METHODOLOGY OF FUZZY CONTROL: AN INTRODUCTION

H.T. NGUYEN
Department of Mathematical Sciences
New Mexico State University
Las Cruces, NM 88003-8001, USA
email `hunguyen@nmsu.edu`

AND

V. KREINOVICH
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
email `vladik@cs.utep.edu`

Abstract. Fuzzy control is a methodology that translates the experience of human operators, experience that is usually described by using words of natural language like "small", "a little bit", etc., into a precise control strategy. This chapter gives an introduction to this methodology.

1. INTRODUCTION: WHY FUZZY CONTROL

Control is necessary. In today's world, many systems can be controlled.

- Some things, such as cars, planes, ships, are actually *designed* to be controlled. In this case, the problem is how to control them so as to achieve certain goals.
- Some systems are not designed by us, but we have learned how to control them: we can, to some extent, control weather, we can control pollution, etc. In these cases, we also need to find out the best ways to control.

Controlling a system means that we *monitor* (i.e., frequently measure) some characteristics $x_1, \ldots, x_n$ of this system, and, depending on the values of these characteristics, we apply different controls $u_1, \ldots, u_m$. An algorithm that transforms the senor inputs $x_1, \ldots, x_n$ into the corresponding control values $u_i(x_1, \ldots, x_n)$ is called a *control strategy.*

1.1. TRADITIONAL CONTROL METHODOLOGY AND ITS LIMITATIONS

Traditional approach to control. The traditional approach to control consists of the following:

- First, we try to describe the behavior of the system in *precise* mathematical terms, i.e., we try to come up with the exact *model* of the system.
  - In some cases, especially when we control a *man-made* system, we may already *know* how exactly this system reacts to different controls. In other words, for each possible initial state $\vec{x} = (x_1, \ldots, x_n)$, and for each possible control values $u_1, \ldots, u_m$, we know how exactly the state will change if we apply this control, i.e., we know how the time derivatives $\dot{x}_i$ depend on x_i and u_j:
  $$\dot{x}_i = f_i(x_1, \ldots, x_n, u_1, \ldots, u_n). \tag{1}$$
  This system of differential equations forms a *model* of the controlled system.
  - In other cases, we *do not know* the differential equation *model.* Therefore, we analyze the system:
    * we apply different controls in different states,
    * determine how the state changes, and thus,
    * determine the desired differential equations model (1).
- Second, we try to describe in precise terms what we want to achieve. We want the control that is the *best*, in the sense of some criterion; for example, we may want to get to the Moon in the shortest possible time, or we may want to design a control strategy for the subway that leads to the most comfortable (smoothest) ride, etc.
  - Some *objectives* (such as "the shortest time to the Moon") are already formulated in well-defined terms.
  - Other *objectives* (such as "the most comfortable ride") need to be re-formulated in these terms.
- Now that the controlled system is described in precise mathematical terms, and the objective function is described in mathematical terms, we can, for each control strategy and for each initial state, determine how exactly the system will change and what the resulting value of the control function will be. Our goal is then to *find the control strategy* for which the resulting value of the objective function is the largest

possible. This is a *well-defined mathematical optimization problem*, and traditional control theory has developed many methods for solving these optimization problems and designing the corresponding control strategies.

Traditional control theory is not always applicable: three possible situations. Traditional control theory has many important applications. There are, however, practical cases when this theory is not applicable. Indeed, to apply the traditional control theory, we must:

- know the *model* of the controlled system (described in terms of differential equations);
- know the *objective function* formulated in precise terms;
- *be able to solve* the corresponding mathematical optimization problem.

If one of these conditions is not satisfied, then traditional control methodology is not applicable.

Sometimes, we know the model and the objective function, but we cannot solve the optimization problem. In some cases, we *know the model and the objective function*, but we are *unable to solve* the corresponding optimization problem. There may be three reasons for this inability:

- It could be that the optimization problem is *very complicated* (e.g., highly non-linear), so many attempts to solve it has *not yet* lead to a successful *algorithm*. This lack of success is not necessarily caused by an insufficient effort: It is known that in general, the problem of finding the optimal control is computationally intractable; see, e.g., (Abello *et al.*, 1994; Smith *et al.*, 1995a).
- In some complicated cases, methods of traditional control theory do provide an algorithm, but this *algorithm is* so *complicated* that its running time exceeds the time that we have to make a control solution.

  The best example of such a situation is weather prediction. In principle, we know a system of integro-differential equations that describes the weather, and we have pretty accurate measurements of the current state of the atmosphere. However, these computations are very time-consuming:

  * even to predict the weather for *a few hours* ahead, we would need to spend quite some time on the world fastest supercomputer; and
  * if we want to make a *long-term* prediction, e.g., a for a month ahead, we would need much more than one month of computations.

If computing predictions for next month takes more than a month, then it is no longer a prediction: we will *observe* the next month's weather *before* the *predicting* computations are over.

- The third possibility is that the *problem is new* and unusual for control theory, and so, algorithms for solving it have not yet been developed.

  For example, *parking a car* is an example of a problem that traditional control theory has not considered until recently.

Sometimes, we know the model, but we do not know the objective function. In some cases, we *know* the exact *model*, but we *do not know* the exact *objective function*.

For example, if we design a control system for subway, the intended goal is to make the subway ride the most comfortable. Unfortunately, there is no well-accepted formalization of what "comfortable" means.

Sometimes we do not even know the model. Finally, in many control situations, we do not even know the model of the controlled system. There are two reasons for that:

- In many practical applications, especially in applications to manufacturing, we can, *in principle*, measure all the possible variable and determine the model exactly, but this will drastically (and unnecessarily) increase the costs.

  For example, the exact form of the differential equations that describe an airplane or a car depend on its load.

  * A failure of a plane can lead to catastrophic consequences. Therefore, a plane is equipped with lots of sensors. When a plane is designed, a model is formed and tested before the actual plane is allowed to fly.
  * For a car, most possible faults are not that dangerous. Therefore, there are usually much fewer sensors, and models, if any, are only *approximate*. It is, in principle, possible to add sensors to a car and to design its exact model, but this would make the car almost as expensive as a small plane.

- In other practical situations, the main goal of the controlled system is to *explore the unknown*:

  • to control a rover on a new planet, with the terrain of unknown type;

  • to control a catheter during a microsurgery, etc.

  In such situations, the entire objective of the control is to learn as much about the system, and we cannot have a precise model of this system before the control is over.

If we cannot apply traditional control methodology, how can we control? A problem. In the case of exploration of the unknown, there was a *fundamental* reason why we cannot use traditional control methods. In other cases, in principle, traditional control methods may be possible at some later moment of time, but as of now, they do not help us to solve the practical problem of control.

So, what can we do if traditional control methods cannot be used?

1.2. WHAT WE CAN USE INSTEAD OF THE CLASSICAL CONTROL MODEL

Often, we have an additional expert knowledge. In many practical control situations, we have *expert operators* who successfully control the desired system: expert astronauts know how to dock and land the Space Shuttle; expert operators know how to operate a chemical plant, etc. Therefore, it is desirable to extract the "control rules" from the experts and use this knowledge in an automated controller.

Comment. Control rules do not always come from the top experts, there are two extreme situations when these rules come from common sense:

- One situation is *everyday control*, e.g., control of a car, when almost every person considers himself an expert. In such situations, instead of the "expert rules" coming from top experts, we must have *common sense* rules.
- Another extreme is designing controllers for new systems for which no controllers have been known before, and for which, therefore, there aren't yet any expert controllers. In such situations, we also have to reply on common sense only. As fuzzy control methodology is applied to new areas, such applications become more and more frequent; see, e.g., (Kosko, 1997).

Why do we need to extract expert knowledge at all? By definition, an expert is a person who is well knowledgeable about the object. An expert helicopter pilot is a person who can control a helicopter well; an expert doctor is a person who can cure different diseases, etc. If we already have such an expert, why do we need an automated system? There are two reasons for that:

- The first reason is that in every area,
  - there are only a *few* top experts, and
  - there are *many* problems to be solved.

  It is, usually, simply physically impossible to have a top expert for each problem: we cannot have a top surgeon for every surgery, we

cannot have a top helicopter pilot for every helicopter in the world. It is therefore desirable to develop a *computer program* that would somehow incorporate the expert knowledge and give advise comparable in quality with the advise of the top experts. In other words, we want to design a computer program that somehow *simulates* the expert.
– Often, it is desirable to avoid using an expert altogether: experts are expensive to use, error-prone, and in some dangerous environments, it is desirable to make completely automatic control systems. E.g., we want to control robots who travel into the volcanos or go to other planets.

In both cases, we need a control program that incorporate expert knowledge. Such programs are called *intelligent control systems.*

Why cannot we simply extract the control strategy $u_i(x_1, \ldots, x_n)$ **from an expert?** At first glance, the problem seems relatively simple: Since the person is a real expert, we simply ask her multiple questions like "suppose that x_1 is equal to 1.2, x_2 is equal to 1.3, …, what is u?" After asking all these questions, we will get many patterns, from which we will be able to extrapolate the function $f(x_1, \ldots, x_n)$ using one of the known methods. For example, we may ask the helicopter pilot about all possible combinations of sensor readings and and write down what exactly control the expert recommends. Alas, there are two problems with this idea:

– First, there is a *computational* problem: Since we need to ask a question for each combination of sensor readings, we may end up having to ask *too many* questions. This fact makes our idea *difficult* to implement but *not* necessarily *impossible*: Indeed, we only need to do this lengthy questioning once, so we may afford to spend years, if necessary (actually, the design of the first expert systems did take years).
– However, there is another, more serious problem with this idea that makes it, in most problems, *impossible* to implement. Indeed, this idea may sound reasonable until we try applying it to a skill in which practically all adults consider themselves experts: driving a car. If you ask a driver a question like: "you are driving at 55 miles per hour when the car in 30 feet in front of you slows down to 47 miles per hour, for how many seconds do you hit the brakes?", nobody will give a precise number.

What can we do? We might install measuring devices into a car (or into a driving simulator), and simulate this situation, but the resulting braking times may differ from simulation to simulation.

The problem is not that the expert has some precise number (like 1.453 sec) in mind that he cannot express in words; the problem is that one time it will be 1.39, another time it may be 1.51, etc.

An expert usually expresses his knowledge by using words from natural language. An expert *cannot*, usually, express his knowledge in *precise* numerical terms (such as "hit the brakes for 1.43 sec"), but he *can* formulate his knowledge by using *words* from natural language. For example, an expert can say "hit the brakes for a while".

So, the knowledge that we can extract from an expert consists of statements like "if the velocity is a little bit smaller than maximum, hit the breaks for a while".

There may be also another uncertainty involved, e.g., an expert may say something like "if a patient has a temperature of 100 degrees, and a headache, and ... ⟨several other symptoms⟩, then, most probably, it is a flu". Here, the words "most probably" are the words from natural language that an expert uses to describe his knowledge.

Enter fuzzy control.

- We *know* expert's control rules formulated by words from natural language.
- We *want* to produce the precise control strategy.

The methodology that transforms the informal ("fuzzy") expert control rules into a precise control strategy is called *fuzzy control*. This methodology was first proposed by E. Mamdani in (Mamdani, 1974; Mamdani, 1977). In this chapter, we will describe how exactly this transformation is done.

2. HOW TO TRANSLATE FUZZY RULES INTO THE ACTUAL CONTROL: A MULTI-STEP PROCEDURE AND ITS EXPLANATIONS

2.1. GENERAL IDEA

Toy example: a thermostat. We will illustrate the main ideas of fuzzy control methodology on the example of another situation in which everyone feels himself an expert: controlling a *thermostat*.

For simplicity, we will consider a *toy*, simplified thermostat in which a thermometer shows the current temperature and a dial allows us to control the temperature:

- turning the dial to the *left* makes it *cooler*;
- turning it to the *right* makes it *warmer*.

The angle on which we turn the dial will be denoted by u.

We will also assume, for simplicity, that we know the comfort temperature T_0 that we try to achieve. Strictly speaking, the input to our control is the temperature T. However, in reality, our control decisions depend not so much on the *absolute* value of T but rather on the *difference* $T - T_0$ between the actual temperature T and the ideal temperature T_0. Since this difference is important, we will use a special notation for it: $x = T - T_0$. Our goal is to describe the appropriate control u for each possible value x, i.e., to describe a function $u = f(x)$.

For such an easy system, we do not need any expert to formulate reasonable rules; we can immediately describe several reasonable control rules:

- If the temperature T is *close* to T_0, i.e., if the difference $x = T - T_0$ between the actual temperature is *negligible*, then no control is needed, i.e., u should also be negligible.
- If the room is slightly overheated, i.e., if x is positive and small, we must cool it a little bit (i.e., u must be negative and small).
- If the temperature is a little lower than we would like it to be, then we need to heat the room a little bit. In other terms, if x is small negative, then u must be small positive.

We can formulate many similar natural rules. For simplicity, in our toy example, we will restrict ourselves to these three. As a result, we get the following three rules:

- if x is negligible, then u must be negligible;
- if x is small positive, then u must be small negative;
- if x is small negative, then u must be small positive.

Toy example re-formulated. Our goal is to make these and similar rules accessible for the computer. Before describing how to do it, let us first reformulate these rules in a way that will somewhat clarify these rules. Namely, in the formulation of these rules, we want to clearly separate the *properties* (like "x is negligible") and *logical connectives* (like "if ... then"). To achieve this separation, let us introduce a shorthand notation for all the properties, and let us use standard mathematical notations for logical connectives:

- $N(x)$ will indicate that x is negligible;
- $SP(x)$ will indicate that x is small positive;
- $SN(x)$ will indicate that x is small negative;
- $A \to B$ is a standard mathematical notation for "if A then B".

In these notations, the above three rules take the form:

$$N(x) \to N(u); \tag{2}$$

$$SP(x) \to SN(u); \tag{3}$$

$$SN(x) \to SP(u). \tag{4}$$

General case. In general, the expert's knowledge about the dependence of u on $x_1, \ldots, x_n$ can be expressed by several rules of the type

If x_1 is A_{r1}, ..., and x_n is A_{rn}, then u is B_r.

Here, $r = 1, \ldots, R$ is the rule number, and A_{ri} and B_r are words from natural language that are used in r-th rule, like "small", "medium", "large", "approximately 1", etc.

- In our toy example, we only had *one* input variable: the temperature T (or, to be precise, the difference x between T and the desired temperature T_0).
- In the general case, we have *several* input variables, so, in addition to the logical connective "if ... then", we need another logical connective "and".

In mathematics, "and" is usually denoted by & (or by $\wedge$). If we use the standard mathematical notation for "if ... then" and "and", we can reformulate the above rules as follows:

$$A_{r1}(x_1) \mathbin{\&} \ldots \mathbin{\&} A_{rn}(x_n) \to B_r(u), \tag{5}$$

where $r = 1, \ldots, R$. The set of rules is usually called a *rule base.*

Comment. In this text, we will only consider the case when a rule base consists of *straightforward* if–then rules. In reality, experts may have some additional information about the object, which may be formulated in a more complicated form. For example, an expert may formulate several different rules bases, and then formulate "meta-rules" that decide which of the rule bases are applicable.

General idea. Our goal is to represent a rule base in a computer. A rule base has a clear structure:

- A rule base consists of *rules.*
- Each rule, in its turn, is obtained from *properties* (expressed by words from natural language) by using *logical connectives.*

In view of this structure, it is reasonable to represent the rule base by first representing the basic elements of the rule base, and then by extending this representation to the rule base as a whole. In other words, it makes sense to follow the following methodology:

- First, we represent the basic properties $A_{ri}(x_i)$ and $B_r(u)$.
- Second, we represent the logical connectives.
- Third, we use the representations of the basic properties and of the logical connectives to get the representations of all the rules.
- Fourth, we combine the representations of different rules into a representation of a rule base.

As a result of these four steps, we get an "advising" (expert) system. For example, if we apply these four steps to the medical knowledge, we ideally, get a system that, given the patient's symptoms, provides the diagnostic and medical advise; e.g., it can say that most probably, the patient has a flu, but it is also possible that he has bronchitis. Such an advice, coming from an expert system, is used by a specialist to make a decision.

In *control*, we want a system to automatically make a decision based on its own conclusions. Therefore, for control situations, we need an *additional*, fifth follow-up step:

– The computer-based system makes a decision.

In the following text, we will describe how these five steps are implemented.

Fuzzy control is indeed very successful. The resulting five-step methodology is indeed very successful in control. The resulting *fuzzy control* is used in various areas ranging from appliances (camcorders, washing machines, etc.) to automatically controlled subway trains in Japan to cement kilns to regulating temperature within the Space Shuttle. The reader who is interested in learning more about fuzzy methods in general and fuzzy control in particular is advised to start, e.g., with one of the following textbooks:

– (Klir *et al.*, 1995) gives a more *engineering*-oriented introduction;
– (Nguyen *et al.*, 1997a) is more oriented towards *mathematical* methods.

2.2. MEMBERSHIP FUNCTIONS AND WHERE THEY COME FROM

Before we represent a property $P(x)$, let us represent the expert's degree of belief in this property for each x. To represent a property like "x is positive small" ($SP(x)$), we must be able, for every possible value of the temperature difference x, to represent the expert's opinion of whether this particular value x is indeed small. To represent this opinion, we must solve the following problem:

– All the properties that are traditionally analyzed in mathematics, such as "x is positive", are *crisp*, in the sense that every real number is either positive, or not.
– On the other hand, properties like "x is small" (in which we are interested) are *not crisp*. To be more precise:
  - Some values x are so small that practically everyone would agree that they are small.
  - Some values x are so huge that practically everyone will agree that they are not small.
  - However, for many intermediate values x, it is difficult to decide whether x is small or not, and experts may disagree:

* For a researcher who performs temperature-sensitive experiments in the lab, the difference of $x = 0.5$ degrees may not be small at all.
* However, for a living room, even the difference of ± 5 degrees is usually not only small, but even negligible.

Such non-crisp properties are called *fuzzy*. This term was introduced by Lotfi Zadeh in his pioneer paper (Zadeh, 1965).

How can we represent fuzzy properties?

– If $P(x)$ is a *crisp* property (like "positive"), then for every x, $P(x)$ is either true or false.

  Representing the corresponding truth value in the computer is easy: "true" is usually represented by 1, and "false" by 0.

– If $P(x)$ is a *fuzzy* property (like "small"), then in general, for a real number x, an expert may not be sure whether x satisfies this property P or not. For example, for "small", the larger the value x, the less confident the expert that x is small. So, for different values x, an expert may have different "degrees of confidence" that x satisfies this property P:

  • For some values x, the expert may be absolutely sure that the statement $P(x)$ is true. In this case, the degree of confidence corresponds to "true".

  • For some other values x, the expert may be absolutely sure that the statement $P(x)$ is false. In this case, the degree of confidence corresponds to "false".

  • For yet other values x, the expert is neither sure that $P(x)$ is true, nor he is sure that $P(x)$ is false. In this case, the expert's degree of confidence is *intermediate* between "true" and "false".

  How can we represent these intermediate degrees of belief in a computer?

  If 0 corresponds to "false", and 1 to "true", then it is natural to represent degrees of confidence that are intermediate between "false" and "true" by numbers from the interval $(0, 1)$.

Thus, to describe *arbitrary* degrees of certainty, we must describe:

– either absolute truth (represented by 1),
– or absolute falsity (represented by 0),
– or intermediate degrees of confidence (represented by numbers from the open interval $(0, 1)$).

Therefore, we arrive at the following conclusion:

We must use real numbers from the interval $[0,1]$.

Historical comments.

- From the *mathematical* viewpoint, what we are doing is a very natural idea: we extend the traditional *2-valued* logic (in which the truth value of each statement is an element of a 2-element set $\{$"true", "false"$\} = \{0,1\}$), to the *interval* $[0,1]$. Mathematicians have been developing the corresponding mathematical formalisms (called *multiple-valued logics*) since the pioneer works of K. Lukasiewicz in the early 1920s.
- In *computer science*, the idea of using numbers from the interval $[0,1]$ to describe different degrees of confidence was also first proposed by Lotfi Zadeh in his pioneer paper (Zadeh, 1965).

Comments.

- We have used the term *degree of confidence*; other words are also used, such as *degree of belief*, *degree of certainty*, *truth value*, *subjective probability*, etc. From our viewpoint, all of these terms are good but not perfect:
    - On one hand, each of these terms bring with itself some intuitive meaning.
    - On the other hand, the intuitive meaning brought by these words may be sometimes misleading. For example, the terms "belief" and "subjective probability" may lead to a confusion with the "objective probability", i.e., crudely speaking, with a *frequency* of a certain event (like tossing a coin).

  Since we cannot select one term as the best, we will use these terms interchangingly.
- The interval $[0,1]$ is probably the most natural to use, but some expert systems use a *different interval* to represent uncertainty. For example, historically the first successful expert system MYCIN (Shortliffe, 1976; Buchanan *et al.*, 1984) developed at Stanford University for diagnosing rare blood diseases, used values from the interval $[-1,1]$ to represent uncertainty.
- The main advantage of using an interval (e.g., $[0,1]$) is that elements from this interval (i.e., real numbers) can be easily represented and easily processed in a computer. However, sometimes, to get a more adequate representation of the expert's degree of confidence, we may need a more sophisticated representation. For example, an expert may not be sure about his degree of confidence. To represent this uncertainty, we may want to describe the expert's degree of confidence not by a *single* number, but by an *in-*

terval of possible numbers (see, e.g., surveys (Nguyen *et al.*, 1995; Nguyen *et al.*, 1997)):

- If an expert has no information about the object, he cannot have any definite degree of confidence. To describe this situation, we will use the entire interval $[0, 1]$ as the description of this expert's degree of confidence.
- If an expert has a certain information, them we may use a narrower interval, or even a number (i.e., a degenerate interval).

There are several even more sophisticated ways of representing uncertainty. In this chapter, we will only consider the simplest possible way of representing degrees of confidence: by using real numbers.

We need to elicit a numerical degree of confidence from the expert. We agreed that for each statement of the type $P(x)$, where:

- P is a fuzzy property (i.e., a property formulated by words from natural language), and
- x is a real number,

a reasonable way to represent the expert's degrees of confidence is to use numbers from the interval $[0, 1]$. The natural next question is: how can we "extract", elicit the value from an expert?

Direct elicitation is impossible. The ideal situation would be if an expert would directly provide us with this number, but this is, unfortunately, not realistic:

- we want numbers from the interval $[0, 1]$, because they are very *natural* for a *computer*; but
- real numbers from the interval $[0, 1]$ are *not natural* for a *human expert*; it is very difficult for most experts to express their degree of confidence in a given statement by a real number.

Since we cannot elicit these numbers *directly*, we have to use *indirect* elicitation techniques.

There are several dozen different elicitation techniques; in this chapter, we will only describe the most frequently used ones.

First elicitation method: selecting on a scale. If we cannot elicit a *real* number from an expert, maybe we can elicit *some* number from him, and then convert the result into a real number from the interval $[0, 1]$. In many cases, this is indeed possible: many polls ask us to estimate our degree of confidence, or our degree of belief, etc., on a certain integer scale, e.g., on a scale from 0 to 5, or on a scale from 0 to 10. So, we can ask an expert to mark his degree of confidence in a given statement $P(x)$ on a given scale 0 to S (0 to 5, 0 to 10, etc). On this scale:

– 0 corresponds to "absolutely confident that $P(x)$ is false";
– S corresponds to "absolutely confident that $P(x)$ is true";
– intermediate marks represent different degrees of uncertainty.

As a result of this procedure, we get an integer from 0 to S. The most natural way to convert these integers $0, 1, 2, \ldots, S$ into real numbers from the interval $[0, 1]$ is to *re-scale* the interval $[0, S]$ into the interval $[0, 1]$ by dividing by S. So, if an expert has chosen the mark s on a scale from 0 to S, we will take s/S as the desired degree of confidence.

For example, if an expert selects, 3 on a scale from 0 to 5, then we take $3/5 = 0.6$ as the desired degree of confidence.

The first elicitation method is not always applicable. The above method seems reasonable and is computationally very simple. The problem with this method is that experts often do not feel comfortable expressing their degree of confidence by numbers on an (integer) scale. Usually, these experts only distinguish between three cases:

– a given statement $P(x)$ is true;
– a given statement $P(x)$ is false;
– we do not know whether a given statement $P(x)$ is true or false.

If we have such experts, how can we elicit their degree of confidence?

Second elicitation method: polling. We have already mentioned that the first elicitation method is taken from the experience of the polls. In addition to the polls that require us to mark a point on a scale, there are other polls in which we only answer "yes", "no", or "undecided". For example, polls conducted before the elections are usually of this type. As a result of each poll of this type, we get a *percentage* of people who answered "yes" ("60% are planning to vote for candidate X"). This percentage represents exactly what we want: a number from the interval $[0, 1]$ (e.g., 60% means 0.6).

So, we arrive at the following *polling* method of eliciting the degree of confidence from the experts: we ask several experts whether $P(x)$ is true or false. Some of these experts may not answer at all, or give "unknown" as an answer; these experts we need not count. If out of N experts who gave a definite answer, M answered "yes" (i.e., "yes, $P(x)$ is true"), then we take the ratio M/N as the desired degree of confidence.

Problems with the second method. For the polling method to give meaningful results, we need many respondents. From the strict mathematical viewpoint, for a pool to give meaningful results (e.g., to give the percentage with a guaranteed 10% accuracy), we must interview at least 1,000 people.

- For an election poll, interviewing 1,000 people is quite possible: millions participate in the elections, and we want to know the opinion of the representative group of people.
- However, in an expert system, we are trying to formalize the knowledge of the top experts. We may not have that many top experts, and even if we do, top experts are usually very busy people, we may not be able to convince all of them to cooperate with this project. Last but not the least, the time of top experts costs money, and we may not have the money to hire 1,000 of them.

Can we somehow ask a single expert and still get a number?

Third elicitation method: using subjective probabilities and bets. Some people feel uncomfortable estimating a probability or marking a value on a scale, but they have a good feel for *bets*. Such people cannot express the probability of their favorite horse winning, but they are absolutely sure that, say, they are willing to bet 4 to 1 but not 5 to 1 that this horse will win.

If we can extract a betting ratio from an expert, then we can easily transform this number into the subjective probability, that will serve as the desired degree of confidence.

For example, if an expert is willing to bet 4 to 1 but not any better that $P(x)$ is true (e.g., that most people will agree that $x = 1$ is a small temperature difference), this means that this experts' subjective probability is $4/(4+1) = 0.8$.

From finitely many degrees of confidence to a membership function: extrapolation is needed. We started this section with a problem of describing a fuzzy property $P(x)$ (of the type "x is small"). Namely, for every possible value x, we would like to know the *degree of confidence* $d(P(x))$ that this value x satisfies the property P. This degree of confidence is usually denoted by $\mu_P(x)$, and the function that transforms a real number x into a value $\mu_P(x) \in [0,1]$ is called a *membership function*, or a *fuzzy set*.

For each value x, we can, using one of the above elicitation techniques, find the value $\mu_P(x)$ for this x. However, this is not sufficient:

- On one hand, we want to know the value $\mu_P(x)$ for all possible real numbers x, i.e., for *infinitely many* different values.
- On the other hand, we can only ask an expert *finitely many* questions and therefore, we can only determine the values of the membership function for *finitely many* different values $x^{(1)}, \ldots, x^{(v)}$.

Thus, after all the elicitation is over, we only know the values $\mu_P(x^{(p)})$ of the desired function $\mu_P(x)$ for v different values $x^{(1)}, \ldots, x^{(v)}$. To reconstruct the desired function, we must use *extrapolation*.

Simple examples of extrapolation. Let us describe a few simple examples of extrapolation. The simplest possible extrapolation is an extrapolation by *piece-wise linear functions.* The simplest case of an extrapolation is when we start with the values x for which the expert is either absolutely sure that x is true, or he is absolutely sure that x is false, i.e., with values for which $\mu_P(x) = 0$ or $\mu_P(x) = 1$. Let us give several examples of such situations.

– Let us first describe the property of the type "x is negligible".

  - The only case when we are 100% sure that x is negligible is when $x = 0$. So, we have the value $\mu_P(0) = 1$.
  - Usually, we also know the value $\Delta > 0$ after which the difference in temperatures is no longer negligible. For example, for a thermostat that controls the room's temperature, we can take $\Delta = 10$. This means that $\mu_P(x) = 0$ for $x \geq \Delta$ and for $x < -\Delta$.

  We know the value of the function $\mu_P(x)$ for $x \leq -\Delta$, for $x = 0$, and for $x \geq \Delta$. We need to use linear extrapolation to find the values of this function for $x \in (-\Delta, 0)$ and for $x \in (0, \Delta)$. In general, the formula for a linear function $f(x)$ that passes through the point $y_1 = f(x_1)$ and $y_2 = f(x_2)$, is

$$f(x) = y_1 + (x - x_1) \cdot \frac{y_1 - y_2}{x_2 - x_1}.$$

  For our case, this formula leads to the following membership function:

  - To get the values $\mu_P(x)$ for $x \in (-\Delta, x)$, we take $x_1 = -\Delta$, $x_2 = 0$, $y_1 = 0$, and $y_2 = 1$. As a result, we get the expression $\mu_P(x) = (x + \Delta)/\Delta$.
  - To get the values $\mu_P(x)$ for $x \in (0, \Delta)$, we take $x_1 = 0$, $x_2 = \Delta$, $y_1 = 1$, and $y_2 = 0$. As a result, we get the expression $\mu_P(x) = 1 - x/\Delta$.

  Thus, the function $\mu_P(x)$ takes the following form:

  - $\mu_P(x) = 0$ for $x \leq -\Delta$;
  - $\mu_P(x) = (x + \Delta)/\Delta$ for $-\Delta \leq x \leq 0$;
  - $\mu_P(x) = 1 - x/\Delta$ for $0 \leq x \leq \Delta$; and
  - $\mu_P(x) = 0$ for $x > \Delta$.

  The graph of this function has the shape of a *triangle* over the x-axis. Therefore, such functions are called *triangular* membership functions. A similar shape describes properties like "close to a", for some fixed a. In this case, the triangular membership function has the form:

  - $\mu_P(x) = 0$ for $x \leq a - \Delta$;

  - $\mu_P(x) = (x - (a - \Delta))/\Delta$ for $a - \Delta \le x \le a$;
  - $\mu_P(x) = 1 - (x - a)/\Delta$ for $a \le x \le a + \Delta$; and
  - $\mu_P(x) = 0$ for $x > a + \Delta$.

- For the same property "x is negligible", in some cases, we know the lower bound δ below which the temperature difference x is indeed negligible. In this case, in addition to knowing that $\mu_P(x) = 0$ for $x \le -\Delta$ and for $x \ge \Delta$, we also know that $\mu_P(x) = 1$ for $-\delta \le x \le \delta$. For this function, linear extrapolation to the intervals $(-\Delta, -\delta)$ and (δ, Δ) results in the following membership function:
  - $\mu_P(x) = 0$ for $x \le -\Delta$;
  - $\mu_P(x) = (x + \Delta)/(\Delta - \delta)$ for $-\Delta \le x \le -\delta$;
  - $\mu_P(x) = 1$ for $-\delta \le x \le \delta$;
  - $\mu_P(x) = 1 - (x - \delta)/(\Delta - \delta)$ for $\delta \le x \le \Delta$; and
  - $\mu_P(x) = 0$ for $x > \Delta$.

  The graph of this function has the shape of a *trapezoid* over the x-axis. Therefore, such functions are called *trapezoidal* membership functions.
- Another frequent example of piece-wise functions are functions that describe properties like "x is large". For these properties, we usually know that values below a certain δ are definitely not large, and that values about a certain $\Delta \gg \delta$ are definitely large. So, we know that $\mu_P(x) = 0$ for $x \le \delta$ and $\mu_P(x) = 1$ for $x \ge \Delta$. To get the values of $\mu_P(x)$ for $x \in (\delta, \Delta)$, we use a linear extrapolation. As a result, we get the following function:
  - $\mu_P(x) = 0$ for $x \le \delta$;
  - $\mu_P(x) = (x - \delta)/(\Delta - \delta)$ for $\delta \le x \le \Delta$;
  - $\mu_P(x) = 1$ for $x \ge \Delta$.

We have described the triangular and trapezoid functions because they are *the simplest*. Interestingly, in fuzzy control, they are also, at present, *the most frequently used membership* functions. This fact has two possible explanations:

- First, fuzzy control is still at its infancy. The first successful application of fuzzy control was produced a little more than 20 years ago (in 1974, by E. Mamdani). Because of that, the potential of simple fuzzy control applications is not yet exhausted. *Maybe, we will need more complicated membership functions* later on, when all simple applications will be used, but so far, we do not seem to need them.
- Second, we are formalizing *approximate* expert knowledge. If all an expert can say about a control is that it should be *small*, or *medium*,

or *large*, it may not be reasonable to try to formalize these notions in a too complicated way. Thus, simple models of expert uncertainty, in particular, simple membership functions, may be more reasonable than complicated ones. From this viewpoint, *we may not need more complicated membership functions at all.*

Probably, the truth lies somewhere in between; but anyway, so far, piecewise-linear functions seem to work just fine.

2.3. FUZZY LOGICAL OPERATIONS

How can we represent logical connectives? An ideal solution. Suppose that an expert system contains statements A and B, and we have elicited the degrees of belief $d(A)$ and $d(B)$ from the experts. Suppose now that a user wants to know the degree of belief in a composite statement $A\&B$.

In principle, knowing only the two numbers $d(A)$ and $d(B)$ is not sufficient to describe the expert's degree of belief in $A\&B$: e.g., if $d(A) = d(B)$,

- it could be that the experts perceive A and B as *equivalent*, in which case $d(A\&B) = d(A)$, or
- it could also be that the experts perceive A and B as *independent* statements, in which case the possibility of A *and* B being true is smaller than the possibility that one of them is true: $d(A\&B) < d(A)$.

So, the *ideal* situation would be to elicit, from the experts, not only the degree of belief in the *basic* statements from the knowledge base, but also the degrees of belief in all possible *logical combinations* of these statements.

This ideal solution is not practically possible. The above described ideal solution is practically impossible: If we have N statements $S_1, ..., S_N$ in the knowledge base, then for each of $2^N - 1$ non-empty subsets $\{S_{i_1}, ..., S_{i_k}\}$ of the knowledge base, we need to elicit the degree of belief in the corresponding AND-statement $S_{i_1}\&...\&S_{i_k}$. For a realistic expert system, N is in hundreds, so asking an expert 2^N questions is impossible.

A practical way to represent logical connectives: logical operations. In view of this practical impossibility, although in *some* cases, we will be able to have the degree of belief $d(A\&B)$ stored in the knowledge base, in *general*, we often have to deal with a following situation:

- we know the degrees of belief $d(A)$ and $d(B)$ in statements A and B;
- we know nothing else about A and B; and
- we are interested in the (estimated) degree of belief of the composite statement $A\&B$.

Since the only information available consists of the values $d(A)$ and $d(B)$, we must compute $d(A\&B)$ based on these values. We must be able to do that for arbitrary values $d(A)$ and $d(B)$. Therefore, we need a *function* that transforms the values $d(A)$ and $d(B)$ into an estimate for $d(A\&B)$. Such a function is called an *AND-operation.* If an AND-operation $f_\& : [0,1] \times [0,1] \to [0,1]$ is fixed, then we take $f_\&(d(A), d(B))$ as an estimate for $d(A\&B)$.

Similarly:

- to estimate the degree of belief in $A \vee B$, we need an *OR-operation* $f_\vee : [0,1] \times [0,1] \to [0,1]$.
- to estimate the degree of belief in the negation $\neg A$, we need a *NOT-operation* $f_\vee : [0,1] \times [0,1]$.

Terminological comment. AND operations are also called *t-norms*, and OR operations are also called *t-conorms.*

Natural properties of logical operations. The logical operations with fuzzy values must satisfy some *natural conditions.*

- For an expert, $A\&B$ and $B\&A$ mean the same. Therefore, the estimates $f_\&(d(A), d(B))$ and $f_\&(d(B), d(A))$ for these two statements should coincide. To achieve that, we must require that $f_\&(a,b) = f_\&(b,a)$ for all a and b; in other words, the operation $f_\&$ must be *commutative.*
- Similarly, from the fact that $A\&(B\&C)$ and $(A\&B)\&C$ mean the same, we can deduce the requirement that $f_\&$ must be *associative*: $f_\&(a, f_\&(b,c)) = f_\&(f_\&(a,b), c)$ for all a, b, and c.
- If A is absolutely false ($d(A) = 0$), then $A\&B$ is also absolutely false, i.e., $f_\&(a, 0) = 0$ for all a.
- If A is absolutely true ($d(A) = 1$), then $A\&B$ is true iff B is true, so, the degree of belief in $A\&B$ must coincide with the degree of belief in B: $f_\&(1, b) = b$ for all b.
- If our belief in A or B increases, then the degree of belief in $A\&B$ must also increase, so $f_\&$ must be *monotonic.*
- If the degree of belief in A changes a little bit, then the degree of belief in $A\&B$ must also change slightly. In other words, $f_\&$ must be *continuous.*

How can we determine AND and OR operations? Since our goal is to describe the expert knowledge, we must elicit these operations from the experts. This can be done in the following manner:

- We form several pairs of statements (A_k, B_k), $k = 1, 2, \ldots$
- For each pair from this set, we elicit, from the experts, the degrees of confidence $d(A_k)$, $d(B_k)$, and $d(A_k\&B_k)$.

– Then, we use an extrapolation procedure to find a function $f_\&(a, b)$ for which $f_\&(d(A_k), d(B_k)) \approx d(A_k \& B_k)$.

Similar procedures enable us to determine OR and NOT operations.

Historical comment. This empirical approach was first implemented and used by the designers of the first successful expert system MYCIN. After spending several years, they came up with AND and OR operations that fit the reasoning of medical experts. Interestingly, it turned out that different medical experts use very similar AND and OR operations. This lead MYCIN designers to a natural hypothesis that these operations are indeed general for human reasoning. Alas, when they tried to apply these same AND and OR operations to another area (geology), the resulting expert system did not lead to good results. It turned out that in different fields, people use different AND and OR operations. This difference is easy to explain:

– In some applications (e.g., in *medicine*), mistakes can be deadly, so more *cautious* estimates are needed (e.g., $f_\&(a, b) = a \cdot b$).
– In other applications (e.g., in *geology*), we cannot measure as many parameters as in medicine, so, we have to rely more on expertise, and hence, experts must take risks. In these applications, more brave, more optimistic estimates are needed: e.g., a geologist starts to drill in the uncertainty in which a surgeon is not likely to start an incision. Therefore, for such applications, we need more *optimistic* estimates for $d(A \& B)$ (e.g., $f_\&(a, b) = \min(a, b)$).

Simple AND, OR, and NOT operations: an idea. There are many possible AND and OR operations, many of them very complicated. However, in most applications, we do not need this complexity:

– Our goal is to apply these operations to the degrees of confidence, that, in their turn, come from the values of membership functions.
– We have already mentioned that in most applications, it is sufficient to use the *simplest* membership functions, that are obtained by applying the *simplest* extrapolation methods to crisp values of the corresponding properties.

Since we start with the simplified expressions for degrees of confidence, it makes sense to consider *simple* AND and OR operations for handling these degrees of confidence, i.e., to consider AND and OR operations that are obtained by applying some *simple* extrapolation techniques (linear or quadratic) to the *crisp* values of these operations.

Simple AND, OR, and NOT operations: results. To describe negation ("not"), we can have a *linear* operation $f_\neg(a)$. The conditions that $f_\neg(0) = 1$ and $f_\neg(1) = 0$ uniquely determine this operation as $f_\neg(a) = 1 - a$.

The resulting formula $d(\neg A) = f_\neg(d(A)) = 1 - d(A)$ for the *degree of belief* in $\neg A$ resembles a formula for the *probability* of $\neg A$: If we know the probability $P(A)$ of an event A, then the probability $P(\neg A)$ that this event will not occur is equal to $P(\neg A) = 1 - P(A)$. Thus, the linear NOT operation is consistent with the two probability-like elicitation procedures for degree of belief: namely, with the procedures that are based on polling and betting.

For AND and OR, linear operations are impossible, but we can have *piece-wise linear* or *quadratic* ones. One can show that there are two possible piece-wise linear AND operations: $f_\&(a,b) = \min(a,b)$ and $f_\&(a,b) = \max(0, a+b-1)$. Similarly, we get two possible OR operations: $f_\vee(a,b) = \max(a,b)$ and $f_\vee(a,b) = \min(a+b,1)$. The only *quadratic* AND and OR operations are $f_\&(a,b) = a \cdot b$ and $f_\vee(a,b) = a + b - a \cdot b$ (they are called *algebraic product* and *algebraic sum*).

The operations $\min(a,b)$, $\max(a,b)$, $a \cdot b$, and $a + b - a \cdot b$ were first proposed in the pioneer 1965 paper of L. Zadeh. The operations $\max(0, a + b - 1)$ and $\min(a+b,1)$ were introduced in (Giles, 1976) under the name of *bold* operations.

Similarly to the linear NOT operation, these six operations are also consistent with the probability-like interpretation: namely, if we know the probabilities $a = P(A)$ and $b = P(B)$ of two events A and B, then:

- The probability $P(A\&B)$ that both events A and B will occur can take any real value from the interval $[\max(a+b-1,0), \min(a,b)]$. In particular, if we know that A and B are independent events, then $P(A\&B) = a \cdot b$.
- The probability $P(A \vee B)$ that at least one of the events A or B will occur can take any real value from the interval $[\max(a,b), \min(a+b,1)]$. In particular, if we know that A and B are independent events, then $P(A \vee B) = a + b - a \cdot b$.

More complicated AND and OR operations. The above-described six AND and OR operations are the ones that are most frequently used in expert systems and in fuzzy control. However, these operations are not the only possible ones. Indeed, we can *re-scale* the scale of degrees of belief, i.e., we can represent the degree of confidence $a = d(A)$ in the statement A not by the original value a, but by a new value $a' = d'(A) = r(a) = r(d(A))$, where $r(a)$ is a monotonic function.

How will an AND operation that has the form $c = f_\&(a,b)$ in the old scale look in the new scale $a' = r(a)$? In other words, if we know the degrees of confidence $a' = d'(A) = r(d(A))$ and $b' = d'(B) = r(d(B))$ in the new scale, what will be the degree of belief $c' = d'(A\&B) = r(A\&B))$ in this new scale? To get the expression for c' in terms of a' and b', we must do the following:

- First, we convert the values a' and b' into the old scale by applying the inverse re-scaling r^{-1}: $a = r^{-1}(a')$ and $b = r^{-1}(b')$.
- Then, we apply the AND operation $f_{\&}(a,b)$ to the values a and b expressed in the old scale. As a result, we get the value $c = f_{\&}(a,b) = f_{\&}(r^{-1}(a'), r^{-1}(b'))$.
- Finally, we convert the value c into the new scale by applying the re-scaling $r(a)$: $c' = r(c)$.

As a result of this three-step procedure, we get a new AND operation

$$f'_{\&}(a',b') = r(f_{\&}(r^{-1}(a'), r^{-1}(b'))).$$

Similarly, we get a new OR operation

$$f'_{\vee}(a',b') = r(f_{\vee}(r^{-1}(a'), r^{-1}(b')))$$

and a new NOT operation

$$f'_{\neg}(a') = r(f_{\neg}(r^{-1}(a'))).$$

The new operations are called *isomorphic* to the old ones, because they represent the same operations but on a different scale.

There are special names for operations that are isomorphic to piecewise linear and quadratic AND and OR operations. (These names may sound somewhat strange for a computer science reader, because they were invented before the computer applications and they describe algebraic properties of the corresponding operations.)

- Operations that are isomorphic to quadratic AND and OR operations, i.e., operations of the type
$$f'_{\&}(a',b') = r(r^{-1}(a') \cdot r^{-1}(b'))$$
and
$$f'_{\vee}(a',b') = r(r^{-1}(a') + r^{-1}(b') - r^{-1}(a') \cdot r^{-1}(b')),$$
for some strictly increasing continuous function $r(a)$, are called *strictly Archimedean* AND and OR operations.
- Operations that are isomorphic to bold AND and OR, i.e., operations of the type
$$f'_{\&}(a',b') = r(\max(r^{-1}(a') + r^{-1}(b') - 1, 0))$$
and
$$f'_{\vee}(a',b') = r(\min(r^{-1}(a') + r^{-1}(b'), 1)),$$
for some strictly increasing continuous function $r(a)$, are called *non-strictly Archimedean* AND and OR operations.

– If we use $f_\&(a,b) = \min(a,b)$ and $f_\vee(a,b) = \max(a,b)$, then, as one can see, for every strictly monotonic function $r(a)$, the isomorphic operations $f'_\&(a',b')$ and $f'_\vee(a',b')$ have exactly the same form: $f'_\&(a',b') = \min(a',b')$ and $f'_\vee(a',b') = \max(a',b')$. These two operations are called *idempotent.*

Most of the AND and OR operations that are actually used belong to one of these three types.

In addition of these three classes of operations, we may consider even more complicated AND and OR operations that are, e.g., isomorphic to an idempotent operation on one subinterval of the interval $[0,1]$ and to a strictly Archimedean one on another subinterval of this interval $[0,1]$. It turns out that this combination covers *all* possible AND and OR operations: namely, for an arbitrary AND and OR operation that satisfies several reasonable properties (e.g., the ones described above), we can subdivide the interval $[0,1]$ into sub-intervals on each of which the operation is isomorphic to an operation from one of the three classes. This general classification result was proven in (Ling, 1965); see (Klir *et al.*, 1995; Nguyen *et al.*, 1997a) for details.

2.4. AGGREGATION INSIDE A RULE

Problems with implication. We started the description of the expert knowledge by mentioning that the expert knowledge is usually represented by "if-then" rules. Since we want to formalize these rules, it seems reasonable to formalize the statements of the type "if A then B", i.e., using a logical term for such statements, to formalize *implication* in the same way as we formalized "and", "or", and "not" operations. There are, however, two problems with this idea:

– First, a *minor* problem: unlike "and", "or", and "not", implication is usually *not directly implemented* in the computers. Our goal is to describe the expert knowledge for a computer. Most computer languages have built-in logical operations "and", "or", and "not", but usually, not implication. Therefore, even if all the statements are crisp, when we formalize these statements for the computer, we will still need to first *reformulate* implication in terms of other logical operations.
– Second, a *major* problem: implication is somewhat *counter-intuitive.* Namely, researchers working in mathematical logic are using a formalization of implication in which implication is defined by a truth table in which $A \to B$ is true in all cases except when A is true and B is false. So, if A is false, then $A \to B$ is true for an arbitrary statement B. For example, we can conclude that "if $2+2=5$, then witches have six wings". Such statement may be mathematically correct, but they

are absolutely *counter-intuitive*, because our intuitive understanding of "if-then" assumes some relation, while in the witches example, there is no relation whatsoever between the assumption (that $2+2=5$) and the conclusion (that witches have six wings).

Due to these problems, most expert systems and intelligent control systems do not *directly* formalize implication but instead, try first to *reformulate* if-then rules in terms of "and", "or", and "not". How can we do that?

Reformulating if-then rules in terms of "and", "or", and "not": example. Let's first consider our toy thermostat example, with three rules (2)–(4). If we know the difference x between the actual and the desired temperature, what control u should we apply? We have three rules that describe when a control is reasonable. Therefore, u is a reasonable control if one of the the three rules is applicable, i.e., when either:

- the first rule is applicable (i.e., x is negligible) and u is negligible; or
- the second rule is applicable (i.e., x is small positive), and u is small negative; or
- the third rule is applicable (i.e., x is small negative), and u is small positive.

Summarizing, we can say that u is an appropriate choice for a control if and only if either (x is negligible *and* u is negligible), *or* (x is small positive *and* u is small negative), etc.

Let us describe this statement in more succinct terms. Let us use the following notations:

- $R_k(x,u)$ will indicate that k-th rule is applicable for a given x, and that this rule recommends to use the control value u;
- $C(x,u)$ will indicate that u is a reasonable control for a given input x.

Then, in the above example, we get

$$C(x,u) \equiv R_1(x,u) \ \vee\ R_2(x,u) \ \vee\ R_2(x,u),$$

where

$$R_1(x,u) \equiv N(x) \,\&\, N(u); \quad R_2(x,u) \equiv SP(x) \,\&\, SN(u);$$

$$R_3(x,u) \equiv SN(x) \,\&\, SP(u).$$

We already know how to formalize the properties (as membership functions, or fuzzy sets), and we know how to formalize the "and" and "or" operations. Thus, for every input x, we can define the degree of belief $d_r(x,u)$ that r-th rule will be fired:

$$d_1(x,u) = f_\&(\mu_N(x), \mu_N(u)); \quad d_2(x,u) = f_\&(\mu_{SP}(x), \mu_{SN}(u));$$

$$d_3(x,u) = f_\&(\mu_{SN}(x), \mu_{SP}(u)).$$

Reformulating if-then rules in terms of "and", "or", and "not": general case. For a general rule base with R rules of the type (5), we get

$$C(x_1,\ldots,x_n,u) \equiv R_1(x_1,\ldots,x_n,u) \vee \ldots \vee R_R(x_1,\ldots,x_n,u), \tag{6}$$

where

$$R_r(x_1,\ldots,x_n,u) \equiv A_{r1}(x_1) \,\&\, \ldots \,\&\, A_{rn}(x_n) \,\&\, B_r(u). \tag{7}$$

Therefore, for each rule, the "firing degree" is equal to:

$$d_r(x_1,\ldots,x_n,u) = f_\&(\mu_{r1}(x_1),\ldots,\mu_{rn}(x_n),\mu_r(u)), \tag{8}$$

where:

- $\mu_{ri}(x_i)$ is the membership function corresponding to the property A_{ri};
- $\mu_r(u)$ is the membership function corresponding to the property $B_r(u)$;
- $f_\&(a,b,c)$ stands for $f_\&(f_\&(a,b),c)$, and, in general, $f_\&(a_1,\ldots,a_n,a_{n+1}) = f_\&(f_\&(a_1,\ldots,a_n),a_{n+1})$.

2.5. COMBINATION OF DIFFERENT RULES

General idea. In the previous section, we have shown how to transform the formula (7) into an algorithm that describes the degree of belief $d_r(x_1,\ldots,x_n,u)$ that each rule is applicable. After we have computed the firing degree of each rule, we can similarly formalize the formula (6) and get a numerical value that describes to what extent each possible control value u is reasonable for a given input $x_1,\ldots,x_n$:

$$\mu_C(x_1,\ldots,x_n,u) = f_\vee(d_1(x_1,\ldots,x_n,u),\ldots,d_R(x_1,\ldots,x_n,u)), \tag{9}$$

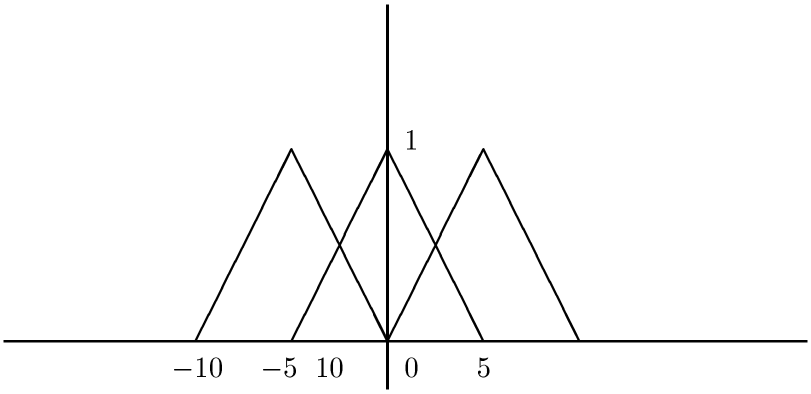
where $f_\vee(a,b,c)$ stands for $f_\vee(f_\vee(a,b),c)$, and, in general, $f_\vee(a_1,\ldots,a_n,a_{n+1}) = f_\vee(f_\vee(a_1,\ldots,a_n),a_{n+1})$.

In particular, in our toy example,

$$\mu_C(x,u) = f_\vee(d_1(x,u),d_2(x,u),d_3(x,u)) =$$

$$f_\vee(f_\&(\mu_N(x),\mu_N(u)), f_\&(\mu_{SP}(x),\mu_{SN}(u)), f_\&(\mu_{SN}(x),\mu_{SP}(u))).$$

Numerical example. Let us assume that all three membership functions are piece-wise linear, and that they are described by the following graph:



What is the degree of confidence $\mu_C(4, -2)$ that for $x = 4^\circ$ the control $u = -2^\circ$ is reasonable? According to our formulas, let us first compute the values of the membership functions. From the above general formula for linear extrapolation, we can find the analytical formulas for these membership functions:

– The term "negligible" is described by the following formulas:
  - $\mu_N(x) = 1 + x/5$ for $-5 \le x \le 0$;
  - $\mu_N(x) = 1 - x/5$ for $0 \le x \le 5$;
  - $\mu_N(x) = 0$ for all other x.
– The term "small positive" is described by the following formulas:
  - $\mu_{SP}(x) = x/5$ for $0 \le x \le 5$;
  - $\mu_{SP}(x) = 2 - x/5$ for $5 \le x \le 10$;
  - $\mu_{SP}(x) = 0$ for all other x.
– The term "small negative" is described by the following formulas:
  - $\mu_{SN}(x) = 2 + x/5$ for $-10 \le x \le -5$;
  - $\mu_{SN}(x) = -x/5$ for $-5 \le x \le 0$;
  - $\mu_{SN}(x) = 0$ for all other x.

If we use $f_\&(a, b) = \min(a, b)$ and $f_\vee(a, b) = \max(a, b)$, then we get $\mu_C(4, -2) = \max(d_1, d_2, d_3)$, where

$$d_1 = \min(\mu_N(4), \mu_N(-2)); \quad d_2 = \min(\mu_{SP}(4), \mu_{SN}(-2));$$

$$d_3 = \min(\mu_{SN}(4), \mu_{SP}(-2)).$$

Here, $\mu_N(4) = 0.2$, $\mu_N(-2) = 0.6$, $\mu_S P(4) = 0.8$, $\mu_{SN}(-2) = 0.4$, and $\mu_{SN}(4) = \mu_{SP}(-2) = 0$. Hence,

$$d_1 = \min(0.2, 0.6) = 0.2; \quad d_2 = \min(0.8, 0.4) = 0.4; \quad d_3 = \min(0, 0) = 0,$$

and

$$\mu_C(4,-2) = \max(0.2, 0.4, 0) = 0.4.$$

2.6. DEFUZZIFICATION

The problem. As a result of applying the previous steps, we get a *fuzzy set* $\mu(u)$ that describes, for each possible control value u, how reasonable it is to use this particular value. In other words, for every possible control value u, we get a degree of confidence $\mu(u)$ that describes to what extent this value u is reasonable to use. In automatic control applications, we want to transform this *fuzzy* information into a *single* value $\bar{u}$ of the control that will actually be applied.

This transformation from a *fuzzy* set to a (*non-fuzzy*) number is called a *defuzzification*. What defuzzification should we apply?

Main idea. We want to select a value $\bar{u}$ that, on average, would lead to the smallest error. If we choose $\bar{u}$, and the best control is u, then the control error is $\bar{u}-u$. Thus, to determine $\bar{u}$, we can use the least squares method; as weights for each square $(\bar{u}-u)^2$, we can take the degree of confidence $\mu(u)$ that u is a reasonable control value. As a result, we get the following formula for determining $\bar{u}$: $\int \mu(u)\cdot(\bar{u}-u)^2\,du \to \min$. Differentiating the minimized function with respect to the unknown $\bar{u}$ and equating the derivative to 0, we get the formula

$$\bar{u} = \frac{\int u \cdot \mu(u)\,du}{\int \mu(u)\,du} \tag{10}$$

that is called *centroid defuzzification*.

Warning: centroid defuzzification does not always work. In most real-life situations, centroid defuzzification leads to a meaningful control, but not always. Let us give a simple example. Suppose that we are designing an automatic controller for a car. If the car is traveling on an empty wide road, and there is an obstacle straight ahead (e.g., a box that fell from a truck), then a reasonable idea is to *swerve* to avoid this obstacle. Since the road is empty, there are two possibilities:

- we can swerve to the right; and
- we can swerve to the left.

For swerving, the control variable u is the angle to which we steer the wheel. Based on the distance to the obstacle and on the speed of the car, an experienced driver can describe a reasonable amount of steering u_0.

> In reality, u_0 will probably be a fuzzy value, but for simplicity, we can assume that u_0 is precisely known.

Thus, as a result of formalizing expert knowledge, we conclude that there are two possible control values: u_0 and $-u_0$, both with degree of confidence 1. For this $\mu(u)$, formula (10) leads to $\bar{u} = 0$, i.e., the car will run directly into the box.

For such cases, more complicated defuzzification methods must be used; see, e.g., (Yen *et al.*, 1991; Yen *et al.*, 1991a; Kreinovich *et al.*, 1992; Yen *et al.*, 1992).

Simplifications. The formula (10) is sometimes too computationally complicated. To simplify this formula, the membership functions $\mu_r(u)$ (that describe the words $B_r(u)$) can be replaced by their defuzzified values u_r. In this case, instead of the general control rule (5), we get a simplified rule

$$A_{r1}(x_1)\,\&\,\ldots\,\&\,A_{rn}(x_n) \to u = u_r. \tag{5a}$$

Here, the function $\mu_C(u)$ is different from 0 only for one one of these values u_r, and the integrals in (10) are reduced to easier-to-compute sums:

$$\bar{u} = \frac{d_1 \cdot u_1 + \ldots + d_R \cdot u_R}{d_1 + \ldots + d_R}, \tag{10a}$$

where

$$d_r = f_\&(\mu_{r1}(x_1), \ldots, \mu_{rn}(x_n)). \tag{11}$$

This version of fuzzy control is, computationally, much simpler than the control based on centroid defuzzification, but still leads to a reasonable control.

Takagi-Sugeno version of fuzzy control. In some situations, for each combination of fuzzy inputs, an expert is able not only to describe a single reasonable control value u, but also to describe how exactly, within the corresponding fuzzy area, the control must depend on the inputs $x_1, \ldots, x_n$. In such situations, we get rules of the type

$$A_{r1}(x_1)\,\&\,\ldots\,\&\,A_{rn}(x_n) \to u = f_r(x_1, \ldots, x_n), \tag{5b}$$

where $f_r(x_1, \ldots, x_n)$ is a function (usually, linear) supplied by the expert. The rule base of the type (5b) was first considered in (Takagi *et al.*, 1985) under the name of a *fuzzy model*. For this fuzzy model, the least squares method similar to the one that we used lead to the following defuzzification formula:

$$\bar{u} = \frac{d_1 \cdot f_1(x_1, \ldots, x_n) + \ldots + d_R \cdot f_R(x_1, \ldots, x_n)}{d_1 + \ldots + d_R}, \tag{10b}$$

where d_r is determined by the formula (11). This control also has many important applications; see, e.g., (Palm *et al.*, 1997).

2.7. THE BASIC STEPS OF FUZZY CONTROL: SUMMARY

- We start with the if-then expert rules of the type "If x is small, and ..., then u is small". In general, each of these rules can be represented by a formula (5), where $x_1, \ldots, x_n$ are inputs, and A_{ri} and B_r are words that describe properties of inputs and output.
- For each words w used in these rules, we pick several values $x^{(1)}, \ldots, x^{(k)}$, and use one of the above-described elicitation techniques described to determine the degrees of confidence $\mu_w(x^{(1)}), \ldots, \mu_w(x^{(n)})$ that these values satisfy the property w.
- Then, we use some extrapolation technique to determine the membership functions $\mu_w(x)$ (that describe the degrees of confidence that different values of x satisfy the property w).
- We choose AND and OR operations $f_\&(a, b)$ and $f_\vee(a, b)$.
- For each rule r, and for each possible values of input and output, we compute the *firing values* $d_r(x_1, \ldots, x_n, u)$ using the formula (8) and then we compute the membership function for control by using the formula (9).
- For every input $x_1, \ldots, x_n$, we get a function $\mu_C(u)$ that describes the degree of confidence that this very u is a reasonable control. To get a single recommended control value $\bar{u}$, we use the formula (10) (or one of its modifications).

2.8. TUNING

Why tuning? It is quite possible that the control resulting from the above-described five-step procedure is sometimes inadequate. There are two possible reasons for this:

- First, when an expert formulates the rules, he usually remembers *specific* rules but sometimes forgets to explicitly mention *common sense* rules that are absolutely evident to any human, but that need to be explicitly spelled out for the computer.
- Second, all knowledge elicitation methods are *approximate*, and if we add distortions caused by this approximate character on each step of fuzzy control methodology, we may end up with a rather distorted representation of expert's control.

In view of this possibility, before implementing this control, we must first *test* it. If it turns out that in some situations, the resulting control is inadequate, we have two options:

- If the inadequacy is *huge*, this probably means that we are missing or misinterpreting some of the rules. In this case, we need to confront the experts with these results. Since the rules that the experts have formulated lead to not adequate results, the experts will be able to either *modify* these rules, or *add* new rules that cover these situations.

- If the inadequacy is *small*, then probably, the experts' rules were adequate, and the inadequacy is caused by the approximate character of the expert system methodology. In this case, to make a control better, we can *tune* the parameters of the resulting control.

Comment. Tuning is especially important if we use fuzzy control methodology in novel applications when there aren't yet any expert operators, and the rules that we use are simply coming from common sense (see, e.g., (Kosko, 1997)). It turns out (Lea *et al.*, 1995) that in this case, the optimal control strategy (optimal in some reasonable sense) can be indeed obtained by an appropriate tuning of common sense rules.

Simple iterative tuning. If we *do not know the exact model* of the controlled object, then the only way to predict the quality of different control strategies is to actually test these strategies on the *actual* object. So, if the original fuzzy control does not work well, we try to modify it a little bit and check whether it becomes better or not. This testing takes lots of time.

Tuning is easier in the situation when we *know the exact model* of the controlled object and the exact objective function (and when the only reason for using fuzzy control is that we do not know how to solve the corresponding optimization problem). In this case, we can *simulate* the controlled object, and test different modifications of the original fuzzy control on this computer simulation. This computer testing is fast and easy, and we can, therefore, test many different modifications and find the best one.

Genetic algorithms for tuning. An even better way of tuning fuzzy control is to use intelligent optimization techniques instead of the simple search for a maximum. One of these techniques is *genetic algorithms* that simulate the survival of the fittest in nature to generate better and better controls.

Neural networks for tuning. In addition to *rules*, we can have *records* of the expert's control. The fact that the original fuzzy control is not completely adequate may mean, in particular, that for the recorded inputs $x_1^{(k)}, \ldots, x_n^{(k)}$, the result of applying the fuzzy control methodology may differ from the recorded control $u^{(k)}$ applied by this expert. It is, therefore, desirable to tune the original fuzzy control so that it will fit with the recorded patterns $(x_1^{(k)}, \ldots, x_m^{(k)}, u^{(k)})$.

Neural networks (computer programs that simulate the way our brain operates) are known to be a universal tool for fitting patterns. Therefore, it makes sense to use neural networks for tuning fuzzy control. Details can be found in a special chapter of this book.

Model-based fuzzy control. When we do not have a *crisp* model for the controlled object, i.e., if we do not exactly how different controls will change the state of the system, we may still be able to extract from the experts some information about these changes. Namely, we may be able to extract the information of the type "If x_1 was small, ..., and we apply a small control u, then x_1 will change slightly". This additional information can be formulated in terms of fuzzy logic, and is called a *fuzzy model* of the controlled object (Takagi-Sugeno model mentioned above can be viewed as a particular case of this information).

There are papers in which such models are used to tune the fuzzy control (Pham *et al.*, 1992; Chen *et al.*, 1994; Bouchon-Meunier *et al.*, 1996).

3. METHODOLOGIES OF FUZZY CONTROL: WHICH IS THE BEST?

Which non-linearity should we choose? In the previous section, we have seen that on each stage of fuzzy control methodology, we have several different choices: we can choose different extrapolation techniques, we can choose different AND and OR operations, we can choose different defuzzifications, etc. Different choices often lead to control strategies of different quality; see, e.g., (Kreinovich *et al.*, 1992). It is, therefore, important to make choices that lead to the *best* control.

- In some situations, we *know* the exact expression for the *objective function*; in such situation, "the best" simply means the control that leads to the largest possible value of this objective function.
- However, in most applications of fuzzy control, we do not have such a precise expression. In such situations, we must choose the fuzzy control methodology based on some reasonable control criterion.

In this section, we will describe reasonable control characteristics and describe what choices of fuzzy control methodology optimize these characteristics.

Stability. The main objective of the control is that it should control. For example, if we control a car on the road, then, for the largest part of the trip, one of the main objectives of this control is to make sure that it stays in its lane with the desired speed, i.e., that whenever it will accidentally deviate from the straight course, the steering control will return it back on course, and when the speed would deviate, the acceleration or deceleration would bring it back to the optimal cruise speed.

In the general case, we want a control that, after an initial deviation, will bring the controlled system back "on track". This property is called *stability* of the control (and of the controlled system).

Stability is a generic term. There are many different particular notions of stability, depending on how big initial deviations we allow (usually, only small ones), whether we want the system to be stabilized for a potentially infinite amount of time or only for a given finite interval, etc.

From the practical viewpoint, we can describe the stability of a control strategy by its *relaxation time*: we start with a deviation of a given size $\Delta > 0$, and we measure the time after which the control brings the deviation down to the pre-defined value $\delta \ll \Delta$. This relaxation time depends on the values Δ and δ.

- If the initial deviation Δ is *large*, then we may not be able to return the system to its desired state at all.
- Thus, we will consider only *small* deviations Δ.

In this sense, we can say that a control strategy $u(x_1, \ldots, x_n)$ is *more stable* than the control strategy $u'(x_1, \ldots, x_n)$ if for all sufficiently small Δ and δ, the relaxation time T corresponding to the strategy u is smaller than the relaxation time corresponding to the strategy u'.

It turns out that for reasonable control rules, the *most stable* control corresponds to $f_{\&}(a, b) = \min(a, b)$ and $f_{\vee}(a, b) = a + b - a \cdot b$ (Kreinovich *et al.*, 1991; Kreinovich *et al.*, 1992; Smith *et al.*, 1995; Kreinovich *et al.*, 1996a).

Smoothness. Stability is not all that we expect from a control.

For example, when driving a car, stability means, in particular, that once the car swerved, it should return to the original trajectory. The faster it returns, the more stable is the system. Therefore, from the viewpoint of stability only, the ideal (optimal) control would be the one that brings the car back on track in the shortest possible time (i.e., with the largest possible acceleration). The resulting driving with sudden accelerations may be good on a racetrack or for a car chase, but it is very uncomfortable for passengers. From the passenger viewpoint, we prefer the resulting trajectory to be *smooth*.

Just like stability, smoothness is a generic notion; there are several different understandings (and formalizations) of what "smooth" means. In mathematical terms, *smoothness* usually means that the time derivative $\dot{x}_i(t)$ is small. To compare different control strategies, we must combine the values $\dot{x}_i(t)$ for different moments of time t into a single numerical criterion that characterizes how smooth is the trajectory (i.e., how small are all these derivatives). To form this numerical criterion, it is reasonable to use the idea of the least squares method and to take $I = \sum_i \int (\dot{x}_i(t))^2 \, dt$.

Similarly to stability, it makes sense to compare the values of this smoothness functional for *small* initial deviations Δ. It turns out that from the viewpoint of this comparison, the best choice of AND and OR operations is $f_{\&}(a,b) = a \cdot b$ and $f_{\vee}(a,b) = \min(a,b)$ (Kreinovich *et al.*, 1991; Kreinovich *et al.*, 1992; Smith *et al.*, 1995).

Comment. At first glance, it may seem that we can require that the control is *both* stable and smooth. However, from the fact that the requirements of smoothness and stability lead to different control strategies, we can conclude that different control requirements are not exactly consistent:

- the most stable control is often not smooth at all;
- the smoothest control is not necessarily very stable.

Therefore, in real life, we must select the control characteristic that is the most adequate for a given control situation, and then choose the fuzzy control methodology that leads to the optimal value of this characteristic.

Computational simplicity. Stability and smoothness are typical examples of the *idealized* goals. When, in mathematical control theory, we look for the optimal control strategy, we look for the optimal mathematical function, without taking into consideration how exactly we are going to implement this function.

In *real life*, however, the computational ability of the processor that actually computes the desired control is limited, so some very good control strategies may be too complicated for this processor. Moreover, in many control situations, we need the control *fast* (e.g., for a car control, if we spend too much time on the computation of the optimal control, the car may, by then, have already wrecked). This necessity is, as we have mentioned, one of the reasons why *fuzzy* control is sometimes used even when the *optimal* control strategy is known.

It can be shown that if we are looking for the control that is the *fastest to compute*, then the best choice is to use $f_{\&}(a,b) = \min(a,b)$ and $f_{\vee}(a,b) = \max(a,b)$ (Kreinovich *et al.*, 1994).

Robustness. In the traditional fuzzy control methodology, we assume, among other assumptions, that:

- we know the *exact* values of the inputs x_i; and
- we know the *exact* values of the membership functions.

In reality, both assumptions are idealizations:

- Measurements are never 100% accurate, so, the measurement results $\tilde{x}_i$ of measuring the input variables x_i (that characterize the current state of the controlled system) are, generally speaking, *different* from the *actual* (unknown) values of these variables.
- Similarly, all methods of eliciting the degrees of confidence from the experts are only *approximate*, so the (approximate) values $\tilde{\mu}_{ri}(x_i)$ and $\tilde{\mu}(u)$ that are used in the fuzzy control algorithm may be somewhat

different from the *ideal* (unknown) values $\mu_{ri}(x_i)$ and $\mu_r(u)$ that characterize the actual expert's degrees of confidence.

We would like to choose the fuzzy control methodology that would make the resulting control the least sensitive to this uncertainty.

There are two ways to describe the possible errors $\Delta x_i = \tilde{x}_i - x_i$ and $\Delta\mu_{ri}(x_i) = \tilde{\mu}_{ri}(x_i) - \mu_{ri}(x_i)$:

- Usually, we know the *upper bound* Δ on these errors, i.e., we know that the error must belong to the interval $[-\Delta, \Delta]$.
- In some cases, we also know the *probabilities* of different values from this interval. These probabilities are usually described by a normal (Gaussian) distribution with 0 average and a given standard deviation σ.

In the *first* case, the interval uncertainty of the inputs leads to the interval uncertainty in the resulting control. So, the minimal sensitivity corresponds to the smallest width of the resulting interval. Similarly to the cases of smoothness and sensitivity, we will consider these widths for sufficiently small deviations Δ. The following control methodology leads to the smallest sensitivity (i.e., to to the narrowest intervals):

- we must choose piecewise-linear membership functions (Nguyen *et al.*, 1995; Nguyen *et al.*, 1995a);
- we must choose $f_\&(a,b) = \min(a,b)$ and $f_\vee(a,b) = \max(a,b)$ (Nguyen *et al.*, 1992; Nguyen *et al.*, 1993; Nguyen *et al.*, 1995; Nguyen *et al.*, 1995a; Nguyen *et al.*, 1997a); and
- we must choose the centroid defuzzification (Kreinovich *et al.*, 1991; Kreinovich *et al.*, 1992; Kreinovich *et al.*, 1996a; Kreinovich, 1997).

In the *second* case, random deviations of the input lead to random deviations of the resulting control. In this case, the minimal sensitivity corresponds to the smallest possible standard deviation of the control value. This smallest value is attained if we use $f_\&(a,b) = a \cdot b$ and $f_\vee(a,b) = a + b - a \cdot b$ (Nguyen *et al.*, 1994; Nguyen *et al.*, 1995; Nguyen *et al.*, 1995a; Nguyen *et al.*, 1997a).

Entropy. Instead of minimizing the *average* error, we can try to minimize the corresponding *entropy* (Ramer *et al.*, 1992; Ramer *et al.*, 1994; Ramer *et al.*, 1994a; Kreinovich, 1996; Kreinovich *et al.*, 1996a; Kreinovich, 1997):

- if we use the *average* entropy (in some reasonable sense), then the best alternative is to use algebraic product for AND and algebraic sum for OR;
- for an appropriately defined *worst-case* entropy the optimal operations are $f_\&(a,b) = \min(a,b)$ and $f_\vee(a,b) = a + b - a \cdot b$.

Comment. These optimization results are in good accordance with the general group-theoretic approach that enables us to classify techniques that are optimal relative to arbitrary reasonable criteria (Kreinovich *et al.*, 1991; Kreinovich *et al.*, 1992; Smith *et al.*, 1995; Bouchon-Meunier *et al.*, 1996).

Acknowledgments. This paper was partly supported by the NSF grant EEC-9322370. We are thankful to numerous friends and colleagues for valuable discussions.

References

J. Abello, V. Kreinovich, H. T. Nguyen, S. Sudarsky, and J. Yen (1994), "Computing an appropriate control strategy based only on a given plant's rule-based model is NP-hard", In: L. Hall, H. Ying, R. Langari, and J. Yen, Eds., *NAFIPS/IFIS/NASA'94, Proceedings of the First International Joint Conference of The North American Fuzzy Information Processing Society Biannual Conference, The Industrial Fuzzy Control and Intelligent Systems Conference, and The NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic, San Antonio, December 18–21, 1994*, IEEE, Piscataway, NJ, 331–332.

B. Bouchon-Meunier, V. Kreinovich, A. Lokshin, and H. T. Nguyen (1996), "On the formulation of optimization under elastic constraints (with control in mind)", *Fuzzy Sets and Systems,* Vol. 81, No. 1, pp. 5–29.

B. G. Buchanan and E. H. Shortliffe, *Rule-based expert systems. The MYCIN experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, MA, Menlo Park, CA.

G. Chen, T. T. Pham, and J. J. Weiss (1994), "Fuzzy modeling of control systems", *IEEE Transactions on Aerospace and Electronic Systems.*

R. Giles (1976), "Lukasiewicz logic and fuzzy set theory", *Internat. J. Man-Machine Stud.*, Vol. 8, pp. 313–327.

G. Klir and B. Yuan (1995), *Fuzzy sets and fuzzy logic: theory and applications*, Prentice Hall, Upper Saddle River, NJ.

B. Kosko (1997), *Fuzzy engineering*, Prentice Hall, Upper Saddle River, NJ.

V. Kreinovich (1996), "Maximum entropy and interval computations", *Reliable Computing*, Vol. 2, No. 1, pp. 63–79.

V. Kreinovich (1997), "Random sets unify, explain, and aid known uncertainty methods in expert systems", in J. Goutsias, R. Mahler, and H. T. Nguyen (eds.), *Applications and Theory of Random Sets*, Springer-Verlag (to appear).

V. Kreinovich, H. T. Nguyen, and E. A. Walker (1996a), "Maximum entropy (MaxEnt) method in expert systems and intelligent control: new possibilities and limitations", In: K. Hanson and R. Silver, Eds., *Maximum Entropy and Bayesian Methods, Santa Fe, New Mexico, 1995*, Kluwer Academic Publishers, Dordrecht, Boston, pp. 93–100.

V. Kreinovich, C. Quintana, and R. Lea (1991), "What procedure to choose while designing a fuzzy control? Towards mathematical foundations of fuzzy control", *Working Notes of the 1st International Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems*, College Station, TX, pp. 123–130.

V. Kreinovich, C. Quintana, R. Lea, O. Fuentes, A. Lokshin, S. Kumar, I. Boricheva, and L. Reznik (1992), "What non-linearity to choose? Mathematical foundations of fuzzy control", *Proceedings of the 1992 International Conference on Fuzzy Systems and Intelligent Control*, Louisville, KY, pp. 349–412.

V. Kreinovich and D. Tolbert (1994), "Minimizing computational complexity as a criterion for choosing fuzzy rules and neural activation functions in intelligent control". In: M. Jamshidi, C. Nguyen, R. Lumia, and J. Yuh (Editors), *Intelligent Automation and Soft Computing. Trends in Research, Development, and Applications. Proceed-*

ings of the First World Automation Congress (WAC'94), August 14–17, 1994, Maui, Hawaii, TSI Press, Albuquerque, NM, Vol. 1, pp. 545–550.

R. N. Lea and V. Kreinovich (1995), "Intelligent Control Makes Sense Even Without Expert Knowledge: an Explanation", *Reliable Computing*, 1995, Supplement (Extended Abstracts of APIC'95: International Workshop on Applications of Interval Computations, El Paso, TX, Febr. 23–25, 1995), pp. 140–145.

C. H. Ling (1965), "Representation of associative functions", *Publ. Math. Debrecen*, Vol. 12, pp. 189–212.

E. H. Mamdani (1974), "Application of fuzzy algorithms for control of simple dynamic plant", *Proceedings of the IEE*, Vol. 121, No. 12, pp. 1585–1588.

E. H. Mamdani (1977), "Application of fuzzy logic to approximate reasoning using linguistic systems", *IEEE Transactions on Computing*, Vol. 26, pp. 1182–1191.

R. R. Mohler (1991), *Nonlinear systems. Vol. 1. Dynamics and control*, Prentice Hall, Englewood Cliff, NJ.

H. T. Nguyen and V. Kreinovich (1995), "Towards theoretical foundations of soft computing applications", *International Journal on Uncertainty, Fuzziness, and Knowledge-Based Systems*, Vol. 3, pp. 341–373.

H. T. Nguyen, V. Kreinovich, B. Lea, D. Tolbert (1992), "How to control if even experts are not sure: robust fuzzy control", *Proceedings of the Second International Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems*, College Station, December 2–4, pp. 153–162.

H. T. Nguyen, V. Kreinovich, B. Lea, and D. Tolbert (1995a), "Interpolation that leads to the narrowest intervals, and its application to expert systems and intelligent control", *Reliable Computing*, 1995, Vol. 3, No. 1, pp. 299–316.

H. T. Nguyen, V. Kreinovich, and D. Tolbert (1993), "On robustness of fuzzy logics". *Proceedings of IEEE-FUZZ International Conference*, San Francisco, CA, March 1993, Vol. 1, pp. 543–547.

H. T. Nguyen, V. Kreinovich, and D. Tolbert (1994), "A measure of average sensitivity for fuzzy logics", *International Journal on Uncertainty, Fuzziness, and Knowledge-Based Systems*, Vol. 2, No. 4, pp. 361–375.

H. T. Nguyen, V. Kreinovich, and Qiang Zuo (1997), "Interval-valued degrees of belief: applications of interval computations to expert systems and intelligent control", *International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems (IJUFKS)*, Vol. 5 (to appear).

H. T. Nguyen and E. A. Walker (1997a), *A first course in fuzzy logic*, CRC Press, Boca Raton, FL.

R. Palm, D. Driankov, and H. Hellendoorn (1997), *Model based fuzzy control*, Springer-Verlag, Berlin, Heidelberg.

T. T. Pham, J. J. Weiss, and G. R. Chen (1992), "Optimal fuzzy logic control for docking a boat", *Proc. of the Second International Workshop on Industrial Fuzzy Control and Intelligent Systems, Dec. 2–4, 1992*, College Station, TX, pp. 66–73.

A. Ramer and V. Kreinovich (1992), "Maximum entropy approach to fuzzy control", *Proceedings of the Second International Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems*, College Station, December 2–4, pp. 113–117.

A. Ramer and V. Kreinovich (1994), "Information complexity and fuzzy control", Chapter 4 in: Abraham Kandel and Gideon Langholtz (Eds.), *Fuzzy Control Systems*, CRC Press, Boca Raton, FL, pp. 75–97.

A. Ramer and V. Kreinovich (1994a), "Maximum entropy approach to fuzzy control", *Information Sciences*, Vol. 81, No. 3–4, pp. 235–260.

E. H. Shortliffe (1976), *Computer-based medical consultation: MYCIN*, Elsevier, New York.

M. H. Smith and V. Kreinovich (1995), "Optimal strategy of switching reasoning methods in fuzzy control", Chapter 6 in H. T. Nguyen, M. Sugeno, R. Tong, and R. Yager (eds.), *Theoretical aspects of fuzzy control*, J. Wiley, N.Y., pp. 117–146.

S. Smith and V. Kreinovich (1995a), "In Case of Interval Uncertainty, Optimal Control is NP-Hard Even for Linear Plants, so Expert Knowledge is Needed", *Reliable Computing*, Supplement (Extended Abstracts of APIC'95: International Workshop on Applications of Interval Computations, El Paso, TX, Febr. 23–25, 1995), 190–193.

T. Takagi and M. Sugeno (1985), "Fuzzy identification of systems and its applications to modelling and control", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 15, No. 1, pp. 116–132.

J. Yen and N. Pfluger (1991), "Path planning and execution using fuzzy logic", In: *AIAA Guidance, Navigation and Control Conference*, New Orleans, LA, 1991, Vol. 3, pp. 1691–1698.

J. Yen and N. Pfluger (1991a), "Designing an adaptive path execution system", *IEEE International Conference on Systems, Man and Cybernetics, Charlottesville, VA, 1991.*

J. Yen, N. Pfluger, and R. Langari (1992), "A defuzzification strategy for a fuzzy logic controller employing prohibitive information in command formulation", *Proceedings of IEEE International Conference on Fuzzy Systems, San Diego, CA, March 1992.*

L. A. Zadeh (1995), "Fuzzy sets", *Inform. and Control*, Vol. 8, pp. 338–353.