Fuzzy Systems

Rule-based Computing

Dr. Petr Musilek

Department of Electrical and Computer Engineering
University of Alberta

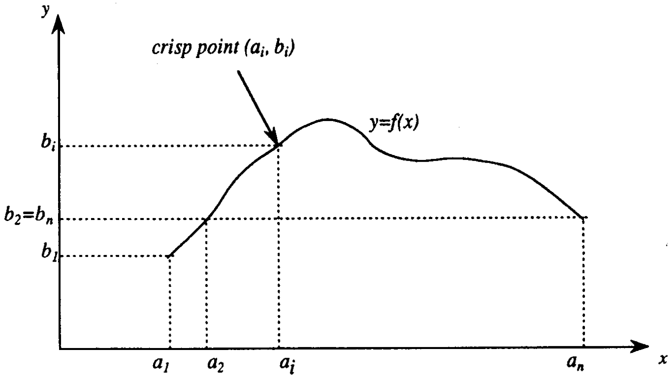
Fall 2019

Rule-based Systems

Rules provide a formal way of representing knowledge in form of directives and strategies. Rule-based systems (RBS) are appropriate when domain knowledge is available as empirical results or experience. RBS also form basis for fuzzy rule-based systems (FRBS) and fuzzy control .

Crisp function

To understand how an IF-THEN statement (and a set of IF-THEN-ELSE statements) can be represented as a relation, consider a crisp function.



Analytical representation

To say "$y = f(x)$" is an analytical form representing the function. Another way of representing this function would be to list all (or a sufficient number of) pairs that describe the function:

$$\begin{gathered}(a_1, b_1) \\ (a_2, b_2) \\ \dots \\ (a_i, b_i) \\ \dots \\ (a_n, b_n)\end{gathered}$$

Linguistic representation

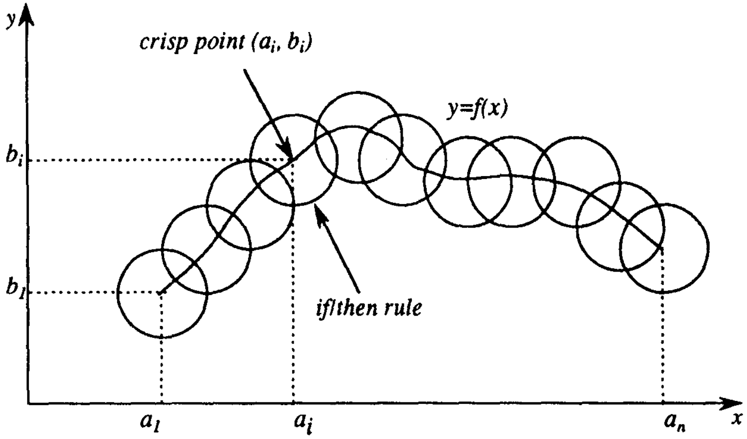
This form can be linguistically represented as

$$
\begin{array}{ll}
\text{IF } x \text{ is } a_1 & \text{THEN } y \text{ is } b_1 \\
\text{IF } x \text{ is } a_2 & \text{THEN } y \text{ is } b_2 \\
\ldots & \\
\text{IF } x \text{ is } a_i & \text{THEN } y \text{ is } b_i \\
\ldots & \\
\text{IF } x \text{ is } a_n & \text{THEN } y \text{ is } b_n
\end{array}
$$

which becomes more “crisply accurate” as n increases. But this is not always necessary (or even desired).

Fuzzy extension

The above concept can be extended to fuzzy input/output (using extension principle)



Fuzzy linguistic representation

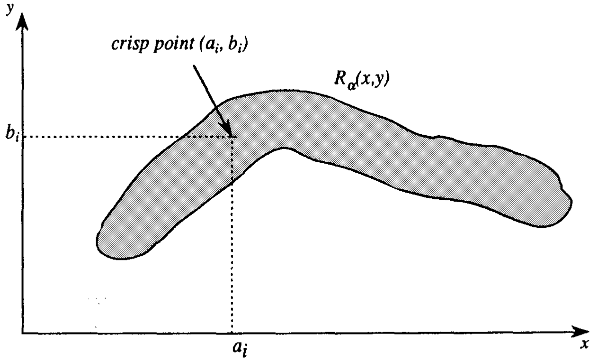
This new representation can be approximated using fuzzy rules as follows

$$
\begin{array}{ll}
\text{IF } x \text{ is } A_1 & \text{THEN } y \text{ is } B_1 \\
\text{IF } x \text{ is } A_2 & \text{THEN } y \text{ is } B_2 \\
\ldots & \\
\text{IF } x \text{ is } A_i & \text{THEN } y \text{ is } B_i \\
\ldots & \\
\text{IF } x \text{ is } A_n & \text{THEN } y \text{ is } B_n
\end{array}
$$

Where A_i is a fuzzy quantity defined on X, and B_i is a fuzzy quantity defined on Y. The analytical form of these rules is a fuzzy relation $R_i(x, y) \rightarrow$ (implication relation, covered shortly), and each rule used to approximate the function has its own implication relation.

Fuzzy algorithm

Merging the rules joined together by ELSE depends upon the manner by which the implication relations are created. Regardless the analytical form, this relation is called *fuzzy algorithm* (rather just a single rule).



Rules

Rule-based systems are based on IF-THEN statements in form

IF	*condition*	THEN	*action*, or alternatively
IF	*premise*	THEN	*conclusion*
IF	*antecedent*	THEN	*consequent*

Fuzzy Rules

Consider a rule in the form

IF x is A AND y is B THEN z is C

with x, y and z being fuzzy quantities; e.g.

IF	*motor temperature* is *high*
AND	*motor speed* is *high*
THEN	*motor current* is *low*

Inference, Modus ponens

Rules are combined with facts (or collections of facts) to make inference. This leads to the sole rule of inference in propositional calculus, called *modus ponens* (from Latin: *mode that affirms*). Modus ponens can be formally expressed as.

$$\begin{array}{ccc} p & & \\ p & \rightarrow & q \\ \hline & & q \end{array}$$

Informally, this expression states that given an implication (rule, $p \rightarrow q$) and the presence of its premise p, the conclusion q can be taken as true.

Generalized modus ponens

In conventional logic, with modus ponens, the proposition *x* is *p* has to be observed to consider the proposition *y* is *b* . In other words, there has to be an exact match between the premise *p* and the antecedent of the rule $p \rightarrow q$.
In fuzzy logic, a proposition *x* is *A*', close to the premise *x* is *A* can be observed to provide a conclusion *y* is *B*', close to the conclusion *y* is *B* *(A* and *B* are fuzzy stes). This leads to *generalized modus ponens*.

Generalized modus ponens

$$\begin{array}{lcl} A^{'} & & \\ A & \rightarrow & B \\ \hline & & B^{'} \end{array}$$

First, $A^{'}$ is matched with A. To find $B^{'}$, the implication relation $R(x, y)$ is composed with $A^{'}$ (this process is sometimes called *forward chaining* or *data-driven inference*)

$$B^{'} = R(x, y)A^{'}$$

Generalized modus tolens

GMT is essentially the reverse of GMP

$$\begin{array}{ccc} & & B^{'} \\ A & \rightarrow & B \\ \hline A^{'} & & \end{array}$$

First, $B^{'}$ is matched to B. To find $A^{'}$, the implication relation $R(y, x)$ is composed with $B^{'}$

$$A^{'} = R(y, x)B^{'}$$

[Note that x and y are now swapped; this is analogous to finding and using $f^{-1}(y)$, the inverse of $f(x)$.]

Linguistic variables and values

Linguistic variables are described by fuzzy sets:

- Primary values of a fuzzy variable typically include terms like "small", "large", "high", etc.
- These can be further modified with *linguistic hedges* as "very", "more or less", "not", e.g.

$$\begin{aligned} \text{not}A(x) &= 1 - A(x) \\ \text{very}A(x) &= A^2(x) \\ \text{more or less}A(x) &= \sqrt{A(x)} \end{aligned}$$

Individual variables can also be connected with AND and OR, using appropriate triangular norms (**t** norms for AND, **s** norms for OR).

Accumulation and usage of knowledge

In fuzzy RBS, accumulation and usage of knowledge are facilitated by constructing relations and performing composition, respectively.
Accumulation of knowledge: It is evident that rules represent functional dependencies and relations between premises and conclusions. As such, they can be represented using Cartesian product. For example two fuzzy sets

$$A : \mathbf{X} \rightarrow [0, 1] \text{ and } B : \mathbf{Y} \rightarrow [0, 1]$$

can be combined using Cartesian product to form a relation $R : \mathbf{X} \times \mathbf{Y} \rightarrow [0, 1]$. This can be modelled by a **t**-norm operation, such as min.

Multiple rules/relations

Because the relation between the sets is generally not uniform, multiple rules/relations are build for multiple segments, k, of the universe of discourse

$$R_k = A_k \times B_k$$

To describe the overall relation, the individual rules R_k are aggregated

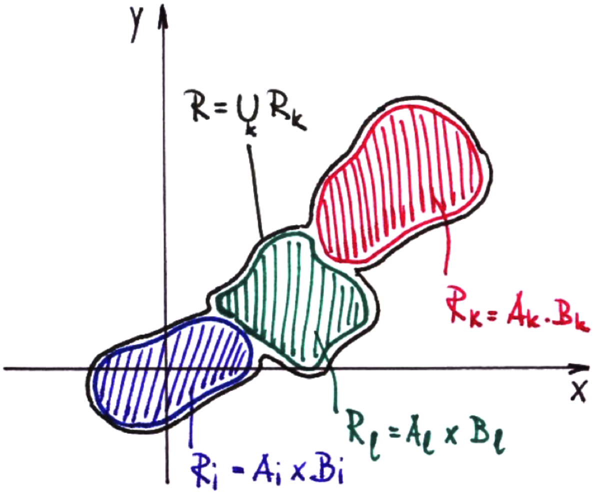
$$R = \cup_{k=1}^{N} R_k$$

This process corresponds to accumulation of knowledge and can be described (assuming min and max operation to model fuzzy intersection and union, respectively) as follows

$$R = \max_{k=1,2,\ldots,N} R_k(x, y) = \max_{k=1,2,\ldots,N} \min[A_k(x), B_k(y)]$$

Fuzzy patches

Graphically, this situation is depicted the following picture.



Accumulation and usage of knowledge

Usage of knowledge: Knowledge accumulated the way indicated above and codified using fuzzy rules can be used to make conclusions. Having described relation R between fuzzy sets A in $\mathbf{X}$ and B in $\mathbf{Y}$, the goal is to infer value of B given a particular value of A. Because the inference in fuzzy systems is driven by generalized modus ponens, exact match between A and one of the antecedentsA_k is not necessary. Using sup–$\mathbf{t}$ composition, fuzzy set B can determined as

$$B^{'} = A^{'} \circ R$$

Usage of knowledge (cont.)

Therefore, the resulting membership function is described as

$$B(y) = \sup_x [A(x) \mathbf{t} R(x, y)]$$

and because the relation $R(x, y)$ is a union of individual relations R_k, we can expand the above expression to (assuming min intersection and max union)

$$B(y) = \sup_x \min[A(x), R(x, y)] =$$

$$= \sup_x \left\{ \min \left[A(x), \max_k (\min (A_k(x), B_k(y))) \right] \right\}$$

Rearranging this expression, we obtain

$$B(y) = \max_k \left\{ \min \left[\sup_x (\min (A(x), A_k(x))) , B_k(y) \right] \right\}$$

Usage of knowledge (cont.)

Term $\sup_x (\min (A(x), A_k(x)))$ is expressing the level of overlap between the antecedent A_k of rule R_k, and the actual value of fuzzy set A, or in other words the possibility of A with respect to A_k

$$\text{Poss}(A, A_k) = \sup_x (\min (A(x), A_k(x)))$$

By substituting $\lambda_k = \text{Poss}(A, A_k)$ we finally obtain the following expression to calculate fuzzy set B for given A

$$B(y) = \max_k \{\min [\lambda_k, B_k(y)]\}$$

Based on max-min composition, this expression is called *compositional rule of inference* (CRI, cf. the brief note on CRI in the previous section).

Implication operators

The results presented in the previous section represent only one specific case of fuzzy implication relation, based on max and min operations. There are, however, numerous other possibilities how implication can be modelled.
There are two general ways to define fuzzy implications:

(i) $A \wedge B$ which corresponds to matching operation; it provides a stronger (local) implication meaning A is coupled with B; examples are Larsen and Mamdani implications (see bellow);

(ii) $\overline{A} \vee B$ which corresponds (NOT A) OR B implication found in binary logic; it provides a weaker (global) implication, meaning "A entails B"; examples include Lukasziewicz and Boolean implications;

(iii) $(A \wedge B) \vee \overline{A}$ which is equivalent to (ii) and has similar properties; an example is Zadeh implication

Implication operators

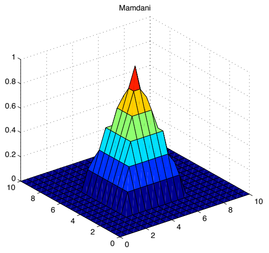
Rule IF x is A THEN y is B can be described in general analytical form of relation

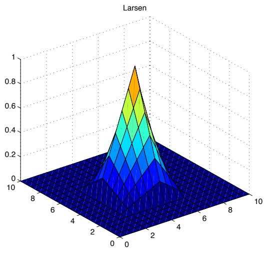
$$R_{A \to B}(x, y) = \Phi[A(x), B(y)],$$

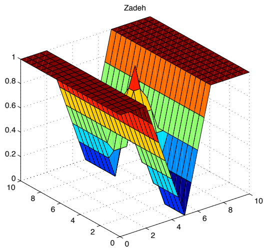
where Φ is the implication operator. A list of commonly used implication operators follows

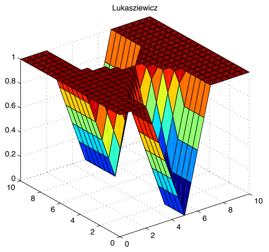
Name	Implication operator $\Phi[A(x), B(y)]$
Mamdani (min)	$\min[A(x), B(y)]$
Larsen (product)	$A(x) \cdot B(y)$
Zadeh (max-min)	$\max\{\min[A(x), B(y)], 1 - A(x)\}$
Lukasziewicz (arithmetic)	$\min[1 - A(x) + B(y), 1]$
Boolean (Dienes-Rescher)	$\max[1 - A(x), B(y)]$

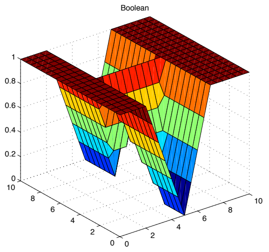
Fuzzy Implication Operators











Multivariate Implication

These elementary fuzzy implications can be extended to multivariate cases, in order to allow decision making or control with more than one independent variable. In such case, we consider a rule in the form

IF x_1 is A_1 AND x_2 is A_2 AND … AND x_m is A_m THEN y is B

The connective AND can be modeled either as min or product operations leading to implication operators in one of the following forms

$$\Phi\{\min[A_1(x_1), A_2(x_2), \ldots, A_m(x_m)], B(y)\}, \text{ or}$$

$$\Phi\{A_1(x_1)A_2(x_2)\ldots A_m(x_m), B(y)\}$$

.

Fuzzy Algorithm

A fuzzy algorithm is a procedure of collecting fuzzy IF-THEN rules. The rules are defined over the same universes (entering the Cartesian product of corresponding fuzzy sets), and are connected by connectives ELSE:

$$\begin{array}{ll}
\text{IF } x \text{ is } A_1 & \text{THEN } y \text{ is } B_1 \text{ ELSE} \\
\text{IF } x \text{ is } A_2 & \text{THEN } y \text{ is } B_2 \text{ ELSE} \\
\ldots & \\
\text{IF } x \text{ is } A_i & \text{THEN } y \text{ is } B_i \text{ ELSE} \\
\ldots & \\
\text{IF } x \text{ is } A_n & \text{THEN } y \text{ is } B_n
\end{array}$$

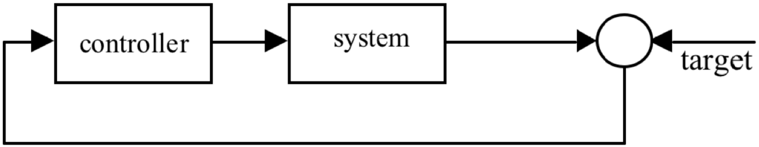
Each rule is represented by a relation whose form depends on particular implication operator, Φ, used. The form of ELSE connective, in turn, depends on the form of the implication, Φ.

Implication operator and ELSE connectives

Name	$\Phi[A(x), B(y)]$	ELSE connectives
Mamdani	$\min[A(x), B(y)]$	OR (**s** -norm)
Larsen	$A(x) \cdot B(y)$	OR (**s** -norm)
Zadeh	$\max\{\min[A(x), B(y)], 1 - A(x)\}$	AND (**t** -norm)
Lukasziewicz	$\min[1 - A(x) + B(y), 1]$	AND (**t** -norm)
Boolean	$\max[1 - A(x), B(y)]$	AND (**t** -norm)

Control Problem

Most typical control scheme is that of *direct control*, where the controller is in the forward path of a feedback control system.



Due to the diversity of systems and control objectives, design of controller is usually nontrivial and includes

(i) identification of the system, i.e. building of mathematical model of system behaviour and its parameters, and
(ii) analytical description of control objectives.

Reality

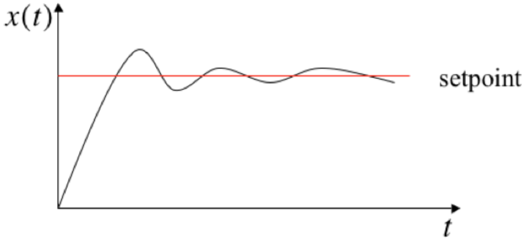
In real situations, mathematical model of the system can be too difficult or too expensive to obtain. This is especially evident in systems that are normally controlled by human, for example: parking a car, driving a car, operating a crane, maintaining comfortable temperature, etc. These situations are characteristic by

- unavailability of mathematical model of the system, and
- no explicitly formulated performance index, but
- availability of experiential knowledge

This last fact (availability of experiential knowledge) is very important because it allows linguistic description of the control and controlled variables (in form of fuzzy sets) and linguistic description of the control strategies (in form of fuzzy rules).

Control problem – example

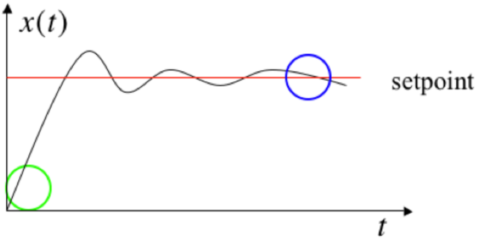
Consider the following simple example of control problem



The objective is to stabilize system very quickly. These requirements are conflicting: we can either approach the setpoint in a short time but with many large oscillations, or avoid oscillations in which case the approach will be very slow.

Control problem – example (cont.)

This problem can be eliminated by considering different strategies in different stages of control, as in the following picture:



Control problem – example (cont.)

By considering variables error = setpoint – x(*t*), chage_of_error = error(*t*)- *error*(*t*-1), and control,

and defining fuzzy sets *positive large* (*PL*), *positive small* (*PS*), *zero* (*Z*), *negative small* (*NS*), *negative large* (*NL*) on these variables,

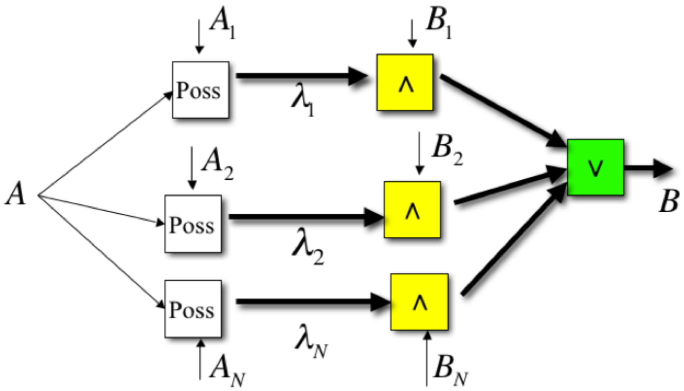
we can formulate the following set of rules:

Control problem – example (cont.)

For phase 1 (green circle)
IF *error* is *PL* and *change_of _error* is *NL* THEN *control is PL*
For phase 2 (blue circle)
IF *error* is *Z* and *change_of _error* is *Z* THEN *control is Z*
By constructing more rules/relations for additional phases/segments of the control process, a smooth control of the system can be achieved.

Fuzzy Controller

A general structure of fuzzy controller corresponds to the compositional rule of inference developed earlier.



Fuzzy Controller (cont.)

The first column of blocks (Poss) performs matching of fuzzy set A with antecedents A_k of individual rules R_k. The second column ($\wedge$) corresponds to min operation of the inner term,

$$\min\left[\lambda_k, B_k(y)\right].$$

The last block ($\vee$) finally aggregates the results of individual rules

$$B(y) = \max_k \min[\lambda_k, B_k(y)]$$

Fuzzy/crisp transformations

Operation of the fuzzy controller can be summarized as follows: given set of rules IF x is A_k THEN y *is* B_k, and fuzzy set A describing the state of the system, fuzzy controller applies CIR to determine control variable in form of fuzzy set B, or formally

$$\begin{array}{llll} & x \text{ is } A & & \\ \text{IF} & x \text{ is } A_k & \text{THEN} & y \text{ is } B_k \\ \hline & & & y \text{ is } B \end{array}$$

Real systems do not operate with fuzzy sets $\rightarrow$ need to transform real (crisp) values to fuzzy sets to be processed by the fuzzy controller, and transforming the resulting fuzzy set into a real value that can be used to actually control the system.

These transformations (encoding and decoding)

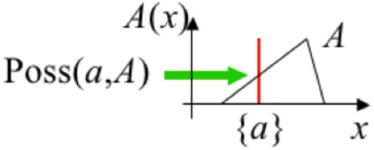
are called fuzzification and dufuzzification, respectively.

Fuzzification

Fuzzification

Encoding of crisp input value is already embedded in the existing structure of the fuzzy controller.

Recall that it is possible to apply possibility measure on a fuzzy singleton $\{a\}$ corresponding to the crisp value of input variable:



In such case, the result of antecedent matching is simply

$$\lambda_k = Poss(a, A_k) = A_k(a)$$

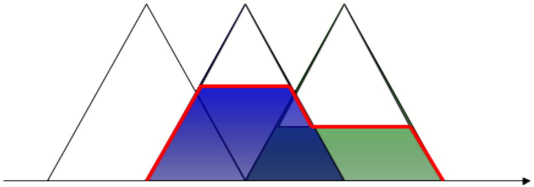
Defuzzification

Defuzzification

Due to the nature of fuzzy rules, its possible that more than one rule applies to a given input value a. As a result, the output fuzzy set can be of complicated shape composed of several consequent fuzzy sets B_k.

Defuzzification (cont.)

This situation is illustrated in the following figure for three consequent fuzzy sets B_1, B_2, and B_3, forming the output set B (indicated by red outline).



The goal of defuzzification is to provide a single value that best represents the output fuzzy set. It is important to consider not only shape/location of the fuzzy sets forming B, but also their height carrying information about relative significance of its corresponding fuzzy rule (i.e. how much is given rule activated – how well is its antecedent matching current input).

Defuzzification (cont.)

One possibility is to determine the center of the resulting area *B*, and project the location of this point to y axis, i.e.

$$b = \mathrm{COG}\,(B(y)) = \frac{\int B(y)ydy}{\int B(y)dy}$$

This approach is called Center of Gravity (COG), center of area, or *centroind*.

Defuzzification (cont.)

Other defuzzification methods include:

- SCOG (Simplified Center of Gravity) $b = \text{SCOG}\,(B(y)) = \frac{\sum B(y)y}{\sum B(y)}$
- BOA (Bisector of Area)
  $b = \text{BOA}\,(B(y)) : \int_{\alpha}^{\text{BOA}} B(y)dy = \int_{\text{BOA}}^{\beta} B(y)dy$, where α and β are, respectively, the left and right boundaries of the output fuzzy set
- MOM (Mean of Maximum) $b = \text{MOM}\,(B(y)) = \frac{\alpha^* + \beta^*}{2}$ where α^* and β^* are, respectively, the left and right boundaries of the maximum level of the output fuzzy set.
- SOM (Smallest of Maximum) $b = \text{SOM}\,(B(y)) = \alpha^*$
- LOM (Largest of Maximum) $b = \text{LOM}\,(B(y)) = \beta^*$

Defuzzification in python/scikit-fuzzy

In python/scikit-fuzzy, there is a function `defuzz`, which implements all basic defuzzification methods:

`b = defuzz(x,B,type)`, where type can be one of the following:

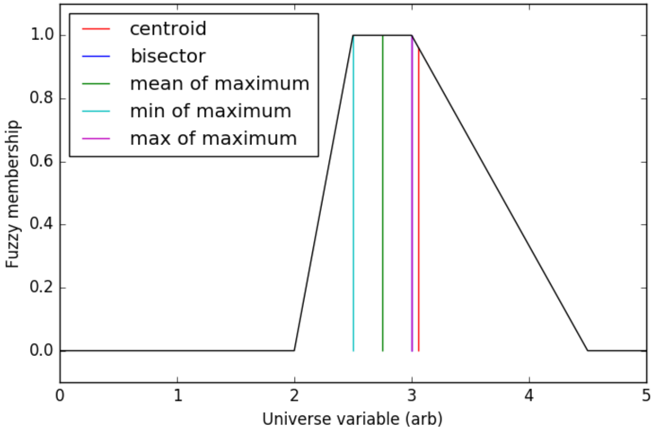
`centroid`, `bisector`, `mom`, `som`, `lom`

Sample code

```
y = -10:0.1:10;
B = trapmf(y,[-10 -8 -4 7]);
b = defuzz(y, B, 'centroid');
```

Defuzzification in python/scikit-fuzzy (cont.)

```
defuzz = fuzz.defuzz(x, mfx, 'centroid')
        {bisector, mom, som, lom}
```



Numerical characteristic of FLC

Numerical characteristic of FLC

A graph depicting the behavior of a FLC from the point of view of controlled system. It shows functional dependence between input and output variable(s) in their crisp (non-fuzzy) form.

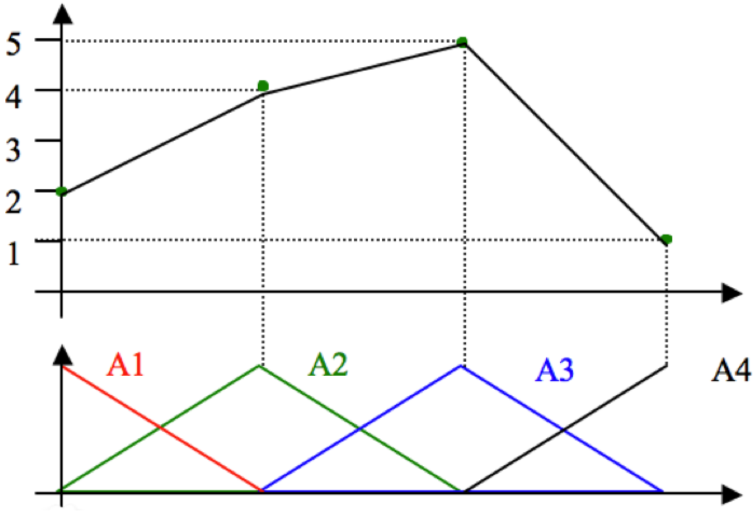
Consider a FLC with input fuzzy sets $A1\{x;0,0,1\}, A2\{x;0,1,2\}, A3\{x;1,2,3\}, A4\{x;2,3,3\}$, output fuzzy singletons $b1=\{1\}, b2=\{2\}, b3=\{4\}, b4=\{5\}$, and the following set of rules:

$$\begin{array}{ll} \text{IF } A_1 & \text{THEN } b_2 \\ \text{IF } A_2 & \text{THEN } b_3 \\ \text{IF } A_3 & \text{THEN } b_4 \\ \text{IF } A_4 & \text{THEN } b_1 \end{array}$$

Numerical characteristic (cont.)

- Constructed by sweep of the values of input variable x over the universe $[0, 4]$, and determining numerical value of the output.
- Determining the activity level of individual rules λ_k for each value of x and defuzzification of the output set B obtained as the union of the output subsets B_k combined with their activities λ_k.
- In our case – this is simplified:
    - fuzzy singletons used in place of output fuzzy subsets
    - input space is partitioned evenly.

Numerical characteristic (cont.)



Numerical characteristic (cont.)

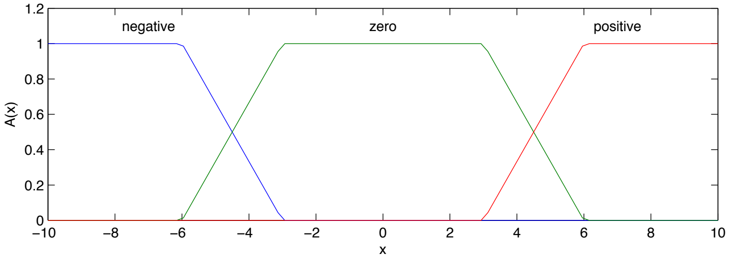
- As two subsequent input fuzzy sets compensate activities of their corresponding rules (i.e. as λ_k decreases λ_{k+1} increases totalling to $\sum kA_k(x) = 1$ for any value of x), these points corresponding to the modal values can be connected by straight lines.
- This example illustrates that using linear concepts (triangular fuzzy sets, even partition) can yield a non-linear behaviour of fuzzy controller (due to nonlinearity of rules).

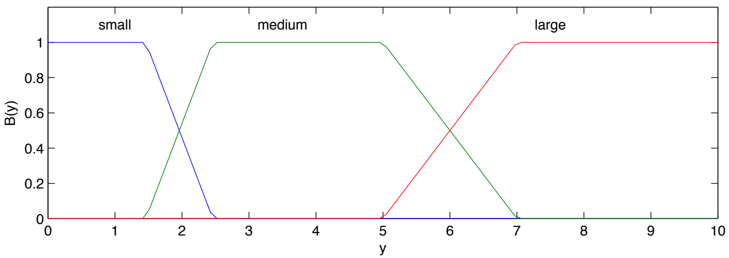
Compiled fuzzy controller

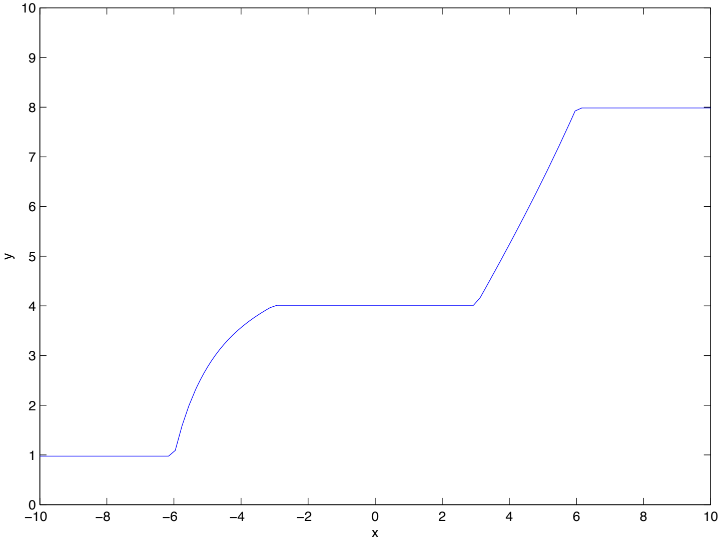
The concept of numerical characteristic can be used to build a compiled version of fuzzy controller stored as a look-up table on particular hardware (ROM, FPGA, etc.).

Example: Single-input Mamdani fuzzy model

IF x is “negative”	THEN y is “small”
IF x is “zero”	THEN y is “medium”
IF x is “positive”	THEN y is “large”


negative
zero
positive
A(x)
1.2
1
0.8
0.6
0.4
0.2
0
−10
−8
−6
−4
−2
0
2
4
6
8
10
x

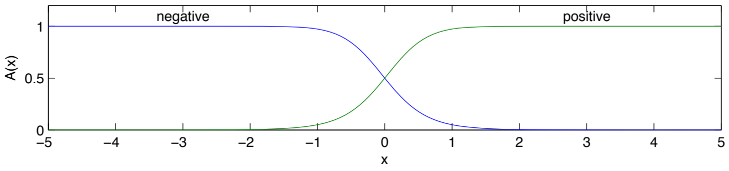

small
medium
large
B(y)
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9
10
y


10
9
8
7
6
5
4
3
2
1
0
y
–10
–8
–6
–4
–2
0
2
4
6
8
10
x

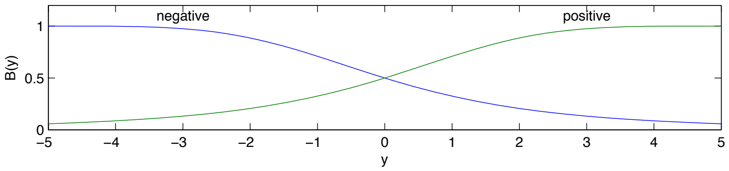
Compiled fuzzy controller (cont.)

Example: Two-input Mamdani fuzzy model

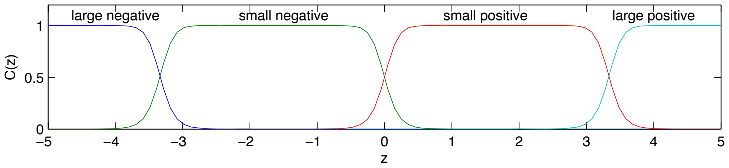
IF x is “negative”	AND y is “negative”	THEN z is “negative large”
IF x is “negative”	AND y is “positive”	THEN z is “negative small”
IF x is “positive”	AND y is “negative”	THEN z is “positive smal”
IF x is “positive”	AND y is “positive”	THEN z is “positive large”



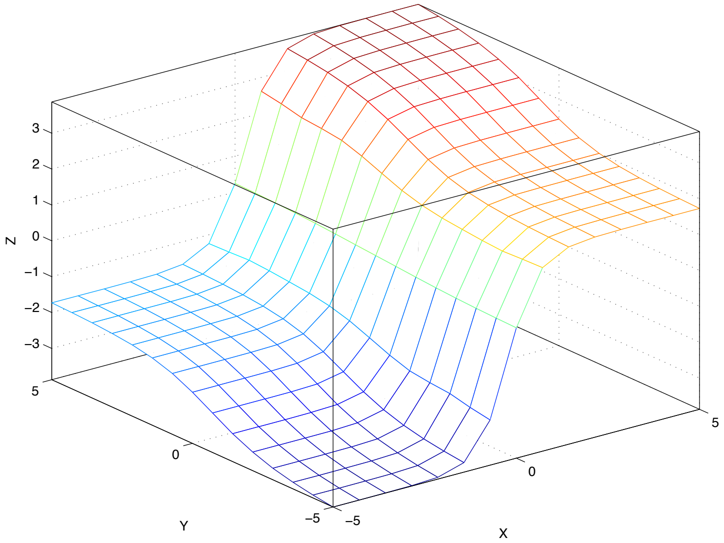
negative
positive
A(x)
1
0.5
0
–5
–4
–3
–2
–1
0
1
2
3
4
5
x



negative
positive
B(y)
1
0.5
0
–5
–4
–3
–2
–1
0
1
2
3
4
5
y



large negative
small negative
small positive
large positive
C(z)
1
0.5
0
–5
–4
–3
–2
–1
0
1
2
3
4
5
z

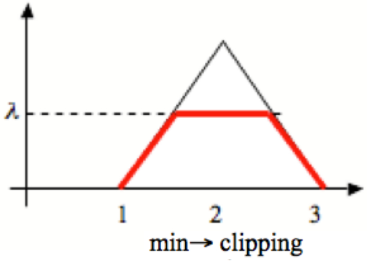

Z
3
2
1
0
−1
−2
−3
5
0
−5
Y
−5
0
5
X

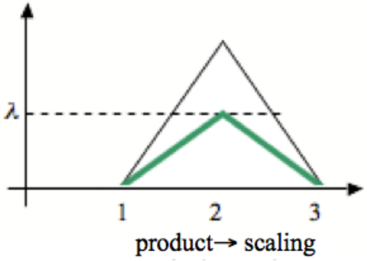
Other types of fuzzy inference

The basic type of fuzzy inference is based on Mamdani fuzzy implication operator (min) and the standard form of rules

IF x is A AND y is B THEN z is C

Its variant is based on rules of the same form, but uses Larsen fuzzy implication operator (product). These two types of fuzzy inference provide clipping and scaling of corresponding output fuzzy set





Selection of defuzzification scheme

The choice of defuzzification scheme depends largely on the requirements for speed of processing and implementation constraints

- the centroid (COG, COA) method is most precise, but also most demanding
- using the sum-product composition, calculation of the centroid can be greatly simplified

$$b = \mathrm{COA}\left(B(y)\right) - \frac{\int B(y)ydy}{\int B(y)dy} = \frac{\sum \lambda_i a_i y_i}{\sum \lambda_i a_i}$$

where $a = \int B(y)dy$ and $y_i = \int B(y)ydy / \int B(y)dy$ are the area and centroid of the consequent membership function $B_i(y)$, resp. These values can be easily determined analytically for most standard membership functions.

Sugeno fuzzy inference

This type of inference modifies the standard form of fuzzy rules

$$\text{IF } x \text{ is } A \text{ AND } y \text{ is } B \text{ THEN } z = f(x, y),$$

i.e. the output of the controller is determined using a crisp function of the input variables. Usually, $f(x, y)$ is polynomial in the input variables x and y, but it can be an arbitrary function that appropriately describes the output of the model within the fuzzy region specified by the corresponding antecedent of the rule.

Sugeno fuzzy inference – order

When $f(x, y)$ is a first-order polynomial, the resulting fuzzy RBS is called a first-order Sugeno fuzzy model. A zero-order Sugeno fuzzy model ($z = k$, a constant) is also often used, and can be viewed as a special case of Mamdani FRBS with consequents specified in form of fuzzy singletons

$$\text{IF } x \text{ is } A \text{ AND } y \text{ is } B \text{ THEN } z = k,$$

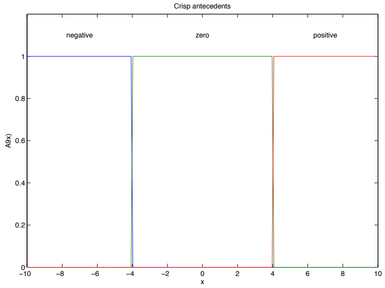
The output of a Sugeno model is smooth as long as the antecedent fuzzy sets have enough overlap (i.e. discontinuity of the consequents does not break the smoothness).

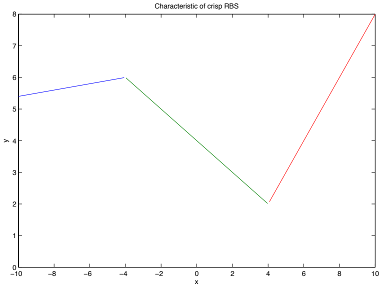
Sugeno fuzzy model

Example: Single-input first-order Sugeno fuzzy model

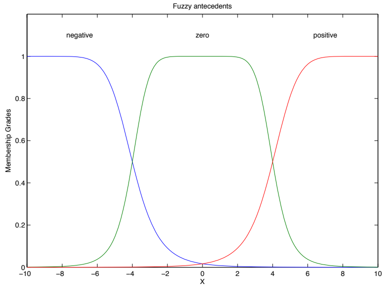
IF x is "negative" THEN $y = 0.1x + 6.4$
IF x is "zero" THEN $y = -0.5x + 4$
IF x is "positive" THEN $y = x - 2$

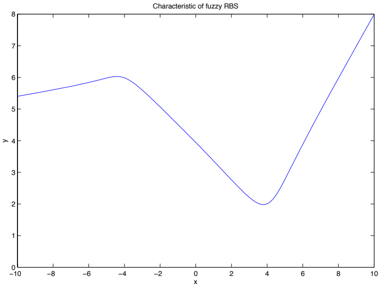
Sugeno - crisp antecedents





Sugeno - fuzzy antecedents



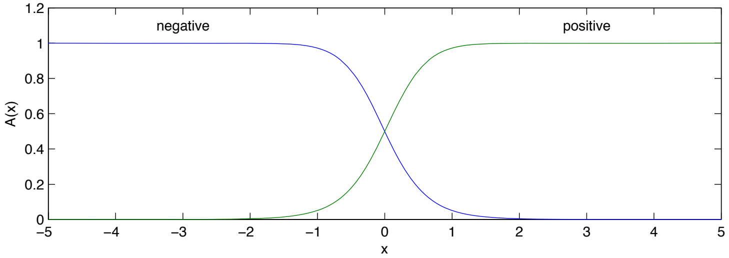


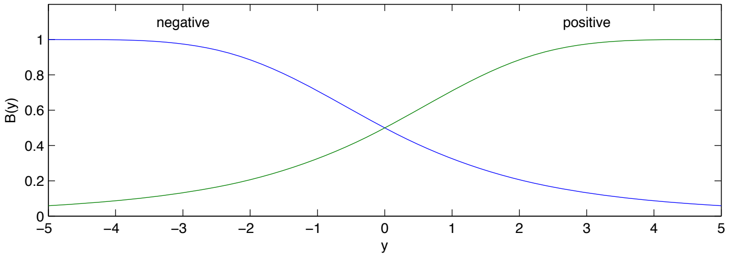
Sugeno fuzzy model

Example: Two-input first-order Sugeno fuzzy model

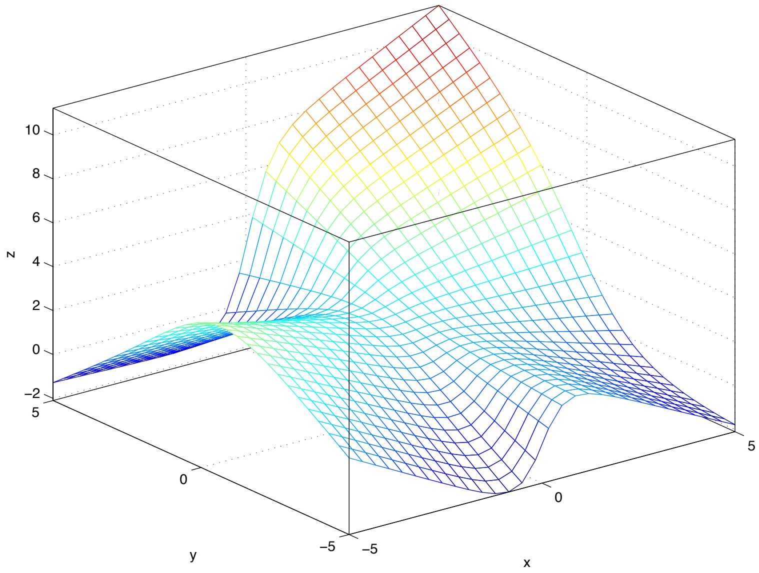
IF x is “negative”	AND y is “negative”	THEN $z = -x + y + 1$
IF x is “negative”	AND y is “positive”	THEN $z = -y + 3$
IF x is “positive”	AND y is “negative”	THEN $z = -x + 3$
IF x is “positive”	AND y is “positive”	THEN $z = x + y + 2$

Sugeno fuzzy model





Sugeno fuzzy model



Properties of Fuzzy Controllers

Some considerations are important in design and development of fuzzy controllers

Static aspects of FLCs

(i) Completeness
(ii) Continuity
(iii) Consistency

Dynamic aspects of FLCs

(iv) No-interactivity
(v) Robustness and stability

Completeness of fuzzy rule base

Rule base is complete

if the fuzzy sets standing in the condition parts of the rules cover the corresponding universes of discourse.

Given a set of rules

$$\text{IF } x \text{ is } A_i \text{ THEN } y \text{ is } B_i,\ \ i = 1, 2, ..., n,$$

the condition of completeness can be formulated as

$$\forall x \in X\ \exists i = 1, 2, \ldots, n:\ \ A_i(x) > 0$$

This condition ensures that a satisfactory control inference is made for any valid set of data (observations). It is a non-strict requirement: less important or unknown conditions can be omitted without seriously affecting the system performance.

Continuity of fuzzy rule base

Rule base is continuous

when there are “no gaps” between any rules with respect to their MF.

More specifically, when the condition fuzzy set overlap, the action fuzzy sets should also overlap. Formally, a rule base is continuous if for some rule, $A_i \rightarrow B_i$, there exist at least one rule

$$A_j \rightarrow B_j;\ j \neq i$$

such that both antecedents and consequents of these rules overlap, i.e.

$$\text{Poss}(A_i, A_j) > 0 \text{ and } \text{Poss}(B_i, B_j) > 0$$

Consistency of fuzzy rule base

Consistency as a very important requirement, ensuring that there are no inconsistent (contradictory) rules in the rule base. Consider the following two pairs of rules:

IF road is crowded	THEN drive slowly
IF road is crowded	THEN drive fast

IF close to obstacle	THEN turn left
IF close to obstacle	THEN turn right

They both represent conflicting rules:

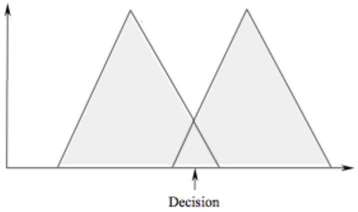
- collected from different experts (the first pair)
- based on different observations (the second pair)

Conceptually, one can envision the following two cases.

Inconsistency - case 1

$$A_i \rightarrow B_i \text{ OR } A_i \rightarrow B_i^{'}$$

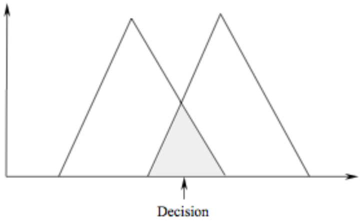
- Two rules with the same antecedent but different consequents, connected with OR
- If the consequents are mutually exclusive, i.e. $B_i \bigcap B_i^{'} = \emptyset$, the rule base is inconsistent.
- For partial overlap, the level of inconsistency decreases, but inference is still inconsistent (multimodal MF).



Inconsistency - case 2

$$A_i \to B_i \text{ AND } A_i \to B_i^{'}$$

- Two rules with the same antecedent but different consequents, connected with AND
- If the consequents are mutually exclusive, i.e. $B_i \bigcap B_i^{'} = \emptyset$, the rule base is inconsistent.
- For partial overlap, the level of inconsistency decreases, but the inference is still inconsistent (although output MF is unimodal, it still lies in the middle of two consequents)



Inconsistency evaluation

Quantitatively, it can be evaluated using Poss - expression of similarity between fuzzy sets. We need to capture situation of similar conditions and different conclusions, i.e. in extreme case

$$\text{Poss}(A_i, A_j) = 1,\ \text{Poss}(B_i, B_j) = 0$$

Consistency between two rules, *i* and *j*, can be evaluated with an implication operator:

$$\text{Cons}(i, j) = \text{Poss}(A_i, A_j) \rightarrow \text{Poss}(B_i, B_j)$$

In the case above:

$$1 \rightarrow 0 = 0$$

Implication operator

An example of implication operator suitable for this purpose has the following form

$$a \to b = \begin{cases} 1 & \text{if} a \le b, \\ 0 & \text{if} a > b. \end{cases}$$

Mutual consistency of all pairs of rules in a rule base can be arranged in the following consistency matrix

$$\begin{bmatrix} \text{Cons}(1,1) & \text{Cons}(1,2) & \text{Cons}(1,3) & \text{Cons}(1,n) \\ \text{Cons}(2,1) & \text{Cons}(2,2) & \cdots & \\ & & & \\ \text{Cons}(n,1) & \text{Cons}(n,2) & & \text{Cons}(n,n) \end{bmatrix}$$

Implication operator

To evaluate the consistency of one particular rule, say *i*, with respect to all other rules in the base, average of mutual consistencies is taken across the *i*-th row of the matrix

$$\mathrm{Cons}(i, \mathrm{RB}) = \frac{1}{n} \sum_{j=1}^{n} \mathrm{Cons}(i, j)$$

After determining consistency of all rules this way, the rules can be sorted according to their consistency value Cons(*i*, *RB*), and most inconsistent rules (from the bottom of the sorted group) manipulated to improve the overall consistency of the rule base.

Enhancing consistency

1. Eliminating inconsistent rules – simple but can cause problems with coverage/continuity of the rule base
2. Modifying inconsistent rules – there are several approaches:
    - Making conditions more specific and/or conclusions more general
    - Adding a (missing) condition

Making conditions more specific and/or conclusions more general

$$\mathrm{Cons}(i, j) = \mathrm{Poss}(A_i, A_j) \rightarrow \mathrm{Poss}(B_i, B_j)$$

This can be achieved, e.g. using linguistic modifiers (hedges), such as "very", "more or less". There are limits to this, as the modal values do not change their location, MFs are only compressed or expanded.

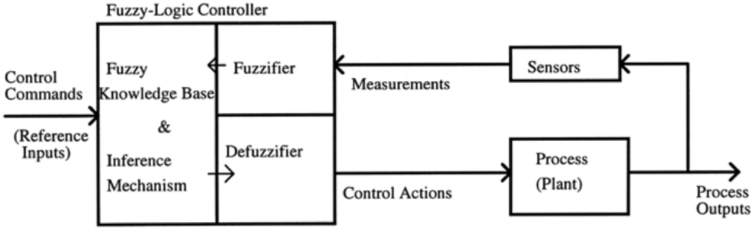
Adding a (missing) condition C

IF road is crowded	AND road is slippery	THEN drive slowly
IF road is crowded	AND road is dry	THEN drive fast

This changes the situation by removing the conflict. This way, we are now evaluating Poss($A_i \wedge C_i, A_j \wedge C_j$) instead of Poss($A_i, A_j$) which leads to lowering the amount of condition overlap, effectively removing the inconsistency.

Fuzzy Control Architectures

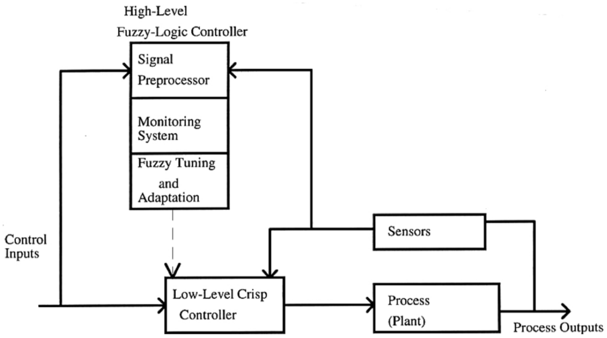
1) The most common, conventional, direct-control architecture. FLC generates low-level direct control signals. This architecture is suitable for low demand control problems (high time constants of controlled process, low control bandwidth) which would be difficult to model analytically and for which there is a sufficient experience with low-level human-in-the-loop control.



Fuzzy Control Architectures (cont.)

2) Hierarchical supervisory control architecture using crisp low-level direct controller and high-level fuzzy controller performing supervisory tasks, i.e.

- process monitoring
- performance assessment
- tuning, adaptation, and restructuring of the low-level controller


High-Level
Fuzzy-Logic Controller
Signal Preprocessor
Monitoring System
Fuzzy Tuning and Adaptation
Control Inputs
Sensors
Low-Level Crisp Controller
Process (Plant)
Process Outputs