_A(x)+\mu_B(y), 1) \\
&= \sup_x \max(\mu_{A'}(x) + \min(1-\mu_A(x)+\mu_B(y), 1) - 1, 0) \\
&= \min(1, \sup_x(\mu_{A'}(x) - \mu_A(x)) + \mu_B(y))
\end{aligned}$$

And this meets criterion (6.40) when *S* equals the conjunction according to Łukasiewicz and:

$$\beta = \sup_x(\mu_{A'}(x) - \mu_A(x))$$

Therefore the method proposed by Magrez and Smets in their article *"Fuzzy modus ponens: a new model suitable for application in knowledge-based systems"* (1989) is functionally equivalent to the combination of the Łukasiewicz implication and the *max-*$T_Ł$ composition.

addition to the basic implication, also represent some gradual notion (for truth-qualifying or gradual rules, see section 6.4.1.3).

6.4.1.3 Truth-qualifying rules

In this section *truth-qualifying rules*, also referred to as *gradual rules*, are addressed. Gradual rules ***if*** x *is* A, ***then*** y *is* B are interpreted as *the* ***more*** x *is* A, *the* ***more*** y *is* B (Dubois and Prade, 1992). The use of *less* instead of *more* requires the use of the complements of the label in the propositions. For example, *the* ***less*** x *is* A can be replaced by *the* ***more*** x *is* ***not*** A, where the membership function of ***not*** A *is* $1 - \mu_A(x)$. A well-known example of a gradual rule is ***if*** *a tomato is red,* ***then*** *a tomato is ripe*, which can be interpreted as *the* ***more*** *a tomato is red, the* ***more*** *a tomato is ripe*. Hence, a tomato which is *very red* can be considered *very ripe*. Two types of truth-qualifying rules are distinguished by Dubois and Prade (1991):

- **core-widening gradual rules**, which interpret the rule ***if*** x *is* A, ***then*** y *is* B as *the* ***more*** x *is* A *and the* ***more*** y *is related to* x *according to the rule, the* ***more*** y *is* B.
- **support-shrinking gradual rules**, which interpret a fuzzy rule as *the* ***more*** x *is* A *and the* ***less*** y *is related to* x *according to the rule, the* ***less*** y *is* B.

The following subsections address these two types of gradual rules in more detail.

Core-widening gradual rules

The **core-widening gradual rules** pose the following restriction on the conditional possibility distribution (Dubois and Prade, 1991):

$$\forall x \in X,\ \forall y \in Y,\ \min(\mu_A(x), \pi_{V|U}(x,y)) \leq \mu_B(y) \tag{6.41}$$

The solution for the conditional possibility distribution $\pi_{V|U}(x,y)$ representing the rule, is the restriction:

$$\forall x \in X,\ \forall y \in Y,\ \pi_{V|U}(x,y) \leq \begin{cases} 1, & \text{if } \mu_A(x) \leq \mu_B(y) \\ \mu_B(y), & \text{otherwise} \end{cases} \tag{6.42}$$

and can be interpreted as ***if*** x *is* A, ***then*** y *is* B *is at least* $\mu_A(x)$*-true*. A closer look at (6.41) reveals the use of an R-implication as defined by (3.11b), in which the T-norm is chosen to be the *min* operation. A generalized restriction for $\pi_{V|U}(x,y)$ could be obtained by defining the conditional possibility distribution as:

$$\pi_{V|U}(x,y) \geq \sup\{\gamma \in [0,1] \mid \mathrm{T}(\mu_A(x),\gamma) \leq \mu_B(y)\} \tag{6.43}$$

which is the definition of an R-implication (see (3.11b) in section 3.2.2 on page 50). This restriction for $\pi_{V|U}(x,y)$ can be loosened to require that the conditional possibility distribution is greater than or equal to an implication function based on partial ordering as defined by (3.11):

$$\pi_{V|U}(x,y) \geq \begin{cases} 1, & \text{if } \mu_A(x) \leq \mu_B(y) \\ 0, & \text{if } \mu_A(x) = 1 \wedge \mu_B(y) = 0 \\ \in [0,1\rangle, & \text{otherwise} \end{cases} \tag{6.44}$$

In section 3.3.2.1, it was explained that in knowledge-based systems local inference is preferred, and calculus on product spaces should be avoided for obvious reasons. However, in section 3.3.2.1, it was discussed that local inference can lead to results being less restrictive in terms of possibility distributions than the results obtained by global inference. Analytical solutions for local inference can be determined for many implications which fulfill (6.43), but analytical solutions for the global inference of a set of parallel core-widening truth-qualifying rules are not generally available.

Support-shrinking gradual rules

The **support-shrinking gradual rules** are characterized by the following restriction for the conditional possibility distribution representing the rule:

$$\forall x \in X,\ \forall y \in Y,\ \max(1-\mu_A(x), \pi_{V|U}(x,y)) \geq \mu_B(y) \tag{6.45}$$

which results in the following solution for the conditional possibility distribution $\pi_{V|U}(x,y)$:

$$\forall x \in X,\ \forall y \in Y,\ \pi_{V|U}(x,y) \leq \begin{cases} 0, & \text{if } \mu_A(x) + \mu_B(y) \leq 1 \\ \mu_B(y), & \text{otherwise} \end{cases} \tag{6.46}$$

This can be interpreted as ***if*** *x is A,* ***then*** *y is B is at least $(1-\mu_A(x))$-true*. Like inequality (6.41) can be obtained by swapping $\pi_{V|U}(x,y)$ and $\mu_B(y)$ in (6.32), we can obtain a restriction for the conditional possibility distribution for support-shrinking gradual rules by swapping $\pi_{V|U}(x,y)$ and $\mu_B(y)$ in (6.45), which results in (6.46). A generalization for the restriction of $\pi_{V|U}(x,y)$ is defined by:

$$\pi_{V|U}(x,y) \leq \inf\{\gamma \in [0,1] \mid \mathrm{S}(1-\mu_A(x),\gamma) \geq \mu_B(y)\} \tag{6.47}$$

which in the extremes (only 0 and 1 are considered) equals the logical *and* operation. This restriction can be loosened even more by requiring:

$$\pi_{V|U}(x,y) \leq \begin{cases} 0, & \text{if } \mu_A(x)+\mu_B(y) \leq 1 \\ 1, & \text{if } \mu_A(x)=1 \wedge \mu_B(y)=1 \\ \in \langle 0,1], & \text{otherwise} \end{cases} \tag{6.48}$$

Clearly, the two-valued logical *and* operation falls in this class. Since support-shrinking truth-qualifying rules are modeled by pseudo-conjunctions, the result of global inference can be obtained by means of local inference. Considering a set of parallel rules of this type, the inference can be done for each individual rule after which the results can be aggregated by means of the *max* operator; see also section 3.3.2.1.

6.4.1.4 Fuzzy control rules in terms of rule types

Considering the results addressed in the previous sections, it is possible to describe fuzzy controllers in terms of possibility and certainty qualifications. From the viewpoint of possibility theory, the fuzzy controller according to Mamdani (1974) can be viewed as being based on rules like (considering a 2-input-1-output controller):

r_k : **if** x_1 is $A_{1,k}$ is *possible*
and x_2 is $A_{2,k}$ is *possible*
then y is B_k is *possible*

and the rule base contains the fuzzy rules:

(r_1 **else** r_2 **else** ... **else** r_k **else** ...)

When applying the restrictions on possibility distributions induced by the propositions, this will result in the following conditional possibility distribution to represent the combination of all rules r_k:

$$\pi_{V|U_1,U_2}(x_1,x_2,y) = \overbrace{\max_k}^{\text{else}} \underbrace{\min}_{\text{if-then}}(\overbrace{\min}^{\text{and}}(\mu_{A_{1,k}}(x_1),\mu_{A_{2,k}}(x_2)),\mu_{B_k}(y)) \tag{6.49}$$

where x_1 and x_2 are assumed noninteractive. For each rule r_k the inference, assuming a general T-norm for the implication, results in:

$$B'_k = \mathrm{T}(\Pi(A; A'), B_k) \tag{6.50}$$

where T is the T-norm used for the implication (possibility rule, see section 6.4.1.1).

The aggregation is done by the *max* operator since the rules can be considered possibility qualifications and the rule base can be seen as a combination of a number of possibility qualifications (r_1 **else** r_2 **else** ...). The left *min* operator in (6.49) is the implication function used by Mamdani and co-workers (and many others). Other combinations of operators for the conjunction in the rule premises and for the implication modeling the rules are possible. However, using, for example, product for conjunction indicates that the controller inputs are considered to be interactive, since it would lead to $\pi_{U,V}(x_1, x_2) < \min(\mu_{A_{1,k}}(x_1), \mu_{A_{2,k}}(x_2)$ for some values (x_1, x_2).

Yager (1994) also considers certainty qualifications in the rule premises. When we still consider the rule as a possibility qualifying rule, this results in:

r_k : **if** x_1 is $A_{1,k}$ is *certain*
and x_2 is $A_{2,k}$ is *certain*
then y is B_k is *possible*

Then, the result of the inference of rule r_k is given by:

$$B'_k = \mathrm{T}(\mathrm{N}(A; A'), B_k) \tag{6.51a}$$
$$= \mathrm{T}(1 - \Pi(\overline{A}; A'), B_k) \tag{6.51b}$$

where $\mathrm{N}(A; A') = 1 - \Pi(\overline{A}; A')$ represents the certainty of the proposition x *is* A.

It is important to note that in case the data A' is a singleton, the different interpretations all reduce to the same type of inference:

$$B'_k = \mathrm{T}(a', B_k) \tag{6.52}$$

where $a' = \mu_A(x')$ is the “fuzzy” representation of the numerical data x'. This reduction of different rule interpretations to the same type of inference in the case of numerical inputs is because $\Pi(A; a') = 1 - \Pi(\overline{A}; a') = \mathrm{N}(A; a')$.

6.4.2 An inference break-up method

It has been noted in previous sections that the application of approximate reasoning on continuous domains poses the problem that analytical solutions of global inference can usually not be obtained when implications are used which comply with the classical implication. In this section, this problem is addressed, and a solution is proposed which minimizes the calculational load and memory requirements which occur when the domains are discretized and all calculations are done on product spaces. This break-up method breaks up the inference of a rule base into the inference of a number of rule bases of which the inference is easier to perform or has analytical solutions.

First, it is shown how the inference of a rule base with complex rules, modeled by implications which are based on the classical implication, can be reduced to the inference of a number of rule bases with simple rules. After this, a number of possible simplifications for the inference of these derived rule bases with simple rules are given. In section 6.4.2.1, a method is discussed to obtain the analytical solution of inference of a rule base with simple rules. Section 6.4.2.2 discusses further simplifications. The last subsection (6.4.2.3) summarizes the results and address some implementation issues.

6.4.2.1 Breaking up the inference

In this section, it is shown how the inference of a rule base with complex rules (rules with two or more independent variables in their premise) can be simplified to a number of inferences of rule bases with simple rules (only one variable in their premise). Mizumoto (1985) showed explicitly for eight different implication functions that complex rules can be broken up in simple rules. Demirli and Turksen (1992) give a more general solution, since they show that a complex rule modeled by a general S- or R-implication can be broken up into a set of simple rules. The proof of Demirli and Turksen (1992) is only valid for *sup-min* composition and the *min* operation for conjunction in the premise, and is based on the fact that S- and R-implications are non-increasing with respect to their first argument. When the rule:

if x_1 is A_1 **and** x_2 is A_2 **then** y is B

is considered, the resulting conditional possibility distribution is given by:

$$\mathrm{I}(A_1 \cap A_2, B) = \mathrm{I}(A_1, B) \cup \mathrm{I}(A_2, B) \tag{6.53}$$

where the extension principle is assumed where necessary. In section F.1, the proof for (6.53) can be found, and it should be noted that the proof can be generalized to cases with more than two variables in the rule premises. However, Demirli and Turksen (1992) considered only one rule although normally a rule base consists of more rules than just one. It is now shown, that the inference of a rule base with more than one rule can be broken up into the inference of a number of rule bases with simple rules. When considering a rule base with two rules and rules as above, the rule base break-up results in:

$$B' = (A'_1 \cap A'_2) \circ \{\mathrm{I}(A_{1,1} \cap A_{2,1}, B_1) \cap \mathrm{I}(A_{1,2} \cap A_{2,2}, B_2)\} \tag{6.54a}$$

$$\begin{aligned} &= A'_1 \circ \{\mathrm{I}(A_{1,1}, B_1) \cap \mathrm{I}(A_{1,2}, B_2)\} \ \cup \\ &\quad A'_2 \circ \{\mathrm{I}(A_{2,1}, B_1) \cap \mathrm{I}(A_{2,2}, B_2)\} \ \cup \\ &\quad \{[A'_1 \circ \mathrm{I}(A_{1,1}, B_1)] \cap [A'_2 \circ \mathrm{I}(A_{2,2}, B_2)]\} \ \cup \\ &\quad \{[A'_1 \circ \mathrm{I}(A_{1,2}, B_2)] \cap [A'_2 \circ \mathrm{I}(A_{2,1}, B_1)]\} \end{aligned} \tag{6.54b}$$

The proof can be found in section F.3. The result of the inference break-up can easily be extended to a larger number of rules or more complex rules. The number of parts resulting from the inference break-up can become quite large*, but many simplifications are possible and are discussed in section 6.4.2.2.

The fuzzy sets A'_1 and A'_2 are assumed to be normalized. When subnormal fuzzy sets are considered, the following modification of A'_1 and A'_2 is necessary:

$$\begin{aligned} A''_1 &= \mathrm{hgt}(A'_2) \wedge A'_1 \\ A''_2 &= \mathrm{hgt}(A'_1) \wedge A'_2 \end{aligned}$$

Since the *min* operator is used for conjunction and the *sup-min* composition is used for inference, the influence of subnormal fuzzy sets is compensated for when using A''_1 and A''_2, instead of A'_1 and A'_2, respectively.

6.4.2.2 Reduction of inference break-up

In section 6.4.2.1, it was shown that the inference of a rule base with complex rules ("complex" rule base) can be divided into the inference of a number of rule bases with simple rules (from now on referred to as "simple" rule bases). Since the number of those simple rule bases can be very large, the calculational load due to the inference phase can be very high. In this section a number of concepts are discussed which can greatly simplify

*The structure of (6.54) can be determined by considering a structure like $(x_{1,1}+x_{2,1}+\cdots+x_{n,1})(x_{1,2}+x_{2,2}+\cdots+x_{n,2})\cdots(x_{1,m}+x_{2,m}+\cdots+x_{n,m})$, in the case of n variables in the premises of the m rules. The number of cross-products is n^m.

the inference. Because of the break-up of the inference, only simple rule bases have to be considered.

Non-overlapping fuzzy sets in premises

It was shown that in section 6.4.2.1 the inference engine only has to consider the rules in the rule base of which the premise overlaps with the data. This results in a conflict set (of rules), and only the rules in the conflict set has to be considered for the inference break-up method. After the breaking-up of the inference a number of simple rule bases can be distinguished

A simplification of the inference of a simple rule base can be achieved when the rule base can be divided into a set of simple rule bases which do not interact. This interaction of rules (or rule bases in this case) is because the premises of the rules overlap. In case a simple rule base can be separated into two or more simple rule bases with rules that do not have overlapping premises, then these separable simple rule bases do not interact, since $I(0, b) = 1$ for the implications considered by the inference break-up method. To explain the possible simplification more clearly, consider the following two rules:

r_1 : **if** x is A_1 **then** y is B_1
r_2 : **if** x is A_2 **then** y is B_2

where it is assumed that $A_1 \cap A_2 = \emptyset$. Because of the empty intersection of the premises, there is no interaction between the rules, which simplifies the inference of this simple rule base:

$$\begin{aligned} B' &= A' \circ (\mathrm{I}(A_1, B_1) \cap \mathrm{I}(A_2, B_2)) && \text{(6.55a)} \\ &= (A' \cap \mathrm{supp}(A_1)) \circ \mathrm{I}(A_1, B_1) \ \cup \\ &\quad (A' \cap \mathrm{supp}(A_2)) \circ \mathrm{I}(A_2, B_2) \ \cup \\ &\quad \mathrm{hgt}(A' \cap \mathrm{supp}(\overline{A}_1 \cap \overline{A}_2)) && \text{(6.55b)} \end{aligned}$$

Hence, rule r_1 is used in combination with the part of A' which overlaps with A_1, rule r_2 is used in combination with the part of A' which overlaps with A_2. The results of the inferences of these two combinations is aggregated by means of a disjunction and an additional indetermination is “added” to the aggregated result by means of a disjunction. The indetermination is determined by the height of the parts of A' which do not overlap with A_1 and A_2.

If no (sets of) noninteractive rules can be distinguished, it is possible to separate the inference of a simple rule base into the inference of a set of simple rule bases by breaking up the data A'. To show this, consider the following three rules:

r_1 : **if** x is A_1 **then** y is B_1
r_2 : **if** x is A_2 **then** y is B_2
r_3 : **if** x is A_3 **then** y is B_3

where $A_1 \cap A_2 \neq \emptyset$, $A_2 \cap A_3 \neq \emptyset$ and $A_1 \cap A_3 = \emptyset$. When the data A' is split up by:

$$\begin{aligned}
A'_1 &= A' \cap \{\mathrm{supp}(A_1) \cap \mathrm{supp}\overline{A}_2 \cap \mathrm{supp}(\overline{A}_3)\} \\
A'_2 &= A' \cap \{\mathrm{supp}(\overline{A}_1) \cap \mathrm{supp}A_2 \cap \mathrm{supp}(\overline{A}_3)\} \\
A'_3 &= A' \cap \{\mathrm{supp}(\overline{A}_1) \cap \mathrm{supp}\overline{A}_2 \cap \mathrm{supp}(A_3)\} \\
A'_{1,2} &= A' \cap \{\mathrm{supp}(A_1 \cap A_2) \cap \mathrm{supp}(\overline{A}_3)\} \\
A'_{2,3} &= A' \cap \{\mathrm{supp}(\overline{A}_1) \cap \mathrm{supp}(A_2 \cap A_3)
\end{aligned}$$

then the inference of the three rules can be written as:

$$\begin{aligned}
B' = A'_1 \circ \mathrm{I}(A_1, B_1) \,\cup\, A'_2 \circ \mathrm{I}(A_2, B_2) \,\cup\, A'_3 \circ \mathrm{I}(A_3, B_3) \,\cup \\
A'_{1,2} \circ \{\mathrm{I}(A_1, B_1) \cap \mathrm{I}(A_2, B_2)\} \,\cup\, A'_{2,3} \circ \{\mathrm{I}(A_2, B_2) \cap \mathrm{I}(A_3, B_3)\}
\end{aligned} \tag{6.56}$$

This shows that in many cases the inference of simple rule bases can be represented by the inference of even simpler rule bases using only parts of the data A'.

When this result is generalized to N_r rules, and the premises of the rules are assumed to take fuzzy sets which form a fuzzy partition and which are convex and normalized, resulting in no more than two overlapping fuzzy sets, then it can be derived that the number of necessary simple rule bases is given by $(2N_A - 1)$, where N_A is the number of fuzzy sets defined on the domain of the variable in question. Here, it is assumed that the rule base is complete and, hence, $N_r = N_A$. This number of necessary rule bases $(2N_A - 1)$ is less than the number of simple rule bases to be used for inference when no fuzzy partition is assumed, which is given by:

$$\sum_{i=1}^{N_A} \binom{N_A}{i} \tag{6.57}$$

Hence, this is the maximum number of rule bases to be used for inference and is only necessary is all N_A fuzzy sets overlap.

Bounded support of data

Because of the *sup-min* composition, only rules of which the premise overlaps with the data, represented by fuzzy set A', have to be considered, since:

$$\sup_{x, \mu_{A'}(x)=0} \min(\mu_{A'}(x), \mu_R(x, y)) = 0$$

Hence, if the data A' has a bounded support, the number of rule premises that overlap with A' is normally less than the total number of rules in a simple rule base (assuming the fuzzy sets in the rule premises also have bounded support, which is normally the case). Thus the inference can be done using a reduced simple rule base and other inference reduction methods can then be applied to it; for example, the possible inference reduction in the case of non-overlapping rule premises as described before.

6.4.2.3 Summary of inference break-up

This section is used to summarize the previously described inference break-up method. In the inference break-up method, the following steps can be distinguished:

1. A complex if-then rule with m conditions in the premise can be broken up into m simple rules (with only one condition in their premises) when the conjunction of the m conditions in the premises is represented by the *min* operator and the implication function complies with the classical implication (Demirli and Turksen, 1992). The subresults of the inference of the simple rules are aggregated by means of disjunctions to obtain the results of the inference of the complex rule.

2. The inference of a rule base with n complex rules, with m conditions in their premises, can be broken up into the inference of $n^2 - 1$ simple rule bases, consisting only of simple rules. The combination of the subresults of the inference of the simple rule bases requires $m^n - m$ conjunctions and $m^n - 1$ disjunctions (for aggregation), all on one-dimensional universes. The same requirements as stated in step 1 are to be met.

3. The inference of a simple rule base in which the Kleene-Dienes implication, defined by $\mathrm{I}(a, b) = \max(1 - a, b)$, is used to model the rules, can be broken up into calculations on one-dimensional universes only. Considering a simple rule base with n rules, the break-up will results in 2^n subresults which have to be aggregated by means of a disjunction.

The three steps as enumerated above provide a method to deal with fuzzy rules in expert systems, which is close to the inference as applied in "conventional" expert systems. First, a *conflict set* is determined, consisting of fireable rules. Next, the rules in this conflict set are used to obtain new knowledge. In conventional expert systems, based on a two-valued logic, one or more rules from the conflict set are selected to fire by means of local inference. In the case of fuzzy rules (certainty rules in this case), the rules in the conflict set are fired in parallel without ignoring the interaction between the fuzzy rules when the break-up method is applied.

An interesting possibility concerning the inference break-up is a dedicated (co)processor which performs composition of a fuzzy set and a (two-dimensional) fuzzy relation. The inference break-up method can be applied off-line and, hence, the run-time calculations can be performed by such a highly optimized processor. Obviously, instead of a dedicated processor, a speed-up of the inference can be achieved by using optimized code to perform the composition of a fuzzy set and a fuzzy relation. Because of the inference break-up, the code optimization can be applied to specific parts (composition in this case) of a software implementation.

6.5 Other and derived approaches to fuzzy reasoning

Various approaches to fuzzy reasoning have been proposed in literature. In this section, we describe those different approaches using the following classification:

- fuzzy reasoning as originally proposed by Zadeh (1975);
- reasoning with *fuzzy truth values* (Baldwin, 1979);
- fuzzy reasoning based on *similarity measures* (Yager, 1980a).

Since fuzzy reasoning according Zadeh (1975) has been described in chapter 3 and previous sections in this chapter, it is not treated in this section. The other two approaches have not yet been described in this thesis and are discussed in the following subsections.

6.5.1 Reasoning with fuzzy truth values

The approach to fuzzy reasoning proposed by Baldwin (1979) is different from the one originally proposed by Zadeh (1975). The reasoning method Baldwin proposed is not based on the composition of relations representing the data and the rule or rule base, but on *fuzzy truth values* as described in section 6.3.2.2 (page 202). Several others have also proposed fuzzy reasoning methods based on fuzzy truth values, which are also described in the following subsections.

6.5.1.1 Baldwin's method

Baldwin (1979) proposed a fuzzy reasoning method based on fuzzy truth values. In the previous section, the concept of fuzzy truth values was addressed. It was shown how a fuzzy proposition x *is* A' can be derived from x *is* A *is* τ_A and vice versa. To explain the reasoning method, consider the following inference scheme:

$$\begin{array}{l} (\textbf{if } x \text{ is } A \textbf{ then } y \text{ is } B) \text{ is } \tau_r \\ (x \text{ is } A') \text{ is } \tau_d \\ \hline y \text{ is } B' \end{array}$$

where τ_r is a fuzzy truth value concerning the rule as a whole and τ_d is a fuzzy truth value concerning the data. Initially τ_r and τ_d were not considered. In the following they are considered, since the case where one or both are not considered are special cases. By taking those fuzzy truth values equal to $\tau_{\text{true}}(t) = t$ they will be ignored. The inference is done by the following steps:

1. use the fuzzy truth value τ_d to obtain a fuzzy set for the data: $\mu_{A''}(x) = \mu_{\tau_d}(\mu_{A'}(x))$;
2. determine τ_A using (6.24) from page 203: $\tau_A = \mu_A(A'')$;
3. determine the relation $I(\tau_{\text{true}}, \tau_{\text{true}})$ where the implication I is an implication as described in section 3.2.2 and apply the fuzzy truth value τ_r: $\tau_r(I(\tau_{\text{true}}, \tau_{\text{true}}))$;
4. determine τ_B by composition: $\tau_B = \tau'_A \circ \tau_r(I(\tau_{\text{true}}, \tau_{\text{true}}))$;
5. calculate $\mu_{B'}(y) = \mu_{\tau_B}(\mu_B(y))$.

If $\tau_d = \textit{true}$, hence $\mu_{\tau_d}(t) = t$, step 2 is obsolete. The included adjustment using τ_r in step 4 is obsolete if $\tau_r = \textit{true}$.

To show the reasoning method according to Baldwin (1979), let us consider an example with the following rule and data:

if x is A **then** y is B
x is A'

where the membership functions of A, A' and B' are given by:

$$\begin{aligned} \mu_A(x) &= \max(1 - \frac{|x-4|}{3}, 0) \\ \mu_{A'}(x) &= \max(1 - \frac{|x-5|}{3}, 0) \\ \mu_B(y) &= \max(1 - \frac{|y-4|}{3}, 0) \end{aligned}$$

and are shown in figures 6.3a and 6.3d, respectively. For the sake of simplicity, fuzzy truth values for the rule (τ_r) and data (τ_d), as denoted on page 221 are not considered. The fuzzy truth value τ_A is determined by:

$$\mu_{\tau_A}(t) = \mu_A(A''), \text{ with} \tag{6.58a}$$

$$\mu_{A''}(x) = \mu_{\tau_{\text{true}}}(\mu_{A'}(x)) = \mu_{A'}(x) \tag{6.58b}$$

and is shown in figure 6.3b. The inference to obtain τ_B is based on the "paper-and-pencil" method (Baldwin and Pilsworth, 1980) shown in figure 6.2. In this example the Kleene-Dienes implication, $I(a, b) = max(1 - a, b)$, is used. The resulting fuzzy truth value τ_B is given in figure 6.3c. The result B' is obtained by:

$$B' = \tau_B(B) \tag{6.59}$$

and is shown in figure 6.3d. The result B' equals the result that is obtained by inference based on composition of relations: the compositional rule of inference.

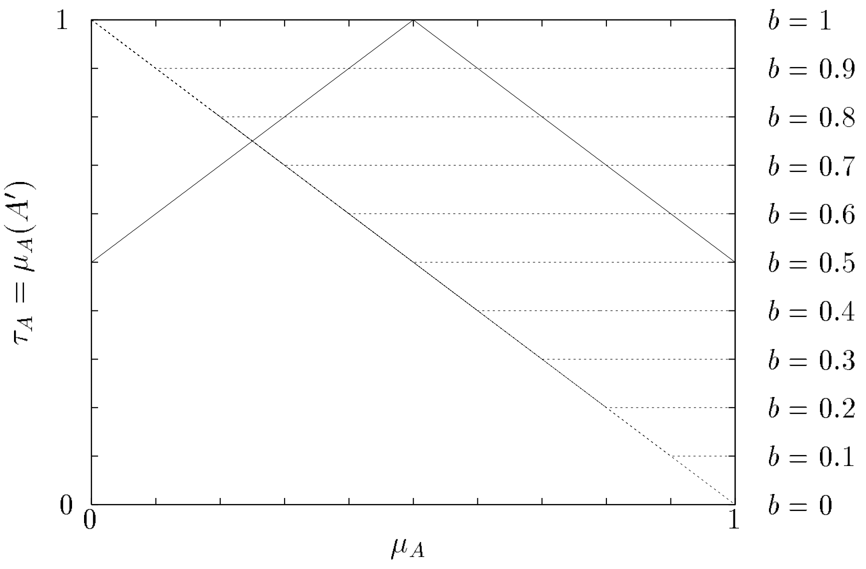
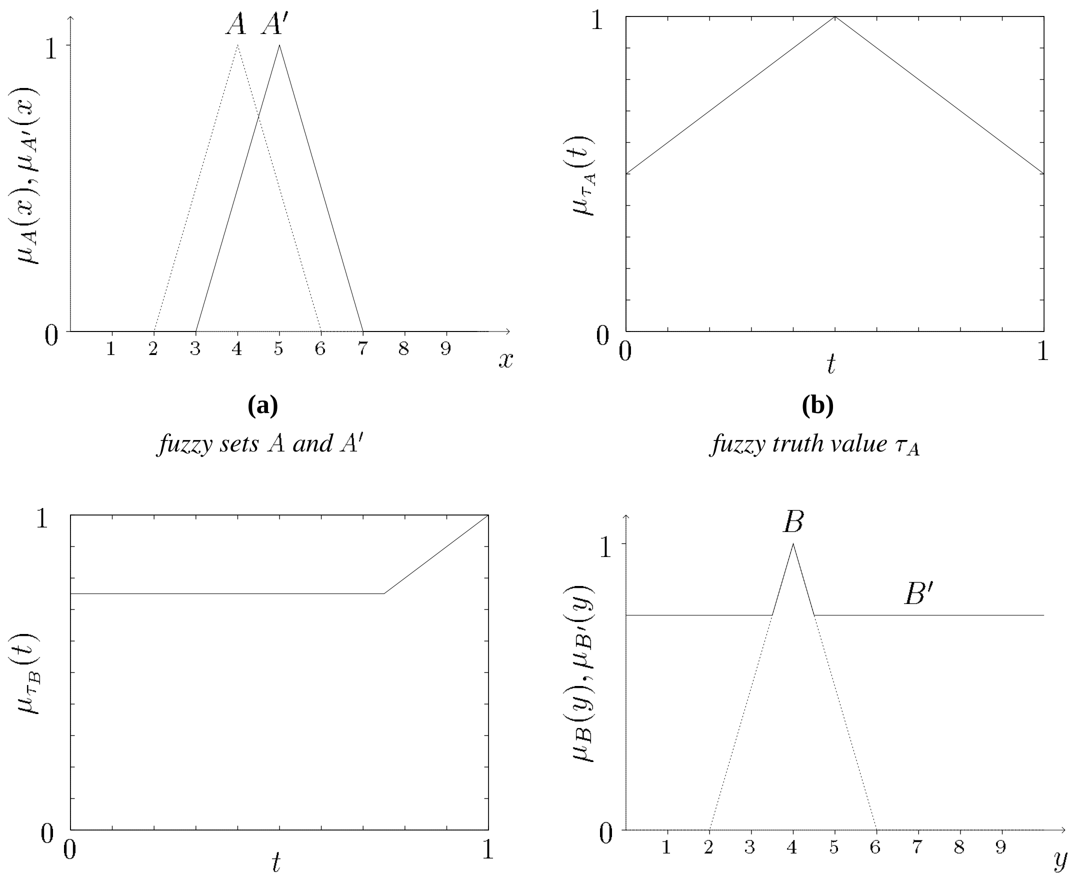


Figure 6.2: *Paper-and-pencil method to determine $\tau_{B'}$ (Baldwin and Pilsworth, 1980).*



(a)
fuzzy sets A and A'

(b)
fuzzy truth value τ_A

(c)
fuzzy truth value τ_B

(d)
fuzzy sets B and B'

Figure 6.3: *Results of inference with fuzzy truth values according to Baldwin (1979).*

6.5.1.2 Tsukamoto's method

The reasoning method proposed by Tsukamoto (1977) is also based on fuzzy truth values, but is based on a different approach to obtain the fuzzy truth value τ_B to determine the result B' of the inference. Consider the following inference scheme:

$$\begin{array}{l} (\textbf{if } x \text{ is } A \textbf{ then } y \text{ is } B) \text{ is } \tau_r \\ x \text{ is } A' \\ \hline y \text{ is } B' \end{array}$$

where no fuzzy truth value for the fuzzy proposition x *is* A' is included, but this can be done without problems; note step 2 in section 6.5.1.1. The fuzzy truth value τ_B is obtained from:

$$\tau_r = \mathrm{I}(\tau_A, \tau_B) \tag{6.60}$$

where it is important to note that the implication I is implemented using fuzzy arithmetic. Since τ_A and τ_r are known, τ_B can be solved from (6.60) by means of the extension principle as described in section 2.1.4:

$$\mu_{\tau_B}(y) = \sup_{\substack{t_1,t_2 \\ t_2=\mathrm{I}(t_1,t)}} \min(\mu_{\tau_A}(t_1), \mu_{\tau_r}(t_2)) \tag{6.61}$$

To solve (6.61), Tsukamoto used a method based on level sets (α-cuts). The fuzzy truth values τ_A and τ_r are represented by their level sets according to:

$$\tau = \cup_{\alpha\in\langle 0,1]}\tau^\alpha = \cup_{\alpha\in\langle 0,1]}[t^{\alpha,l}, t^{\alpha,r}] \tag{6.62}$$

The fuzzy truth value τ_B is determined by its level sets which are obtained from:

$$\tau_B^\alpha = \{t \mid F^{-1}(t) \cap (\tau_A^\alpha \cap \tau_r^\alpha) \neq \emptyset\} \tag{6.63}$$

where $F^{-1}(t)$ is the inverse of $F(t_1, t_2)$, which is the solution of (6.60) for t when numerical truth values are considered. The inverse $F^{-1}(t)$ of $F(t_1, t_2)$ is, by definition, given by:

$$\forall t \in [0,1],\ F^{-1}(t) \triangleq \{(t_1, t_2) \in [0,1] \times [0,1] \mid t \in F(t_1, t_2)\} \tag{6.64}$$

To clarify the method of Tsukamoto (1977), consider the Kleene-Dienes implication as was used previously. For numerical truth values, the solution for t is given by:

$$F(t_1, t_2) = \begin{cases} t_2, & \text{if } 1 - t_1 < t_2 \\ [0, 1 - t_1], & \text{if } 1 - t_1 = t_2 \\ \emptyset, & \text{otherwise} \end{cases} \tag{6.65}$$

When the fuzzy truth value τ_r is considered to be τ_{true}, with $\mu_{\tau_{\text{true}}}(t_2) = t_2$, its level sets are given by:

$$\tau_r^\alpha = [t_2^{\alpha,l}, t_2^{\alpha,r} = 1] \tag{6.66}$$

and the following solution for τ_B^α can be obtained (Tsukamoto, 1977):

$$\tau_B^\alpha = \begin{cases} [0, 1], & \text{if } t_1^{\alpha,l} + t_2^{\alpha,l} \leq 1 \\ [t_2^{\alpha,l}, 1], & \text{otherwise} \end{cases} \tag{6.67}$$

When the data of the example in the previous section is assumed, the final results for τ_B and B' are the same as the results obtained by Baldwin's method, shown in figure 6.3.

6.5.1.3 Mizumoto's method

Another fuzzy reasoning method based on fuzzy truth values is the method proposed by Mizumoto (1981). In this method the implication function is also considered to be based on fuzzy arithmetic. Considering the following inference scheme:

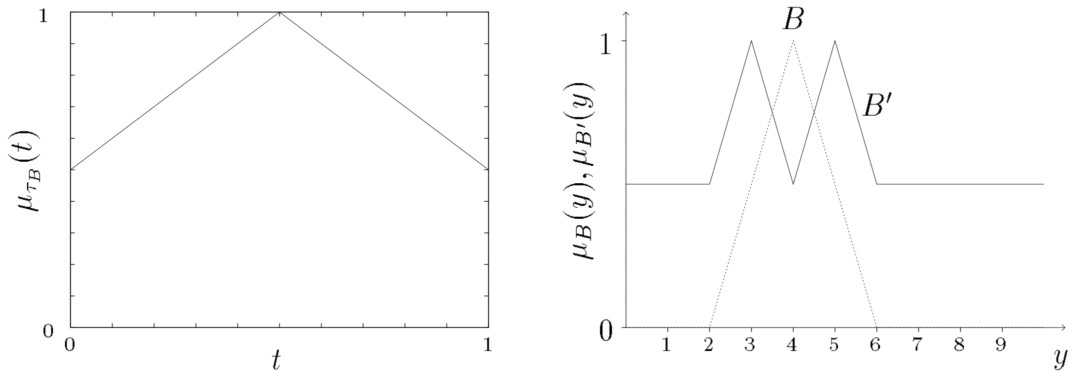
$$\begin{array}{l} \textbf{if } x \text{ is } A \textbf{ then } y \text{ is } B \\ x \text{ is } A' \\ \hline y \text{ is } B' \end{array}$$

then τ_B is given by:

$$\tau_B = \tau_A \wedge \mathrm{I}(\tau_{\text{true}}, \tau_{\text{true}}) \tag{6.68}$$

where $\mathrm{I}(\tau_{\text{true}}, \tau_{\text{true}})$ is based on fuzzy arithmetic and is thus a fuzzy truth value itself. Hence, this fuzzy reasoning method is purely based on fuzzy number/interval calculus. This entails a straightforward implementation when approximations by means of level sets are used (see also previous section). Extension of the inference scheme above to the ones considered in sections 6.5.1.1 and 6.5.1.2 can be done by adjusting the two parts on the right-hand side. The main idea of this method is denoted in (6.68): using fuzzy number/interval calculus to obtain τ_B and hence B' using (6.25).

When we consider the same example that was used in the previous sections, the obtained result is different than that was obtained by the methods described so far. Based on the data shown in figure 6.3a, the results τ_B and B' for this method are shown in figure 6.4. The result B' has the undesired property that it is not convex.



(a) *fuzzy truth value τ_B*

(b) *fuzzy sets B and B'*

Figure 6.4: *Results of inference with fuzzy truth values according to Mizumoto (1981).*

6.5.2 Fuzzy reasoning based on similarity measures

In this section, another approach to fuzzy reasoning is addressed: fuzzy reasoning based on similarity measures. We describe briefly the work of Yager (1980a) and Turksen and Zhong (1990). Both methods overlap with the fuzzy reasoning according to Zadeh as described in chapter 3. Additionally, a method based on domain scaling is described.

6.5.2.1 Yager's method

Yager (1980a) proposed a method for fuzzy reasoning in which the following similarity measures are defined:

$$\mathrm{SM}_c = \frac{\mathrm{hgt}(A \cap A')}{\mathrm{hgt}(A')} \tag{6.69}$$

$$\mathrm{SM}_f = \int_0^1 \mu_{A'}(x)/\mu_A(x) \tag{6.70}$$

The first similarity measure *SM*$_c$ results in a crisp number $\in [0, 1]$, the second one *SM*$_f$ results in a fuzzy measure. This second measure *SM*$_f$ is known as the compatibility of A' with A (Zadeh, 1981b) and was also described in section 6.3.2.2 (page 202) where *SM*$_f$ was referred to as a fuzzy truth value. Note that if A' is normalized $(hgt(A') = 1)$, the similarity measure *SM*$_c$ equals the possibility measure $\Pi(A; A')$ as defined by (6.11a) on page 195.

Yager introduced the fuzzy implication $I(a, b) = b^a$ which is used in the fuzzy reasoning method he proposed. The result B' of inference is given by:

$$B' = B^{\mathrm{SM}} \tag{6.71}$$

where *SM* is either the similarity measure *SM*$_c$ or *SM*$_f$. When *SM*$_f$ is used, the resulting B' will be a type-2 fuzzy set. So far this type of fuzzy sets have not been addressed in this thesis. Type-2 fuzzy sets are fuzzy sets of which the membership grades are fuzzy themselves.[*] When practical applicability is considered, the use of type-2 fuzzy sets is less attractive and we will focus on the case of the crisp similarity measure *SM*$_c$. Focusing on similarity measure *SM*$_c$, the resulting membership function for B' is given by:

$$\mu_{B'}(y) = \mu_{B'}^{\mathrm{SM}_c}(y) \tag{6.72}$$

Hence, when *SM*$_c = 0$ the result is *unknown* and when *SM*$_c = 1$ it results in $B' = B$. From (6.72), the resemblance to the powered hedges described in section 2.2.1 is clear. The method based on the crisp similarity measure can be summarized by stating that the result B' is a modification of the consequent B by means of a modification function using the obtained similarity between premise A and data A'. In the next section, the method proposed by Turksen and Zhong (1990) is described, where also other similarity measures and modification functions are used.

[*]The notion of type-2 fuzzy sets can be extended to type-*n* fuzzy sets (Dubois and Prade, 1980). A "standard" fuzzy set can be regarded as a type-1 fuzzy set: membership grades are crisp numbers.

6.5.2.2 Turksen and Zhong's method

Using the idea of a similarity measure and a modification function, Turksen and Zhong (1990) proposed the *approximate analogical reasoning schema* (AARS). This fuzzy reasoning method uses a similarity measure *SM* which is obtained from a *distance measure DM* by:

$$\mathrm{SM} = \frac{1}{1 + \mathrm{DM}} \tag{6.73}$$

Turksen and Zhong give a number of different distance measures. We do not list all the different distance and derived similarity measures, but show the measures which overlap with other methods to emphasize how this method fits in the collection of other fuzzy reasoning methods. The relation with other fuzzy reasoning methods becomes clear when considering the following similarity measure:

$$\mathrm{SM}_1(A, A') = \mathrm{hgt}(A \cap A') \tag{6.74}$$

This measure is derived from what is referred to as the *disconsistency measure* by Turksen and Zhong. The similarity measure is used by a *modification function MF* which entails a modification of the rule consequent to obtain the result. The modification functions Turksen and Zhong list, stating there are many possible modification functions, are:

$$\mathrm{MF}_1 : \mu_{B'}(y) = \min(1, \mu_B(y)/\mathrm{SM}) \tag{6.75}$$

$$\mathrm{MF}_2 : \mu_{B'}(y) = \mu_B(y) * \mathrm{SM} \tag{6.76}$$

The modification function MF_1 is referred to as the *more or less form* since it results in $B \subset B'$. The second modification function, MF_2, is named the *membership value reduction form* since it results in $B' \subset B$. Note the resemblance with fuzzy reasoning based on T-norm implications if the similarity measure according to (6.74) is used, namely $B' = MF_2(B, SM_1) = B * SM_1$ which is similar to the result obtained when the product operator is used for the implication as described in section 4.2.3.2, and where it was noted that this method does not have a counterpart in fuzzy inference based on the composition of relations.

Turksen and Zhong give a complete algorithm to evaluate a set of parallel rules. In their algorithm, it is possible to assign threshold values to avoid firing fuzzy rules of which the similarity between the premise and data is below such a threshold value. We do not go further into detail about the reasoning method, but note that in this algorithm a clear distinction between the matching phase, determining similarity measures, and the modification phase is made. This approach also characterizes the practical approach to fuzzy control as discussed in section 4.2.1.1.

6.5.2.3 Reasoning with domain scaling

In this section, a method of fuzzy reasoning is described which uses a similarity measure to determine B' from B by means of domain scaling. The result B' is determined by:

$$\mu_{B'}(y) = \mu_B(f_B(y, \beta)) \tag{6.77}$$

where $f_B(y, \beta)$ is a function which performs a scaling of y relative to a characteristic point of $\mu_B(y)$. The value of β is a similarity measure representing the matching of the data with the premise of the rule. This is similar to the scaled hedges approach as described in section 2.2.3. Note that the method proposed by Yager (1980b) is similar to the powered hedges (section 2.2.1) and Hellendoorn (1990) proposed a fuzzy inference method for increasing notions which is closely related to the shifted hedges approach (section 2.2.2).

Let us show an example of a triangular-shaped $\mu_B(y)$ as shown in figure 6.5a. The inferred result B' is given by:

$$\mu_{B'}(y) = \mu_B(\beta y + (1 - \beta)b) \tag{6.78}$$

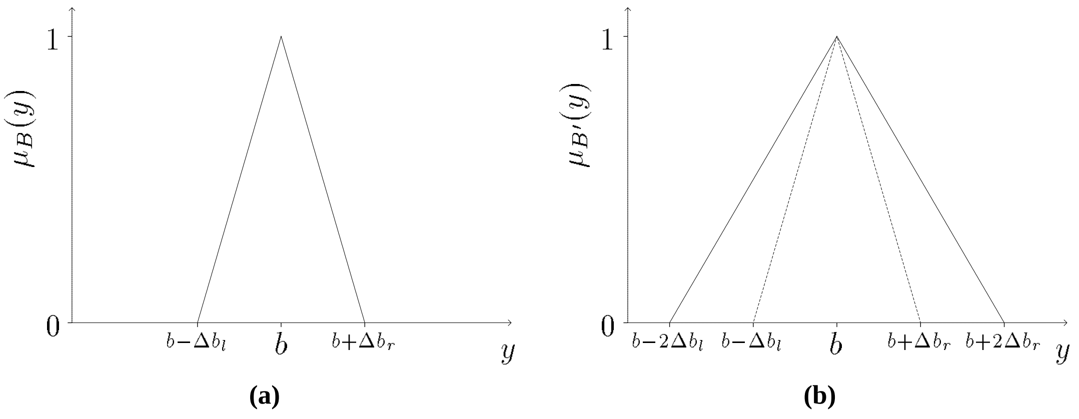


Figure 6.5: *Possibility distributions of original B (a) and inferred B' (b) with $\beta = \frac{1}{2}$, resulting in $\mu_{B'}(y) = \mu_B(\frac{1}{2}y + \frac{1}{2}b)$.*

where b is the center of $\mu_B(y)$ as shown in 6.5a. In figure 6.5b one can see the resulting $\mu_{B'}(y)$ if $\beta = \frac{1}{2}$. From (6.78) the following can be seen:

- if the matching of the data with the premise of the rule is maximal ($\beta = 1$), then $B' = B$;
- when there is no matching between the data and the rule premise ($\beta = 0$), then $B' = \textit{unknown} \ \ (\mu_{B'}(y) = 1)$;
- the less the matching is between the data and the premise of the rule, ($\beta \downarrow 0$) the more "uncertain" is B';
- the more A' is a subset of A, the more B' is a subset of B.

Whether the *law of noncontradiction* is not violated, depends on how the matching between the data and the premise of the rule is determined. Like the other fuzzy reasoning methods described in this section, a clear distinction is made between the matching and the modification phase.

6.5.3 Reasoning with linguistic qualifiers

In this section we briefly address reasoning with linguistic qualifiers. In section 6.3.2.2 different qualifying statements were described. It was also shown that inference with, for example, linguistic probabilities, is not attractive from a practical point of view. Also reasoning with possibility distributions suffers from this, although alternative approaches based on numerical possibility and certainty qualifications are known (see sections 6.2.1.3 and 6.4.1). Dubois and Prade (1991) wrote the following about a fixed set of fuzzy truth qualifiers with respect to fuzzy reasoning:

> *. . . may prove useful for an efficient implementation of the generalized modus ponens, provided that the class of invariant truth-values τ is rich enough to express various types of input facts "x is A" . . .*

This approach is used for linguistic truth, possibility and probability qualifications in the fuzzy first-order logic proposed by Rhodes and Menani (1991) and is described in the following. Although this method can be regarded as a multi-valued logic it is addressed here, because of its relation with the topics addressed before: reasoning with fuzzy truth values (section 6.5.1.1) and qualifier rules in approximate reasoning (section 6.3.2.2).

Rhodes and Menani (1991, 1992) proposed a propositional fuzzy logic which is analogous to traditional two-valued logic. They use truth, possibility and probability qualifiers which are limited to a fixed set of qualifiers. The truth qualifiers are given in table 6.1 and the corresponding membership functions are shown in figure 6.6. One can distinguish "negative" and "positive" qualifiers, where the negative qualifiers have monotonic decreasing membership functions and the positive qualifiers have monotonic increasing membership functions. Negative qualifiers are related to positive qualifiers by the following:

$$\mu_{\text{neg}}(x) = \mu_{\text{pos}}(1-x) \tag{6.79}$$

which resembles the antonym defined by (6.23). The membership functions of the probability and possibility qualifiers are chosen equal to the ones for truth qualifiers; *likely* and *possible* correspond to *true*, *unlikely* and *impossible* correspond to *false*.

Table 6.1: *Truth qualifiers and corresponding membership functions chosen by Rhodes and Menani (1992). Membership functions based on the variable n are limited by $n \to \infty$. In examples given by Rhodes and Menani (1991) a value of $n = 10$ is chosen. See figure 6.6 for the membership functions.*

truth qualifier	membership function
absolutely false	$\mu_{\text{af}}(x) = 1 - \sqrt[n]{1-(1-x)^n}$
very false	$\mu_{\text{vf}}(x) = 1 - \sqrt{1-(1-x)^2}$
false	$\mu_{\text{f}}(x) = 1 - x$
fairly false	$\mu_{\text{ff}}(x) = \sqrt{1-x^2}$
not absolutely true	$\mu_{\text{nat}}(x) = \sqrt[n]{1-x^n}$
not absolutely false	$\mu_{\text{naf}}(x) = \sqrt[n]{1-(1-x)^n}$
fairly true	$\mu_{\text{ft}}(x) = \sqrt{1-(1-x)^2}$
true	$\mu_{\text{t}}(x) = x$
very true	$\mu_{\text{vt}}(x) = 1 - \sqrt{1-x^2}$
absolutely true	$\mu_{\text{at}}(x) = 1 - \sqrt[n]{1-x^n}$

Also five fuzzy quantifiers are defined (Rhodes and Menani, 1992), namely *for all*, *most*, *all*, *few* and *there exists*, which have membership functions corresponding to the

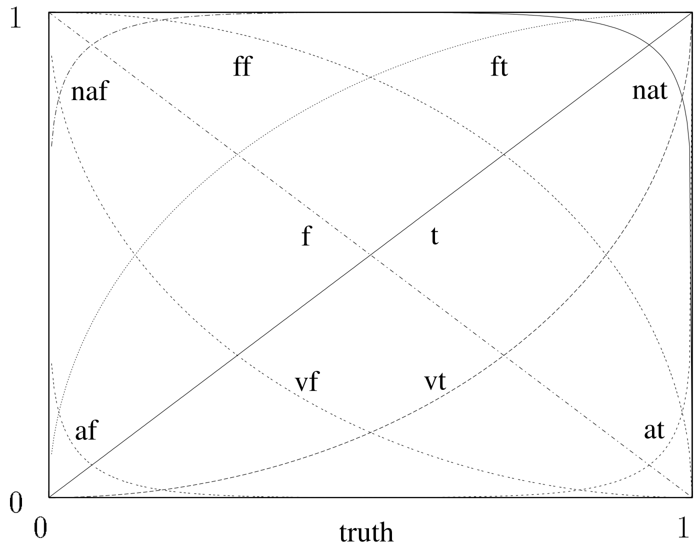


Figure 6.6: *Membership functions of truth qualifiers as defined by Rhodes and Menani (1991). See table 6.1 for the definition of membership functions and abbreviations.*

"positive" probability qualifiers. Hence, a conversion from fuzzy quantifying propositions to probability qualifying propositions is possible (see also section 6.3.2.2, page 203, on this topic):

for all x is A $\Longrightarrow$ x is A is **absolutely likely**
there exists x is A $\Longrightarrow$ x is A is **not absolutely unlikely**

The first-order fuzzy logic proposed by Rhodes and Menani (1992) can be considered a sort of multi-valued logic, since they focus on a fixed set of truth, possibility and probability qualifiers. Indeed, logical operations (conjunction, disjunction, implication and equivalence) concerning truth qualifiers can be defined in truth tables (Rhodes and Menani, 1991). The operations for possibility qualifiers are equal to the ones for truth qualifiers. The probability tables, however, also include intervals, identified by two of the possible probability qualifiers.

The advantage of this approach is that the implementation is not difficult since the predefined tables with results can be used for logical operations. A disadvantage is the fact that the matching of the data with the rule premises has to be "rounded off" to the nearest linguistic qualifier from the fixed set of possible qualifiers to be able to use the predefined tables for logical operations. However, interpolation between entries in the tables to approximate the matching of data and rule premises is possible. When chaining of rules is considered, this poses a problem since the fuzzy qualifiers no longer matches one the predefined fixed set of fuzzy qualifiers.

6.5.4 Remarks and considerations

In the previous sections, various approaches to fuzzy reasoning were described. Two main approaches can be distinguished: those based on fuzzy truth values and those based on similarity measures. Both approaches overlap with the "conventional" approach to fuzzy reasoning, based on composition of fuzzy relations. A characteristic property of both methods is that they are based on local inference. These different approaches are possible solutions to the problem of the practical applicability of approximate reasoning as described in sections 6.3 and 6.4.

The *fuzzy truth value* approaches of Baldwin (1979) and Tsukamoto (1977) produce the same results as those obtained from composition-based fuzzy reasoning, but are completely based on local inference. Hence, this is a major drawback when considering the inference of a set of parallel rules, since this yields less restrictive results than could be expected from the available knowledge and information. The methods proposed by Mizumoto (1981) and Tsukamoto (1977) are seldomly referred to in literature; most applications based on fuzzy truth values are according to Baldwin's method. With respect to practical applicability, it can be stated that the method according to Mizumoto (1981) is the most attractive one, but it should be noted that the results obtained from this fuzzy reasoning method are questionable; see section 6.5.1.3 for an example.

In the approach according to Yager (1980a), an implication is used which complies with the classical implication. The similarity measure is close to a possibility measure and the used implication function has the property that the support of the consequent does not change, except when the similarity measure equals 0. In that case the results is *unknown*. When the *min* operator is used for aggregation in the inference of a set of parallel rules, the result in the case of local inference can be less restrictive than the result possible based on the available data. In the case of the similarity measure approach by Turksen and Zhong (1990), this is not a disadvantage since the approach is similar to fuzzy reasoning where conjunctions are used for implications. This is also the case for the reasoning based on domain scaling.

Linguistic qualifiers can provide an easy-to-implement fuzzy reasoning method. A generalization can be considered where the qualifiers are not chosen from a predefined fixed set of linguistic truth, possibility or probability qualifiers, but are not predefined fuzzy sets.

The disadvantage is that in that case the reasoning becomes less simple, and is similar to the methods based on reasoning with fuzzy truth values as addressed before.

6.6 Conclusions and remarks

In this chapter a number of approaches to the application of fuzzy logic in expert/knowledge-based systems have been addressed. The following summarizes this chapter and contains the main conclusions.

Approaches to fuzzy reasoning based on global inference have the disadvantage that severe calculational efforts and memory requirements are necessary. There are a number of cases where the results of local inference equal the results of global inference. One of those cases is where (pseudo-)conjunctions are used to model implications, which entails that the results of local inference equal the results of global inference. Also reasoning schemes based on similarity measures fall into this category (see section 6.5.2). In some cases analytical solutions are possible, for example, the inference break-up method described in section 6.4.2. This inference break-up method can also be used to reduce the complexity of global inference (section 6.4.2.3).

Many practical approaches to fuzzy reasoning are based on local inference. This entails a loss of specificity in many cases. Reasoning methods based on fuzzy truth values (section 6.5.1) suffer from this also. The main advantage of reasoning with fuzzy truth values is the fact that it reduces the calculus on product spaces. Different approaches to reasoning with fuzzy truth values exist and some even reduce the inference to pure fuzzy number arithmetic (section 6.5.1.3).

Within the framework of approximate reasoning there exists the concept of linguistic qualifiers. Three types of linguistic qualifiers are distinguished: truth, possibility and probability qualifiers (section 6.3.2.2). The truth-qualifiers are directly related to the fuzzy sets used to represent the data and the rule premises. Possibility and probability qualifications induce possibility distributions on possibility and probability distributions, respectively. In case of possibility distributions, alternative approaches exist (sections 6.2.1.3 and 6.4.1). Choosing a fixed set of possible qualifiers provide a straightforward implementation of the reasoning scheme as described in section 6.5.3. It should be noted, however, that the set of linguistic qualifiers is rich enough to express the various input data with respect to the rule premises.

Hence, there is a conflict between what can be seen as "correct" from the theory of approximate reasoning and possibility theory, and practical applicability. Many examples given in literature are based on fuzzy sets defined on discrete domains. This, of course, leads to straightforward implementation. In the case of continuous domains, approximations can be made, or in some restricted cases analytical solutions exist. Inference based on fuzzy truth values according to Baldwin (1979) and Tsukamoto (1977) is interesting from a practical point of view, since approximations of the results can be obtained using level sets. In that case the membership functions (or possibility distribution functions) are discretized instead of the universes. When the inference is broken up according to the method described in 6.4.2, the remaining problem is the inference of a simple rule base in which the rules interact. It was argued in section 6.4.2.3 that dedicated and (highly) optimized software or hardware can play a role in this, since only composition on two-dimensional universes has to be considered.

Using the capability to model vague and imprecise propositions provides the possibility to "communicate" with humans in a way that is close to the way humans communicate. The acceptance of expert systems can be improved when the interfacing with humans is more human-like. Currently, a natural language communication system called FLINS[*] is developed at LIFE[†]. The final goal of FLINS is *"to implement a lingual computer that can communicate and learn, both by being taught and on its own, through the use of a fuzzy natural language"* (Tano et al., 1994; Okamoto et al., 1994). Looking at the still increasing power of computers these days, the severe memory requirements and calculational loads are becoming (relatively) less severe. This provides the means to solve by brute force the problems which cannot be solved analytically.

[*] FLINS stands for Fuzzy LINgual System.

[†] The Laboratory for International Fuzzy Engineering in Japan.

7 Conclusions and suggestions

Conclusions and some suggestions for further research are given in this final chapter. The aim of this thesis is to provide a clear view on fuzzy control: a demystification and at the same time a profilation of the topic. Hopefully, we accomplished in doing so, by describing fuzzy control in general, analyzing the working of a fuzzy controller and considering its place within the framework of approximate reasoning. In the following we only focus on the conclusions and suggestions (a summary is given on page 291).

Fuzzy control provides a method to develop controllers by means of a linguistically expressed control algorithm. This can be regarded as a high level of programming and controller design. We can distinguish between a theoretical (relation-based) approach and a practical (rule-based) approach. The first approach implies global inference, the second approach implies local inference. Local inference means that each rule in inferred and the results of the inferences of the individual rules are aggregated afterwards. Global inference means that the rules are aggregated and used for inference as a whole.
The inference in practical rule-based approach can be characterized by **matching the rule premise with the available data** and **modification of the rule consequent** based on that matching (section 4.2.1.1). Additionally, defuzzification is necessary to obtain numerical controller outputs.

In the field of fuzzy control (and modeling) two types of fuzzy rules can be distinguished: Mamdani rules (section 4.3.1) and Sugeno rules (section 4.3.2). The Mamdani rules have fuzzy propositions as consequents and, therefore, conform more to fuzzy set theory. The Sugeno rules have (linear) functions of the inputs as consequents and are based on a practical approach to fuzzy modeling and control. The **simplest representations of both types of rules are "functionally" equal** in combination with the fuzzy-mean defuzzification method (section 4.3.3). This simple representation is used in many fuzzy control applications.

In fuzzy control, the implication functions to model the rules are normally triangular norms (T-norms), which means that the implication is interpreted as a conjunction (section 4.2.1.2). The rules in fuzzy systems for which a conjunction is used to model the implication can be regarded as possibility qualifying rules from the viewpoint of possibility theory (section 6.4.1.4). If the rules are modeled by implications which comply with the classical implication, a number of problems in the application of fuzzy control can be noticed. The most important **disadvantage of classical-implication-based implications in fuzzy control is the possible indetermination of the output** (sections 4.2.1.3 and 4.2.1.4). A characteristic property of S-implications is the decision-making character of the inference. In general, it can be concluded that for direct fuzzy control, conjunction-based implications are the better choice.

When a fuzzy controller is considered as a input-output mapping which is characterized by tuples in a hyperspace, where each tuple represents a fuzzy rule (modeled by a conjunction-based implication) in the rule base, the fuzzy inference performs interpolation between these tuples in combination with the defuzzification method. This interpolation can exhibit nonlinearities which cannot be influenced by altering the rule base, but are the results of choices for the used operators and membership functions of the fuzzy sets. A number of conclusions concerning these nontrivial nonlinearities can be extracted from the analysis performed in this thesis:

- The center-of-gravity defuzzification method introduces nonlinearities when the aggregation operator is not a summation (section 4.2.4.1).
- The mean-of-maxima defuzzification method introduces discontinuities in the control hypersurface (section 4.2.4.2).
- Other than trapezoidally-shaped membership functions of fuzzy sets for the input result in (nontrivial) nonlinear interpolation (section 4.6.1.3).
- Other operators than the product operator for the conjunction in rule premises result in (nontrivial) nonlinear interpolation (section 4.6.2).

Furthermore, it has been shown that using **more than two overlapping fuzzy sets on input universes yield a "filtering" of the control hypersurface** (section 4.6.1.2). This means that fuzzy rules are not solely responsible for a tuple in the control hyperspace and altering a rule does not have a trivial effect on this control hypersurface.

The nontrivial nonlinearities of the interpolative character of a fuzzy controller can be avoided by meeting the following conditions:

- normal and trapezoidally-shaped fuzzy sets;
- no more than two overlapping fuzzy sets;
- product operator for conjunction in rule premises;
- representing the implication by a conjunction;
- using the fuzzy-mean defuzzification method.

Meeting these requirements entails that the fuzzy controller can be regarded as a look-up table and the inference is performed by means of a weighted-sum interpolation between the elements of that look-up table (section 4.5). If the elements of the look-up table, which represent the (numerical) conclusions of the fuzzy rules, are chosen according to a linear function of the inputs, a linear controller can be “emulated” by a fuzzy controller (section 4.5.2).

In the field of *adaptive fuzzy control*, several approaches can be distinguished: self-organizing fuzzy control, fuzzy relations as associative memories, adaptation by fuzzy supervisors and adaptation by means of gradient-descent optimization.
A characteristic property of the *self-organizing controller* according to Procyk and Mamdani (1979) is that the adaptation is local which hopefully results in global optimization (section 5.1.1). In the case of nonlinear processes and using the error and its differences as controller inputs, this is a major disadvantage, since a different set-point requires a new “learning phase”.
Using *fuzzy relations as associative memories* provides a method to control a process and obtain a process model at the same time (section 5.2). A disadvantage is the fact that in order to obtain a control action, a model is needed to predict future process outputs in case the process exhibits time delays or dynamics which are represented by time delays. This property also plays a role in the field of predictive control.
Fuzzy supervisors can be used to perform adaptations of direct controllers, which can be conventional or fuzzy controllers. Fuzzy supervisors for PID controllers are well-known examples of fuzzy supervisory systems. In some case fuzzy supervisors do not perform “global” adaptation and can be reduced to in-line (direct) fuzzy controllers based on Sugeno rules.
Adaptation by means of *gradient-descent optimization* (section 5.4) is a technique which

is used in "fuzzy neural networks". The membership functions of the fuzzy sets on the input universes and the (numerical) consequents of the fuzzy rules are adapted by a gradient-descent learning rule. In many cases the linguistic interpretability is lost after adaptation. A gradient-descent learning rule which maintains fuzzy partitions on the input universes does preserve linguistic interpretability (section 5.4.3).
Similarities between *adaptive fuzzy systems and other "learning" systems* exist. Comparison of a fuzzy system and RBFN[*] shows that the two systems are different (section 5.5.1). They are functional equivalent when the RBFN performs normalization. A GCMAC[†] system can be regarded as a fuzzy system of which the rule base contains many rules with overlapping premises (section 5.5.2). This results in "filtering" of the control hypersurface as concluded before.
There exist few applications of adaptive fuzzy control.

Because of the defuzzification, a fuzzy controller (or more general, a fuzzy system) performs interpolation between tuples in a hyperspace, where each tuple is represented by a fuzzy rule. Therefore, **a fuzzy controller can be "simplified" to a combination of a look-up table and an interpolation method** (section 4.5). The rule base is represented by the look-up table and the fuzzy inference is represented by the interpolation method.
When this approach is considered with respect to adaptive fuzzy controllers, the following can be derived. The concept of the *self-organizing controller* proposed by Procyk and Mamdani (1979) can be simplified to the interpolation between the elements of a look-up table of which the elements are adapted.
Fuzzy controllers or models which are adapted by means of *gradient-descent optimization* can be simplified to the interpolation between the elements of a look-up table of which the elements as well as the index vectors are adapted. This approach maintains fuzzy partitions for the inputs universes.
The above-given simplifications can reduce the fuzzy aspect of fuzzy control to a user-interfacing concept during the design stage. Together with the fact that fuzzy control uses only a small part of the framework of approximate reasoning where it resides in, one can question whether the success of fuzzy control is because of the use of fuzzy logic.

The application of *approximate reasoning in knowledge-based systems* suffers from a number of problems. When the implications to model the rules are based on conjunction, the inference can be analytically solved. Using conjunctions to implement the implication is usually the case in fuzzy control. For classical-implication-based fuzzy implications an analytical solution is usually not possible. This is due to the fact that the aggregated results of local inference of the individual rules do not entail the same result as that would be obtained when applying global inference using all rules together.
Global inference, however, requires calculations with multi-dimensional functions (fuzzy relations, possibility distributions) in combination with nonlinear operations. There are

[*]Radial Basis Function Network.
[†]Generalized Cerebellar Model Articulation Controller.

two ways to tackle this problem: either use approximations based on the implementation level by means of discretizations, or use local inference instead of global inference.
Approximating fuzzy relations using discretizations results in severe memory requirements or calculational load (section 6.3.3). Using only local inference of individual rules and aggregation of the results can lead to less restrictive results than could be obtained based on the data and knowledge.
In some cases analytical solutions are possible or inference of a rule base can be simplified to the inference of simpler rule bases. An example of this is the break-up of the inference of a rule base as presented in section 6.4.2. Within this method one can distinguish three types of inference break-up:

- the inference of a complex rule can be broken up in the inference of a number of simple rules, provided that the *min* operator is used for conjunction;
- the inference of a complex rule base can be broken up in the inferences of a number of simple rule bases, provided that the *min* operator is used for conjunction;
- the inference of a simple rule base can be broken up in the inference of a number of simple rules, provided that the *min* operator is used for conjunction and the Kleene-Dienes implication is used to model the rules.

This method can simplify the inference of complex rule bases. In case the fuzzy sets defined for the variables used in the premises of the rules meet some requirements, further simplifications of the inference are possible (section 6.4.2.2). The inference break-up method does not provide a general solution but offers a mechanism which enables simplification of inference for some cases. Because of this simplification one can think of dedicated hardware or highly optimized software to perform inference of simple rule bases (section 6.6).

Reasoning with fuzzy truth values (section 6.5.1) has been initiated by Baldwin (1979). This reasoning method does not propagate fuzzy sets or possibility distributions on the universes of discourse of the used variables, but reasons with fuzzy truth values which are fuzzy sets defined for the interval $[0, 1]$; the numerical values 0 and 1 correspond to *absolutely false* and *absolutely true*, respectively. Several approaches to reasoning with fuzzy truth values are known from literature, including methods which reduce the inference to fuzzy number calculus. The characteristic property of reasoning with fuzzy truth values is that local inference is performed. This entails that in many cases the result is less restrictive than it can be, based on the available data and knowledge. Using inference break-up, this problem can only partly be solved.

Another approach to fuzzy reasoning is reasoning in which a clear distinction is made between *a matching phase and a modification phase* (section 6.5.2). These approaches are close, and in some cases equal, to the type of reasoning used in fuzzy control. Different approaches can be distinguished and they are based on local inference. The local inference does not pose problems in most cases since implications in these approaches are normally represented by conjunctions.

Considering the performed analysis of fuzzy control and the framework of approximate reasoning and derived reasoning schemes, it can be stated that fuzzy control is based on only a small part of the much broader field of approximate reasoning. Today, considering the numerous applications, fuzzy control is a more or less accepted type of control. However, applications of approximate reasoning or derived reasoning schemes can barely be found in literature. In our opinion approximate reasoning could and should be used, since it can provide a user-friendly knowledge representation and a reasoning method which can model human reasoning in higher level expert systems. In the field of control, this entails planning, scheduling and plant-wide supervision. To apply approximate reasoning in these fields, (more) software tools should be developed and become available for industry.

So, finally we return to the issue of stability of fuzzy controllers which, thus far, was only addressed in the introduction of this thesis. When a fuzzy controller is regarded as a static nonlinearity, for example represented by a look-up table and interpolation method, this can provide a starting point for stability analysis if a stability analysis is desired. However, we still support Mamdani's view as presented by the quote on page 9. Considering higher level expert systems for control, requiring a stability analysis is not realistic and "stability" should be interpreted as "common sense" stability and "proved" by working prototypes.

A

Fuzzy logic operators

In this appendix fuzzy set and fuzzy logic operators are summarized. Various fuzzy complement operators are given in table A.1. Table A.2 lists fuzzy intersection and union operators. Fuzzy implications are summarized in table A.3. The references in the tables are partly from Bandler and Kohout (1980), Dubois and Prade (1991) and Gupta and Qi (1991b).

Table A.1: *Fuzzy complements* $\mathrm{c}(a)$.

$\mathrm{c}(a)$	parameters	references
$1 - a$		Zadeh (1973)
$\dfrac{1-a}{1+\lambda a}$	$\lambda > 1$	Sugeno (1977)
$\sqrt[p]{(1-a^p)}$	$p > 0$	Yager (1980b)

Table A.2: *Fuzzy intersections* $\mathrm{T}(a,b)$ *and unions* $\mathrm{S}(a,b)$.

$\mathrm{T}(a,b)$	$\mathrm{S}(a,b)$	parameters	references
$\min(a,b)$	$\max(a,b)$		Zadeh (1973)
ab	$a+b-ab$		Bandler and Kohout (1980)
$\max(a+b-1,0)$	$\min(a+b,1)$		Łukasiewicz, Giles (1976)
$\begin{cases} x, & \text{if } y=1 \\ y, & \text{if } x=1 \\ 0, & \text{otherwise} \end{cases}$	$\begin{cases} x, & \text{if } y=0 \\ y, & \text{if } x=0 \\ 1, & \text{otherwise} \end{cases}$		Weber (1983)
$\dfrac{ab}{\gamma+(1-\gamma)(a+b-ab)}$	$\dfrac{a+b-(2-\gamma)ab}{1-(1-\gamma)ab}$	$\gamma>0$	Hamacher (1978)
$\dfrac{ab}{\max(a,b,\alpha)}$	$\dfrac{a+b-ab-\min(a,b,1-\alpha)}{\max(1-a,1-b,\alpha)}$	$\alpha\in[0,1]$	Dubois and Prade (1986)
$\left[1+\sqrt[\lambda]{(\frac{1}{a}-1)^{\lambda}+(\frac{1}{b}-1)^{\lambda}}\right]^{-1}$	$\left[1+\sqrt[\lambda]{(\frac{1}{a}-1)^{-\lambda}+(\frac{1}{b}-1)^{-\lambda}}\right]^{-1}$	$\lambda>0$	Dombi (1982)
$1-\sqrt[p]{(1-a)^p+(1-b)^p-(1-a)^p(1-b)^p}$	$\sqrt[p]{a^p+b^p-a^pb^p}$	$p>0$	Schweizer and Sklar (1961)
$\max(1-\sqrt[p]{(1-a)^p+(1-b)^p},0)$	$\min(\sqrt[p]{a^p+b^p},1)$	$p\geq 1$	Yager (1980b)
$\max(\dfrac{a+b-1+\lambda ab}{1+\lambda},0)$	$\min(a+b+\lambda ab,1)$	$\lambda\geq -1$	Weber (1983)
${}^{w}\log\left(1+\dfrac{(w^a-1)(w^b-1)}{w-1}\right)$	$1-{}^{w}\log\left(1+\dfrac{w^{1-a}+w^{1-b}}{w-1}\right)$	$w>0, w\neq 1$	Turksen (1986)

Table A.3: *Fuzzy implications* $\mathrm{I}(a, b)$.

$\mathrm{I}(a, b)$	references
$\min(1 - a + b, 1)$	Łukasiewicz
$\min(1 - a, \min(a, b))$	Zadeh (1975)
$\max(1 - a, b)$	Kleene (1938)
$1 - a + ab$	Reichenbach
$\begin{cases} b, & \text{if } a > b \\ 1, & \text{otherwise} \end{cases}$	Gödel
$\begin{cases} 1 - a, & \text{if } b = 0 \\ b, & \text{if } a = 1 \\ 1, & \text{otherwise} \end{cases}$	Dubois and Prade
$\begin{cases} 1, & \text{if } a = 0 \\ \min(\frac{b}{a}, 1), & \text{otherwise} \end{cases}$	Goguen (1969)
$\begin{cases} 1, & \text{if } a \leq b \\ 0, & \text{otherwise} \end{cases}$	Gaines (1976)
$\begin{cases} 1, & \text{if } a \leq b \\ \min(1 - a, b), & \text{otherwise} \end{cases}$	Wu (1986)
$\begin{cases} 0, & \text{if } a < b \\ b, & \text{otherwise} \end{cases}$	Wu (1986)
b^a	Yager (1980a)
$\min(\max(1 - a, b), \max(a, 1 - b, \min(b, 1 - a)))$	Willmott (1980)
$\min(a, b)$	Mamdani (1974)
ab	Larsen (1980)

B

Linear controller $\subset$ fuzzy controller proof

In the following, it will be shown that a fuzzy controller can "emulate" a linear controller. The proof assumes that the criteria are fulfilled which were given in section 4.5.2. These criteria are:

L-1 the membership functions of the fuzzy sets on the universe of discourse of the inputs are triangularly shaped and normal;

L-2 the fuzzy sets for each input form a fuzzy partition: the sum of the membership functions equals 1;

L-3 the fuzzy rule base is complete;

L-4 a T-norm is used for the implication function (T-implication);

L-5 the operator for the conjunction in the premises of the fuzzy rules is the *product* operator;

L-6 the *(bounded) sum* operator (union according to Łukasiewicz) is used for the aggregation and for the *or* connective if it is used;

L-7 the defuzzified consequents (constant numerical representations) of the individual fuzzy rules are chosen according to equation (4.51);

L-8 the fuzzy-mean defuzzification method is used; this implies the choice for the aggregation operator in L-6.

Because the aggregation is a summation, the fuzzy controller output y' is described by applying the fuzzy-mean defuzzification directly by using the consequents of all (contributing) fuzzy rules:

$$y' = \frac{\sum_{k=1}^{N_r} \beta_k b_k}{\sum_{k=1}^{N_r} \beta_k} \tag{B.1}$$

where b_k is the numerical consequent of fuzzy rule r_k; numerical representation of the fuzzy consequent B_k in case of Mamdani rules. Furthermore, because of criteria L-2 to L-4 and L-6 we can write:

$$\begin{aligned}
\sum_{k=1}^{N_r} \beta_k &= \sum_{k=1}^{2^{N_X}} \beta_k && \text{(B.2a)}\\
&= \sum_{k=1}^{2^{N_X}} \prod_{i=1}^{N_X} \alpha_{i,k} && \text{(B.2b)}\\
&= (\alpha_{1,l} + \alpha_{1,r}) \sum_{k=1}^{2^{N_X-1}} \prod_{i=2}^{N_X} \alpha_{i,k} && \text{(B.2c)}\\
&= \ldots\\
&= \prod_{i=1}^{h} (\alpha_{1,l} + \alpha_{1,r}) \sum_{k=1}^{2^{N_X-h+1}} \prod_{i=h}^{N_X} \alpha_{i,k} && \text{(B.2d)}\\
&= \ldots\\
&= \prod_{i=1}^{N_X} (\alpha_{i,l} + \alpha_{i,r}) && \text{(B.2e)}\\
&= \prod_{i=1}^{N_X} 1 && \text{(B.2f)}\\
&= 1 && \text{(B.2g)}
\end{aligned}$$

where $\alpha_{i,l}$ is the “left” fuzzy set and $\alpha_{i,r}$ is the “right” fuzzy set as depicted in figure B.1.

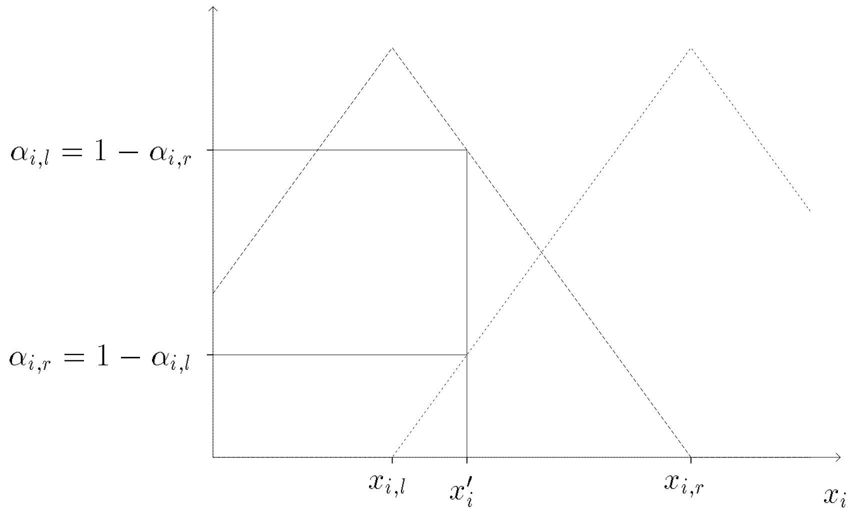

$\alpha_{i,l} = 1 - \alpha_{i,r}$
$\alpha_{i,r} = 1 - \alpha_{i,l}$
$x_{i,l}$
x'_i
$x_{i,r}$
x_i

Figure B.1: *Determining* $\alpha_{i,l}$ *and* $\alpha_{i,r}$.

What is left to proove is:

$$\sum_{k=1}^{2^{N_X}} \beta_k b_k = \boldsymbol{c}^T \boldsymbol{x}' + d \tag{B.3}$$

of which the left-hand side can be written as:

$$\sum_{k=1}^{2^{N_X}} \beta_k b_k = \sum_{k=1}^{2^{N_X}} \beta_k (\boldsymbol{c}^T \boldsymbol{x}_k + d) \tag{B.4a}$$

$$= \sum_{k=1}^{2^{N_X}} \beta_k \boldsymbol{c}^T \boldsymbol{x}_k \ + \ \sum_{k=1}^{2^{N_X}} \beta_k d \tag{B.4b}$$

$$= c^T \sum_{k=1}^{2^{N_X}} \beta_k \boldsymbol{x}_k \ + \ d \sum_{k=1}^{2^{N_X}} \beta_k \tag{B.4c}$$

$$= c^T \sum_{k=1}^{2^{N_X}} \beta_k \boldsymbol{x}_k \ + \ d \tag{B.4d}$$

Hence, the proof is reduced to proving:

$$\sum_{k}^{2^{N_X}} \beta^k \boldsymbol{x}^k = \boldsymbol{x}' \tag{B.5}$$

For the i^{th} input this can written as:

$$\sum_{k=1}^{2^{N_X}} \beta_k x_{i,k} = \sum_{k=1}^{2^{N_X}} \prod_{i=1}^{N_X} \alpha_{i_k} x_{i_k} \tag{B.6a}$$

$$= \sum_{k=1}^{2^{N_X-1}} \left\{ \alpha_{i,l} x_{i,l} \prod_{\substack{h=1\\h\neq i}}^{N_X} \alpha_{h_k} + \alpha_{i,r} x_{i,r} \prod_{\substack{h=1\\h\neq i}}^{N_X} \alpha_{h_k} \right\} \tag{B.6b}$$

$$= (\alpha_{i,l} x_{i,l} + \alpha_{i,r} x_{i,r}) \sum_{k=1}^{2^{N_X-1}} \prod_{\substack{h=1\\h\neq i}}^{N_X} \alpha_{h_k} \tag{B.6c}$$

$$= \alpha_{i,l} x_{i,l} + \alpha_{i,r} x_{i,r} \tag{B.6d}$$

which is equal to x_i', because $\alpha_{i,r} = 1 - \alpha_{i,l}$ and:

$$\alpha_{i,l} = \frac{x_{i,r} - x_i'}{x_{i,r} - x_{i,l}} \tag{B.7}$$

This means that any linear system represented by a static functional description can be emulated by a fuzzy system. The fuzzy system can be viewed as a look-up table and (linear) interpolation is performed between the elements of the look-up table. The elements fulfill the static functional description representing the linear controller (or model).

C

Derivation of restricted learning rule

The derivation of the restricted learning rule as described in section 5.4.3 is given in this appendix. This learning rule maintains fuzzy partitions on the input universes. By maintaining these fuzzy partitions on the input universes, the interpretability of the rules after (and during) adaptation is maintained.

The following criterion is supposed to be minimized (Guély and Siarry, 1993):

$$E = \sum_{n=1}^{N_s} E_n \tag{C.1a}$$

$$= \sum_{n=1}^{N_s} \tfrac{1}{2}(\tilde{y}_n - y_n)^2 \tag{C.1b}$$

where $\tilde{y}_n$ is the n^{th} reference for output y_n and N_s is the number of samples. Parameters p are updated according to:

$$\Delta p = -\frac{K_p}{N_s}\frac{\partial E}{\partial p} \tag{C.2a}$$

$$= -\frac{K_p}{N_s}\sum_{n=1}^{N_s}\frac{\partial E_n}{\partial p} \tag{C.2b}$$

where K_p is the learning factor (speed factor) and N_s is the number of samples in a batch used for learning. For simplification e, y and $\tilde{y}$ are used instead of E, y_p and $\tilde{y}_p$. Thus $\partial E_n/\partial p$ is denoted by:

$$\frac{\partial e}{\partial p} = \frac{\partial e}{\partial y}\frac{\partial y}{\partial p} \tag{C.3a}$$

$$= (y - \tilde{y})\frac{\partial y}{\partial p} \tag{C.3b}$$

So this leaves me to determine the partial derivatives of y to the parameters that are to be adapted. Before determining the partial derivative $\partial y/\partial b^k$ and $\partial y/\partial a_i^j$ the membership functions have to be known. The triangularly-shaped membership functions used in the restricted learning method are defined by:

$$\mu_{A_{i,j}}(x_i) = \begin{cases} \dfrac{x_i - a_{i,j-1}}{a_{i,j} - a_{i,j-1}}, & \text{if } a_{i,j-1} \le x_i \le a_{i,j} \\ \dfrac{x_i - a_{i,j+1}}{a_{i,j} - a_{i,j+1}}, & \text{if } a_{i,j} \le x_i \le a_{i,j+1} \\ 0, & \text{otherwise} \end{cases} \tag{C.4}$$

Note that these membership functions are convex and form a fuzzy partition on each universe. Because the *product* operator is used for *and* connective, the *sum* operator is used for aggregation and the fuzzy-mean method is used for defuzzification, a complete rule base results in:

$$\sum_{k=1}^{N_r}\beta_k = 1 \tag{C.5}$$

First the partial derivative of y to b_k is derived:

$$\frac{\partial y}{\partial b_k} = \frac{\beta_k}{\sum_{k=1}^{N_r}\beta_k} = \beta_k \tag{C.6}$$

The partial derivative $\partial y/\partial a_{i,j}$ is determined by:

$$\frac{\partial y}{\partial a_{i,j}} = \frac{\partial y}{\partial \mu_{A_{i,j-1}}}\frac{\partial \mu_{A_{i,j-1}}}{\partial a_{i,j}} + \frac{\partial y}{\partial \mu_{A_{i,j}}}\frac{\partial \mu_{A_{i,j}}}{\partial a_{i,j}} + \frac{\partial y}{\partial \mu_{A_{i,j+1}}}\frac{\partial \mu_{A_{i,j+1}}}{\partial a_{i,j}} \tag{C.7}$$

where $\partial y/\partial \mu_{A_{i,j}}$ is given by:

$$\frac{\partial y}{\partial \mu_{A_{i,j}}} = \frac{\sum\limits_{k'=1}^{N_{A_{i,j}}} \beta^{k'} b^{k'} - y \sum\limits_{k'=1}^{N_{A_{i,j}}} \beta^{k'}}{\mu_{A_{i,j}}(x_i) \sum\limits_{k=1}^{N_r} \beta^k} = \mu_{A_{i,j}}^{-1}(x_i) \sum_{k'=1}^{N_{A_{i,j}}} \beta^{k'}(b^{k'} - y) \tag{C.8}$$

This leaves us to derive the following partial derivatives:

$$\frac{\partial \mu_{A_{i,j-1}}}{\partial a_{i,j}} = \begin{cases} \dfrac{\mu_{A_{i,j}}(x_i)}{a_{i,j} - a_{i,j-1}}, & \text{if } a_{i,j-1} < x_i < a_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{C.9}$$

$$\frac{\partial \mu_{A_{i,j}}}{\partial a_{i,j}} = \begin{cases} \dfrac{-\mu_{A_{i,j}}(x_i)}{a_{i,j} - a_{i,j-1}}, & \text{if } a_{i,j-1} < x_i < a_{i,j} \\ \dfrac{-\mu_{A_{i,j}}(x_i)}{a_{i,j} - a_{i,j+1}}, & \text{if } a_{i,j} < x_i < a_{i,j+1} \\ 0, & \text{otherwise} \end{cases} \tag{C.10}$$

$$\frac{\partial \mu_{A_{i,j+1}}}{\partial a_{i,j}} = \begin{cases} \dfrac{\mu_{A_{i,j}}(x_i)}{a_{i,j} - a_{i,j+1}}, & \text{if } a_{i,j} < x_i < a_{i,j+1} \\ 0, & \text{otherwise} \end{cases} \tag{C.11}$$

Now, all necessary partial derivatives are derived and result in the learning rules as given by (5.17) on page 175:

$$\Delta a_{i,j} = \begin{cases} \dfrac{K_a(y - \tilde{y})}{(a_{i,j} - a_{i,j-1})} \left[\dfrac{\mu_{A_{i,j}}(x_i)}{\mu_{A_{i,j-1}}(x_i)} \sum\limits_{k'=1}^{N_{A_{i,j-1}}} \beta_{k'}(b_{k'} - y) \right. \\ \qquad \left. - \sum\limits_{k'=1}^{N_{A_{i,j}}} \beta_{k'}(b_{k'} - y) \right], & \text{if } a_{i,j-1} < x_i < a_{i,j} \\ \dfrac{K_a(y - \tilde{y})}{(a_{i,j} - a_{i,j+1})} \left[\dfrac{\mu_{A_{i,j}}(x_i)}{\mu_{A_{i,j+1}}(x_i)} \sum\limits_{k'=1}^{N_{A_{i,j+1}}} \beta_{k'}(b_{k'} - y) \right. \\ \qquad \left. - \sum\limits_{k'=1}^{N_{A_{i,j}}} \beta_{k'}(b_{k'} - y) \right], & \text{if } a_{i,j} < x_i < a_{i,j+1} \\ 0, & \text{otherwise} \end{cases} \tag{C.12a}$$

$$\Delta b_k = K_b \beta_k (y - \tilde{y}) \tag{C.12b}$$

Clearly, $a_{i,j-1} < a_{i,j} < a_{i,j+1}$ should be maintained. Application of this learning rule can be considered as adaptation of the elements as well as the index vectors of a look-up table.

D

GCMAC: Generalized Cerebellar Model Articulation Controller

In section 5.5.2, a comparison was described between a fuzzy system and a CMAC using kernel functions with values in the interval $[0, 1]$ instead of values restricted by $\{0, 1\}$. This appendix will give a small overview of GCMAC, a generalized CMAC (Krijgsman and Jager, 1993b).

The CMAC algorithm as proposed by Albus (1975b, 1975a), can be seen as a look-up table with additional extensions. These extensions are the following (Krijgsman, 1993):

- **Generalization** of inputs is performed. This causes "neighboring" input vectors to be considered when a certain input vector is considered. Hence, **distributed storage** is used to store learned input-output combinations.
- **"Random" mapping** is used to map elements in a virtual memory (not bounded in size) to the actual memory (bounded in size). Collisions can occur because the virtual memory is mapped onto a smaller actual memory. Because of the randomness of the mapping, the chances of (systematic) collisions due to this are minimized.

The working of the (G)CMAC is described by the following:

- The input vector $\boldsymbol{x}$ is generalized into a set of (weighted) input vectors $\boldsymbol{x}_g$ by means of a kernel function $\Phi(\boldsymbol{x}, \boldsymbol{x}_g)$. Albus (1975b) used a binary kernel function in the original CMAC:

  $$\Phi(\boldsymbol{x}, \boldsymbol{x}_g) = \begin{cases} 1, & \text{if } \| \boldsymbol{x} - \boldsymbol{x}_g \|_\infty < \frac{\rho}{2} \\ 0, & \text{otherwise} \end{cases} \tag{D.1}$$

  where ρ is a generalization factor. The *generalization of the CMAC* allows kernel functions to have values in the interval $[0, 1]$. Hence, (D.1) is generalized to:

  $$\Phi(\boldsymbol{x}, \boldsymbol{x}_g) \in [0, 1] \tag{D.2}$$

  An example is the following kernel function:

  $$\Phi(\boldsymbol{x}, \boldsymbol{x}_g) = \max(1 - \alpha \| \boldsymbol{x} - \boldsymbol{x}_g \|_n, 0) \tag{D.3}$$

  where $\alpha \in \mathbb{R}$ is a scaling factor and $\| \cdot \|_n$ is the n-norm and. Examples of such kernel functions are shown in figure D.1.

- Each vector $\boldsymbol{x}_g$ is mapped into a memory cell c_g of the actual memory C by means of "random" mapping. Collisions occur when different input vectors $\boldsymbol{x}_g$ address the same memory cell c_g.

- An output is retrieved by:

  $$y = \frac{\sum_{g=1}^{N_g} \Phi(\boldsymbol{x}, \boldsymbol{x}_g)\, c_g}{\sum_{g=1}^{N_g} \Phi(\boldsymbol{x}, \boldsymbol{x}_g)} \tag{D.4}$$

  where g is used to index the input vectors $\boldsymbol{x}_g$ within the generalization of input vector $\boldsymbol{x}$.

- Learning is performed according to a Widrow-Hoff updating rule (Krijgsman, 1993). Each cell c_g is updated by:

$$\Delta c_g = \lambda_g \, (\tilde{y} - y) \, \Phi(\boldsymbol{x}, \boldsymbol{x}_g) \tag{D.5}$$

  where $\lambda_g \in [0, 1]$ is the learning factor for memory cell c_g and $\tilde{y}$ is the desired output.

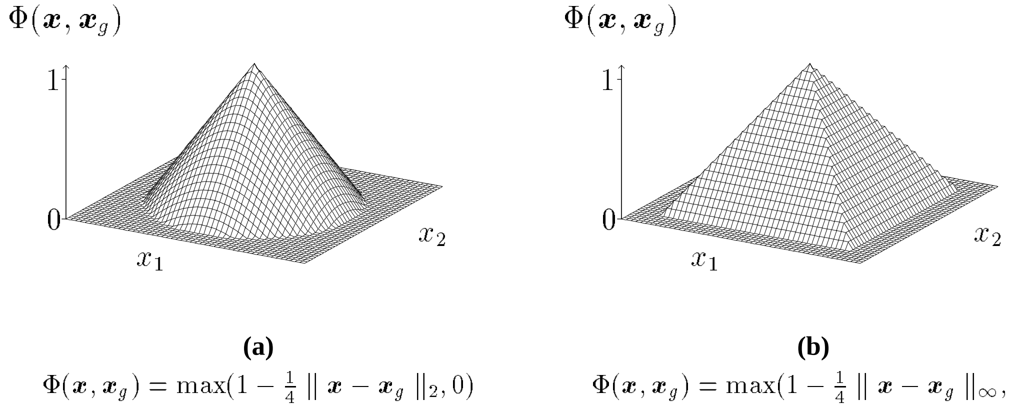


Figure D.1: *Examples of "fuzzy" kernel functions.*

To test the generalized CMAC, the GCMAC algorithm has been implemented in C++[*]. The GCMAC implementation has been used for quite a number of experiments (Krijgsman and Jager, 1993b; van Kats, 1993; Welling, 1994). These experiments showed that "fuzzy" kernel functions result in improvements with respect to learning performance compared to binary kernel functions.

[*]The software can be obtained from the author.

E

RICE: Routines for Implementing C Expert systems

In this appendix, a short description of RICE[*] is given. RICE stands for *Routines for Implementing C Expert systems* and is a C software library which provides an inference engine and supporting tools. RICE has been used at the Control laboratory, as well as outside the laboratory (Peitsman and van Duyvenvoorde, 1993; Peitsman, 1993; Li et al., 1994). First the inference engine and supporting tools are described in the following subsection. The use of RICE in simulation or control environments is shown by a simple example in section E.2.

E.1 The inference engine and supporting tools

The fuzzy inference engine is capable of dealing with two ways of knowledge representation: *rules* and *relations*. Both approaches influence a grade of possibility or truth

[*]RICE is available from the author and, among other sites, the CMU Artificial Intelligence Repository (`http://www.cs.cmu.edu/Web/Groups/AI/html/repository.html`).

(corresponding with grade of membership in case of inputs and outputs) of symbolic statements, for example *"error is negative big"* or *"weather is nice"*. The inference engine has several features to perform inference using the knowledge base, containing rules and relations, and available data. The following list gives an overview of the capabilities of the RICE library:

- **Rules** are based on simple *if-then* statements. They have an antecedent, consisting of conditions, and a consequent, consisting of actions. The conditions can be combined by an intersection (*and*) or a union (*or*). The actions are only fired in case the antecedent of the rule is *not more false than true*. The level *not more false than true* can be adjusted for each action within a rule. Thus, a fuzzy rule as used in fuzzy control can be implemented by choosing this level equal to false (0). The actions in the consequent of the rule can be of the *then* or the *else* type.

- **Relations** do not have an antecedent and a consequent but only consist of *dimensions*. The use of the word *dimensions* has its origin in the multi-dimensional functions representing fuzzy rules theoretically. This results in a kind of reversible fuzzy rules, which can be used by the inference engine to infer one dimension from the rest of the dimensions by applying a intersection on those. For example, a bi-implication as used in first-order predicate logic can be achieved by defining a relation with just two dimensions.

- **Forward reasoning**. Also known as bottom-up or data-driven reasoning. The inference engine tries to prove as much as possible given an initial set of data. In case of fuzzy control a control signal will be inferred using the controller inputs and the knowledge base.

- **Backward reasoning**, also known as top-down or goal-driven reasoning, is used by the inference engine to prove a priori known goals. In fuzzy control the goal is to obtain a control signal each sampling instant. To derive a new controller output, the inference engine uses the knowledge base and the given data (controller inputs).

- **Progressive reasoning** is provided for real-time applications (Lattimer Wright et al., 1986; Jager et al., 1990; Krijgsman et al., 1991). For the application of progressive reasoning all knowledge in the knowledge base can hierarchically be divided in knowledge layers, which are inferred in a specific sequence. A knowledge layer can be seen as a knowledge base which is an extension of "lower" knowledge layers: every "higher" knowledge layer contains "higher/deeper" knowledge and "includes" the knowledge (static as well as dynamic) of the lower ones.

- **Explanation utilities**, although primitive, provide some tools for debugging. Implemented are the so-called *how* and *why* facilities.

- **Focus of attention** is provided by means of activation or deactivation of knowledge layers. Using knowledge layers for this purpose does not necessarily imply a hierarchy among the knowledge layers.

- **Knowledge base compilation** is used to build an internal representation of the knowledge base. This prevents run-time search through the knowledge base and thus faster execution. Incremental compilation of knowledge bases is possible.

- **Linking symbolic statements with C code** provides a way to connect the system to the outside world, a symbolic statement inside the knowledge base can be "linked" with source code, written in the C programming language. Argument passing by symbolic statements in the knowledge base to the C code is possible. This offers the ability to reuse symbolic statements, like programmed routines, and thus limit the number of symbolic statements and C code. The source code can be used to perform, for example, classifications of measurements and defuzzification of fuzzy outputs of the system.

The symbolic statements in the knowledge base are used to represent the "dynamic" knowledge, the implications and relations are used to represent the "static" knowledge. The terms "dynamic" and "static" are not to be taken literally, because within the knowledge base it is, for example, possible to alter the *grade of possibility/truth* of actions of implications and/or dimensions of relations. This provides on-line adaptation or tuning of rules.

In the application of rules as well as relations it is possible to have different interpretations of the intersection and union in the fuzzy inference engine. Several T-norms and S-norms, like the ones according to Zadeh, Łukasiewicz or probability theory are already implemented, as well as functions for using several "families" of T- and S-norms (see section 2.3.1) are offered. Those families or other, user supplied, types of T-norms and S-norms can be used in the inference engine to represent the *and* and *or* operation.

When using the fuzzy inference engine, it is possible that the conditions of rules are actions of other rules. The first question that arises is: "What happens with the intermediate results?". The relevance of this question depends on the way the rule base is composed. Rules with actions which are used as conditions in other rules can be regarded as a kind of short-hand notation for implementing more complex rules. In another perspective one can conclude that such a rule actually has defined a new imaginary membership function for a specific input. The correctness of this interpretation depends on the conditions and actions defined in the rule. Whether or not the result of the fuzzy inference engine is in agreement with the compositional rule of inference, depends on the correctness of the fuzzy rules implemented and the coherence between them.

Another possibility is the fact that a rule does not have a condition for every input available, which is however not relevant when assuming the following:

- the antecedent of every rule should imaginary contain a classification of every input;
- the membership grade of a "missing" input classification in the antecedent of the rule is assumed 1 (true).

From this point of view, rules are defined for every possible symbolic combination of the inputs, although not every rule is really individually provided in the rule base. Because missing input classification are assumed to have a membership grade of 1, in fact the union according to Łukasiewicz is applied on all classifications of the input in question, assuming consistent choices of membership functions for those classifications; see also section 4.6.2 on this topic. So, in fact, a large imaginary rule base is defined which has a fuzzy rule for every situation, but only the relevant parts are actually provided. In this way it is possible to design fuzzy controllers which use different inputs in different situations: the inputs which are not relevant are simply ignored by the system.

E.2 Examples using RICE in simulation and control

In this section an example is given in which the fuzzy inference engine is used in a control system for the level control of a water column, as shown in figure E.1. Although it is a simple control problem, it is sufficient for showing how the fuzzy inference engine can be easily embedded in a simulation and/or real-time control environment. First we set up the knowledge base as shown in listing E.1. The `rem` keyword is used for comment (remarks).

Input as in listing E.1 is an easy way of setting up a knowledge base. To do so, two macro's were defined to transform this to the maximum number of 49 rules, expected by the compiler of the fuzzy inference engine. In the file included by the knowledge base, the two necessary macro's are defined (see listing E.2).

The addition `run` to the `and` and `then` keywords makes the link between symbolic statements and code in the C programming language. Examples of such links are shown in listing E.3, where the `link` and `grade` are keywords. The `link` keyword links a symbolic statement with C code. The `grade` keyword represents the grade of membership or truth, which can be assigned a value (in a condition) or which its value can be used (in an action). The first `link` statement determines the membership value of the error in case of a trapezoidal (`trap(...)`) membership function, the second stores the grade of truth,

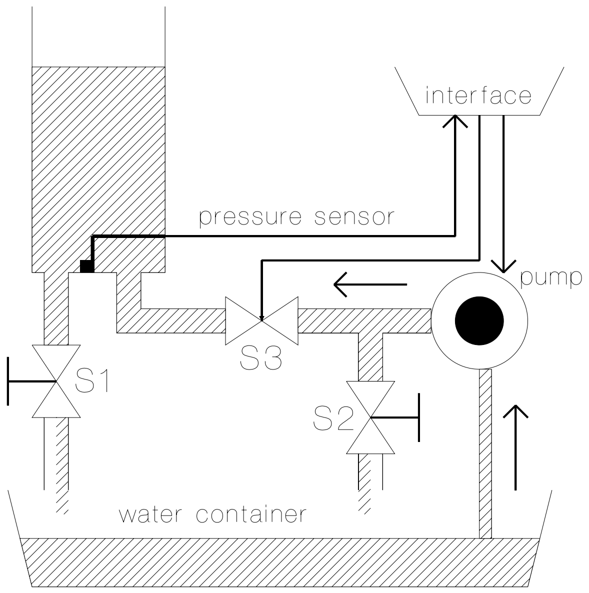


Figure E.1: *Standpipe used in level control problem.*

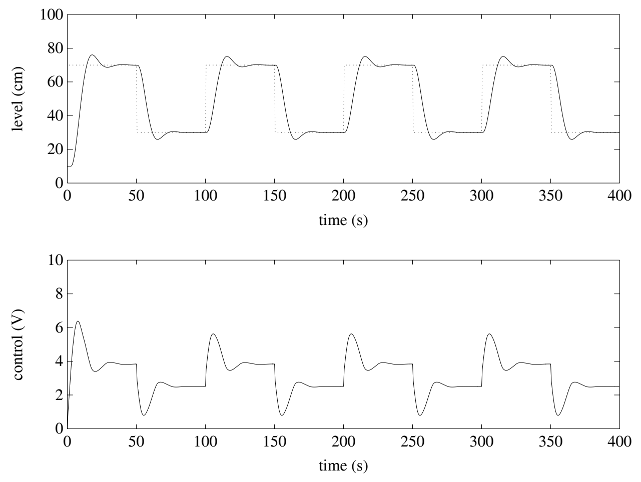
determined by the fuzzy inference engine, in an array (`du_grade[]` in this example). This array can be used to determine the crisp value for the control signal change by means of defuzzification. All symbolic statements in the knowledge base can be linked this way, if necessary. In this example a total 21 (3×7) linkages between symbolic statements and C code were defined. This could be limited to 3 when using the possibility to work with argument passing from symbolic statements in the knowledge base to the C code.

For simulation experiments the fuzzy inference engine was used in the simulation package CSIM[*] on an IBM-compatible computer (40486DX/33Mhz). The result of the simulated level control of the water column is shown in figure E.2a. In order to perform the real-time experiments, the fuzzy inference engine was used in a simulation and real-time control environment, called MUSIC[†] (Cser et al., 1986), running on VAX/VMS[‡] stations. In fact the fuzzy inference engine can be used on any platform where a ANSI-C compiler is available (ports of RICE to real-time operating system DECelx and Unix environments are known to work). In figure E.2b the results of the real-time experiment of the level control are shown.

[*]CSIM is developed at the Control laboratory.

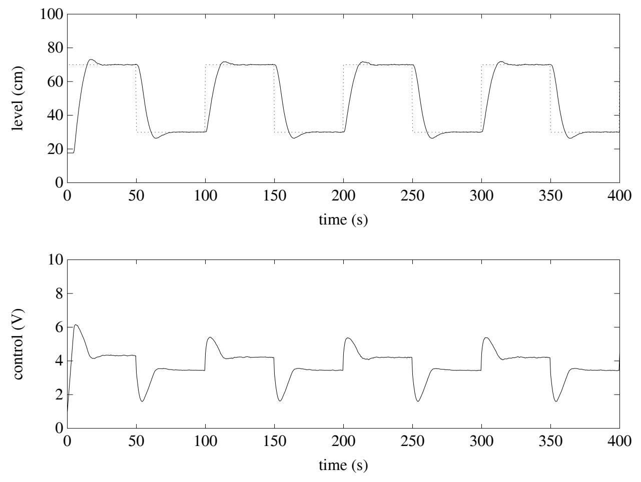
[†]MUSIC is an acronym for MUlti-purpose SImulation and Control.

[‡]VAX is a trademark of Digital Equipment.



(a)

simulation



(b)

real-time control

Figure E.2: *Results of water-level control by fuzzy PI controller implemented in RICE in combination with simulation package CSIM* **(a)** *and real-time control package MUSIC* **(b)**.

```
rem the knowledge base as a matrix
rem horizontal : error (e)
rem vertical   : error change (ce)
rem contents   : control change

include fuzzypi.ki

rem     ce\e : NB : NM : NS : AZ : PS : PM : PB
rem     ..   : .. : .. : .. : .. : .. : .. : ..
PI-row  PB   : AZ : PS : PM : PB : PB : PB : PB
PI-row  PM   : NS : AZ : PS : PM : PB : PB : PB
PI-row  PS   : NM : NS : AZ : PS : PM : PB : PB
PI-row  AZ   : NB : NM : NS : AZ : PS : PM : PB
PI-row  NS   : NB : NB : NM : NS : AZ : PS : PM
PI-row  NM   : NB : NB : NB : NM : NS : AZ : PS
PI-row  NB   : NB : NB : NB : NB : NM : NS : AZ
```

Listing E.1: *The file* `fuzzypi.kb` *containing the knowledge base for the fuzzy PI controller.*

The example and experiments, as discussed in this section, give an idea of the relation of the developed fuzzy inference engine and the application of fuzzy control. Fuzzy controllers are easily implemented within a fuzzy inference expert system shell, because they use only a small part of the capacity of such a tool. The advantage of a fuzzy expert system shell is the ability to “add” knowledge, exceptions, daemons, etc.

```
rem a macro for a fuzzy rule
rem - Zadeh-type of AND-operation (min)
rem - weight/truth value of 1 (true)
rem - logical threshold of 0 (false)

define PI-rule
zandrun error is #1
zandrun error change is #2
pthenrun change contol #3 : 1 : 0

rem a macro for a ’row’ of the fuzzy rule base

define PI-row
PI-rule NB : #1 : #2
PI-rule NM : #1 : #3
PI-rule NS : #1 : #4
PI-rule AZ : #1 : #5
PI-rule PS : #1 : #6
PI-rule PM : #1 : #7
PI-rule PB : #1 : #8
```

Listing E.2: *The included file* `fuzzypi.ki` *with macro definitions.*

```
link("error is NM",
   grade = trap(e, -4, -3, -2.5, 1);
);

link("change control PS",
   du_grade[4] = grade;
);
```

Listing E.3: *Examples of linking symbolic statements with C code, where* `grade` *and* `link` *are keywords.*

F

Proofs for inference break-up method

In this appendix, the proofs are given for the inference break-up method as described in section 6.4.2. First it is shown how a rule can be broken up. After this, the break-up of the inference of a single rule is addressed. The last section deals with the inference break-up of a rule base.

F.1 Rule break-up

Demirli and Turksen (1992) proved for general S- and R-implications that:

$$\mathrm{I}(A_1 \cap A_2, B) = \mathrm{I}(A_1, B) \cup \mathrm{I}(A_2, B) \tag{F.1}$$

where the cylindrical extension is implicit, and which can be considered element-wise by $f(a_{1,i_1} \wedge a_{2,i_2}, b_j)$ for any i_1, i_2, j. Since S- and R-implications are nonincreasing with respect to their first arguments and the *min* operation is used for the conjunction in the premise of the rule, the following two cases can be distinguished:

1. $f(a_{1,i_1} \wedge a_{2,i_2}, b_j) = f(a_{1,i_1}, b_j)$, when $a_{1,i_1} \leq a_{2,i_2}$, because of the *min* operation for conjunction, and $f(a_{1,i_1}, b_j) \geq f(a_{2,i_2}, b_j)$, because of the fact that S- and R-implications are nonincreasing with respect to their first arguments;

2. $f(a_{1,i_1} \wedge a_{2,i_2}, b_j) = f(a_{2,i_2}, b_j)$, when $a_{1,i_1} \geq a_{2,i_2}$ and $f(a_{1,i_1}, b_j) \leq f(a_{2,i_2}, b_j)$, for the same reasons as before.

The combination of these two cases results in $f(a_{1,i_1} \wedge a_{2,i_2}, b_j) = f(a_{1,i_1}, b_j) \vee f(a_{2,i_2}, b_j)$, which proves (F.1).

Since the proof is based on the property of general S- and R-implications that they are non-increasing with respect to their first arguments, it can be stated that (F.1) is correct for any implication which has that property. Hence, (F.1) is correct for all implications reflecting partial ordening of propositions, including R-implications, but also including the implication according to Gaines (1976) and several others, among which the implication proposed by Yager (1980a), $I(a, b) = b^a$, since $b^a \leq b^{a+\varepsilon}$ with $\varepsilon > 0$. It should be noted that implications based on the implication from quantum logic (QL-implications for short), defined by $I(a, b) = S(1 - a, T(a, b))$ (where S and T are c-dual), do not have the required property.

F.2 Rule inference break-up

In the previous section it was shown that a complex rule can be represented by a number of simple rules in case the *min* operation is used for conjunction in the premises of the rules. In this section it will be shown that this rule break-up can be used to simplify the inference of a complex rule. Demirli and Turksen (1992) proved that:

$$B' = (A'_1 \cap A'_2) \circ \mathrm{I}(A_1 \cap A_2, B) \tag{F.2a}$$

$$= \{A'_1 \circ \mathrm{I}(A_1, B)\} \ \cup \ \{A'_2 \circ \mathrm{I}(A_2, B)\} \tag{F.2b}$$

$$= B1' \cup B2' \tag{F.2c}$$

The proof given by Demirli and Turksen (1992) assumes that A'_1 and A'_2 are normalized (*hgt*$(A'_1) = 1$ and *hgt*$(A'_2) = 1$) and is based on (F.1). When for simplicity it is assumed that A_1, A_2 and B are discretized by a_i, b_j and c_k, respectively, then the following proof is given by Demirli and Turksen (1992):

$$\begin{aligned}
\forall k,\ c'_k &= \bigvee_{i,j}[(a'_i \wedge b'_j) \ \wedge \ f(a_i \wedge b_j, c_k)] \\
&= \bigvee_{i,j}[(a'_i \wedge b'_j) \ \wedge \ \{f(a_i, c_k) \vee f(b_j, c_k)\}] \\
&= \bigvee_{i,j}[\{a'_i \wedge b'_j \wedge f(a_i, c_k)\} \ \vee \ \{a'_i \wedge b'_j \wedge f(b_j, c_k)\}]
\end{aligned}$$

$$
\begin{aligned}
&= \bigvee_j [\bigvee_i \{a'_i \wedge b'_j \wedge f(a_i, c_k)\} \;\vee\; \bigvee_i \{a'_i \wedge b'_j \wedge f(b_j, c_k)\}] \\
&= \bigvee_j [\bigvee_i \{a'_i \wedge b'_j \wedge f(a_i, c_k)\} \;\vee\; \{b'_j \wedge f(b_j, c_k)\}] \\
&= \bigvee_i [\bigvee_j \{a'_i \wedge b'_j \wedge f(a_i, c_k)\} \;\vee\; \bigvee_j \{b'_j \wedge f(b_j, c_k)\}] \\
&= \bigvee_i [a'_i \wedge f(a_i, c_k)] \;\vee\; \bigvee_j [b'_j \wedge f(b_j, c_k)] \\
&= c1'_k \vee c2'_k
\end{aligned}
$$

The above given proof by Demirli and Turksen (1992) assumes that A'_1 and A'_2 are normalized. However, this assumption is not necessary when modifying the sets A'_1 and A'_2 used in (F.2) in the following way:

$$A''_1 = \mathrm{hgt}(A'_2) \wedge A'_1 \tag{F.3}$$

$$A''_2 = \mathrm{hgt}(A'_1) \wedge A'_2 \tag{F.4}$$

Due to the *min* operation for conjunction and the *sup-min* composition, this modification will compensate for subnormal data (*hgt*(A'_1)*hgt*$(A'_2) < 1$).

F.3 Rule base inference break-up

The proof in the previous section showed that it is possible to break-up the inference of a rule, which is modeled by an implication that is nonincreasing with respect to its first arguments, when the conjunction is represented by a *min* operation and the *sup-min* composition is used. In this section it is shown that under the same conditions the inference of a rule base with rules (not just one rule) can be broken-up to the inference of simple rule bases (rule bases in which the rule have only one variable addressed in their premises). Next it is assumed that the data $A' = A'_1 \cap A'_2$ is normalized. If not, a similar modification of the data as explained at the end of the previous section will compensate for subnormal data. The proof will be done for a rule base with two rules with two-dimensional premises. Extension to more rules and more variables in their premises is straightforward. For the assumed inference, the following can be derived:

$$
\begin{aligned}
B' &= (A'_1 \cap A'_2) \circ \{\mathrm{I}(A_{1,1} \cap A_{2,1}, B_1) \;\cap\; \mathrm{I}(A_{1,2} \cap A_{2,2}, B_2)\} \qquad \text{(F.5a)} \\
&= A'_1 \circ \{\mathrm{I}(A_{1,1}, B_1) \;\cap\; \mathrm{I}(A_{1,2}, B_2)\} \;\cup
\end{aligned}
$$

$$\begin{aligned}
&A'_2 \circ \{\mathrm{I}(A_{2,1}, B_1) \,\cap\, \mathrm{I}(A_{2,2}, B_2)\} \,\cup \\
&\{[A'_1 \circ \mathrm{I}(A_{1,1}, B_1)] \,\cap\, [A'_2 \circ \mathrm{I}(A_{2,2}, B_2)]\} \,\cup \\
&\{[A'_1 \circ \mathrm{I}(A_{1,2}, B_2)] \,\cap\, [A'_2 \circ \mathrm{I}(A_{2,1}, B_1)]\}
\end{aligned} \tag{F.5b}$$

Assuming that A_1, A_2 and B are discretized by a_i, b_j and c_k, respectively, the proof is given by:

$$\begin{aligned}
\forall k,\; c'_k &= \bigvee_{i,j} [(a'_i \wedge b'_j) \,\wedge\, \{f(a_{1,i} \wedge b_{1,j}, c_{1,k}) \,\wedge\, f(a_{2,i} \wedge b_{2,j}, c_{2,k})\}] \\
&= \bigvee_{i,j} [(a'_i \wedge b'_j) \,\wedge\, \{[f(a_{1,i}, c_{1,k}) \vee f(b_{1,j}, c_{1,k})] \,\wedge \\
&\qquad\qquad [f(a_{2,i}, c_{2,k}) \vee f(b_{2,j}, c_{2,k})]\}] \\
&= \bigvee_{i,j} [(a'_i \wedge b'_j) \,\wedge\, \{[f(a_{1,i}, c_{1,k}) \wedge f(a_{2,i}, c_{2,k})] \,\vee \\
&\qquad\qquad [f(a_{1,i}, c_{1,k}) \wedge f(b_{2,j}, c_{2,k})] \,\vee \\
&\qquad\qquad [f(b_{1,j}, c_{1,k}) \wedge f(a_{2,i}, c_{2,k})] \,\vee \\
&\qquad\qquad [f(b_{1,j}, c_{1,k}) \wedge f(b_{2,j}, c_{2,k})]\}] \\
&= \bigvee_{i,j} [\{\{(a'_i \wedge b'_j) \,\wedge\, [f(a_{1,i}, c_{1,k}) \wedge f(a_{2,i}, c_{2,k})]\} \,\vee \\
&\qquad \{(a'_i \wedge b'_j) \,\wedge\, [f(a_{1,i}, c_{1,k}) \wedge f(b_{2,j}, c_{2,k})]\} \,\vee \\
&\qquad \{(a'_i \wedge b'_j) \,\wedge\, [f(b_{1,j}, c_{1,k}) \wedge f(a_{2,i}, c_{2,k})]\} \,\vee \\
&\qquad \{(a'_i \wedge b'_j) \,\wedge\, [f(b_{1,j}, c_{1,k}) \wedge f(b_{2,j}, c_{2,k})]\}\}] \\
&= \bigvee_{i,j} \{(a'_i \wedge b'_j) \,\wedge\, [f(a_{1,i}, c_{1,k}) \wedge f(a_{2,i}, c_{2,k})]\} \,\vee \\
&\quad \bigvee_{i,j} \{(a'_i \wedge b'_j) \,\wedge\, [f(a_{1,i}, c_{1,k}) \wedge f(b_{2,j}, c_{2,k})]\} \,\vee \\
&\quad \bigvee_{i,j} \{(a'_i \wedge b'_j) \,\wedge\, [f(b_{1,j}, c_{1,k}) \wedge f(a_{2,i}, c_{2,k})]\} \,\vee \\
&\quad \bigvee_{i,j} \{(a'_i \wedge b'_j) \,\wedge\, [f(b_{1,j}, c_{1,k}) \wedge f(b_{2,j}, c_{2,k})]\} \\
&= [\bigvee_{i} \{a'_i \wedge [f(a_{1,i}, c_{1,k}) \wedge f(a_{2,i}, c_{2,k})]\}] \,\vee \\
&\quad [\bigvee_{i} \{a'_i \wedge f(a_{1,i}, c_{1,k})\} \,\wedge\, \bigvee_{j} \{b'_j \wedge f(b_{2,j}, c_{2,k})\}] \,\vee \\
&\quad [\bigvee_{j} \{b'_j \wedge f(b_{1,j}, c_{1,k})\} \,\wedge\, \bigvee_{i} \{a'_i \wedge f(a_{2,i}, c_{2,k})\}] \,\vee
\end{aligned}$$

$$[\bigvee_j \{b'_j \wedge [f(b_{1,j}, c_{1,k}) \wedge f(b_{2,j}, c_{2,k})]\}]$$

From this it can be seen that it is inevitable to perform inferences on product spaces. The break-up of the inference of a rule base results in a combination of inferences of simple rule bases. From practical point of view, this inference break-up will limit the necessary product-space calculations to at most 2-dimensional.

References

ALBUS, J.S. (1975a). Data storage in the cerebellar model articulation controller (CMAC). *Transactions of the ASME*, 228–233.

ALBUS, J.S. (1975b). A new approach to manipulator control: the cerebellar model articulation controller (CMAC). *Transactions of the ASME*, 220–227.

ASSILIAN, S. (1974). *Artificial Intelligence Techniques in the Control of Real Dynamic Systems*. Ph.D. thesis, Queen Mary College, University of London, London, UK.

ÅSTRÖM, K.J. AND B. WITTENMARK (1984). *Computer Controlled Systems: theory and design*. Prentice Hall. ISBN 0-13-164302-9.

BABUŠKA, R., R. JAGER AND H.B. VERBRUGGEN (1994). Interpolation issues in Sugeno-Takagi reasoning. See IEEE (1994), pp. 859–863.

BALDWIN, J.F. (1979). A new approach to approximate reasoning using fuzzy logic. *Fuzzy Sets and Systems 2*, 309–325.

BALDWIN, J.F. AND B.W. PILSWORTH (1980). Axiomatic approach to implication for approximate reasoning with fuzzy logic. *Fuzzy Sets and Systems 3*, 193–219.

BALDWIN, J.F. AND S.Q. ZHOU (1984). Fril - a fuzzy relational inference language. *Fuzzy Sets and Systems 14*, 155–174.

BANDLER, W. AND L.J. KOHOUT (1980). Fuzzy power sets and fuzzy implication operators. *Fuzzy Sets and Systems 4*, 13–30.

BATUR, C. AND V. KASPARIAN (1991). Predictive fuzzy expert controllers. *Computers in Industrial Engineering 20*(2).

BELLMAN, R.E. AND M. GIERTZ (1973). On the analytic formalism of the theory of fuzzy sets. *Information Sciences 5*, 149–156.

BERSINI, H., J-P. NORDVIK AND A. BONARINI (1993). A simple direct adaptive fuzzy controller derived from its neural equivalent. See IEEE (1993), pp. 345–350.

BROEDERS, H.M.T., P.M. BRUIJN AND H.B. VERBRUGGEN (1989). Real-time direct expert control. *Engineering Applications of Artificial Intelligence 2*(2).

BROUWN, G.G. (1993). Adaptive single-layer networks for nonlinear system identification. Twaio report, Delft University of Technology, Department of Electrical Engineering, Control Laboratory, Delft, The Netherlands. TWAIO93.049.

BROWN, M. (1990). First year progress report, in developing an adaptive piloting system for an autonomous land vehicle. Technical report, Aeronautics and Astronautics, Southampton University.

BROWN, M., R. FRASER, C.J. HARRIS AND C.G. MOORE (1991). Intelligent self-organising controllers for autonomous guided vehicles. See IEE (1991), pp. 134–139.

BROWN, M. AND C.J. HARRIS (1991). A nonlinear controller: a comparison between fuzzy logic control and neurocontrol. *IMA Journal of Mathematical Control and Information 8*(3), 239–265.

BUCHANAN, B.G. AND E.H. SHORTLIFFE (1984). *Rule-Based expert Systems*. Readings (MA), USA: Addison–Wesley.

CHIEN, Y.C. AND C.C.TENG (1993). Fuzzy modelling using neural networks. In *Proceedings of the 2nd European Control Conference*, Groningen, The Netherlands, pp. 503–508. ECCA.

CSER, J., P.M. BRUIJN AND H.B. VERBRUGGEN (1986). Music: a tool for simulation and real-time control. In *Proceedings IFAC/IFIP 4th Symposium on Software for Computer Control*, Graz, Austria.

DEMIRLI, K. AND I.B. TURKSEN (1992). Rule break up with compositional rule of inference. See IEEE (1992), pp. 949–956.

DESPRÉS, S. (1989). GRIF: a guide for representing fuzzy inferences. In *Proceedings of the 3rd International Fuzzy Systems Association Congress*, Seattle (WA), USA, pp. 353–356. IFSA.

DOMBI, J. (1982). A general class of fuzzy operators, the De Morgan class of fuzzy operators and fuzziness induced by fuzzy operators. *Fuzzy Sets and Systems 8*, 149–163.

DRIANKOV, D. (1987). Inference with a single fuzzy conditional proposition. *Fuzzy Sets and Systems* (24), 51–63.

DRIANKOV, D., H. HELLENDOORN AND M. REINFRANK (1993). *An introduction to fuzzy control*. New York, USA: Springer–Verlag.

DUBOIS, D. AND H. PRADE (1980). *Fuzzy sets and systems: theory and applications*, Volume 144 of *Mathematics in science and engineering*. Academic Press. ISBN 0-12-222750-6.

DUBOIS, D. AND H. PRADE (1984). Fuzzy logics and the generalized modus ponens revisited. *International Journal on Cybernetics and Systems* (15), 293–331.

DUBOIS, D. AND H. PRADE (1986). New results about properties and semantics of fuzzy set-theoretic operators. In P.P. Wang and S.K. Chang (Eds.), *Fuzzy Sets*, pp. 59–75. New York, USA: Plenum Press.

DUBOIS, D. AND H. PRADE (1988). *Possibility theory - an approach to computerized processing of uncertainty*. New York, USA: Plenum Press. ISBN/ISSN 0-306-42520-3.

DUBOIS, D. AND H. PRADE (1990). An introduction to possibilistic and fuzzy logic. See Shafer and Pearl (1990), pp. 742–761. ISBN 1-55860-125-2.

DUBOIS, D. AND H. PRADE (1991). Fuzzy sets in approximate reasoning, part 1: Inference with possibility distributions. *Fuzzy Sets and Systems 40*, 143–202.

DUBOIS, D. AND H. PRADE (1992). Possibility theory as a basis for preference propagation in automated reasoning. See IEEE (1992), pp. 821–832.

DUBOIS, D. AND H. PRADE (1994a). Can we inforce full compositionality in uncertainty calculi? In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, Washington, USA, pp. 149–154. AAAI.

DUBOIS, D. AND H. PRADE (1994b). On the validation of fuzzy knowledge bases. See Tzafestas and Venetsanopoulos (1994), Chapter 2, pp. 31–49. ISBN 0-7923-2643-1.

DUBOIS, D., H. PRADE AND P. SMETS (1994). Partial truth is not uncertainty - fuzzy logic versus possibilistic logic. *IEEE Expert - A Fuzzy Logic Symposium 9*(4), 15–19.

DUBOIS, D., H. PRADE AND R.R. YAGER (Eds.) (1993). *Readings in Fuzzy Sets for Intelligent Systems*. Morgan Kaufmann Publishers. ISBN 1-55860-257-7.

DUBOIS, D., H. PRADE AND R.R. YAGER (1993a). Basic notions in fuzzy set theory. See Dubois, Prade and Yager (1993), Chapter 2, pp. 27–64. ISBN 1-55860-257-7.

DUBOIS, D., H. PRADE AND R.R. YAGER (1993b). Introduction. See Dubois, Prade and Yager (1993), Chapter 1, pp. 1–19. ISBN 1-55860-257-7.

ELITE (1993). *Proceedings First European Congress on Fuzzy and Intelligent Technologies*, Aachen, Germany.

ELKAN, C. (1993). The paradoxical success of fuzzy logic. In *Proceedings of the National Artificial Intelligence Conference*, pp. 698–703. AAAI: MIT and AAAI Press.

ELKAN, C. (1994). The paradoxical success of fuzzy logic. *IEEE Expert - A Fuzzy Logic Symposium 9*(4), 3–8.

E.SANCHEZ AND M.M. GUPTA (Eds.) (1983). *Proceedings of IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis*, Marseilles, France. IFAC: Pergamon Press.

FILEV, D.P. AND R.R. YAGER (1991). A generalized defuzzification method via BAD distributions. *International Journal of Intelligent Systems 6*, 687–697.

FUKAMI, S., M. MIZUMOTO AND T. TANAKA (1980). Some considerations on fuzzy conditional inference. *Fuzzy Sets and Systems 4*, 243–273.

GAINES, B.R. (1976). Foundations of fuzzy reasoning. *International Journal of Man-Machine Studies 8*, 623–668.

GILES, R. (1976). Łukasiewicz logic and fuzzy set theory. *International Journal of Man-Machine Studies 8*, 313–327.

GOGUEN, J.A. (1969). The logic of inexact concepts. *Synthese 19*, 325–373.

GOLDBERG, A. AND D. ROBSON (1983). *Smalltalk-80: The Language and its Implementation*. Reading (MA), USA: Addison–Wesley.

GRAAFEILAND, VAN, R. (1992). Hybride modelvorming. Master's thesis, Delft University of Technology, Department of Electrical Engineering, Control Laboratory, Delft, The Netherlands. A92.044 (611).

GRAHAM, I. (1991). Fuzzy logic in commercial expert systems - results and prospects. *Fuzzy Sets and Systems 40*(3), 451–472.

GUÉLY, F. AND P. SIARRY (1993). Gradient descent method for optimizing various fuzzy rule bases. See IEEE (1993), pp. 1241–1246.

GUPTA, M.M., A. KANDEL, W. BANDLER AND J.B. KISZKA (Eds.) (1985). *Approximate Reasoning in Expert Systems*. Amsterdam, The Netherlands: Elsevier Science Publishers.

GUPTA, M.M. AND J. QI (1991a). Design of fuzzy logic controllers based on generalized t-operators. *Fuzzy Sets and Systems 40*, 473–489.

GUPTA, M.M. AND J. QI (1991b). Theory of t-norms and fuzzy inference methods. *Fuzzy Sets and Systems 40*, 431–450.

HAMACHER, H. (1978). Über logische verknupfungen unscharfer aussagen und deren zugehörige bewertungsfunktionen. In G.J. Klir R. Trappl and L. Ricciardi (Eds.), *Progress in Cybernetics and Systems Research*, Volume 3. Washington (DC), USA: Hemisphere.

HARRIS, C.J. AND C.G. MOORE (1989). Intelligent identification and control for autonomous guided vehicles using adaptive fuzzy-based algorithms. *Engineering Applications of Artificial Intelligence 2*, 267–285.

HARRIS, C.J., C.G. MOORE AND M. BROWN (1993). *Intelligent control - aspects of fuzzy logic and neural nets*. Singapore: World Scientific.

HELLENDOORN, J. (1990). *Reasoning with Fuzzy Logic*. Ph.D. thesis, Delft University of Technology, Delft, The Netherlands.

HORIKAWA, S., T. FURUHASHI AND Y. UCHIKAWA (1993). On identification of structures in premises of a fuzzy model using a fuzzy neural network. See IEEE (1993), pp. 606–611.

IEE (1991). *Proceedings IEE Control 91*, Edinburgh, Scotland, UK.

IEEE (1992). *Proceedings of 1st IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '92)*, San Diego (Ca), USA.

IEEE (1993). *Proceedings of 2nd IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '93)*, San Fransisco (Ca), USA.

IEEE (1994). *Proceedings of 3rd IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '94)*, Orlando (Fl), USA.

ISHIBUCHI, H., K. NOZAKI AND H. TANAKA (1993). Empirical study on learning in fuzzy systems. See IEEE (1993), pp. 606–611.

JAGER, R. (1992). Adaptive fuzzy control. In L. Boullart, A.J. Krijgsman and R.A. Vingerhoeds (Eds.), *Application of artificial intelligence in process control*, pp. 368–387. Oxford, UK: Pergamon Press. Lecture notes ERASMUS intensive course, ISBN 0-08-042016-8/0-08-042017-6.

JAGER, R., A.J. KRIJGSMAN, H.B. VERBRUGGEN AND P.M. BRUIJN (1990). Direct real-time control using knowledge-based techniques. See SCS (1990), pp. 61–65.

JAGER, R., H.B. VERBRUGGEN AND P.M. BRUIJN (1991). Adaptive fuzzy control. In *Proceedings 1991 European Simulation Symposium*, Ghent, Belgium, pp. 140–144. SCS.

JAGER, R., H.B. VERBRUGGEN AND P.M. BRUIJN (1992). The role of defuzzification methods in the application of fuzzy control. In *Preprints IFAC Symposium on Intelligent Components and Instrumentation for Control Applications (SICICA) '92*, Málaga, Spain, pp. 111–116. IFAC.

JAGER, R., H.B. VERBRUGGEN AND P.M. BRUIJN (1993b). Fuzzy linear control. See ELITE (1993), pp. 925–929.

JAGER, R., H.B. VERBRUGGEN AND P.M. BRUIJN (1993a). Fuzzy linear modelling. In *Preprints IMACS Symposium on Mathemathical and Intelligent Models and System Simulation 1993*, Brussels, Belgium. IMACS.

JAGER, R., H.B. VERBRUGGEN, P.M. BRUIJN AND A.J. KRIJGSMAN (1991). Real-time fuzzy expert control. See IEE (1991), pp. 966–970.

JANG, J.-S.R. (1993). ANFIS: Adaptive-network-based fuzzy inference systems. *IEEE Transactions on Systems, Man, and Cybernetics 23*(3), 665–685.

JANG, J.-S.R. AND C.-T. SUN (1993). Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks 4*(1), 156–159.

KANDEL, A. (1986). *Fuzzy Mathematical Techniques with Applications*. Addison–Wesley.

KATS, VAN, J.J. (1993). Cmac and the fuzzy concept. Master's thesis, Delft University of Technology, Department of Electrical Engineering, Control Laboratory, Delft, The Netherlands. A93.019 (628).

KECMAN, V. AND B.M. PFEIFFER (1994). Exploiting the structural equivalence of learning fuzzy systems and radial basis function neural networks. In *Proceedings European Congress on Fuzzy and Intelligent Technologies*, Aachen, Germany. ELITE.

KESTEREN, VAN, A. (1991). Kennisgestuurde vage regelsystemen. Master's thesis, Delft University of Technology, Department of Electrical Engineering, Control Laboratory, Delft, The Netherlands. A91.032 (565).

KICKERT, W.M. AND E.H. MAMDANI (1978). Analysis of fuzzy logic controllers. *Fuzzy Sets and Systems* (1), 29–44.

KLEENE, S.C. (1938). On a notation for ordinal numbers. *The Journal of Symbolic Logic 3*, 150–155.

KÓCZY, L.T. AND K. HIROTA (1993). Interpolative reasoning with insufficient evidence in sparse fuzzy rule bases. *Information Sciences 71*, 169–201.

KOSKO, B. (1992). Fuzzy systems as universal approximators. See IEEE (1992), pp. 153–162.

KRIJGSMAN, A.J. (1993). *Artificial Intelligene in Real-time Control*. Ph.D. thesis, Delft University of Technology, Delft, The Netherlands.

KRIJGSMAN, A.J. AND R. JAGER (1993a). DICE: a real-time toolbox. In *Proceedings 1992 IFAC/IFIP/IMACS Symposium on Artificial Intelligence in Real-Time Control*, Number 6 in IFAC Symposia Series, Delft, The Netherlands. IFAC/IFIP/IMACS: Pergamon Press.

KRIJGSMAN, A.J. AND R. JAGER (1993b). Modelling and control using neural networks and use of associative memories: Fuzzy CMAC. See ELITE (1993), pp. 919–924.

KRIJGSMAN, A.J., R. JAGER, H.B. VERBRUGGEN AND P.M. BRUIJN (1991). DICE: a framework for intelligent real-time control. In *Proceedings 3$^{\text{th}}$ IFAC Workshop on Artificial Intelligence in Real-Time Control*, Napa (Ca), USA.

KRIJGSMAN, A.J., H.B. VERBRUGGEN, P.M. BRUIJN AND E.G.M. HOLWEG (1990). DICE: a real-time intelligent control environment. See SCS (1990).

KUIPERS, P. (1986). Qualitative simulation. *Artificial Intelligence* (29), 289–338.

LAKOFF, G. (1973). Hedges: a study in meaning criteria and the logic of fuzzy concepts. *Journal of Philosophical Logic 2*, 458–508.

LARSEN, P.M. (1980). Industrial applications of fuzzy logic control. *International Journal of Man-Machine Studies 12*(1), 3–10.

LATTIMER WRIGHT, M., M.W. GREEN, G. FIEGL AND P.F. CROSS (1986). An expert system for real-time control. *IEEE Software*, 16–24.

LEE, C.C. (1990a). Fuzzy logic in control systems: fuzzy logic controller - part i & ii. *IEEE Transactions on Systems, Man, and Cybernetics 20*(2), 404–435.

LEE, C.C. (1990b). Fuzzy logic in control systems: fuzzy logic controller - part ii. *IEEE Transactions on Systems, Man, and Cybernetics 20*(2), 419–435.

LEUNG, K.S. AND M.H. WONG (1992). Fuzzy concepts in an object oriented expert system shell. *International Journal of Intelligent Systems 7*, 171–192.

LI, M.X., P.M. BRUIJN AND H.B. VERBRUGGEN (1994). Tuning cascade pid controllers using fuzzy logic. *Mathematics and Computers in Simulation* (37), 143–151.

LINKENS, D.A. AND S.B. HASNAIN (1991). Self-organising fuzzy logic control and application to muscle relaxant anaesthesia. In *IEE Proceedings-D*, Volume 138, pp. 274–284.

LUGER, G.F. AND W.A. STUBBLEFIELD (1989). *Artificial intelligence and the design of expert systems*. Redwood City (Ca), USA: The Benjamin/Cummings Publishing Company, Inc.

MAGREZ, P. AND P. SMETS (1989). Fuzzy modus ponens: a new model suitable for applications in knowledge-based systems. *International Journal of Intelligent Systems 4*, 181–200.

MALKI, H.A., H. LI AND G. CHEN (1994). New design and stability analysis of fuzzy proportional-derivative control systems. *IEEE Transactions on Fuzzy Systems 2*(4), 245–254.

MAMDANI, E.H. (1974). Applications of fuzzy algorithm for simple dynamic plant. *Proceedings IEE 121*(12), 1585–1588.

MAMDANI, E.H. (1993). Twenty years of fuzzy control: experiences gained and lessons learnt. See IEEE (1993), pp. 339–344.

MAMDANI, E.H. (1994). Fuzzy control - a misconception of theory and application. *IEEE Expert - A Fuzzy Logic Symposium 9*(4), 27–28.

MAMDANI, E.H. AND S. ASSILIAN (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies 7*, 1–13.

MARTIN, T.P., J.F. BALDWIN AND B.W. PILSWORTH (1987). The implementation of fprolog - a fuzzy prolog interpreter. *Fuzzy Sets and Systems 23*, 119–129.

MATSUOKA, H. (1991). A simple fuzzy simulation model for nuclear reactor system dynamics. *Nuclear Technology 94*, 228–241.

MIZUMOTO, M. (1981). Note on the arithmetic rule by zadeh for fuzzy conditional inference. *Cybernetics and Systems* (12), 247–306.

MIZUMOTO, M. (1985). Extended fuzzy reasoning. See Gupta, Kandel, Bandler and Kiszka (1985).

MYLOPOULOS, J. AND H.J. LEVESQUE (1984). An overview of knowledge representation. In M.L. Brodie, J. mylopoulos and J.W. Schmidt (Eds.), *On Conceptual Modelling*. New York: Springer–Verlag.

NAUTA LEMKE, VAN, H.R. AND W. DE-ZHAO (1985). Fuzzy PID supervisor. In *Proceedings of the 24th IEEE Conference on Decision and Control*, Fort Lauderdale, Florida, USA.

NAUTA LEMKE, VAN, H.R. AND A.J. KRIJGSMAN (1991). Design of fuzzy PID supervisors for systems with different performance requirements. In *Proceedings IMACS '91*, Dublin, Ireland.

NEYER, DE, M., D. STIPANICEV AND R. GOREZ (1990). Intelligent self-organizing controllers and their application to the control of dynamic systems. In *IMACS Annals on Computing and Applied Mathematics Proceedings - MIM-S*^{2}.

NIE, J. AND D.A. LINKENS (1993). A fuzzified cmac self-learning controller. See IEEE (1993), pp. 500–505.

NOMURA, H., I. HAYASHI AND N. WAKAMI (1991). A self-tuning method of fuzzy control by descent method. In *Proceedings of the IFSA '91*, Brussels, Belgium, pp. 155–158.

NRC/KSL (1994). *FuzzyCLIPS Version 6.02 - User's Guide*. Ottawa, Ontario Canada K1A 0R6 (e-mail: `fzclips@ai.iit.nrc.ca`, WWW: `http://ai.iit.nrc.ca/fuzzy/fuzzy.html`): Knowledge Systems Laboratory, Instistute for Information Technology, National Research Council Canada.

OKAMOTO, W., S. TANO, T. IWATANI AND A. INOUE (1994). Inference method for natural language propositions involving fuzzy quantifiers in FLINS. See IEEE (1994), pp. 2082–2087.

O'REILLY, C.A. AND A.S. CROMARTY (1986). Fast is not "real-time". In *Designing Effective Real-Time AI Systems, Applications of Artificial Intelligence II 548*, Bellingham, pp. 249–257.

ÖSTERGAARD, J.J. (1990). FUZZY II – the new generation of high level kiln control. *Zement–Kalk–Gips* (11).

PEARL, J. (1990). Fusion, propagation, and structuring in belief networks. See Shafer and Pearl (1990), pp. 366–414. ISBN 1-55860-125-2.

PEDRYCZ, W. (1989). *Fuzzy control and fuzzy systems*. Chichester: Research Studies Press/J. Wiley & Sons.

PEDRYCZ, W. (1993). *Fuzzy control and fuzzy systems - second, extended, edition*. Taunton, Summerset, England: Research Studies Press Ltd.

PEITSMAN, H. (1993). First experience with RICE 4.0 coupled to the AHU plant. IEA Annex 25, fourth meeting, Zürich, Switzerland.

PEITSMAN, H. AND A.C. VAN DUYVENVOORDE (1993). Expert system RICE 4.0 - examples for novices. IEA Annex 25, third meeting, Tokyo, Japan.

PÉTIEAU, A.M., A. MOREAU AND D. WILLAEYS (1990). Tools for approximate reasoning in expert systems. In *Proceedings TIMS/ORSA Conference*, Las Vegas, USA.

PRADE, H. (1983). Approximate and plausible reasoning. See E.Sanchez and Gupta (1983).

PROCYK, T.J. AND E.H. MAMDANI (1979). A linguistic self-organising process controller. *Automatica 15*, 15–30.

QUILLIAN, M.R. (1968). Semantic memory. In M.L. Minsky (Ed.), *Semantic Information Processing*. Cambridge, USA: MIT Press.

REE, VAN DE, R. (1994). *Supervisory Control Systems*. Ph.D. thesis, Delft University of Technology, Delft, The Netherlands.

RHODES, P.C. AND S.M. MENANI (1991). Towards a fuzzy logic programming system: a fuzzy propositional logic. *Knowledge-Based Systems 4*(1), 52–62.

RHODES, P.C. AND S.M. MENANI (1992). Towards a fuzzy logic programming system: a 1st-order fuzzy logic. *Knowledge-Based Systems 5*(2), 106–116.

RINE, D.C. (1991). Design of fuzzy object-oriented software components data bases. *International Journal on Cybernetics and Systems* (22), 531–551.

RUNKLER, T.A. AND M. GLESNER (1993). Defuzzification with improved static and dynamic behavior: extended center of area. See ELITE (1993), pp. 845–851.

SCHANK, R. AND R. ABELSON (1977). *Scripts, Plans, Goals and Understanding*. Hillsdale (NJ), USA: Erlbaum.

SCHWEIZER, B. AND A. SKLAR (1961). Associative functions and statistical triangle inequalities. *Publicationes Mathematicae Debrecen 8*, 169–186.

SCS (1990). *Proceedings 1990 Europian Simulation Symposium*, Ghent, Belgium.

SHAFER, G. (1976). *A mathematical theory of evidence*. Princeton (NJ), USA: Princeton University Press. ISBN 0-691-08175-1/10042-X.

SHAFER, G. (1987). Belief functions and possibility measures. In J.C. Bezdek (Ed.), *Analysis of Fuzzy Information - Mathematics and Logic*, Volume I, pp. 51–84. Boca Raton (FL), USA: CRC Press. ISBN 0-8493-6296-2.

SHAFER, G. AND J. PEARL (Eds.) (1990). *Readings in uncertain reasoning*. The Morgan Kaufmann series in representation and reasoning. San Mateo (Ca), USA: Morgan Kaufmann Publishers. ISBN 1-55860-125-2.

SHAO, S. (1988). Fuzzy self-organising controller and its application for dynamic processes. *Fuzzy Sets and Systems 26*, 151–164.

SOSNOWSKI, Z.A. (1990). FLISP: a language for processing fuzzy data. *Fuzzy Sets and Systems* (37), 23–32.

SOWA, J.F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Reading (MA), USA: Addison–Wesley.

SUGENO, M. (1977). Fuzzy measrues and fuzzy integrals: a survey. In M.M. Gupta, G.N. Sardis and B.R. Gaines (Eds.), *Fuzzy Automata and Decision Processes*, pp. 89–102. Amsterdam, The Netherlands: North-Holland Publishers.

SUGENO, M. AND K. MURAKAMI (1984). Fuzzy parking control of model car. In *Proceedings IEEE 23rd Conference on Decision and Control*, Las Vegas, USA. IEEE.

SUGENO, M. AND K. MURAKAMI (1985). An experimental study on fuzzy parking control using a model car. In M. Sugeno (Ed.), *Industrial applications of fuzzy control*, pp. 125–138. Amsterdam, The Netherlands: Elsevier Science Publishers.

SUGENO, M. AND M. NISHIDA (1985). Fuzzy control of model car. *Fuzzy Sets and Systems 16*, 103–113.

SUGIYAMA, K. (1988). Rule-based self-organising controller. In M.M. Gupta and T. Yamakawa (Eds.), *Fuzzy Computing - Theory, Hardware and Applications*, pp. 341–353. Amsterdam, The Netherlands: Elsevier Science Publishers. ISBN 0-444-70449-3.

TAKAGI, T. AND M. SUGENO (1983). Derivation of fuzzy control rules from human operator's control actions. See E.Sanchez and Gupta (1983), pp. 55–60.

TANO, S., W. OKAMOTO, T. IWATANI AND A. INOUE (1994). Basic structure of three-layered fuzzy inference in FLINS - Fuzzy Lingual System. See IEEE (1994), pp. 446–451.

TRILLAS, E. AND L. VALVERDE (1985). On mode and implication in approximate reasoning. See Gupta, Kandel, Bandler and Kiszka (1985), pp. 157–166.

TSUKAMOTO, Y. (1977). An approach to fuzzy reasoning method. In M.M. Gupta, R.K. Ragade and R.R. Yager (Eds.), *Advances in Fuzzy Set Theory and Applications*, pp. 137–149. Amsterdam, The Netherlands: North-Holland Publishers.

TURKSEN, I.B. (1986). Interval valued fuzzy sets based on normal forms. *Fuzzy Sets and Systems 20*, 191–210.

TURKSEN, I.B. AND Z. ZHONG (1990). An approximate analogical reasoning scheme based on similarity measures and interval-valued fuzzy sets. *Fuzzy Sets and Systems 34*, 323–346.

TZAFESTAS, S. AND N.P. PAPANIKOLOPOULOS (1990). Incremental fuzzy expert PID control. *IEEE Transactions on Industrial Electronics 37*(5), 365–371.

TZAFESTAS, S.G. AND A.N. VENETSANOPOULOS (Eds.) (1994). *Fuzzy Reasoning in Information, Decision and Control Systems*. Kluwer Academic Publishers. ISBN 0-7923-2643-1.

UEHARA, K. AND M. FUJISE (1993). Mutistage fuzzy inference formaulated as linguistic-truth-value propagation and its learning algorithm based on back-propagating error information. *IEEE Transactions on Fuzzy Systems 1*(3), 205–221.

VERBRUGGEN, H.B. AND K.J. ÅSTRÖM (1989). Artificial intelligence and feedback control. In *Proceedings 2$^{\text{nd}}$ Workshop AIRTC '89*, China. IFAC.

WAKILEH, B.A.M. AND K.F. GILL (1990). Robot control using self-organising fuzzy logic. *Computers in Industry 15*, 175–186.

WEBER, S. (1983). A general concept of fuzzy connectives, negations and implications based on t-norms and t-co-norms. *Fuzzy Sets and Systems 11*, 115–134.

WELLING, H.L.J. (1994). Identification of dynamic systems using cmac. Master's thesis, Delft University of Technology, Department of Electrical Engineering, Control Laboratory, Delft, The Netherlands. A94.015 (651).

WILLMOTT, R. (1980). Two fuzzier implication operators in the theory of fuzzy power sets. *Fuzzy Sets and Systems 4*, 31–36.

WU, W. (1986). Fuzzy reasoning and fuzzy relatinal equations. *Fuzzy Sets and Systems 20*, 67–78.

YAGER, R.R. (1980a). An approach to inference in approximate reasoning. *International Journal of Man-Machine Studies 13*, 323–338.

YAGER, R.R. (1980b). On a general class of fuzzy connectives. *Fuzzy Sets and Systems 4*, 235–242.

YAGER, R.R. (1983). An introduction to applications of possibility theory. *Human Systems Management 3*, 246–269.

YAGER, R.R. (1994). Alternative structures for knowledge representations in fuzzy logic controllers. In A. Kandel and Langholz (Eds.), *Fuzzy Control Systems*, Chapter 5, pp. 99–137. CRC Press. ISBN 0-8493-4496-4.

YAGER, R.R. AND D.P. FILEV (1993). SLIDE: a simple adaptive defuzzification method. *IEEE Transactions on Fuzzy Systems 1*(1), 69–78.

YAMAMOTO, S. (1994). Software representation of fuzzy sets and logic. See Tzafestas and Venetsanopoulos (1994), Chapter 3, pp. 52–68. ISBN 0-7923-2643-1.

ZADEH, L.A. (1965). Fuzzy sets. *Information and Control 8*, 338–353.

ZADEH, L.A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics SMC-3*, 28–44.

ZADEH, L.A. (1975). Calculus of fuzzy restrictions. In L.A. Zadeh, K.-S. Fu, K. Tanaka and M. Shimura (Eds.), *Fuzzy Sets and Their Applications to Cognitive and Decision Processes*, pp. 1–39. New York, USA: Academic Press.

ZADEH, L.A. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems 1*, 3–28.

ZADEH, L.A. (1979). A theory of approximate reasoning. In J.E. Hayes, D. Mitchie and L.I. Mikulich (Eds.), *Machine Intelligence*, Volume 9, pp. 149–94. New York: Wiley.

ZADEH, L.A. (1981a). Possibility theory and soft data analysis. In L. Cobb and R. Thrall (Eds.), *Mathematical Frontiers of Social and Policy Sciences*, pp. 69–129. Boulder: Westview Press.

ZADEH, L.A. (1981b). Pruf – a meaning representation language. In E.H. Mamdani and B.R. Gaines (Eds.), *Fuzzy reasoning and its applications*, Computer and People Series, pp. 1–66. Academic Press.

ZADEH, L.A. (1992). Knowledge representation in fuzzy logic. In R.R. Yager and L.A. Zadeh (Eds.), *An introduction to fuzzy logic applications in intelligent systems*, The Kluwer International series in engineering and computer science, pp. 1–25. Kluwer Academic Publishers.

ZADEH, L.A. (1994a). Soft computing and fuzzy logic. *IEEE Software*, 48–56.

ZADEH, L.A. (1994b). Why the success of fuzzy logic is not paradoxical. *IEEE Expert - A Fuzzy Logic Symposium 9*(4), 43–46.

ZIMMERMANN, H.-J. (1985). *Fuzzy Set Theory and its Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

List of symbols

The following **general symbols, operators and functions** are used throughout this thesis. Symbols which represent functions are denoted with an additional $(\cdot)$. Some symbols are also used to denote parameters, but this should be clear from the context.

$\cap$	(fuzzy) set intersection/conjunction
$\cup$	(fuzzy) set union/disjunction
$\wedge$	intersection, logical and
$\vee$	union, logical or
$\overline{A}$	complement/negation of A
$*$	product
$\circ$	*sup-min* composition
$\circ_{\mathrm{T}}$	*sup-T* composition, where *T* is a general T-norm
$\circ_{\mathrm{m}}$	*sup-m* composition, where *m* is a pseudo-conjunction
∂	partial derivative
γ	linguistic possibility value
λ	linguistic probability value
$\mu(\cdot)$	membership function
$\mathrm{N}(\cdot)$	possibility measure
$\pi(\cdot)$	possibility distribution function
$\mathrm{p}(\cdot)$	probability distribution function
P	probability distribution
Π	possibility distribution
$\Pi(\cdot)$	possibility measure

τ	linguistic truth value

In this thesis a number of **short-hand notations to denote properties of or operations on fuzzy sets** are used. In text these short-hand notations are printed italic, abbreviations in capitals. A summary is provided in the following list:

$\mathrm{badd}(A, \delta)$	BADD defuzzification of fuzzy set A
$\mathrm{cext}(A; X)$	cylindrical extension of A on X
$\mathrm{coa}(A)$	centre-of-area (COA) defuzzification of fuzzy set A
$\mathrm{cog}(A)$	centre-of-gravity (COG) defuzzification of fuzzy set A
$\mathrm{core}(A)$	core of fuzzy set A
$\mathrm{dfz}(A)$	defuzzification of fuzzy set A
$\mathrm{fm}(A)$	fuzzy-mean (FM) defuzzification of fuzzy set A
$\mathrm{hgt}(A)$	heigt of fuzzy set A
$i\mathrm{dfz}(A)$	indexed defuzzification of fuzzy set A
$\mathrm{mom}(A)$	mean-of-maxima (MOM) defuzzification of fuzzy set A
$\mathrm{pow}(A)$	power of fuzzy set A
$\mathrm{proj}(A; X)$	projection of A on X
$\mathrm{slide}(A, \alpha, \beta)$	SLIDE defuzzification of fuzzy set A
$\mathrm{supp}(A)$	support of fuzzy set A
$\mathrm{xcoa}(A, \gamma)$	extended centre-of-area (XCOA) defuzzification of fuzzy set A
$\mathrm{wfm}(A)$	weighted fuzzy-mean (WFM) defuzzification of fuzzy set A

For **fuzzy systems**, including fuzzy controllers and models, the following variables are used:

x_i	i^{th} numerical input variable
x'_i	i^{th} numerical (measured) input value
X_i	i^{th} input universe
A_i	fuzzy set for i^{th} input universe
$A_{i,k}$	fuzzy set for i^{th} input universe in premise of k^{th} rule
A'_i	fuzzy set representing (measured) value for i^{th} input
r_k	k^{th} fuzzy rule
R	fuzzy relation
R_k	fuzzy relation representing k^{th} fuzzy rule
y_j	j^{th} numerical output
Y_j	j^{th} output universe
B_j	fuzzy set for j^{th} output universe

$B_{j,k}$	fuzzy set for j^{th} output universe in premise of k^{th} rule
B'_j	fuzzy set for j^{th} output universe resulting from inference

Specific to **fuzzy controllers** in some cases the following variables are used:

e	error, difference between reference signal and process output
Δe	first difference of error
u	control signal
Δu	control signal change

The **notation of variables** in text as well as equations are conform the following:

a	scalar or fuzzy singleton representing a scalar
A	(fuzzy) set
$\boldsymbol{a}$	vector
$\boldsymbol{A}$	matrix

List of abbreviations

The following list summarizes the abbreviations used throughout this thesis:

AI	**A**rtificial **I**ntelligence
ARMAX	**A**uto-**R**egressive **M**odel with e**X**ogeneous inputs
BADD	**BA**sic **D**efuzzification **D**istribution(s)
CMAC	**C**erebellar **M**odel **A**rticulation **C**ontroller
COA	**C**entre-**O**f-**A**rea
COG	**C**entre-**O**f-**G**ravity
CRI	**C**ompositional **R**ule of **I**nference
DICE	**D**elft **I**ntelligent **C**ontrol **E**nvironment
DOF	**D**egree **O**f **F**ulfillment
ES	**E**xpert **S**ystem
FM	**F**uzzy-**M**ean
FNN	**F**uzzy **N**eural **N**etwork
GCMAC	**G**eneralized **C**erebellar **M**odel **A**rticulation **C**ontroller
GMP	**G**eneralized **M**odus **P**onens
GMT	**G**eneralized **M**odus **T**ollens
KB	**K**nowledge **B**ase
KBS	**K**nowledge-**B**ased **S**ystem
MP	**M**odus **P**onens
MT	**M**odus **T**ollens
MIMO	**M**ultiple-**I**nput **M**ultiple-**O**utput
MISO	**M**ultiple-**I**nput **S**ingle-**O**utput

MOM	**M**ean-**O**f-**M**axima
MRAC	**M**odel **R**eference **A**daptive **C**ontrol
MSF	**M**ember**S**hip **F**unction
OOKR	**O**bject-**O**riented **K**nowledge **R**epresentation
RBF	**R**adial **B**asis **F**unction
RBFN	**R**adial **B**asis **F**unction **N**etwork
RICE	**R**outines for **I**mplementing **C** **E**xpert systems
SIMO	**S**ingle-**I**nput **M**ultiple-**O**utput
SISO	**S**ingle-**I**nput **S**ingle-**O**utput
SLIDE	**S**emi-**LIN**ear **DE**fuzzification
SO(F)C	**S**elf-**O**rganizing (**F**uzzy) **C**ontrol(ler)
SV	**S**upport **V**alue
TMS	**T**ruth-**M**aintenance **S**ystem
WFM	**W**eighted-**F**uzzy-**M**ean
XCOA	e**X**tended **C**entre-**O**f-**A**rea

Summary

Fuzzy control is a hot topic, considering the large amount of publications on this topic. However, there still exists a lot of misunderstanding of fuzzy control. In this thesis fuzzy control is analyzed which hopefully results in a demystification on the one hand, and a profilation on the other.

Fuzzy sets were introduced by Zadeh (1965) although the underlying concept has been recognized long before that; an overview is given by (Dubois et al., 1993b). Fuzzy sets provide the ability to model the ambiguity and impreciseness of human classification. Many operations on fuzzy sets have been defined. They reduce to operations from classical set theory when classical sets are involved instead of fuzzy sets. The most important concept within fuzzy set theory is the *extension principle*.

Fuzzy logic is based on fuzzy set theory. Set-theoretic operations from classical set theory have no unique equivalents in fuzzy set theory and a problem is to choose operators which are suited the most to represent the logical operation in a certain context. Although some general guidelines can be given, there are no clear rules on when to apply which "type" of an operation. Fuzzy propositions are the basis of *fuzzy reasoning*. Within the field of fuzzy reasoning, two approaches can be distinguished: *local inference* and *global inference*. Local inference means that each rule in inferred and the results of the inferences of the individual rules are aggregated afterwards. Global inference means that the rules are aggregated and used for inference as a whole. When implications are modeled by conjunctions, the results of local and global inference are equal when the aggregation operator equals the combination operator of the composition method.

A *fuzzy controller* is basically the application of the compositional rule of inference. When the inputs and outputs of the fuzzy controller are numerical, sensor and actuator signals, translation from numerical values to fuzzy set representations and vice versa are necessary. The first is known as fuzzification, the latter as defuzzification. In the application of fuzzy control, the following stages can be distinguished: matching of data with rule premises (includes fuzzification), determination of degrees of fulfillment for the rules, aggregation of results of individual rules and defuzzification to obtain a numerical controller output. Because of fuzzification and defuzzification, a fuzzy controller can be regarded as an input-output mapping. Regarding a fuzzy controller (or model) as such a mapping, it can be shown that the mapping is characterized by tuples in a hyperspace and each tuple represents a fuzzy rule. The fuzzy reasoning performs an interpolation between these tuples in that hyperspace, resulting in a (nonlinear) input-output mapping.

Based on this idea of interpolation, it can be shown that many non-trivial nonlinearities can be introduced by the used operators, defuzzification method, shape of membership functions of fuzzy sets and the relations between fuzzy sets on a universe. If the non-linearity of the mapping should (only) be defined by the fuzzy rules, the choices for the fuzzy controller parameters should be as follows: product for conjunction, summation for disjunction and aggregation, fuzzy-mean defuzzification, fuzzy partitions on the input universes and a triangular norm for the implication. Using triangular norms for implications, in fact, means that the implication is interpreted as a conjunction. When using an implication which complies with the classical implication, the problem of indetermination of the output can occur. This is highly undesired in in-line control. Therefore, in direct control, it is better to use conjunctions for implications in fuzzy control applications.

Different approaches to *adaptive fuzzy control* exist. Well known is the *self-organizing controller*. This type of adaptive fuzzy control adjusts the rule base by means of a rule base modifier which uses the result of a performance table and a minimal model. Typical for the self-organizing controller is that it is based on local optimization and global optimization is not guaranteed.
Another approach is modeling a process to be controlled by a fuzzy relation and using this fuzzy relation to derive control actions by means of causality inversion: *fuzzy associative memories*. When the process exhibits time delays or system dynamics which are modeled by time delays, a process model is needed to estimate future process outputs. These estimations are necessary to derived control action by causality inversion.
Adaptation by means of *gradient-descent optimization* is a technique which is used in "fuzzy neural networks". The membership functions of the fuzzy sets on the input universes and the (numerical) consequents of the fuzzy rules are adapted by a gradient-descent learning rule. In many cases the linguistic interpretability is lost after adaptation.
Adaptation of fuzzy or conventional controllers by *fuzzy supervisors* is also possible. A number of examples of this approach exists. In some cases a translation to an in-line fuzzy controller (without supervisor) is possible, but the main advantage of a conventional

controller supervised by a fuzzy supervisor is that it can provide a better “interface” to the control strategy.

Fuzzy control can be regarded as only a small part of the much broader framework of *approximate reasoning and possibility theory*. Approximate reasoning provides a method for modeling human classification and reasoning, including natural language modeling and understanding. Although much research in this field has been done, there are only a few applications known these days. A major disadvantage is the lack of practical applicability due to severe calculational effort and/or memory requirements to perform the reasoning according to the theory of approximate reasoning.
Analytical solution to inference are only possible in a few restricted cases. An example of such a case is the inference break-up when the Kleene-Dienes implications is used. The main problem is that the results of local inference are usually less restrictive (equal at best) that the results of global inference. An inference break-up can reduce the inference of a complex rule base to inference of a number of simple rule bases.

Considering the fact that fuzzy control uses only a small part of the framework it resides in, one can question whether the success of fuzzy control is because of the use of fuzzy logic. In many cases a fuzzy controller can be simplified to a look-up table and an interpolation method to provide the “fuzzy inference”. The same idea can be used to simplify adaptive fuzzy controllers. The self-organizing controller can be simplified by a look-up table and an interpolation method where the elements of the look-up table, representing the (numerical) consequents of the fuzzy rules, are adapted. The adaptive fuzzy systems based on gradient-descent optimization can be simplified to the adaptation of a look-up table of which the elements and the index vectors, representing the centers of the fuzzy sets on the input universes, are adapted. Hence, these simplifications can reduce the fuzzy aspect of fuzzy control to a user-interfacing concept during the design stage.

Vage logica in de regeltechniek

Fuzzy control geniet veel aandacht in deze tijd, gezien het grote aantal publikaties over dit onderwerp. Er zijn echter nog steeds een groot aantal misverstanden wat betreft fuzzy control. In dit proefschrift is fuzzy control geanaliseerd, hetgeen hopelijk resulteert in een “demystificatie” aan de ene kant en een “profilering” aan de andere kant.

Fuzzy sets zijn geïntroduceerd door Zadeh (1965), hoewel het onderliggende concept als zodanig al een lange tijd hiervoor bekend was; een overzicht hiervan is gegeven door Dubois et al. (1993b). Fuzzy sets maken het mogelijk om de dubbelzinnigheid en onnauwkeurigheid van menselijke classificatie(s) te modelleren. Er zijn vele operaties op fuzzy sets gedefinieerd. Deze operaties reduceren tot operaties uit de klassieke set theorie wanneer klassieke sets in plaats van fuzzy sets worden gebruikt. Het belangrijkste concept in the fuzzy set theorie is het *“extension principle”*.

Fuzzy logic is gebaseerd op fuzzy set theorie. Set-theoretische operaties uit de klassieke set theorie hebben geen unieke equivalenten in fuzzy set theorie en het is een probleem om te bepalen welke operator het best geschikt is binnen een bepaalde context. Hoewel een aantal vage richtlijnen kunnen worden opgesteld, zijn er geen algemene vuistregels voor het gebruik van bepaalde operatoren. Fuzzy proposities zijn de basis elementen voor *fuzzy redeneren*.
Bij fuzzy redeneren kunnen twee werkwijzen worden onderscheiden: *lokale inferentie* en *global inferentie*. In geval van lokale inferentie worden alle rules individueel gebruikt voor

inferentie, waarna de resultaten worden samengenomen. In geval van globale inferentie worden eerst alle rules samengenomen en daarna wordt er inferentie toegepast. Indien implicaties gemodelleerd zijn door conjuncties en de aggregatie operator is dezelfde als de combinatie operator in de compositie methode, is het resultaat van lokale en globale inferentie gelijk.

Een *fuzzy controller* is in feite de applicatie van de “compositional rule of inference”. Als de ingangen en uitgangen van de regelaar numeriek zijn (sensor en actuator signalen), zijn omzettingen van numerieke waarden naar fuzzy set representaties en vice versa nodig. Het eerste staat bekend als fuzzificatie, het laatste als defuzzificatie. In fuzzy control applicaties kunnen de volgende fasen worden onderscheiden: het matchen van data en de rule premises (inclusief fuzzificatie), het bepalen van de “degrees of fulfillment” voor elke rule, aggregatie van de resultaten van de rules en defuzzificatie om een numerieke regelwaarde te verkrijgen.
Vanwege de fuzzificatie en defuzzificatie kan een fuzzy controller worden gezien als een input-output mapping. Indien een fuzzy controller (of model) wordt beschouwd als een input-output mapping, kan deze mapping gekarakteriseerd worden door een aantal punten in een hyperruimte en elk punt hierin representeert een rule. De fuzzy inferentie zorgt voor een interpolatie tussen deze punten, hetgeen resulteert in een (niet-lineaire) input-output mapping.

Wanneer men dit idee van interpolatie beschouwt, kan aangetoond worden dat er vele niet-triviale niet-lineariteiten worden veroorzaakt door de gebruikte operatoren, defuzzificatie methoden, vormen van fuzzy sets en de relaties tussen fuzzy sets op een (input) domein. Als de niet-lineariteiten van de input-output mapping enkel door de rules bepaald dienen te worden, dient er voldaan te worden aan de volgende criteria: product voor conjunctie, sommatie voor disjunctie en aggregatie, fuzzy-mean defuzzificatie, fuzzy paritities voor de input domeinen en een triangular norm voor de implicatie. Het gebruik van triangular norms voor implicaties betekent, in feite, dat de implicatie word opgevat als een conjunctie. Wanneer implicaties worden gebruikt die op de klassieke implicatie zijn gebaseerd, kan het probleem van een onbepaalde regeluitgang ontstaan. Dit is zeer ongewenst in in-line control. Daarom zijn op conjunctie gebaseerde implicaties de betere keuze in (direct) fuzzy control.

Verschillende types van adapatieve fuzzy control kunnen worden onderscheiden. Een bekend voorbeeld is de *self-organizing controller*. Deze adaptieve fuzzy controller adapteert de rule base door middel van een rule base modifier, gebruik makend van een performance tabel en een minimaal model. Karakteristiek voor dit type controller is dat het is gebaseerd op lokale optimalisatie en dat globale optimalisatie niet gegarandeerd is.
Een andere aanpak is het modelleren van het te regelen process door een fuzzy relatie en deze relatie te gebruiken om regelakties te bepalen door middel van causaliteitsinversie: *fuzzy associatieve geheugens*. Als het proces dode tijd heeft of dynamica welke gemodel-

leerd kan worden door dode tijd, dan is er een proces model nodig om toekomstig proces gedrag te voorspellen. Deze voorspellingen zijn nodig voor het bepalen van regelakties vanwege de causaliteitsinversie.
Adaptatie door middel van *gradient-descent optimalisatie* is een techniek die gebruikt wordt in “fuzzy neural networks”. De membership functies van de fuzzy sets voor het ingangsdomein en de (numerieke) consequents van de rules worden geadapteerd door een gradient-descent leerregel. In veel gevallen gaat de linguistische interpreteerbaarheid door de adaptatie verloren.
Adaptatie van fuzzy of conventionele regelaars door *fuzzy supervisors* is ook een mogelijkheid. Er bestaan een aantal voorbeelden van deze aanpak. In sommige gevallen is er een vertaling naar een in-line, op Sugeno rules gebaseerde, fuzzy regelaar, maar het belangrijkste voordeel van een conventionele regelaar in combinatie met een fuzzy supervisor is dat het een betere “interface” van de regelstrategie kan opleveren.

Fuzzy control kan worden beschouwd als slechts een klein gedeelte van het veel grotere raamwerk van *approximate reasoning en possibility theorie*. Approximate reasoning voorziet in een methode om menselijke classificatie en redeneren te modelleren (inclusief modellering en begrip van natuurlijke taal). Hoewel veel onderzoek op dit gebied is gedaan, zijn er slechts weining applicaties bekend vandaag de dag. Een groot nadeel is de praktische toepasbaarheid door de grote benodigde rekencapaciteit en computergeheugen om approximate reasoning toe te passen volgens de theorie.
Analytische oplossingen voor de inferentie zijn slechts in een aantal gelimiteerde gevallen mogelijk. Een voorbeeld van zo’n geval is wanneer de “inference break-up” methode wordt toegepast. Het voornaamste probleem is dat de resultaten van lokale inferentie over het algemeen minder informatief zijn dan de resultaten van globale inferentie. Een “inference break-up” kan de inferentie van een complexe rule base reduceren tot de inferentie van een aantal simpele rule bases.

Gezien het feit dat fuzzy control slechts een klein gedeelte is van het raamwerk waar het onderdeel van is, kan men zich afvragen of het succes van fuzzy control te danken is aan het gebruik van fuzzy logic. In veel gevallen kan een fuzzy controller gereduceerd worden tot een look-up table en een interpolatie methode welke de “fuzzy inferentie” verzorgt. Hetzelfde idee kan gebruikt worden om adaptieve fuzzy controllers te versimpelen. De self-organizing controller kan vereenvoudigd worden tot een look-up table en een interpolatie methode, waarbij de elementen (numerieke consequents van de rules) van de look-up table worden geadapteerd. De adaptieve fuzzy systemen die adapteren door middel van gradient-descent optimalisatie kunnen vereenvoudigd worden tot een look-up table en een interpolatie methode, waarbij niet alleen de elementen van de look-up table, maar ook de index vectoren worden geadapteerd. Deze simplificaties reduceren het fuzzy aspect van fuzzy controllers tot een user-interfacing concept tijdens de ontwerpfase.

Curriculum vitae

René Jager was born in Amsterdam, the Netherlands, on April 13, 1967. He had his pre-university education at the R.S.G. in Purmerend, from which he graduated in 1985. He studied Electrical Engineering at the Delft University of Technology, Delft, the Netherlands, from which he graduated with honors in 1990 on a thesis on a real-time expert system for process control with minimal process knowledge.
In April 1990 he became a research assistant at the Control Laboratory of the Electrical Engineering Department of the same university, where he worked on his Ph.D. thesis in the field of fuzzy logic and control.

Acknowledgments

Acknowledging the support of others for finishing my Ph.D. work poses the problem whether to focus on technical or moral support. To avoid this problem, I thought it was best to fall back on classical set theory and define a set with people I am grateful to with respect to accomplishing this Ph.D. work. In figure F.1 a classical set is given with the names of those I think all have contributed, one way or another, to this work. Besides those mentioned in figure F.1, I would like to thank Mrs. J.B. Zaat-Jones for her revisions of my American text, and students that contributed to my research.

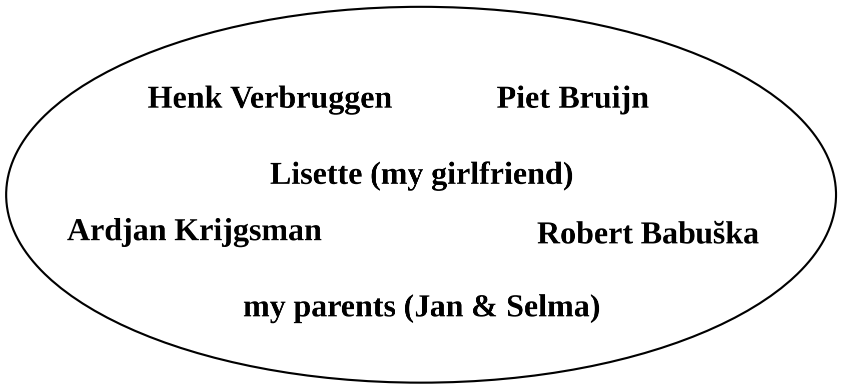


Figure F.1: *Classical set with people I am grateful to with respect to accomplishing this Ph.D. work.*

Author index

Subject index