

School of Artificial Intelligence  
Division of Informatics, University of Edinburgh

**AI3 Large Practical**  
**A Comparison of the**  
**Interpretation Methods for Fuzzy Inference**

Tutor: Jeroen Keppens

author: Lucas Dixon (lucasd)

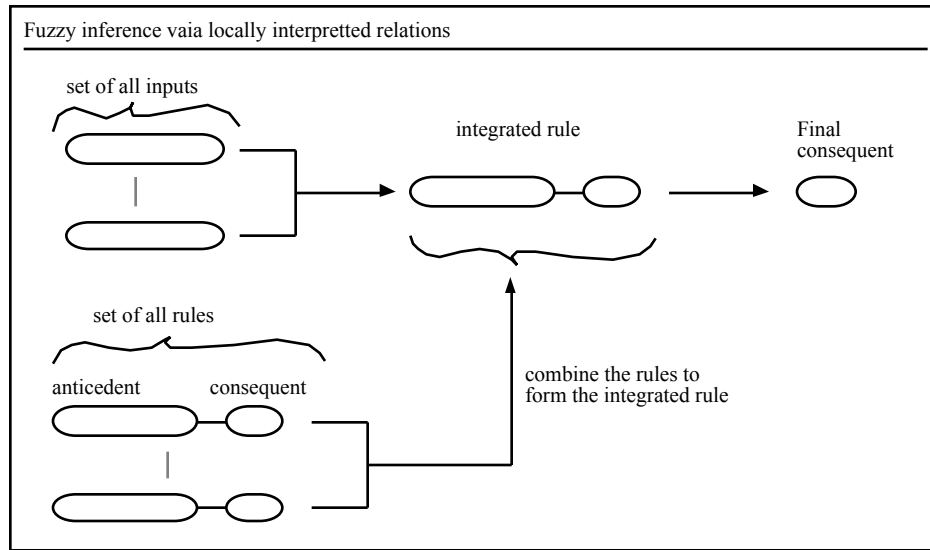
# Contents:

<b>Contents</b>	<b>2</b>
<b>Introduction</b>	<b>3</b>
fuzzy logic	4
fuzzy Sets	5
fuzzy dimensions	5
domain of discourse	5
fuzzy values	5
operations and connective's on fuzzy sets	6
attributes of fuzzy sets	8
fuzzy relations	8
multi-dimensional fuzzy sets	9
fuzzy rules and fuzzy inference	9
the need for fuzzification and defuzzification	11
<b>Details</b>	<b>12</b>
ambiguity	12
normalization and denormalisation	12
Inference via locally interpreted relations	13
Inference with overall interpreted relation	14
<b>Analysis and Criteria</b>	<b>15</b>
performance criteria	15
representation	15
representation by plateaus	16
analysis of methods and problem	17
variables	19
Proof of mathematical equality	20
<b>Design</b>	<b>21</b>
data structures & representation	21
the interface	22
fuzzification	23
defuzzification	25
tools	26
knowledge compiler	27
inference	28
future improvements of the program	28
<b>program testing</b>	<b>29</b>
methodology	29
results	30
conclusions	31
<b>traffic intersection control : a case for fuzzy inference</b>	<b>32</b>
<b>A comparison of two methods of defuzzification</b>	<b>35</b>
<b>Bibliography</b>	<b>37</b>

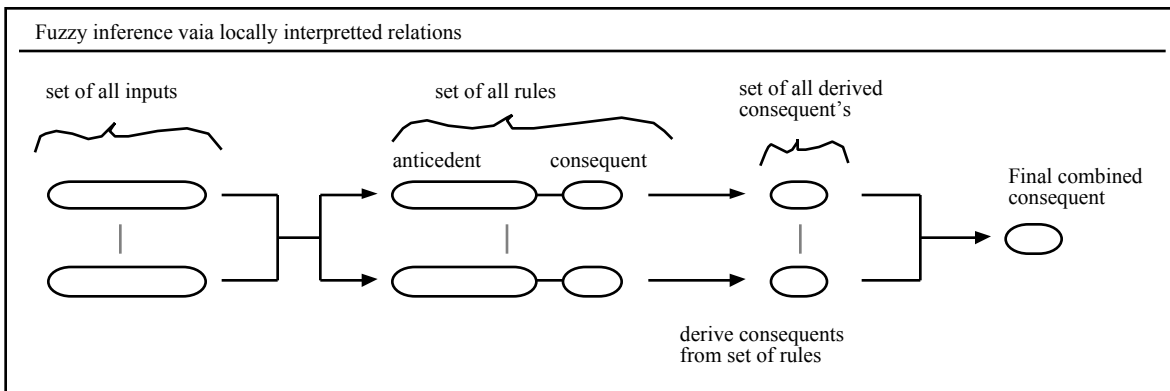
# Introduction

The process of fuzzy inference over a set of rules, with the same consequent domain of discourse, to produce a single consequent, has two basic solutions: to derive the consequent from each rule, and combine the set of consequent's; or to combine the rules into an integrated rule, and the use the integrated rule to derive a single consequent. This paper is a comparison of these two basic interpretations methods for fuzzy inference.

'Inference with overall interpreted relation' : combine the rules into an integrated rule, and the use the integrated rule to derive a single consequent.



'Inference via locally interpreted relations' : derive the consequent from each rule, and combine the set of consequent's to form a single consequent



The requirements to perform a comparison of interpretation method's for fuzzy inference, are:

- an understanding of the problem domain of fuzzy logic ( Fuzzy Logic: Section 1);
- a testable and complete definition, including knowledge representation details, for the interpretations method's (Implementation Details : Section 2);
- a definitions of the performance criteria with which to compare the methods, and how these criteria can be tested (Performance Criteria : Section 3).

The testing of an implementation of an interpretation method will provide results for that implementation of the method, however a careful analysis of the implementation and method is required to arrive at conclusions applicable to more than the specific implementation. An analysis of the interpretation methods will also provide hypothesis for the testing (Problem Analysis : Section 4).

To make testing of the implementation's of the inference method's feasible a more complete fuzzy logic implementation was developed, including fuzzification, defuzzification, knowledge compilation and a corresponding interface. ( Complete Implementation Design : Section 5 ). With a complete implementation, testing of the selected performance criteria can be performed (Results : Section 6 ), to obtain results from which a conclusion can be given. (Conclusion : Section 7).

# Fuzzy Logic

'Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth - truth values between "completely true" and "completely false". It was introduced by Dr. Lotfi Zadeh of UC/Berkeley in the 1960's as a means to model the uncertainty of natural language.' - comp.ai.fuzzy FAQ.

## from Boolean to fuzzy logic - degrees of membership

The essential difference between fuzzy logic and Boolean logic is the range of values defining the possible degrees of membership. The degree of membership of an element, in a domain of discourse, is a value representing the a certainty that the element is a member of the domain of discourse. Where in Boolean logic the degree of membership is either 0, or 1, representing exclusion (false) and inclusion (true) respectively; in fuzzy logic the degree of membership is a value in the interval [0,1] (real numbers between zero (false) and one (true)). This allows representation of uncertainty or quantitative membership in a domain of discourse.

Example of degree of membership:

the degree of membership of the height 'John' in the domain of discourse: 'tall people', in Boolean logic could be: 0, (representing: 'John' is not a member of the set 'tall people'). In fuzzy logic the degree of membership might be 0.2 (representing: 'John' is almost certainty not a member of 'tall people', this could also be interpreted as representing the linguistic construct: John is not very tall).

## fuzzy Sets

A fuzzy set for a domain of discourse is a set of elements and their degree's of membership. For example, for the above domain of discourse 'tall people' there may be a fuzzy set:

{ (element:John, degree of membership:0.2), (element:Fred, degree of membership:0.8) }

## fuzzy dimensions

A fuzzy dimension<sup>1</sup> is a domain of discourse and a mapping of elements to the real numbers. For example 'heights of people' is a fuzzy dimension, with the mapping as the height in feet. Note that for a domain of discourse there may be many different mappings to the real numbers, and so many different fuzzy dimensions. Thus a fuzzy dimension is a representation of a domain of discourse.

## domain of discourse

the domain of discourse is a set of elements that statements are being made about. For example a domain of discourse might be: people that are very tall. A linguistic concept of a domain of discourse from which fuzzy inference is performed is referred to in this document as a real domain, as opposed to fuzzy domain which is used for the fuzzy set representation of a domain of discourse. For example the real domain: 'heights of people' is a linguistic concept of a domain of discourse, hence a real domain, however 'heights of people' could also have a fuzzy representation, so it can also be a fuzzy domain. 'heights of people, measured in feet' is also a real domain and a fuzzy domain, but because provides a function mapping the domain of discourse to a real line, it is also a fuzzy dimension.

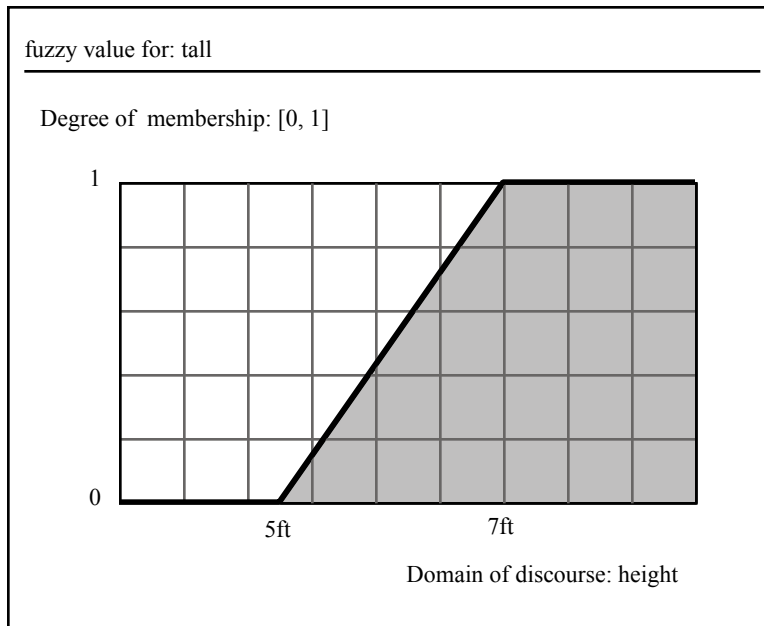
## fuzzy values

A fuzzy value is a fuzzy set representing a linguistic variable.

for example: in the fuzzy dimension of 'heights of people', the linguistic variable 'tall' might have the fuzzy value defined by the graph:

---

<sup>1</sup> (this is not official terminology, it has been used in light of nothing more suitable)



### operations and connective's on fuzzy sets

There are two common binary operations on fuzzy sets<sup>2</sup> : minimum, and maximum. These two operations are used to represent the effect of binary connective's on terms. A binary connective, over two elements in the same domain of discourse, has a degree of membership in the domain of discourse defined by a function for that connective.

The general functions defining degree's of membership for connective's are:

Given: A, B are members of domain's of discourse.  $M_x$  is the degree of membership of X.

Then:

The degree of membership of  $\text{and}(A, B)$  is defined by:  $\text{minimum}(M_A, M_B)$ .

The degree of membership of  $\text{or}(A, B)$  is defined by:  $\text{maximum}(M_A, M_B)$ .

The degree of membership of  $\text{implies}(A, B)$  is defined by:  $\text{minimum}(M_A, M_B)$ .

The degree of membership of  $\text{not}(A)$  is defined by:  $1 - M_A$ .

A binary connective 'c', with a function 'f' defining it's degree of membership, applied to fuzzy sets in the same domain of discourse:

Given Fuzzy sets:

(A fuzzy sets represented as a set of pair's of elements and their degree of membership).

$A_c = \{ (a_1, M_{c1}), (a_2, M_{c2}), \dots, (a_n, M_{cn}) \}$

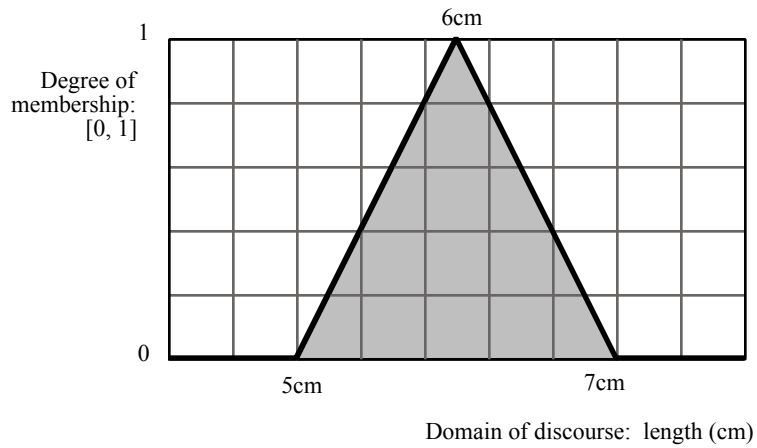
$A_d = \{ (a_1, M_{d1}), (a_2, M_{d2}), \dots, (a_n, M_{dn}) \}$

Then:  $c(A_c, A_d) = \{ (a_1, f(M_{c1}, M_{d1})), (a_2, f(M_{c2}, M_{d2})), \dots, (a_n, f(M_{cn}, M_{dn})) \}$

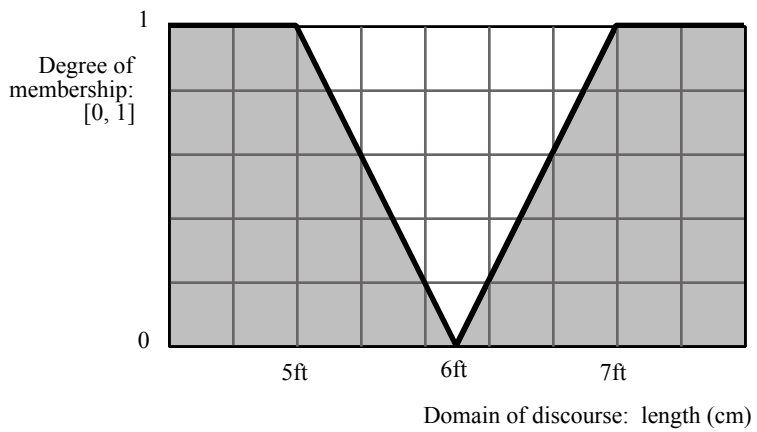
Sometimes an implicit representation of fuzzy sets is used, where any element not included in a set is assumed to have a degree of membership of 0, the above formula assumes complete explicit representation.

<sup>2</sup> Note that there is some variation in the functions representing connective's, within this paper only the functions defined are used.

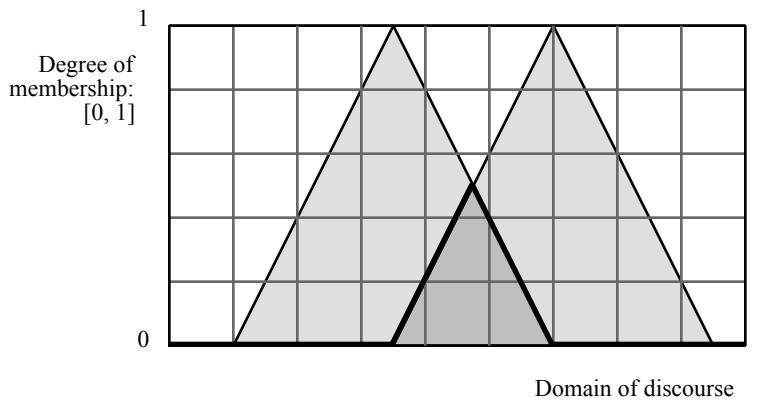
fuzzy value for: about 6cm long

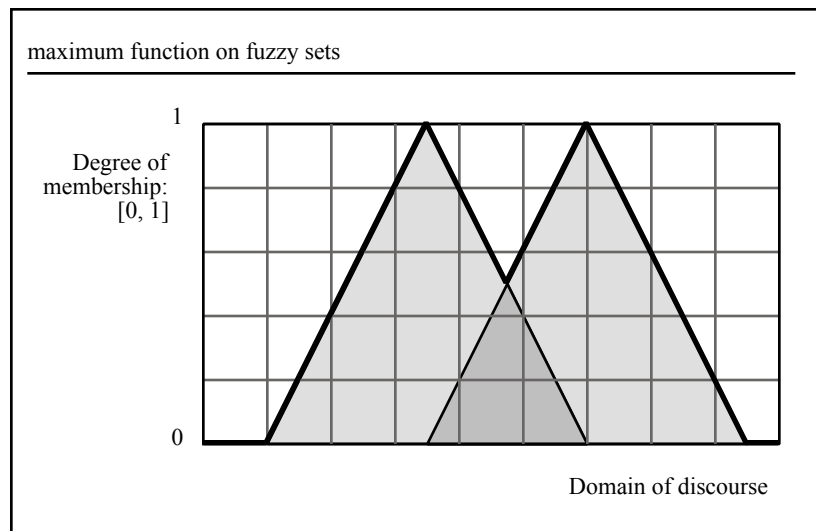


fuzzy value for: not (about 6cm long)



minimum function on fuzzy sets





### attributes of fuzzy sets

An attribute of a fuzzy set is a fuzzy set in a domain of discourse that makes up the domain of discourse of the complete fuzzy set.

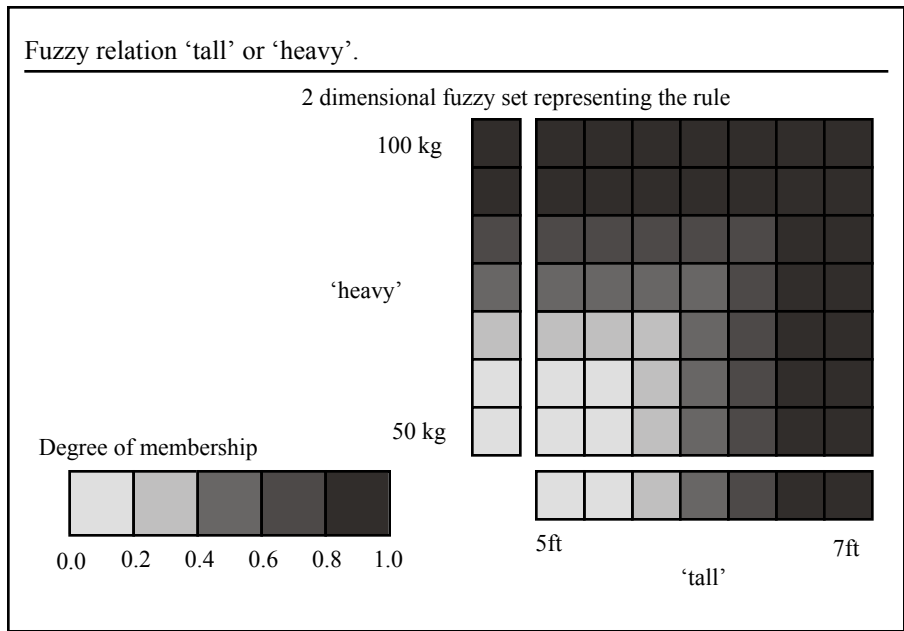
For example the fuzzy set: tall and heavy has attributes (attribute domain's): height, and weight.

### fuzzy relations

If a connective is applied to fuzzy sets with attributes from different domain's of discourse, then the result is the representation of the linguistic connective's relation over the fuzzy sets in the domain's of discourse. More simply, connective's in fuzzy logic can relate linguistic concepts from different domain's of discourse.

In the application of fuzzy logic there are a number of domain's of discourse, however often a logical sentence will not specify all of them, in this case it is taken as an implicit assumption that the sentence is asserted for every element in the domain's of discourse not explicitly represented in the logical sentence.

For example: The fuzzy values defined by the linguistic variables: 'tall', and 'heavy', could be connected using the 'or' connective: 'tall' or 'heavy'. This would result in a two dimensional fuzzy domain, representing the linguistic concept: 'tall or heavy'.



A binary connective 'c', with a function 'f' defining its degree of membership, applied to fuzzy sets in the different domain's of discourse gives a result in the form of a multi dimensional fuzzy set:

(A multidimensional fuzzy set is represented as a set of pair's of related-sub-element's and the degree of membership of the related-sub-element's in the multidimensional domain of discourse).

Given multidimensional fuzzy sets:

$S_a = \{ (a_1, Ma_1), (a_2, Ma_2), \dots, (a_n, Ma_n) \}$

$S_b = \{ (b_1, Mb_1), (b_2, Mb_2), \dots, (b_n, Mb_n) \}$

Then:  $c( A_c, A_d ) =$

$\{ (c(a_1, b_1), f(Ma_1, Mb_1)), (\{a_1, b_2\}, f(Ma_1, Mb_2)), \dots, (\{a_1, b_n\}, f(Ma_1, Mb_n)) \}$

$(c(a_2, b_1), f(Ma_2, Mb_1)), (\{a_2, b_2\}, f(Ma_2, Mb_2)), \dots, (\{a_2, b_n\}, f(Ma_2, Mb_n)) \}$

...

$(c(a_n, b_1), f(Ma_n, Mb_1)), (\{a_n, b_2\}, f(Ma_n, Mb_2)), \dots, (\{a_n, b_n\}, f(Ma_n, Mb_n)) \}$

### multi-dimensional fuzzy sets

a multi dimensional fuzzy set is fuzzy set who's domain of discourse is defined as a relation of a set of fuzzy dimensions. multi dimensional fuzzy sets are created by integrating fuzzy relations on different domains of discourse. For example: multi-dimensional fuzzy set: (height and weight) is the 2 dimensional fuzzy set of a relation between height and weight.

### fuzzy rules and fuzzy inference

In fuzzy logic control, a fuzzy rule is a logical sentence upon which a derivation can be performed. The act of performing a derivation is referred to as inference. There are a number of commonly used derivational rules in Boolean logic, MPP, MTT, &-Introduction, &-Elimination etc. However for fuzzy logic control only one derivation is commonly used: MPP (Modus Ponendo Ponens).

In Boolean logic MPP:

$$\frac{A, A \text{ implies } C}{C}$$

In Boolean logic, the derivation assumes that A has membership degree 1 (A is true), likewise the rule is also true. Given this the derivation concludes that B has membership degree 1 (B is true).

In fuzzy logic MPP is defined more abstractly:

$$\frac{A, R_A \text{ implies } R_C}{C}$$

The domain of discourse of  $R_A$  and  $A$ , must be the same otherwise the derivation is not applicable. The derived consequent  $C$  is of the same domain of discourse as  $R_C$ .

Given:

a fuzzy set  $A$  with membership degree  $M_A x$  for an element  $X$  in  $A$ ;  
 a fuzzy rule  $R$ , with Antecedent Fuzzy set  $R_A$ , and consequent fuzzy set  $R_C$  ( $R_A$  implies  $R_C$ ).

The fuzzy set representing  $\text{implies}(R_A, R_C)$  will be:

$R_{Cx}$  = an element  $X$  in the set  $R_C$ .  
 $R_{Ax}$  = an element  $X$  in the set  $R_A$ .  
 $M_{R_A X}$  = degree of membership of a element  $X$  in  $R_A$ .  
 $M_{R_C X}$  = degree of membership of a element  $X$  in  $R_C$ .

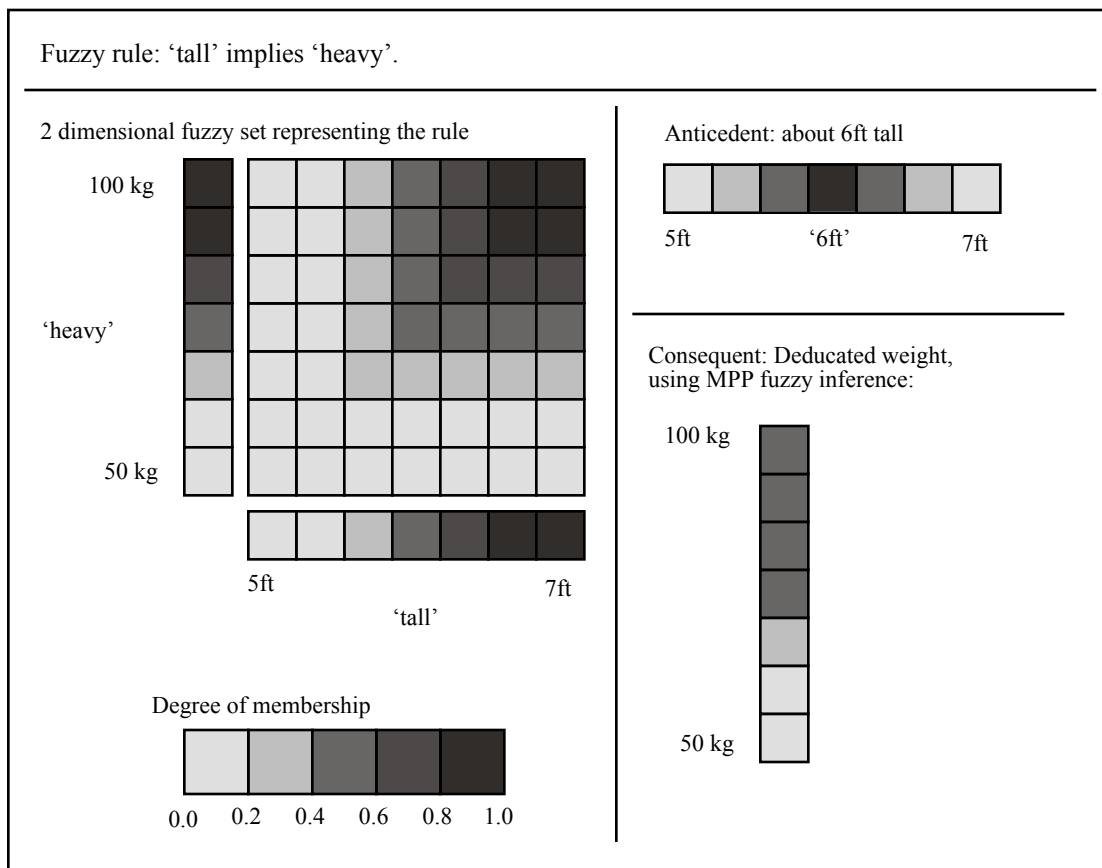
$\{ ( \{ R_{Cx}, R_{Ay} \}, \text{minimum}(M_{R_C X}, M_{R_A Y}) ) ; \text{for all } (R_{Cx}, R_{Ay}) \}$

The fuzzy set  $C$ , will be

$\{ (R_{Cx}, M_{Cx}) ; \text{for all } R_{Cx} \text{ where } M_{Cx} = \max( \{ \min( M_{Ay}, \text{minimum}(M_{R_C X}, M_{R_A Y}) ) ; \text{for all } y \} ) \}$

Example:

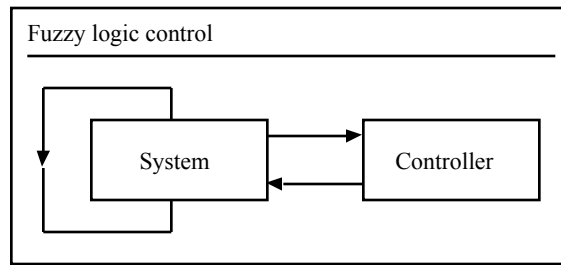
$A$  = 'about 6ft',  $R$  = fuzzy rule: 'tall' implies 'big'.  
 $C$  = quite heavy, probably more than 75kg



## about fuzzy logic control

The purpose of control is to influence the behavior of a system by changing some attributes of the system. Fuzzy logic

can be used to control a system by encoding a set of rules that define the behavior of the controller. In fuzzy logic control an observation of some properties of the system are taken as input to the fuzzy logic controller, which uses a process of inference to define a function from the given input's to the output's of the controller, so changing some attributes of the system.

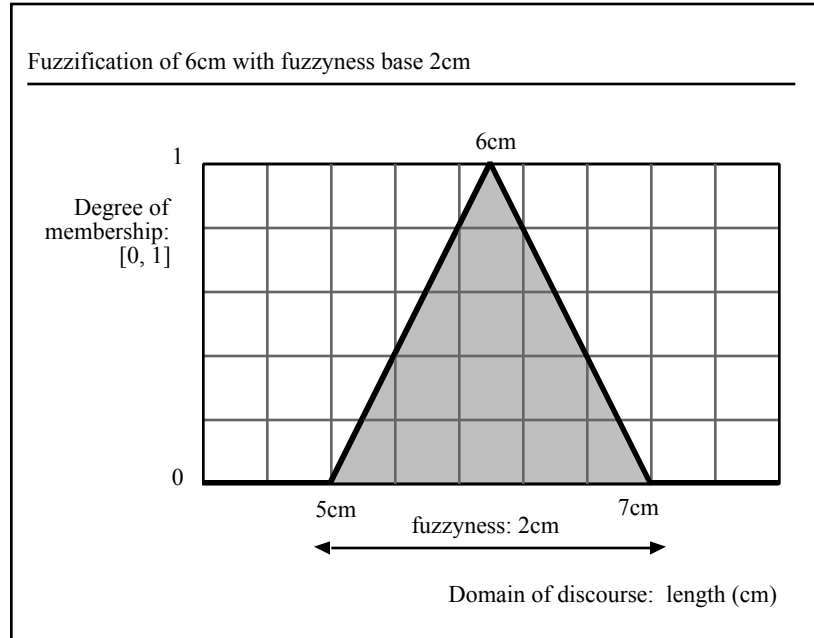


### the need for fuzzification and defuzzification

Fuzzification is required in fuzzy logic control as often inputs are not in the form of a fuzzy set, but instead are a single value. Fuzzification is a function from a discrete value, in a domain of discourse, to a fuzzy dimension. This process creates a fuzzy value from a 'crisp' value. The process of fuzzification can encode both the concepts of uncertainty and relative membership. Uncertainty of input is encoded by having high membership of other likely input element, generally this is input elements close to the received input element as mapped to in the fuzzy dimension, but not necessarily. Relative membership is the concept that if an element has degree of membership  $X$ , then elements close to it should have a degree of membership close to  $X$ .

Fuzzification in generally uses the concept of an input element and a fuzziness, where the fuzziness defines the relative degree of membership of elements at a distance from the input element. The use of a fuzzy triangle is very common:

Example fuzzification of 6cm in the domain of discourse of length, with fuzziness 2cm, using triangular fuzzification:

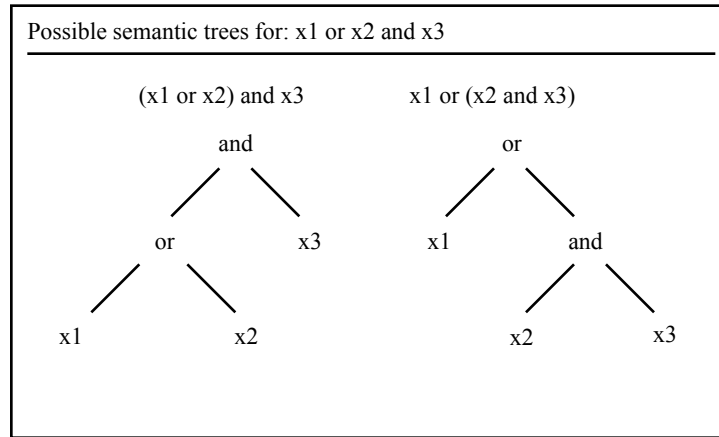


In fuzzy logic control the result of a fuzzy inference is of use in the real domain, so a defuzzification function from the consequent fuzzy set to a single element in the domain of discourse is required. This is a function from a fuzzy set to a single discrete ('crisp') value. An example of a defuzzification function is: first maxima, which takes first peak in a fuzzy set as the defuzzified value. For the fuzzified value of 6ft above, there the first peak is as 6ft, so the defuzzified value of the fuzzy set is 6ft.

# Details

## ambiguity

The distinction between multi-dimensional-domain's of discourse is important. consider:



There are two representations of the sentence: x1 or x2 and x3, with the same elements, but having different semantics. Writing the sentence with brackets eliminates the ambiguity. However while both complete sentences contain the same set of domain's of discifnce, the domain of discifnce of the sentences as a whole are different. This is an important factor in fuzzy inference, as given a rule to perform a derivation on, the sentence matched to the antecedent must have the same domain of discifnce as the antecedent. Thus given a set of input's to be inferred from and a rule, the input's must be changed into the a form matching the antecedent attribute.

Example:

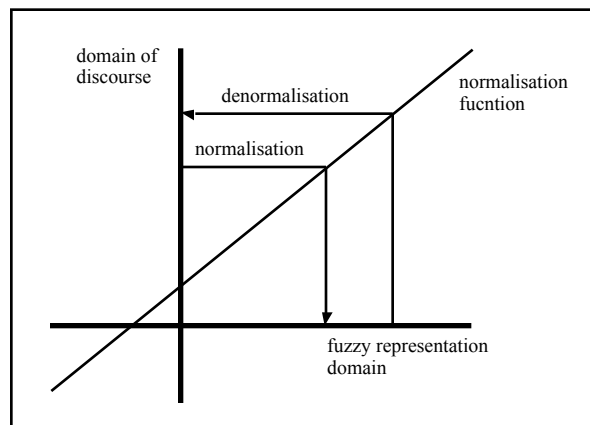
Rule: 'about 6ft' and 'about 20' implies 'about 85kg'

domain's of discifnce: height, age, and weight, respectively. An input to this rule must a multi-domain of discifnce of: 'height and weight'.

and inputs set: { about 5ft5, about 24 }, the input must be converted into the domain of discifnce: 'height and weight', so the input is related to form: 'about 5ft5 and about 24', which can then be used in a deduction with the rule.

## normalization and denormalisation

Normalization is a function that maps a single element from a domain of discifnce to the equivalent single element in the fuzzy representation domain. denormalisation is the inverse function of normalization, a mapping from a single element in the fuzzy representation domain, to a single element in the domain of discifnce. Normalization and denormalisation are used when the range, and spread of values in the domain of discifnce is not convenient for representation directly as a fuzzy set. Often a strait line is used for normalization, for example:



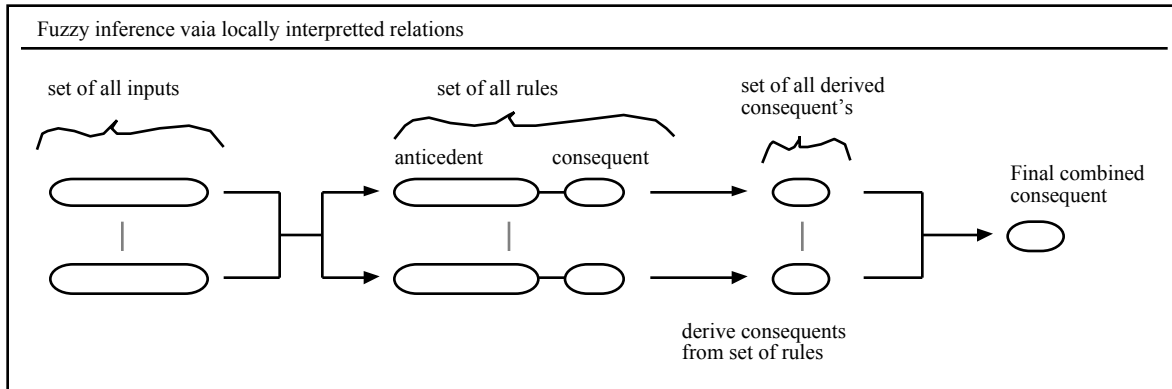
## Inference via locally interpreted relations

The antecedent of a fuzzy rule is a multi-dimensional fuzzy set. An antecedent attribute is a single fuzzy dimension (domain of discourse) of the multi-dimensional antecedent fuzzy set.

Set of all rules: a number of rules which are applicable to the input and have the same consequent domain of discourse. An applicable rule is a rule where: a subset of the set of all inputs provides a fuzzy-multidimensional space that has the same domain of discourse as the antecedent of that rule.

Set of all inputs: is the set of: fuzzy sets corresponding to antecedent attributes in the set of all rules. The only fuzzy sets that have an effect in the inference are those which correspond to a fuzzy dimension of some antecedent attribute.

Derive the consequent from each rule, and combine the set of consequent's to form a single consequent



Set of derived consequent: is the set of all derivations from the set of all rules using the set of all inputs.

The final combined consequent: is the combination of the set of derived consequent. Semantically the final consequent is the 'or' of the set of derived consequent from each rule.

Given:  $R$  = a rule in the set of all rules.  
 $R_{CX}$  = the degree of membership of element  $X$  in the rule  $R$  consequent.  
 $R_{AJ}$  = the degree of membership of element  $J$  in the rule  $R$  antecedent.  
 $I_Y$  = the degree of membership of an element  $Y$  in the related input set  $I$ .  
 $D(Z)$  = the domain of discourse of  $Z$ .  
 $C_X$  = the degree of membership of the derived consequent element  $X$ .

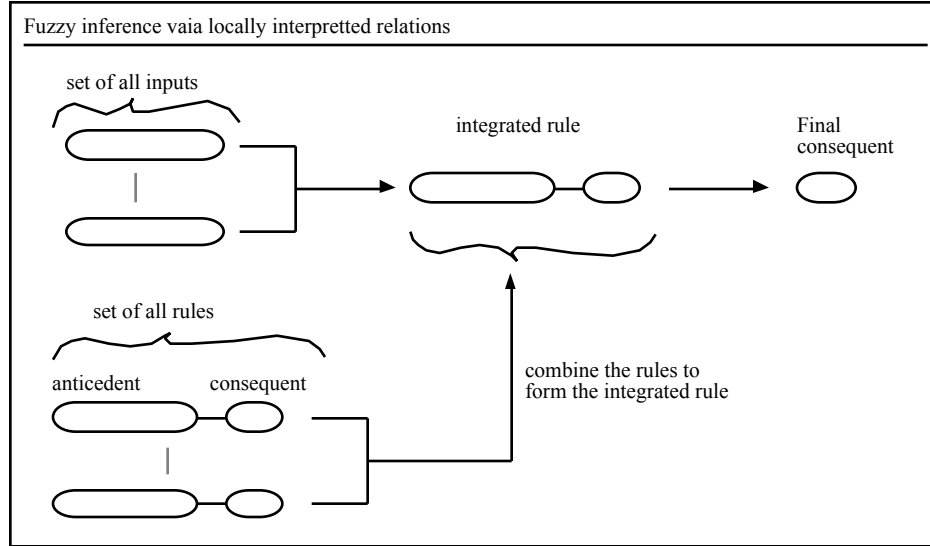
$C_X = \text{or } \{ \text{derived degree of membership from rule } R \text{ and an applicable input ; } R \}$

$C_X = \max \{ \max \{ \min \{ R_{CX}, \{ I_Y, R_{AJ} ; Y \text{ st } D(Y) = D(J) \} \} \} ; R \}$

final combined consequent fuzzy set =  $\{ (X, C_x) ; X \}$ ,  
 where  $X$  is an element in the consequent domain.

## Inference with overall interpreted relation

combine the rules into an integrated rule, and then use the integrated rule to derive a single consequent.



The final consequent is the result of the inference process on the appropriate multidimensional fuzzy set of the set of all inputs, such that multidimensional set has the same domain of discourse as the antecedent of the integrated rule.

The Integrated relation is formed by the semantic ‘or’ of the set of all rules. However to do this either the antecedent of every rule, in the set of all rules, must have the same domain of discourse, or the antecedents must be integrated appropriately. This is of concern only if within the representation of rules it is allowed to represent implicitly all elements in a non-specified domain of discourse, by omitting the attribute relating to the non-specified domain of discourse. Thus every rule is specifying, either implicitly or explicitly, the same complete set of antecedent attributes.

Example of implicit attribute representation:

domain's of discourse : height, age, weight.

rule: tall implies heavy.

This implicitly assumes that the rule is valid for all age's.

The integrated relation of the rules is defined as:

- Given:  $R$  = a rule in the set of all rules.  
 $IR$  = the integrated rule.  
 $IR_{CX}$  = the degree of membership of element  $X$  in the integrated rule  $IR$  consequent.  
 $IR_{AJ}$  = the degree of membership of element  $J$  in the integrated rule  $IR$  antecedent.  
 $R_{CX}$  = the degree of membership of element  $X$  in the rule  $R$  consequent.  
 $R_{AJ}$  = the degree of membership of element  $J$  in the rule  $R$  antecedent.  
 $I_Y$  = the degree of membership of an element  $Y$  in the related input set  $I$ .  
 $D(Z)$  = the domain of discourse of  $Z$ .  
 $C_X$  = the degree of membership of the derived consequent element  $X$ .

$C_X$  = derived degree of membership using  $IR$  with input  $I$ ;

$$C_X = \max \{ \min \{ IR_{CX}, \{ I_Y, IR_{AJ}; Y \text{ st } D(Y) = D(J) \} \} \}$$

$$C_X = \max \{ \min \{ \max \{ R_{CX}; R \}, \{ I_Y, \max \{ R_{AJ}; R \}; Y \text{ st } D(Y) = D(J) \} \} \}$$

final combined consequent fuzzy set =  $\{ (X, C_X); X \}$ ,

where  $X$  is an element in the consequent domain.

# Analysis and Criteria

## performance criteria

To make a comparison between methods, some criteria to are required upon which the two methods can be compared. A criteria is a function from the the method to a some other domain that is of interest. The application of fuzzy inference is generally done via a computer program, to solve a problem. The context of solving problems using fuzzy inference in a computer program has been chosen as the source from which to define the criteria, as this is common use for fuzzy inference. The set of measure for comparing two methods can be divided into:

Versatility and Performance. Versatility is a measure of the size and scope of problems solvable (a definition of the class of problems the method solves), and Performance is a measure of how well these problems are solved.

Both interpretation methods for fuzzy inference have the same function, and work on the same data. They fulfill the same role, and solve exactly the same class of problems giving the same result, the versatility of the two methods is identical. The mathematical equivalence of the two methods is proved below.

In the context of computer programs, where a comparison is interesting, there is two measures for performance of an algorithm: time based performance, memory based performance.

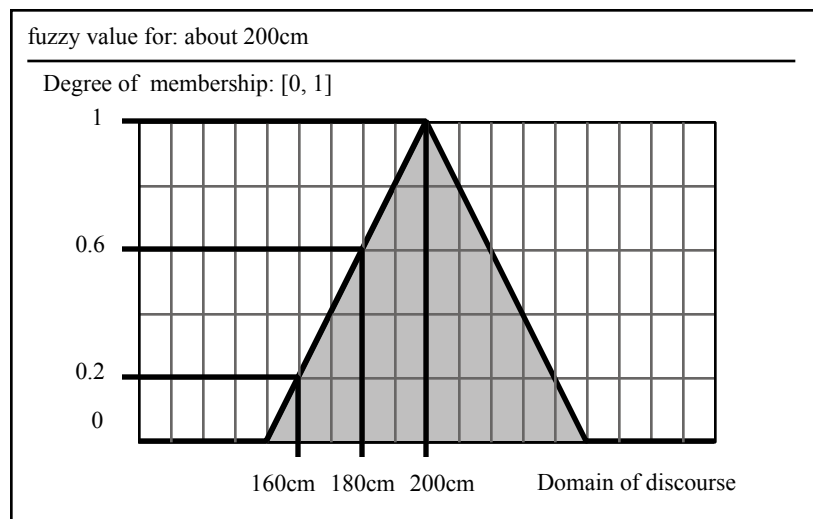
1. time based performance can be measured by the processor time taken to perform the method, measured in seconds.
2. memory based performance can be measured by the memory requirements of the program, measured in KB.

These performance criteria are dependent on an implementation. So the superior interpretation method will have a best implementation which according the the performance criteria, is superior to the other method's best implementation. However it is impossible to know the best implementation for the method's. So the testing of an implementation will only be making a claim about the set of implementation's it represents. It is still interesting as it does give a measure for the two method's for a particular implementation. In practice there there are sets of common implementations sharing common knowledge representation forms, as thus an implementation is also of interest in the question of which form of implementation is most suitable for which types of problems.

## representation

Because it would be very complicated to represent fuzzy sets as continuous functions, a different form of representation is used. Fuzzy sets are represented by a set of discrete points in the domain of discourse, with their degree of membership.

for example: the fuzzy value 'about 200cm' can be represented by the fuzzy set:  
{ ... (140cm, 0), (160, 0.2), (180, 0.6), (200, 1), (220, 0.6), (240, 0.2), (260, 0) ... }



Many dimensions represented in fuzzy logic are infinite, for example: the domain of height. However in a computational representation the domain must be represented in a finite space. The classical solution is to select a finite range of space, in the domain of discourse, to represent. However there is another more flexible solution: to represent an infinite area of space implicitly.

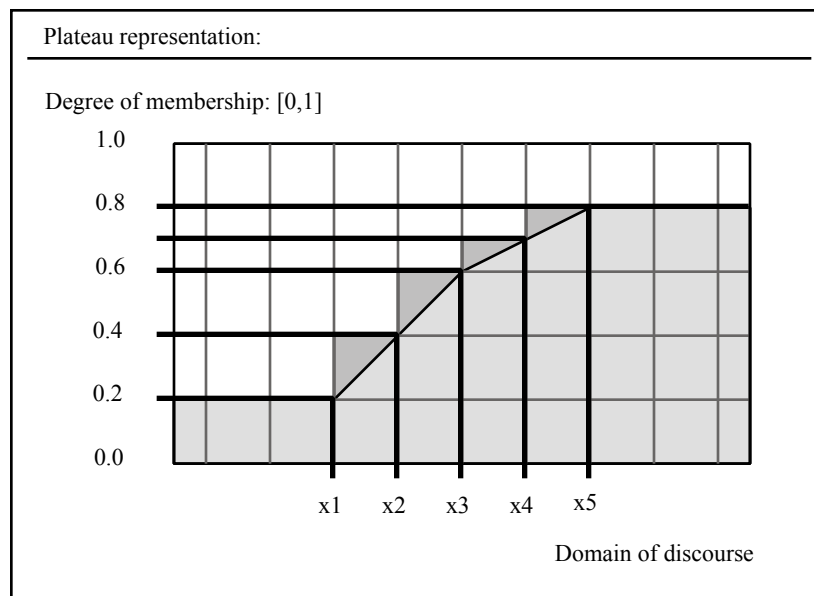
Most uses of fuzzy values real a plateau of degree of membership. Consider the example above: ‘about 200cm’ the fuzzy value has two plateaus of degree of membership that are both infinite in size: all element’s below 150cm and above 250cm have degree of membership 0. Because most of the problems dealt with have fuzzy values with plateaus infinite space it makes sense to implicitly represent the plateaus.

## representation by plateaus

every element-certainty pair in a fuzzy set is a representation for all elements between itself and the previously represented element-certainty pair. The final element in a set’s representation is the representation for the degree of membership for all element’s after it.

This imposes a semantic on the position of element’s in a fuzzy set, thus representing a fuzzy set as a fuzzy list, where the order has a meaning.

example: fuzzy set:  $\{ (x_1, 0.2), (x_2, 0.4), (x_3, 0.6), (x_4, 0.7), (x_5, 0.8) \}$



## analysis of methods and problem

an examination of the tasks involved, for both interpretation methods, follows:

Some basic performances:

- Time to relate  $N$  fuzzy sets, with each  $M$  elements:  $M^N$ . This is because every element of each antecedent attribute has to be related to every other element of each antecedent attribute.
- Memory for a related  $N$  fuzzy sets, with each  $M$  elements:  $M^N$ . This is because at most every combination of related element’s will have to be represented. In practice this is likely to be less as the result of relating attributes may make two adjacent attributes in the representation have the same degree of membership, and so the first element can be implicitly represented by the second.
- Time to perform a binary membership function over two fuzzy set  $A, B$ : at least  $|A| + |B|$  as the binary function will possibly have to be performed on each element  $A$  will be against an implicit element in  $B$ , and each element in  $B$  against an implicit element in  $A$ . However if the function is performed on an element in  $A$  against an implicit element in  $B$ , then this element will have the same degree of membership as the next explicitly represented element in  $B$ , so it isn’t required to be explicitly represented. So in theory only  $\max(|A|, |B|)$  functions need to be

performed.

- Memory for the result of a binary membership function over two fuzzy set A, B: the result of the function over each pair of elements will have to be recorded as the result, thus the memory requirements, are equal to the above stated operations: optimally  $\max(|A|, |B|)$ . however if a less intelligent implementation is used no more than:  $|A| + |B|$ .

Given:

$M_v$  = the memory usage of for a fuzzy value will be directly proportional to the number of represented points in fuzzy space.

$|A|$  = the number of antecedent attributes.

Assuming the consequent of the rule has fuzzy set: C, with size  $|C|$ .

$n$  = the size of set of all rules.

I = the input to a fuzzy rule,  $|I|$  = the size of the input set.

deducing a consequent from a rule will have:

1. time for relating inputs I:  $T_i = |I|^{|A|}$

2. time for performing the inference:  $T_p$ :

$T_p$  = time for:  $\max \{ \min \{ \text{binary function over the related input and the rule antecedent sets} \} \}$

$\Leftrightarrow T_p$  = time for:  $\max \{ \min \{ f(I, A) \} \}$ ; for each consequent element of the rule }

$\Rightarrow T_p$  = time for:  $|C|$  iterations of:  $\max \{ \min \text{bin\_func}(I, A) \}$

$\Rightarrow T_p$  = time for:  $|C|$  iterations of:  $\max \{ \max(|I|, |A|) \text{ operations} \}$

$\Rightarrow T_p$  = time for:  $|C|$  iterations of:  $\max(|I|, |A|)$  operations.

$\Rightarrow T_p = |C| \cdot \max(|I|, |A|)$

Thus total time for performing a rule will be:  $|I|^{|A|} + |C| \cdot \max(|I|, |A|)$

The process of changing the rules into their fuzzy set representation is referred to in the document as knowledge compilation. For a rule set of size  $n$ , compiling the fuzzy rule set would have the expected time usage to compute the fuzzy set representation for each rule:

$n \cdot |A|^{M_v}$

Combining the output from each rule:

Assuming all consequent have the same size consequent set:

$|C|$  = the number of elements of the derived consequent fuzzy set from each rule, thus given  $n$  rules, there the function to combine the consequent will have to perform  $n \cdot |C|$  operations.

Integrating a set of rules to form an integrated rule:

given each rule has a fuzzy set of size  $|A|^{M_v}$

there are  $n$  rules to be related. All rules relate the same set of  $D$  dimensions. Note that  $D = |A|$ .

Assuming that each rule contains representations for exactly the same set of elements, then the function is performed on  $n$  sets of  $|A|^{M_v}$ , therefore  $n \cdot |A|^{M_v}$  operations are required, and also  $n \cdot |A|^{M_v}$  memory requirements. However in practice the rules will be relating elements from different spaces of the domain's, so this will be higher.

A fuzzy inference procedure assumes that the input's are already in the form of fuzzy sets, so measurement for fuzzification will not be examined. It will not be assumed that the input's have already been related such that they fit the antecedents of the rule's. It's not clear whether to assume or not, that the rule's have already been represented in fuzzy sets. This issue is even less clear when it comes to assessing inference with an overall interpreted relation. As such the estimated performance for knowledge compilation has been kept separate from the estimated performance of the inference.

Given the above analysis of the approximate order of the time and memory requirements, an approximation of the order for the time and memory performance can be deduced, providing a hypothesis for testing.

inference via locally interpreted relations:

- knowledge compilation:

time: compiling rules:  $n \cdot |A|^{M_v}$

memory: compiled rule set:  $n \cdot |A|^{M_v}$

- inference from input:

time: number of rules • inference for a rule + time to combine consequent.

$((n) \cdot (|I|^{|A|} + |C| \cdot \max(|I|, |A|))) + (n \cdot |C|)$

memory: memory to hold rules + memory to hold output:

$$n \cdot |A|^{Mv} + |C|$$

Inference with overall interpreted relation:

- knowledge compilation:
  - time: compiling rules + integrating rules:  $n \cdot |A|^{Mv} + n \cdot |A|^{Mv}$
  - memory: compiled rule set + integrated rule:  $n \cdot |A|^{Mv} + n \cdot |A|^{Mv}$
- inference from input:
  - time: inference on integrated rule (note, time to relate the input is:  $(n \cdot |I|^{|A|})$ )  
 $((n \cdot |I|^{|A|}) + |C| \cdot \max(|I|, |A|))$ .
  - memory: memory to integrated rule + memory to hold output:  
 $|A|^{Mv} + |C|$

However, in practice if a rule set contain's rule's which implicitly represent unspecified dimensions, then the  $\max(|I|, |A|)$  for the inference via a locally interpreted relation, will decrease as  $|I|$  and  $|A|$  are smaller. For if all the Inference with overall interpreted relation, if all the antecedent of the rules are in the same domain of discourse, then the time taken to related the input is not  $(n \cdot |I|^{|A|})$ , but instead  $|I|^{|A|}$ , as relating the input's for each rule with 'or' will give  $I$  or  $I$  or ... or  $I = I$ .

## variables

The variables effecting the methods are:

- the number of rules in the set of all rules.
- the number of inputs to be derived from.
- the size of input fuzzy sets.
- the representation of fuzzy sets
- the representation of multi-dimensional fuzzy sets.

The performance should be measured in for these variables.

## Proof of mathematical equality

Both interpretation methods provide the same result given the same input, they are mathematically equivalent. The proof follows:

- Given:  $R$  = a rule in the set of all rules.  
 $IR$  = the integrated rule.  
 $IR_{CX}$  = the degree of membership of element X in the integrated rule IR consequent.  
 $IR_{AJ}$  = the degree of membership of element J in the integrated rule IR antecedent.  
 $R_{CX}$  = the degree of membership of element X in the rule R consequent.  
 $R_{AJ}$  = the degree of membership of element J in the rule R antecedent.  
 $I_Y$  = the degree of membership of an element Y in the related input set I.  
 $D(Z)$  = the domain of discourse of Z.  
 $C_X$  = the degree of membership of the derived consequent element X.

for: Inference with overall interpreted relation

$$C_X = \max \{ \min \{ \max \{ R_{CX}; R \}, \{ I_Y, \max \{ R_{AJ}; R \}; Y \text{ st } D(Y) = D(J) \} \} \}$$

$$\Leftrightarrow C_X = \max \{ \min \{ \max \{ R_{CX} \}, \{ I_Y, \max \{ R_{AJ} \}; Y \text{ st } D(Y) = D(J) \} \}; R \}$$

for: Inference via locally interpreted relations

$$C_X = \max \{ \max \{ \min \{ R_{CX}, \{ I_Y, R_{AJ}; Y \text{ st } D(Y) = D(J) \} \}; R \}$$

$$\Leftrightarrow C_X = \max \{ \min \{ R_{CX}, \{ I_Y, R_{AJ}; Y \text{ st } D(Y) = D(J) \} \}; R \}$$

$$R_{CX}; R \quad \max \{ R_{CX}; R \}$$

$$R_{AJ}; R \quad \max \{ R_{AJ}; R \}$$

=>

$$\{ \min \{ R_{CX}, \{ I_Y, R_{AJ}; Y \text{ st } D(Y) = D(J) \} \}; R \}$$

$$\{ \min \{ \max \{ R_{CX} \}, \{ I_Y, \max \{ R_{AJ} \}; Y \text{ st } D(Y) = D(J) \} \}; R \}$$

however, because  $\max \{ R_{AJ}; R \}$ , is the maximum of  $\{ R_{AJ}; R \}$ , and  $\max \{ R_{CX}; R \}$  maximum of  $R_{CX}$

$$\max\{ \min \{ R_{CX}, \{ I_Y, R_{AJ}; Y \text{ st } D(Y) = D(J) \} \}; R \} =$$

$$\max\{ \min \{ \max\{ R_{CX} \}, \{ I_Y, \max \{ R_{AJ} \}; Y \text{ st } D(Y) = D(J) \} \}; R \}$$

=>  $C_x$  for Inference with overall interpreted relation =  
 $C_x$  for Inference via locally interpreted relations

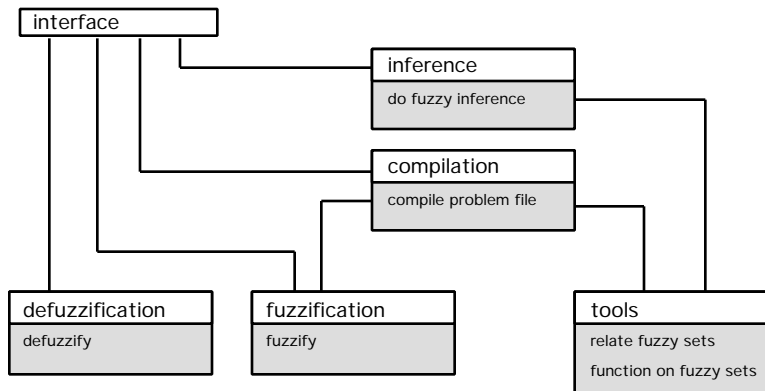
# Design

The program designed to test the interpretation methods for fuzzy inference was designed so that it could be extended and used to do further tests and comparison on aspects of fuzzy logic. As such it was designed to be modular.

The the 'load.pl' includes directives to load all the modules, and prints a simple message about the predicate to start the interface, the 'go' predicate.

The program was written for SWI Prolog (run in Unix using 'pl' command, and uses standard Edinburgh syntax). SWI Prolog is free and under Unix if more user friendly than sicstus (allows use of arrows to select and edit previous commands), it has some minor differences in system calls, but is otherwise, very similar to Sicstus. SWI Prolog is available at: <ftp://SWI.psy.uva.nl/pub/SWI-Prolog/>

The program usage structure is as follows:



## data structures & representation

Important to every module is the data structure used to represent fuzzy sets, rules, values. It was decided that the implicit plateau representation form would provide the best solution in terms of the performance as it can represent large spaces with a single value. The other advantage of this is that edge effects don't exist, and no artificial limit on domain space must be imposed, although it could be deemed useful. The removal of the artificial limits on domain space removes with it problems of fuzzy values appearing completely outside the limit, and hence not being represented.

(note: Degree of membership is sometimes referred to as Certainty.)

The grammar defining Fuzzy Sets under this implementation is:

```
FuzzySet      -->  d( DomainSpec, [ ValueList ] )
FuzzySubSet   -->  d( DomainSpec, [ ValueList ] )
ValueList     -->  Value
               |   Value, ValueList
Value         -->  v( Element, Certainty )
               |   v( Element, FuzzySubSet )
DomainSpec    -->  <a term defining the domain>
Element       -->  <a term representing an element>
Certainty     -->  <a real number in the interval [0, 1]>
```

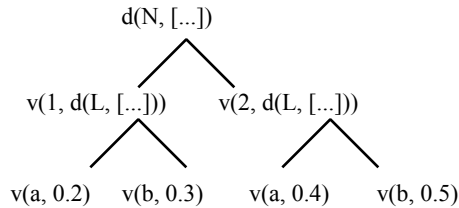
See below for an example.

Important characteristics of this representation:

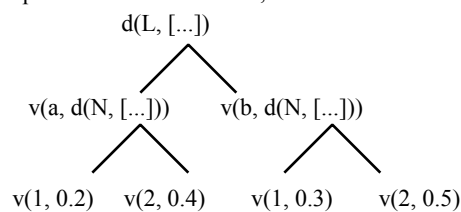
- This representation describes a fuzzy domain as sorted a tree structure. For example consider:

Program representation of the multi-dimensional fuzzy set:  
 $\{ ((1,a),0.2), ((2,a),0.4), ((1,b),0.3), ((2,b),0.5) \}$   
 With (x,y) representing elements from the domain's of discourse N, L respectively.

Representation with root: N, sub dimension L



Representation with root: L, sub dimension N



Because of the tree representation there is more than one way to represent the same information. The advantage of representing the information in a tree form is that as the number of dimensions (domain's of discourse) increase, instead of the search time in a list increasing exponentially as the number of element's increases exponentially, the tree form allows search time to increase logarithmically. The disadvantage is that it is more complex to perform operations on a tree than on a list. However this problem has been largely overcome in the program by providing low level manipulation tools of fuzzy sets.

- Because implicit plateau representation was used, the ordering of elements is semantically important, and must be preserved.

## the interface

Public predicates in this module:

- go/0 starts the interface with no problem file opened.
- go/1 starts the interface and attempts to open the problem file specified by the argument.

The interface is a high level usage tool for linking the different modules together. It makes an assumption about the usage of the program as a whole: that in general a problem will be specified in a file defining domain's of discourse, rules and values. Once a problem file has been specified inference will be performed giving fuzzy output values that are then defuzzified. However while this is the assumption made in the program at this point, it is a trivial task to add high level functions to the interface, although it does require some programming in Prolog.

The interface is implemented using a recursive event loop, which exists upon either 'q' or '^d' (control-d, the end of file input character, as standard in Unix). The interface also has some error handling, which although not very advanced is of some use. Any command which is in the wrong form failed with an error message, but doesn't cause the program to crash.

One feature of the implementation of the interface is that it uses the Prolog cut (!) operator to manage memory usage. The cut is used after every successful interface operation, to stop Prolog from keeping the previous command and the commands details in the choice local stack. If this was not done, memory would never be freed and the program would soon run out after a few commands.

The interface includes functions for performing sets and ranges of inference. These are fairly sophisticated in that a range for a number of sets of input domain's can be specified, and the interface will computer every permutation of the input's and provide a table of the results. This can be done for any number of dimensions of inputs.

The other fairly substantial feature of the interface is the 'prityprint' predicate that prints out in a table form multi-dimensional fuzzy sets, fuzzy rules, and lists of either of these.

The interface also performs measurement of time and memory usage for all lower level operations, thus making the comparison's of performance in these criteria easy.

## fuzzification

The public predicates for this module are:

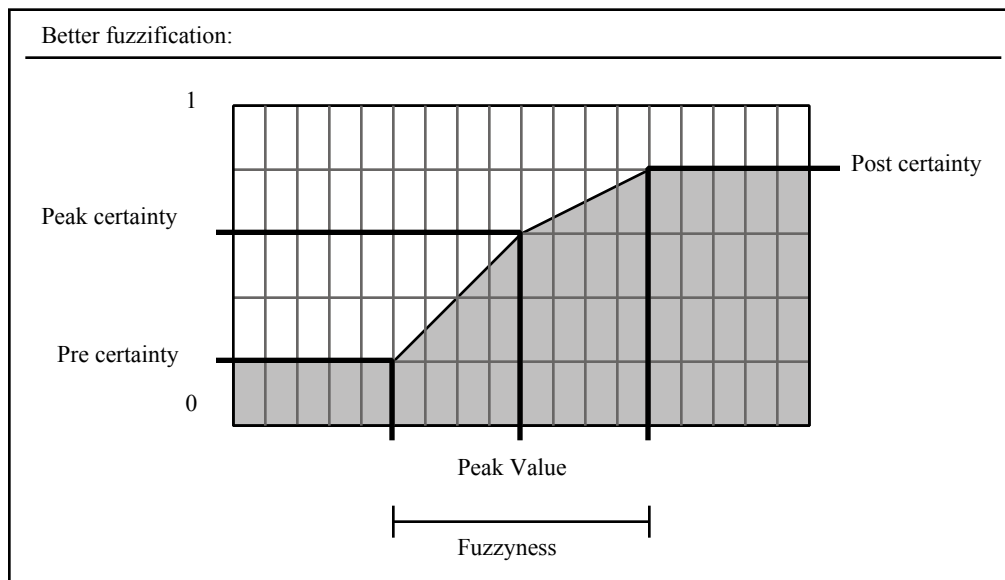
- fuzzify/2 - the most basic fuzzification interface, uses simple fuzzification, first argument is value to fuzzify (in input form), second argument is fuzzified value.
- fuzzify/3 - high level general fuzzification, specify fuzzification type, input, and predicate provides output fuzzified value.
- fuzzify\_list/3 - as fuzzify/3, except that input and output are lists.
- dofuzzyfy/6 - low level usage of crappy fuzz
- dofuzzyfy/8 - low level of betterfuzz

The fuzzification module implements two types of fuzzification, one of which is simpler and a subset of the other. The fuzzification function also has a level of normalization built in, a scale value is passed that defines the minimum interval between discrete representations of the dimension being represented.

The more complex fuzzification method is 'betterfuzz', it performs a fuzzification from:

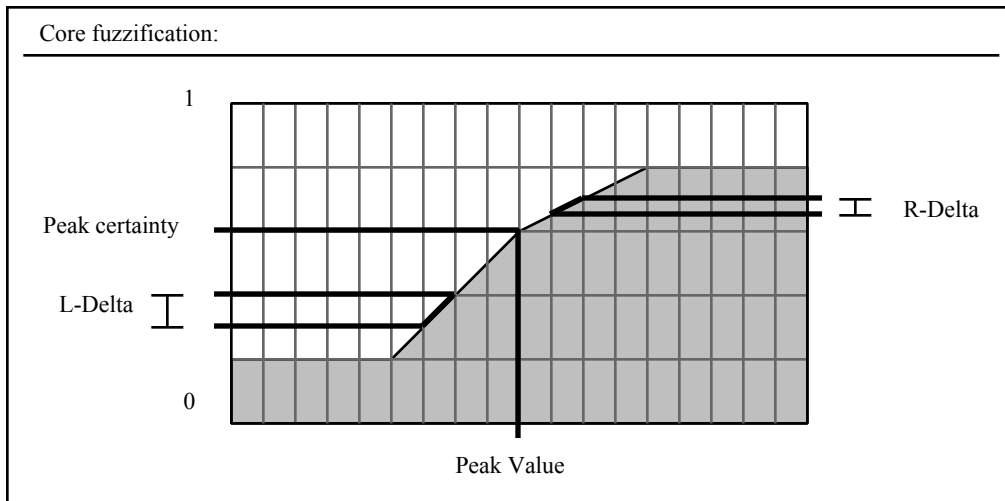
- peak degree of membership (peak certainty)
- peak element (peak value)
- fuzziness
- post certainty
- pre certainty
- scale

Scale is the scale value as described above for normalization. betterfuzz first normalizes the fuzziness and peak element to the discrete fuzzy dimension, then computes:



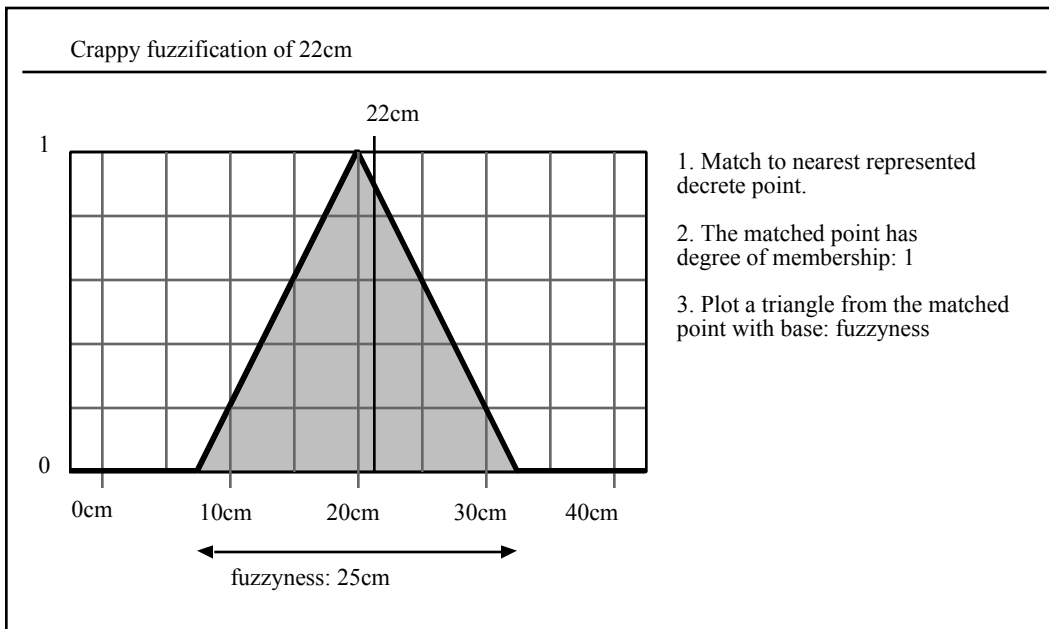
two lines, both from (peakvalue, peak certainty) for distance, one to (peakvalue + Fuzziness / 2, post certainty), the other to (peakvalue - Fuzziness / 2, pre certainty). This allows representation of triangle fuzzy functions, as well as plateaus of certainty.

Betterfuzz fuzzification function is computed using a more basic fuzzification function:



Which takes a peak value and certainty with two gradients, one on the left, and one to the right. This is the most basic level of fuzzification coded. The reason for this low level is that other more interesting fuzzification functions can be made using these basic tools.

Crappy Fuzzification is the simpler, less versatile fuzzification coded using the lower levels.



Crappy fuzzification is implemented by using betterfuzz called after matching the element to fuzzify to the nearest discrete representation. betterfuzz is passed: post certainty: 0, pre certainty: 0, peak certainty: 1, peak value: matched discrete point, the scale passed to crappy fuzz if also passed to better fuzz.

## defuzzification

The public predicates for this module are:

- defuzzy/4 - defuzzyify a fuzzy set, specify type of defuzzification, scale, fuzzy set, predicate attempt to instantiate last argument with the defuzzified value.
- defuzzy\_list/4 - as defuzzy/4, except that defuzzyify a set of fuzzy sets. Used for batch defuzzification.

defuzzification includes partial normalization: the scale value passed to defuzzification specifies the minimum interval between discrete representations of the dimension being represented. denormalisation is performed on the result of defuzzification prior to the returned result.

Two types of defuzzification are implemented:

Mean of maxima:

defuzzified element = sum of all maxima / number of maxima

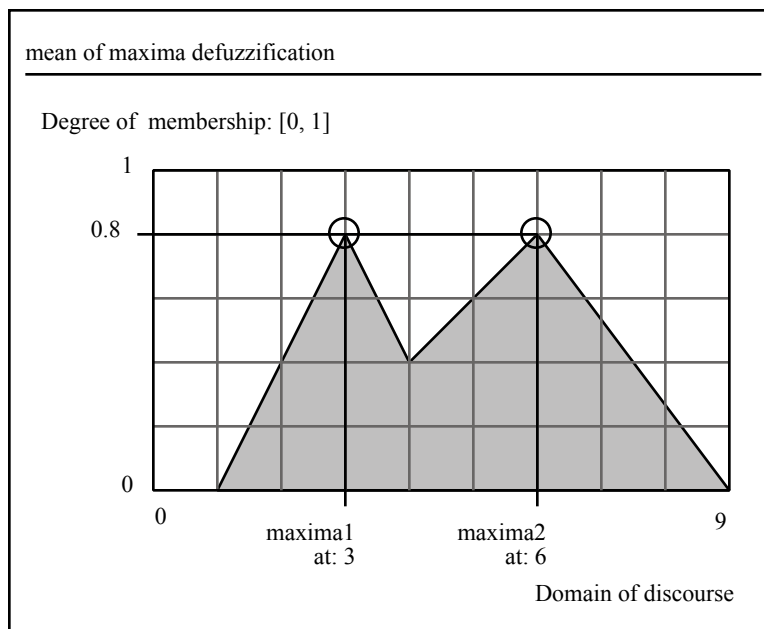
sum of all maxima = the sum of all points maxima points

number of maxima = the number of points maxima points

a maxima point = an element which shares the highest degree of membership

example: fuzzy set:

{ (0,0), (1, 0), (2, 0.4), (3, 0.8), (4, 0.4), (5, 0.6), (6, 0.8), (7, 0.575), (8, 0.225), (9, 0) }  
has two maxima: { (3, 0.8), (5, 0.8) }, the mean of maxima is: 4.



Center of gravity:

Given: degree of membership:  $M_x$  for an element:  $X$

defuzzified element =  $(\sum x \cdot M_x) / (\sum M_x)$  for all  $X$  in the fuzzy set.

Example: fuzzy set: as above.

has center of gravity:

$$\begin{aligned} & (2 \cdot 0.4 + 3 \cdot 0.8 + 4 \cdot 0.4 + 5 \cdot 0.6 + 6 \cdot 0.8 + 7 \cdot 0.575 + 8 \cdot 0.225) / (0.4 + 0.8 + 0.4 + 0.6 + 0.8 + 0.575 + 0.225) \\ & = 18.425 / 3.4 \\ & = 5.4 \end{aligned}$$

## tools

The tools module provides basic functions useful in dealing with fuzzy sets.

the public predicates are:

- `fuzzy_rel/4` - form: `fuzzy_rel(Set1, Set2, Function, Result)` - computes the fuzzy relation using the function 'Function' over the two input Sets, 'Set1' and 'Set2', 'Result' is the related fuzzy set. 'Function' can currently be either max or min, however any binary function on the degree's of membership can be added.
- `binfunc_fval/4` - form: `binfunc_fval(Set1, Set2, Function, Result)` - computes the resultant set of performing a binary function 'Function' on every member of Set1 and Set2, to produce the Result set. Function can currently be either min, or max. however any binary function on the degree's of membership can be added.
- `binsum_func_fval/4` - form: `binsum_func_fval(Set1, Set2, Function, Result)` - performs a summary function on the two sets, Set1, and Set2. Result is the result of the summary. A summary function is a function that takes the previous summary, the next value, and creates the next summary. Currently the only summary function is: `max_of_min`, which computes the maximum of the minimum binary function of the membership of the corresponding elements from the two sets. This is used in fuzzy inference.

Within the tool's source code file, it can be specified to either use optimized, or non optimized fuzzy relation. The optimized fuzzy relation minimizes the result set as it is generated, so that it is represented in the minimal implicit plateau form.

The exact logic of the algorithm is specified by the Prolog implementation's. The general definition of the Result of these three low level tools are:

### **fuzzy\_rel(Set1, Set2, Function, Result)**

Result is a dimension space with the dimension of Set2 as the nested second dimension.

An element at location: `Set1[i]`, `Set2[j]`, in Result has membership defined by the function specified by:

Function : `f( Membership_of(Set1[i]), Membership_of(Set2[j]) )`.

example: `fuzzy_rel( d(d1, [v(1, 0.2), v(2, 0.4)]), d(d2, [v(a, 0.3), v(b, 0.2)]), min, Result )`  
Result = `d(d1, [ v(1, d(d2, [v(a, 0.2), v(b, 0.2)])), v(2, d(d2, [v(a, 0.3), v(b, 0.2)])) ])`

### **binfunc\_fval(Set1, Set2, Function, Result)**

Result is a dimension space with the dimension of Set1, which is also the dimension of S2.

An element at location: `Result[i]` is defined by the function specified by : Function.

Function : `f( Membership_of(Set1[i]), Membership_of(Set2[j]) )`.

example: `binfunc_fval( d(d1, [v(1, 0.2), v(2, 0.4)]), d(d1, [v(1, 0.3), v(2, 0.2)]), min, Result )`  
Result = `d(d1, [ v(1, [v(1, 0.2), v(2, 0.2)]) ])`.

### **binsumfunc\_fval(Set1, Set2, Function, Result)**

Result is a single term specified by the summary function 'Function', which is initially called with undefined, and consequently with the previous summary and the membership of the elements currently being examined:

`Membership_of(Set1[i]), Membership_of(Set2[j])`.

example: `binsumfunc_fval( d(d1, [v(1, 0.2), v(2, 0.4)]), d(d1, [v(1, 0.3), v(2, 0.2)]), max_of_min, Result )`  
Result = 0.2

## knowledge compiler

Knowledge compilation allows a problem to be specified in a file, as a set of rules and values, which than then be compiled into fuzzy knowledge ready to be used in the inference process.

knowledge compiler has two public predicates:

`do_fuzzycompile/3` - compiles to the level specified by the first argument, the filename is the second argument, and the 3rd argument is instantiated to the compiled file.

`getcompiled_data/2` - is used to access the compiled file, the first argument is the attribute to access, with a free variable to be instantiated, the second argument is the a compiled file.

The following are specified in a compiled file:

- values, and information on how to crate appropriate fuzzy values.
- rules in terms of ‘and’, ‘or’, ‘implies’, ‘not’ and other values specified in the file.
- Domain’s of discourse, are specified by name, and have associated Scale attribute.

Fuzzy rules which are specified tree’s with terminal symbols as identifies to fuzzy values. Grammar for rules:

```
Rule      --> rrule( RuleSpec, Connectives ).
RuleSpec  --> < a term specifying the rule >
Connectives --> ValueSpec
           | or( Connectives, Connectives )
           | and( Connectives, Connectives )
           | imp( Connectives, Connectives )
           | not( Connectives )
ValueSpec --> < a term specifying the fuzzy value to use >
```

example:

```
rrule( rule1, imp( and( val1, val2 ), outval) ).
```

Rule’s can be built up from this grammar. The grammar has no ambiguity, as it defines the associativity and precedence of the connective's, by their use. Note that `rrule` simply defined a logical sentence, for the rule to be used in a derivation, the main connective (the first connective) must be ‘imp’.

The compilation process is fairly simple:

1. Fuzzify all the specified fuzzy values.
2. Create the fuzzy rules.

Creating the fuzzy rules from the the `rrule` specification is done by a depth first search and performing an appropriate relation at every connective as the search backtracks.

Values are specified as existing in a rule file by:

```
rvalue( ValueName ).
```

a value must also have a number of attributes associated so that it can be fuzzified to produce corresponding fuzzy value. These attributes are specified by:

```
rvalue_attr( ValueName, AttributeName, AttributeValue ).
```

The minimal required set of attributes are:

`domain_name` - which is the specifier of the domain of discourse that the value belong to.

`contdomainvalue` - is the element in the domain of discourse that is being fuzzified.

`confuzzyness` - is the fuzziness in the domain of discourse used to fuzzify the `contdomainvalue`.

Given these characteristics `CrappyFuzz` can be used to fuzzify the element.

The domain information in a problem file must also be specified by:

```
rdomain( Domainspecifier )
```

‘Domainspecifier’ is the specifier that is referred to in the ‘`domain_name`’ value attribute.

attributes of a domain are specified by:

```
rdomain_attr( Domainspecifier, AttributeName, AttributeValue )
```

Every domain that is fuzzified from or to must have appropriate normalization attributes. Currently only one normalization attribute exists which must be specified:

`scale_value` - which defined the scaling from the domain of discourse to the fuzzy domain.

## inference

The inference module consists of methods of inference and has 2 public predicates:

`finfer/4` - performs inference of type defined by the first argument, on fuzzy input set specified by the second argument, using compiled knowledge specified by the third argument. The fourth argument is the resultant fuzzy set.

`finfer_list/4` - performs `finfer/4` on every element in a list. Arguments are as `finfer/4`, with the second argument as a list of fuzzy input sets.

Current types of inference are:

`join_all_rules` - which generates the integrated rule, and then uses it with the input set to arrive at a consequent. This is an implementation of inference with overall interpreted relation. This is currently done by performing the binary function over every rule, this produces the consequent rule, which is then used for inference in the low level, private inference predicate: `do_fuzzy_infer/3`

`join_all_outputs` - an implementation of Inference via locally interpreted relations. This inference process uses the `all_rules` inference type to perform every inference on the input's with the rules, obtaining all possible consequent. These consequent are then combined using the binary function over fuzzy sets.

`all_rules` - performs fuzzy inference - performs every inference possible with a set of input's producing a set of fuzzy consequent values. This uses the `findall` predicate and the `some_rule` inference procedure.

`some_rule` - performs fuzzy inference on a member of the ruleset with the some match of the input.

There is some use of the cut ('!') in the inference process to minimize memory usage, this is done only for that purpose and has no effect on the result of the inference processes. The cut enables memory usage to be limited by allowing Prolog to dispose of backtracking information.

## future improvements of the program

There are a number of changes that would make the program more versatile. There are also a number of relatively easily fixable bugs.

### Improvements to the program:

- change the binary and summary operations to take instead of degree's of membership, the pair of the element, and it's degree of membership. This would extend the versatility of the basic tools, and would take only a short time to implement. An example of a use this would have is: enable defuzzification modules to be written as summary functions. the standard binary function on fuzzy sets could also be implemented as a summary function. This would require a small amount of recoding of the tools.pl file only.
- It would be useful in the knowledge compilation module to be able to specify the level and type of compilation. The framework for this is in place, however currently only one level of compilation is implemented.
- Add full normalization to the program, so that normalization information is specified in the problem file. This will require a small change to the knowledge compiler and fuzzification engine.
- Add other derivational tools. Currently inference is done using a derivation of MPP, it would be interesting to try encoding other derivations, like & introduction and & elimination. This would allow the engine to be used as a general purpose logical inference engine.

### Known Bugs:

- When performing fuzzy inference (deriving a consequent), the input is not correctly related if any connective other than 'and' is used. This would require some rewriting of the knowledge compiler to include information in a rule about the nature of the antecedent domain of discourse, and an interpretation of this in the fuzzy inference code.
- Relation's don't work on fuzzy sets with a common domain of discourse. To fix this an amendment would have to be made to the `fuzzy_rel` predicate, and the catch to stop this from being tried at present in the fuzzy inference code would have to be removed. This would also only require a small alteration.
- The binary function on fuzzy sets: `binfunc_fvals/4`: doesn't currently work if the two input sets have different ordering of sub-dimensions, even if both sets are in the same domain of discourse. fixing this would require only a change to the `binfunc_fvals/4` predicate in tools.pl and would have no adverse effects on any other part of the program. This is the reason why, when the antecedent attributes are specified in different orders, integration of rules fails.

# program testing

## methodology

The theory and definition of a hypothesis has already been defined. The hypothesis:

inference via locally interpreted relations:

- knowledge compilation:
  - time: compiling rules:  $n \cdot |A|^{Mv}$
  - memory: compiled rule set:  $n \cdot |A|^{Mv}$
- inference from input:
  - time: number of rules • inference for a rule + time to combine consequent.  
 $((n) \cdot (|I|^{|A|} + |C| \cdot \max(|I|, |A|))) + (n \cdot |C|)$
  - memory: memory to hold rules + memory to hold output:  
 $n \cdot |A|^{Mv} + |C|$

Inference with overall interpreted relation:

- knowledge compilation:
  - time: compiling rules + integrating rules:  $n \cdot |A|^{Mv} + n \cdot |A|^{Mv}$
  - memory: compiled rule set + integrated rule:  $n \cdot |A|^{Mv} + n \cdot |A|^{Mv}$
- inference from input:
  - time: inference on integrated rule (note, time to relate the input is:  $(n \cdot |I|^{|A|})$ )  
 $((n \cdot |I|^{|A|}) + |C| \cdot \max(|I|, |A|))$
  - memory: memory to integrated rule + memory to hold output:  
 $|A|^{Mv} + |C|$

Because testing this for all variables would require a large amount of time, only one variable has been examined: n: The number of rules. Because all other attributes are kept constant, the time for knowledge compilation for both inference methods, should increase by a constant gradient in a straight line. For inference testing was done without the overall interpreted relation being precompiled. Given enough time all attributes should be tested, with all levels of compilation, unfortunately time did not permit this.

A set of files was created with, each file contained exactly the same number of dimension's (3) and fuzzy values (3), each fuzzy value had the same size explicit representation (10 elements).

Justification: All variables other than the one being tested must be kept constant. the dimension's and initial characteristics should also be near the range problem's exist in, and also give answers in reasonable time. It was these considerations that were the basis for selecting constant attributes.

Each rule was exactly the same:

implies( and( values1 in Dimension 1 and value2 in dimension2 ), value 3 in dimension 3).

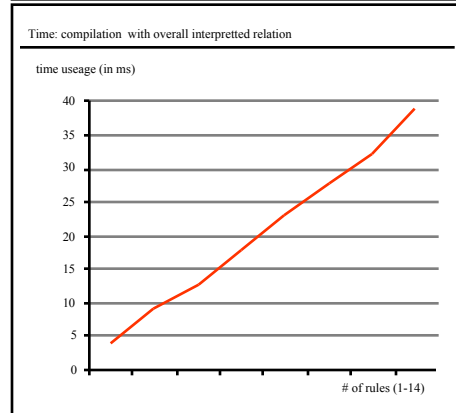
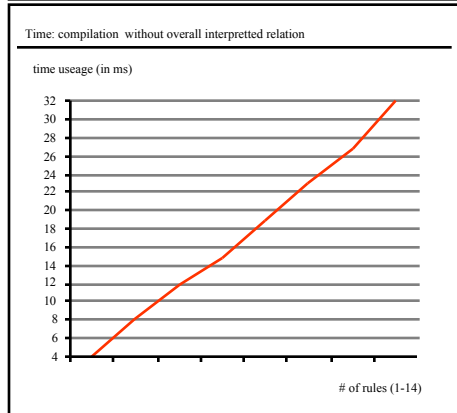
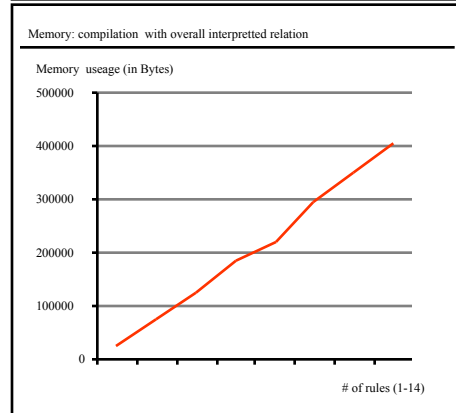
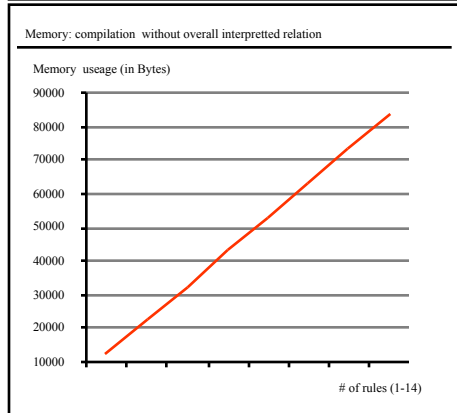
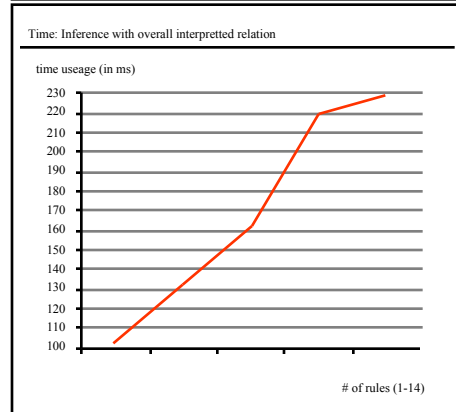
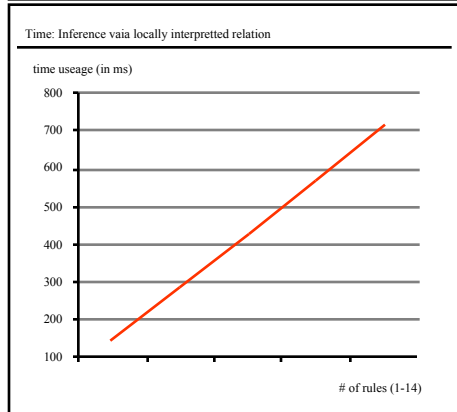
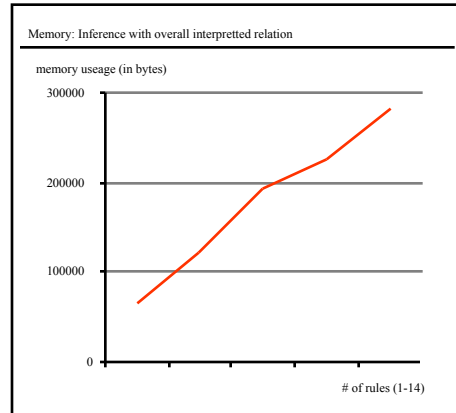
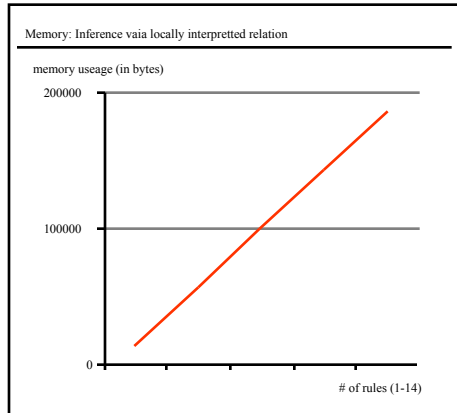
This was done so that each rule has the same antecedent set, and each added rule will have the same effect.

The tests were performed from 1 rule, to 14 rules.

A constant input was defined of( {values1, values2}, with fuzziness 10, thus covering exactly the same area as the rule's. Doing this eliminates any effect's of implicit representation used.

The inference was then performed 100 time's, and the mean measurement taken to eliminate the effect's of UNIX's time sharing between processes. Also in aid of reducing the effect of UNIX's time distribution between processes the test were run a workstation not connected to a network, running minimal background tasks. The Tests were performed on a PPC G3/233Mghz, running LinuxPPC5.

# results



## conclusions

There are two particularly interesting events in these results:

1• Inference with an overall interpreted relation has bend in the graph at about 8 rules, or equivalently 200KB. Likewise for knowledge compilation at about 200KB, a change occurs both in the memory usage, and the time taken. The 'bends' are not particularly large, but it is noticeable. This was unexpected, and not part of the hypothesis.

At first it was assumed that the change in timing was caused by a background activity, however upon rerunning this test with, the same result was arrived at. However this is unlikely to be the cause as it doesn't explain why the gradient of the memory usage decreased.

A factor that had not been considered in the testing was the effect of Prolog's built in garbage collection, this is done when memory can be disposed of, and tends to be quite slow. The effect of this on the program's time measurements is that as the memory usage decreases as Prolog frees memory, however the time increases as garbage collection is a relatively slow task.

Garbage collection seems to explain the results, and also provides a reason why the same result appears at the same point in knowledge compilation. This is because Prolog attempts to free memory at the point when memory is getting low. Another convincing piece of evidence is that Prolog's initial memory heap was set to 200KB.

However to be certain of this the initial heap size should be changed and the experiment re-performed.

2• Inference with the overall interpreted relation, was significantly quicker than without, which is contradictory too the hypothesis.

This is a little confusing as the theory suggests that because the rule would require the input to be related and interpreted in the form of each rule's antecedent, thus making the time increase inference with an overall interpreted relation slower, given more explicitly represented elements in the rule's antecedent than the consequent. This is the case in the problem file. However after an examination of the inference code, and a reexamination of the unimplemented addition still to be made, it can be seen that this is due to the way the input is related interpreted from the rules.

Because all the rules have the same antecedent domain of discourse, the input need's only to be related once. Thus it is as if the input is being run on only a single rule, not an integrated rule. This changes the time function for inference with an overall integrated relation to:

$$n \cdot |A|^{Mv} + n \cdot |A|^{Mv} + |I|^{|A|} + |C| \cdot \max(|I|, |A|).$$

And given that  $|A|$  and  $|I|$  are constant, time  $n \cdot 2(|A|^{Mv}) + |I|^{|A|} + |C| \cdot \max(|I|, |A|)$

And given that the time for inference via locally interpreted relation  $((n) \cdot (|I|^{|A|} + |C| \cdot \max(|I|, |A|))) + (n \cdot |C|)$ , and that  $|C| = 10$ , and  $|A| = 2$ , and  $|I| = 10$ ,

inference via locally interpreted relation  $n \cdot (10 + 10 * 10) + n \cdot 10 = 120n$   
and inference with an overall integrated relation to:  $n \cdot 20 + 10 + 10 * 10. = 20n + 110.$

This rough estimate seems to explain the results fairly closely. However a much more informative result would be to perform formal big-O representation for the function and derive a more accurate estimate for time and memory

Memory usage on the whole was as expected, compilation of an integrated relation used more memory than not doing so.

Some problems with the tests: the testing of memory was done with statistics(globalused) which is not very accurate and is susceptible to garbage collection by Prolog, as was found in the experiments. Again a formal estimation of memory usage could be given by a mathematical analysis of the forms of inference.

# traffic intersection control : a case for fuzzy inference

The traffic intersection control problem has been represented as: an input file for the fuzzy logic engine designed above:

```
%-----
% - Traffic intersection control: a case for fuzzy inference
%-----

%-----
% - The domains that exist
rdomain( d_queuelength ).
rdomain( d_arrivalrate ).
rdomain( d_out_time ).

%-----
% - Fuzzy Domains & Attr
fdomain( d_queuelength ).
fdomain( d_arrivalrate ).
fdomain( d_out_time ).

fdomain_attr( d_queuelength, scaling_value, 1 ).
fdomain_attr( d_arrivalrate, scaling_value, 1 ).
fdomain_attr( d_out_time, scaling_value, 1 ).

%-----
% - Values
% - Input Values for queue Length
fvalue( in_q_v_small, d(d_queuelength, [v(0, 1), v(1, 1), v(2, 0.7), v(3, 0.3),
v(4, 0)] ) ).
fvalue( in_q_small,
    d(d_queuelength, [v(0, 0), v(2, 0.3), v(3, 0.7), v(4, 1), v(5, 0.7), v(6, 0.3),
v(7, 0)] ) ).
fvalue( in_q_medium,
    d(d_queuelength, [v(4, 0), v(5, 0.3), v(6, 0.7), v(7, 1), v(8, 0.7), v(9, 0.3),
v(10, 0)] ) ).
fvalue( in_q_long,
    d(d_queuelength, [v(7, 0), v(8, 0.3), v(9, 0.7), v(10, 1)] ) ).

% - values for arrival rate
fvalue( in_a_almost_none,
    d(d_arrivalrate, [v(0, 1), v(1, 1), v(2, 0.7), v(3, 0.3), v(4, 0)] ) ).
fvalue( in_a_few,
    d(d_arrivalrate, [v(0, 0), v(2, 0.3), v(3, 0.7), v(4, 1), v(5, 0.7), v(6, 0.3),
v(7, 0)] ) ).
fvalue( in_a_many,
    d(d_arrivalrate, [v(4, 0), v(5, 0.3), v(6, 0.7), v(7, 1), v(8, 0.7), v(9, 0.3),
v(10, 0)] ) ).
fvalue( in_a_toomany,
    d(d_arrivalrate, [v(7, 0), v(8, 0.3), v(9, 0.7), v(10, 1)] ) ).

% - values for output wait time
fvalue( o_v_short,
    d(d_out_time, [v(0, 1), v(1, 1), v(2, 0.7), v(3, 0.3), v(4, 0)] ) ).
fvalue( o_short,
    d(d_out_time, [v(0, 0), v(2, 0.3), v(3, 0.7), v(4, 1), v(5, 0.7), v(6, 0.3), v(7, 0)]
) ).
fvalue( o_medium,
    d(d_out_time, [v(4, 0), v(5, 0.3), v(6, 0.7), v(7, 1), v(8, 0.7), v(9, 0.3),
v(10, 0)] ) ).
```

```

fvalue( o_long,
  d(d_out_time, [v(7, 0), v(8, 0.3), v(9, 0.7), v(10, 1)] ) ).

%-----
% - Rules
rrule( r11, imp( and( in_q_v_small, in_a_almost_none ), o_v_short) ).
rrule( r12, imp( and( in_q_small, in_a_almost_none ), o_v_short) ).
rrule( r13, imp( and( in_q_medium, in_a_almost_none ), o_v_short) ).
rrule( r14, imp( and( in_q_long, in_a_almost_none ), o_v_short) ).

rrule( r21, imp( and( in_q_v_small, in_a_few ), o_short) ).
rrule( r22, imp( and( in_q_small, in_a_few ), o_v_short) ).
rrule( r23, imp( and( in_q_medium, in_a_few ), o_v_short) ).
rrule( r24, imp( and( in_q_long, in_a_few ), o_v_short) ).

rrule( r31, imp( and( in_q_v_small, in_a_many ), o_medium) ).
rrule( r32, imp( and( in_q_small, in_a_many ), o_short) ).
rrule( r33, imp( and( in_q_medium, in_a_many ), o_v_short) ).
rrule( r34, imp( and( in_q_long, in_a_many ), o_v_short) ).

rrule( r41, imp( and( in_q_v_small, in_a_toomany ), o_long) ).
rrule( r42, imp( and( in_q_small, in_a_toomany ), o_medium) ).
rrule( r43, imp( and( in_q_medium, in_a_toomany ), o_short) ).
rrule( r44, imp( and( in_q_long, in_a_toomany ), o_v_short) ).

/* - testing code:

% -- inference method 2, defuzz 1
tableinf( 1, 2, 1, [input( d_queuelength, from(0,10), 4 ), input(
d_arrivalrate, from(0,10), 4 )] ).

% -- inference method 1, defuzz 1
tableinf( 1, 1, 1, [input( d_queuelength, from(0,10), 4 ), input(
d_arrivalrate, from(0,10), 4 )] ).

% -- inference method 2, defuzz 2
tableinf( 1, 2, 2, [input( d_queuelength, from(0,10), 4 ), input(
d_arrivalrate, from(0,10), 4 )] ).

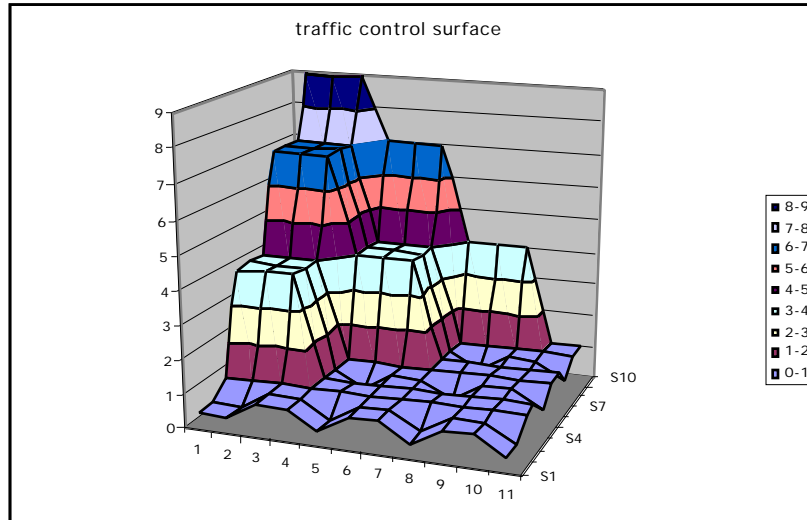
% -- inference method 1, defuzz 2
tableinf( 1, 1, 2, [input( d_queuelength, from(0,10), 4 ), input(
d_arrivalrate, from(0,10), 4 )] ).

*/

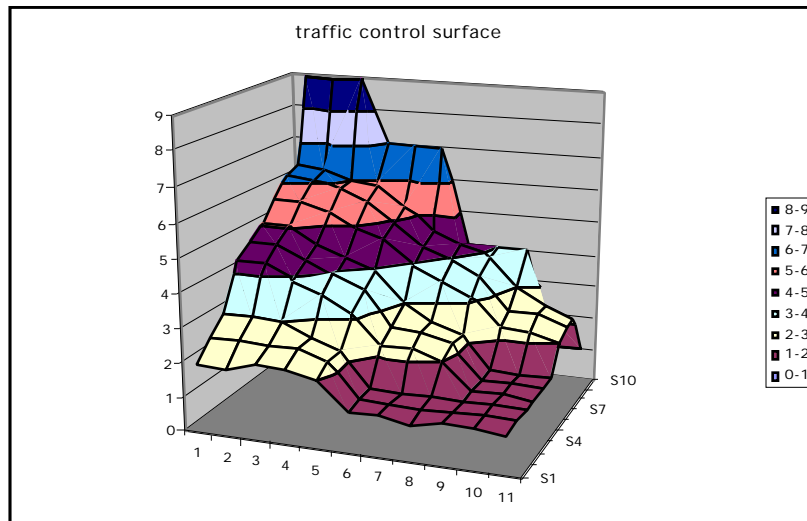
% -- end

```

The resulting control surface has been plotted for both implemented types of defuzzification:



Using mean of maxima defuzzification.



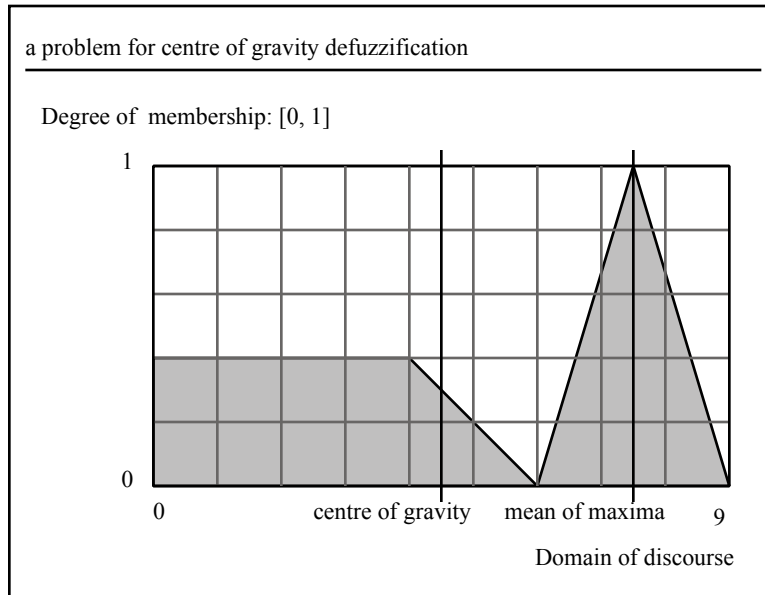
Using mean of maxima defuzzification

# A comparison of two methods of defuzzification: Center of Gravity and Mean of Maxima

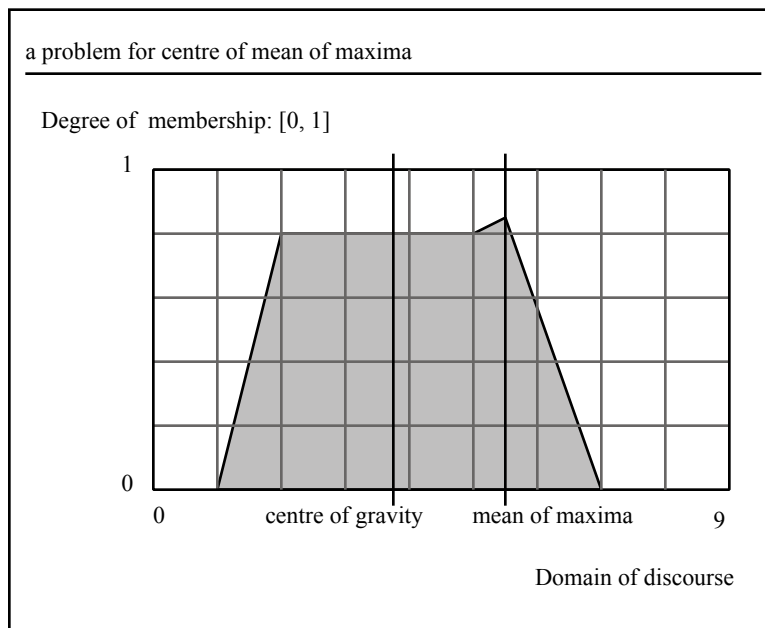
A comparison in terms of performance and nature of result's obtained from two basic types defuzzification: Center of Gravity and Mean of Maxima.

## nature of results

The nature of these two types of defuzzification differs, center of gravity takes into account small effect's of rules, and so is susceptible to noise. It would provide poor results is there was a large quantity of noise in a result, for example:

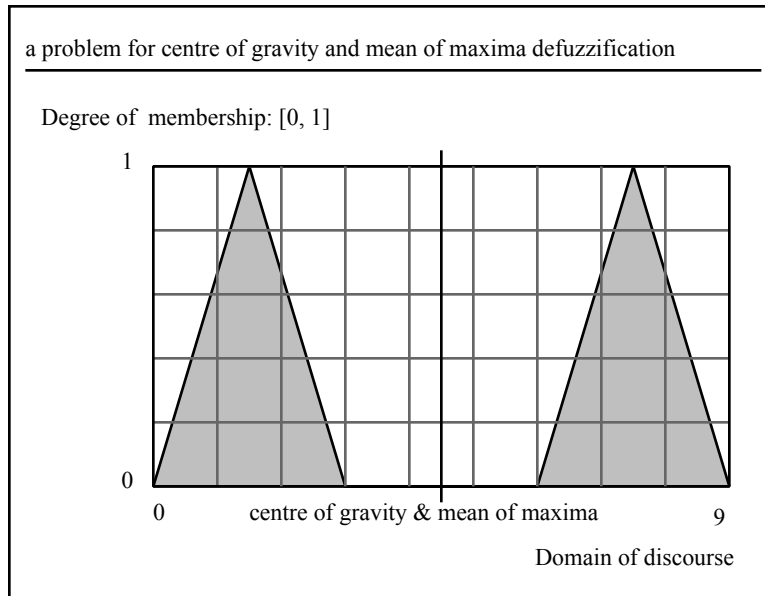


However Mean of maxima is susceptible to being distracted by a minor higher peak:



Some fuzzy sets are delt with very poorly by both meethods of defuzzification. In particular problems with distinct

peaks at distant intervals. For example:



There is no real need for testing the implementations of these forms of defuzzification as they are quite trivial and the results obvious. However from the examination of where the methods do poorly, it can be concluded that each form of defuzzification is more apt for a particular range of fuzzy sets. Center of gravity is most apt when there is small changes the peak heights, and where there is a little noise consequent degree's of membership. mean of maxima defuzzification is gives a better result when there is distinct peaks, and a tendency for general background noise, as it will select the distinct peaks.

# Bibliography

AI/CS3, AI/M3, AI/SE3 Large Practical 1999-2000: A comparison of the Interpretation Methods for fuzzy inference: Qiang Shen (1999)

Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I: Chuen Chien Lee (1990)

AI3 - KRI Lecture Notes by John Levine (1999)

Answers to Questions about Fuzzy Logic and Fuzzy Expert Systems:  
<http://www.cs.cmu.edu/Web/Groups/AI/html/faqs/ai/fuzzy/part1/faq.html>

Kosko, Bart, "Fuzzy Thinking: The New Science of Fuzzy Logic", Warner, 1993 [For technical details, see Kosko, Bart, "Fuzzy cognitive maps", International Journal of Man-Machine Studies 24:65-75, 1986.]

McNeill, Daniel, and Freiburger, Paul, "Fuzzy Logic: The Discovery of a Revolutionary Computer Technology", Simon and Schuster, 1992. ISBN 0-671-73843-7. [Mostly history, but many examples of applications.]

Brubaker, D.I., "Fuzzy-logic Basics: Intuitive Rules Replace Complex Math," EDN, June 18, 1992.