# FUZZY LOGIC LIBRARY 1.2 IN NETLOGO 5.2

Luis R. Izquierdo, Doina Olaru & Segismundo S. Izquierdo

## Outline of the document

# List of functions

**fuzzy-start**

## RETRIEVAL

**fuzzy-set-with-label** *label*

## EVALUATION

**fuzzy-evaluation** *x*

## PLOTTING

**fuzzy-plot** *set*

## DEFUZZIFICATION

**fuzzy-FOM**
**fuzzy-LOM**
**fuzzy-MOM**
**fuzzy-MeOM**
**fuzzy-COG**

## OPERATIONS WITH SETS

**fuzzy-min (= fuzzy-and)** *list-of-sets*
**fuzzy-max (= fuzzy-or)** *list-of-sets*
**fuzzy-sum** *list-of-sets*
**fuzzy-prob-or** *list-of-sets*
**fuzzy-not** *set*
**fuzzy-truncate** *set number*
**fuzzy-prod** *set number*

## HEDGES

**fuzzy-power** *set number*

## MEMORY

**fuzzy-die**

## CREATION OF FUZZY SETS FROM SCRATCH

**fuzzy-discrete-numeric-set** [[$x_1$ $y_1$] [$x_2$ $y_2$] ... [$x_n$ $y_n$]]
**fuzzy-discrete-numeric-set-with-label** *label* [[$x_1$ $y_1$] [$x_2$ $y_2$] ... [$x_n$ $y_n$]]

**fuzzy-piecewise-linear-set** [[$x_1$ $y_1$] [$x_2$ $y_2$] ... [$x_n$ $y_n$]]
**fuzzy-piecewise-linear-set-with-label** *label* [[$x_1$ $y_1$] [$x_2$ $y_2$] ... [$x_n$ $y_n$]]

**fuzzy-trapezoidal-set** [a b c d e f height]
**fuzzy-trapezoidal-set-with-label** *label* [a b c d e f height]

**fuzzy-logistic-set** [$x_0$ a b [lower-limit upper-limit]]
**fuzzy-logistic-set-with-label** *label* [$x_0$ a b [lower-limit upper-limit]]

**fuzzy-gaussian-set** [m s [lower-limit upper-limit]]
**fuzzy-gaussian-set-with-label** *label* [m s [lower-limit upper-limit]]

**fuzzy-exponential-set** [a b c [lower-limit upper-limit]]
**fuzzy-exponential-set-with-label** *label* [a b c [lower-limit upper-limit]]

**fuzzy-interval-with-points-set** [ [[a b] v] [[$x_1$ $y_1$] [$x_2$ $y_2$] ... [$x_n$ $y_n$]] ]
**fuzzy-interval-with-points-set-with-label** *lbl* [ [[a b] v] [[$x_1$ $y_1$]... [$x_n$ $y_n$]] ]

## RULES

**fuzzy-truncate-rule (= fuzzy-rule)** *list-of-2-items consequent-fuzzy-set*
**fuzzy-prod-rule** *list-of-2-items consequent-fuzzy-set*
**fuzzy-min-truncate-rule (= fuzzy-and-rule)** *list-of-2-item-lists consequent-fuzzy-set*
**fuzzy-max-truncate-rule (= fuzzy-or-rule)** *list-of-2-item-lists consequent-fuzzy-set*
**fuzzy-min-prod-rule** *list-of-2-item-lists consequent-fuzzy-set*
**fuzzy-max-prod-rule** *list-of-2-item-lists consequent-fuzzy-set*

# Basic introduction to fuzzy sets

Fuzzy sets are generalizations of conventional sets (also called "crisp" sets).

Note that a conventional (crisp) set *S* can be defined by a Membership Function $MF_S(x)$ that specifies, for each possible element *x* in the universe *X* (or space of all objects), whether the element belongs to *S* or not. If an element *x* belongs to *S*, then the membership function of set *S* applied to element *x* equals 1. If an element *x* does not belong to *S*, then the membership function of set *S* applied to element *x* equals 0.

$$MF_S(x) = \begin{cases} 1 & if\ x \in S \\ 0 & if\ x \notin S \end{cases}$$

For instance, the crisp set *EvenNumbers* contains the elements {2, 4, 6, …}, so its membership function equals 1 when applied to each of such numbers (i.e. $MF_{EvenNumbers}(2) = 1$). By contrast, the membership function of the crisp set *EvenNumbers* equals 0 when applied to numbers such as 1, 3, or 5 (i.e. $MF_{EvenNumbers}(5) = 0$).

The sharp dichotomy in the concept of "belonging" in conventional sets (i.e. an element either belongs to the set, or it does not) is relaxed with fuzzy sets, which allow for partial membership. This can be useful to define and reason with fuzzy concepts, such as e.g. "Tall". We may be reluctant to specify a precise and definite threshold value *h* such that every adult person with a height below *h* is "not tall" and every adult person with a height greater or equal to *h* is "tall". This would imply, in particular, that a person with a height just below *h* would be definitely "not tall", but someone only 1 mm higher would be definitely "tall".

Even though our concept of "Tall" does not seem to be perfectly precise, we may still find the concept useful in logical reasoning, e.g. from the general premise

"*Parents that are tall tend to have children who will be tall*"

and the specific premise

"*Mary and John are tall*",

we may conclude that

"*Mary and John's child will likely be tall*".

Note that we, humans, have developed a remarkable (and very useful) tolerance for imprecision. We seem to feel fairly comfortable dealing with rules, instructions and logical statements defined with fuzzy concepts (e.g. "If you drive very fast, your fuel consumption will be high"); however, computers using Boolean logic may not find it so straightforward. Using fuzzy logic, we can use computers to model and exploit this type of fuzzy reasoning.

Fuzzy sets are implemented by allowing membership functions to give intermediate values in between 0 and 1. Thus, the membership function MF of a fuzzy set assigns to each element *x* of a certain universe *X* a degree of membership $MF(x) \in [0,1]$. This mapping is usually denoted:

$$MF(x): X \rightarrow [0,1]$$

As an example, one could define the fuzzy set *Tall*, with a membership function whose argument is a number $x \in [0,250]$ representing height measured in cm (see e.g. Fig. 1). The universe *X* (or space of all objects) in this example would be the continuous interval [0,250].[1]

[1] Remember that the *illusion* of continuity is implemented in most computer platforms (NetLogo, in particular) using floating point numbers (see http://ccl.northwestern.edu/netlogo/docs/programming.html#math).

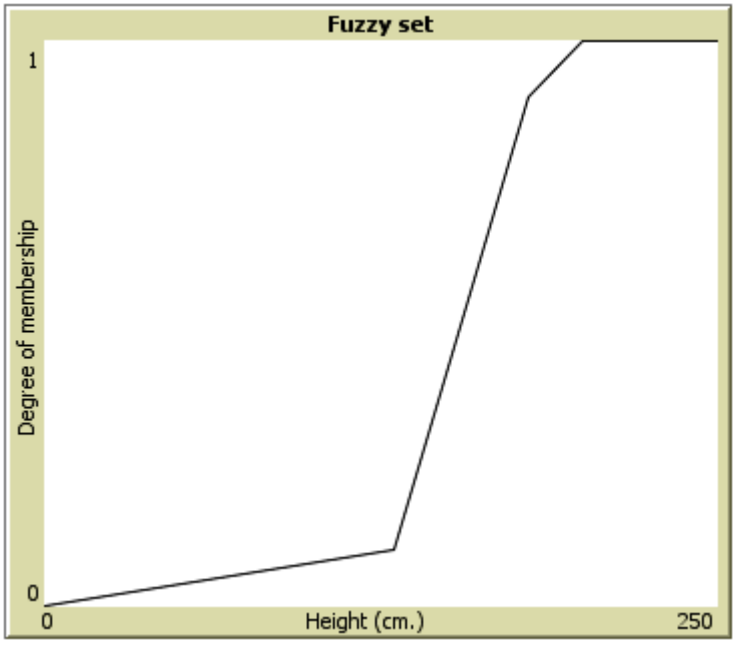


**Fig. 1. Membership function of a fuzzy set created running the code:**
**fuzzy-plot fuzzy-piecewise-linear-set [[0 0] [130 0.1] [180 0.9] [200 1] [250 1]]**

The fuzzy set *Tall* shown in Fig. 1 would assign a degree of membership 0 to height 0 cm, a degree of membership 0.1 to height 130 cm, and a degree of membership 0.82 to height 175 cm.

The library described in this document provides several procedures that assist in modeling fuzzy systems within NetLogo. In particular, using this fuzzy logic library it will be straightforward to implement agent-based models where individual agents implement fuzzy rules such as: “If both parents are tall, then the son is tall”.

## How to include and initialize the fuzzy logic library

To be able to use all the procedures explained in this document you must save the file named “fuzzy-logic.nls” in the same directory where your NetLogo code lies, and also include the file from your code by typing the following line at the top of your code:

```
__includes ["fuzzy-logic.nls"]
```

Then, to initialize the library, you have to type once:

```
fuzzy-start
```

The procedure **`fuzzy-start`** initializes the fuzzy functionality, so it is a good idea to run **`fuzzy-start`** just after running clear-all.

## Visualization

```
fuzzy-plot A
```

Procedure run by any agent (i.e. the observer, a turtle, a patch or a link); it draws the membership function of fuzzy set *A* on the current plot.

```
let my-set fuzzy-piecewise-linear-set [[0 0][1 0][2 0.8][4 1][6 1]]
set-current-plot "Fuzzy set"   ;; make sure this plot exists!
set-plot-x-range 0 6
fuzzy-plot my-set              ;; The membership function is plotted
                               ;; in the current plot. See Fig. 2 below.
```

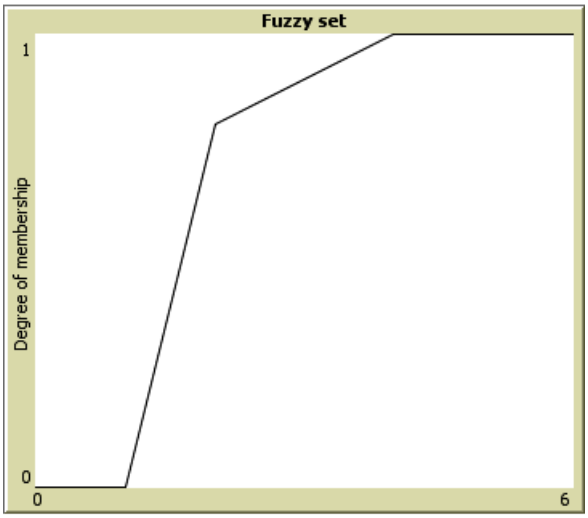


**Fig. 2. Membership function of a piecewise linear fuzzy set created running the code: fuzzy-plot fuzzy-piecewise-linear-set [[0 0][1 0][2 0.8][4 1][6 1]]**

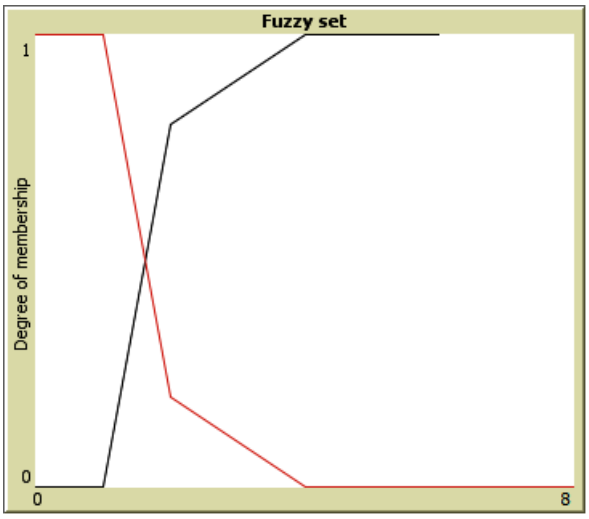


**Fig. 3. Membership functions of two piecewise linear fuzzy sets. By default, the procedure "fuzzy-plot" does not clear the plot. The color of the plot can be changed using the primitive "set-plot-pen-color".**

By default, the procedure **fuzzy-plot** will not clear the graph. This is done to allow you to plot several membership functions on the same graph.

```
let my-set1 fuzzy-piecewise-linear-set [[0 0][1 0][2 0.8][4 1][6 1]]
let my-set2 fuzzy-piecewise-linear-set [[0 1][1 1][2 0.2][4 0][8 0]]
set-plot-x-range 0 8
fuzzy-plot my-set1             ;; The membership function is the current plot
set-plot-pen-color red         ;; changes the color of the pen we use to plot
fuzzy-plot my-set2             ;; See Fig. 3.
clear-plot                     ;; clears the plot
```

## Evaluation of membership functions

Fuzzy sets can respond to a procedure named **fuzzy-evaluation**, which computes the degree of membership of any input value (either a number or another fuzzy set) on the set.

```
fuzzy-evaluation x
```

Reporter run by a fuzzy set; it returns the value of the Membership Function of the set running the procedure on input value *x* (either a number or another fuzzy set). The procedure **fuzzy-evaluation** can also take a list as an argument; in that case, it returns a list with the evaluation of the Membership Function on each item in the input list.

Let us start assuming *x* is a number. If the membership function is not defined at input number *x*, the procedure **fuzzy-evaluation** returns the value of the global variable **$undefined-degree-of-membership**, which equals "undefined" by default, but can be changed modifying the code of **fuzzy-start**.

```
let my-set fuzzy-piecewise-linear-set [[0 0][1 0][2 0.8][4 1][6 1]]
fuzzy-plot my-set                              ;; This produces Fig. 2.
ask my-set [show fuzzy-evaluation 1.5]         ;; my-set shows 0.4
ask my-set [show fuzzy-evaluation [1.5 2 3 4 7]]
                                              ;; shows [0.4 0.8 0.9 1 "undefined"]
show [fuzzy-evaluation 1.5] of my-set         ;; syntax to evaluate in any context
let a [fuzzy-evaluation [1.5 2 3 4 7]] of my-set
                                          ;; a equals [0.4 0.8 0.9 1 "undefined"]
```

The input value *x* can also be another fuzzy set. In that case, **fuzzy-evaluation** returns the degree of fulfilment between the two sets, i.e. the maximum value *y* such that there exists a number with degree of membership greater or equal to *y* in both sets. This also works if the input is a list of fuzzy sets.

```
let my-set1 fuzzy-piecewise-linear-set [[0 0][1 0][2 0.8][4 1][6 1]]
let my-set2 fuzzy-piecewise-linear-set [[0 1][1 1][2 0.2][4 0][8 0]]
fuzzy-plot my-set1     ;; draws the membership function in the current plot
set-plot-pen-color red         ;; changes the color of the pen we use to plot
fuzzy-plot my-set2             ;; See Fig. 3.
ask my-set1 [show fuzzy-evaluation my-set2]           ;; my-set1 reports 0.5
ask my-set2 [show fuzzy-evaluation my-set1]           ;; my-set2 reports 0.5
ask my-set1 [show fuzzy-evaluation list my-set1 my-set2]   ;; reports [1 0.5]
```

If there is no number at which the two sets are defined, the procedure **fuzzy-evaluation** returns the value of the global variable **$undefined-degree-of-fulfilment**, which equals "undefined" by default, but can be changed modifying the code of **fuzzy-start**.

```
let my-set1 fuzzy-piecewise-linear-set [[0 0][1 0][2 0.8][4 1][6 1]]
let my-set2 fuzzy-piecewise-linear-set [[7 1][8 0.5]]
ask my-set1 [show fuzzy-evaluation my-set2]   ;; my-set1 reports "undefined"
ask my-set2 [show fuzzy-evaluation my-set1]   ;; my-set2 reports "undefined"
ask my-set1 [show fuzzy-evaluation list my-set1 my-set2]
                                              ;; my-set1 reports [1 "undefined"]
```

Finally, the input list can contain numbers and sets.

```
let my-set1 fuzzy-piecewise-linear-set [[0 0][1 0][2 0.8][4 1][6 1]]
let my-set2 fuzzy-piecewise-linear-set [[7 1][8 0.5]]
ask my-set1 [show fuzzy-evaluation (list my-set1 my-set2 0 2 3 10)]
               ;; my-set1 reports [1 "undefined" 0 0.8 0.9 "undefined"]
```

## Retrieval of fuzzy sets

1. A variable (or several variables) can store a pointer to a fuzzy set.

```
let one-var fuzzy-piecewise-linear-set [[0 0][1 0][2 0.8][4 1][6 1]]
let another-var one-var                ;; another-var points to the new fuzzy set
fuzzy-plot another-var                 ;; This produces Fig. 2.
show one-var = another-var             ;; this shows true
ask another-var [fuzzy-die]            ;; the new fuzzy set ceases to exist
show one-var                           ;; this shows nobody
show another-var                       ;; this shows nobody
```

2. You can also retrieve a fuzzy set if you know its label, by using the procedure **fuzzy-set-with-label**.

```
fuzzy-set-with-label label
```

Reports one existing fuzzy set that has its label equal to the parameter `label`. If there is more than one fuzzy set with label equal to `label`, then a user message notifies this anomaly, and one of the fuzzy sets with label equal to `label`, is reported at random.

```
let one-var fuzzy-piecewise-linear-set-with-label "hot" [[0 0][40 1]]
fuzzy-plot fuzzy-set-with-label "hot"          ;; The membership function is
                                               ;; drawn in the current plot.
show one-var = fuzzy-set-with-label "hot"      ;; this shows true
ask fuzzy-set-with-label "hot" [fuzzy-die]
                                               ;; the new fuzzy set ceases to exist
show one-var                                   ;; this shows nobody
```

3. Finally, since fuzzy sets are turtles, they also have a who number, so you can use it to retrieve them.

## Creation of fuzzy sets from scratch

`fuzzy-discrete-numeric-set [[x1 y1] [x2 y2] ... [xn yn]]`

Creates and reports a new discrete fuzzy set. The parameter must be a list of numbers of the form: $[[x_1,y_1]\ [x_2,y_2]\ \ldots\ [x_n,y_n]]$, i.e., a list of 2-number lists. The membership function of the reported fuzzy set is a discrete function that assigns the degree of membership $y_i$ to input $x_i$. For values of $x \neq x_i$, the membership function is not defined.

```
let my-set fuzzy-discrete-numeric-set [[1 0.3][2 0.8][4 0]]
ask my-set [show fuzzy-evaluation 2]          ;; my-set shows 0.8
show [fuzzy-evaluation 3] of my-set           ;; "undefined" is shown
fuzzy-plot my-set                             ;; This produces Fig. 4 below.
```

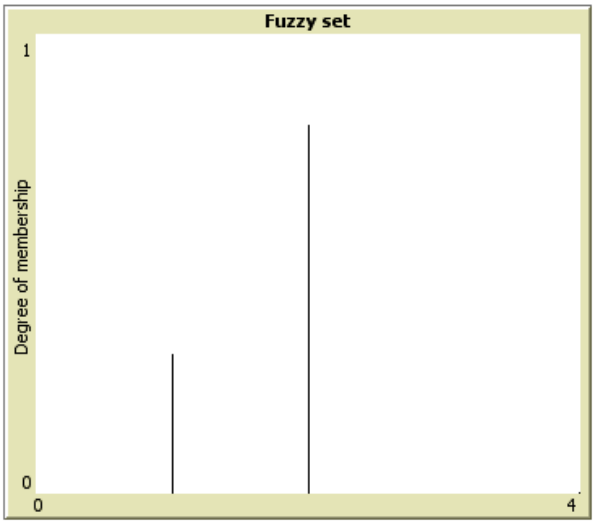


**Fig. 4. Membership function of a discrete fuzzy set created running the code: fuzzy-plot fuzzy-discrete-numeric-set [[1 0.3][2 0.8][4 0]]**

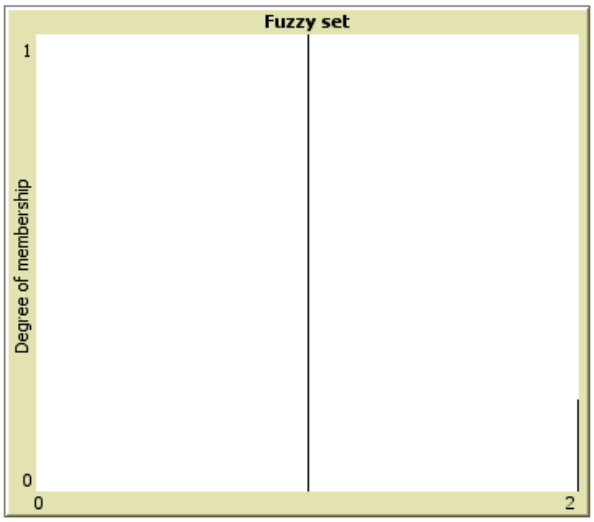


**Fig. 5. Membership function of a discrete fuzzy set created running the code: fuzzy-plot fuzzy-discrete-numeric-set [[1 1][2 0.2]]**

`fuzzy-discrete-numeric-set-with-label label [[x1 y1] [x2 y2] ... [xn yn]]`

Creates and reports a new discrete fuzzy set with label *label*. The parameter `label` can be used to retrieve the fuzzy set at a later stage by running the procedure **fuzzy-set-with-label**. See **fuzzy-discrete-numeric-set**.

```
let my-set fuzzy-discrete-numeric-set-with-label "low" [[1 1][2 0.2]]
ask fuzzy-set-with-label "low" [show fuzzy-evaluation 1]     ;; my-set shows 1
show [fuzzy-evaluation 3] of fuzzy-set-with-label "low"
                                              ;; "undefined" is shown
show my-set = fuzzy-set-with-label "low"      ;; true is shown
fuzzy-plot my-set                             ;; This produces Fig. 5 above.
```

**fuzzy-piecewise-linear-set** [[$x_1$ $y_1$] [$x_2$ $y_2$] ... [$x_n$ $y_n$]]

Creates and reports a new piecewise linear fuzzy set. The parameter must be a list of numbers of the form: [[$x_1$,$y_1$] [$x_2$,$y_2$] ... [$x_n$,$y_n$]], i.e., a list of 2-number lists. The membership function of the reported fuzzy set is a piecewise linear function that joins all the points [$x_i$,$y_i$] (previously sorted on $x_i$).

```
let my-set fuzzy-piecewise-linear-set [[0 0][1 0][1 0.3][2 0.8][4 0]]
fuzzy-plot my-set                                  ;; This produces Fig. 6 below.
show [fuzzy-evaluation 2] of my-set                ;; 0.8 is shown
show [fuzzy-evaluation 3] of my-set                ;; 0.4 is shown
show [fuzzy-evaluation 5] of my-set                ;; "undefined" is shown
```

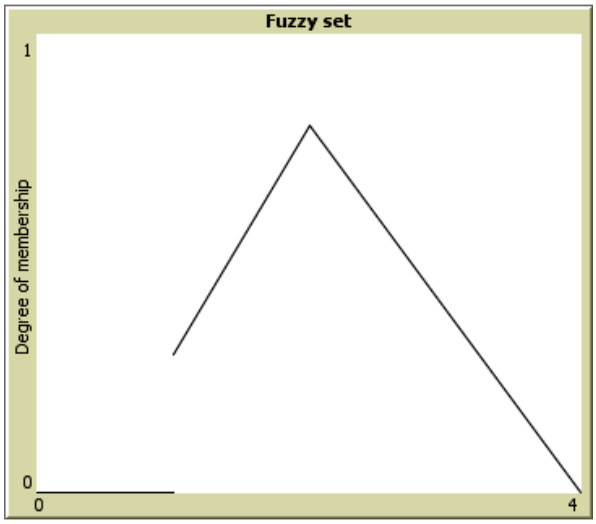


**Fig. 6. Membership function of a piecewise linear fuzzy set created running the code: fuzzy-plot fuzzy-piecewise-linear-set [[0 0][1 0][1 0.3][2 0.8][4 0]]**

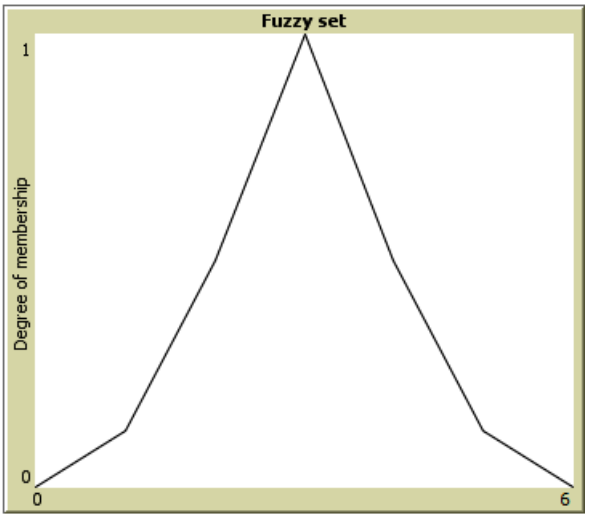


**Fig. 7. Membership function of a piecewise linear fuzzy set created running the code: fuzzy-plot fuzzy-piecewise-linear-set [[0 0][1 0.125] [2 0.5][3 1][4 0.5][5 0.125][6 0]]**

**fuzzy-piecewise-linear-set-with-label** *label* [[$x_1$ $y_1$] [$x_2$ $y_2$] ... [$x_n$ $y_n$]]

Creates and reports a new piecewise linear fuzzy set with label *label*. The parameter `label` can be used to retrieve the fuzzy set at a later stage by running the procedure **fuzzy-set-with-label**. See **fuzzy-piecewise-linear-set**.

```
let my-set fuzzy-piecewise-linear-set-with-label "about 3"
                        [[0 0][1 0.125][2 0.5][3 1][4 0.5][5 0.125][6 0]]
set-plot-x-range 0 6
fuzzy-plot my-set                                  ;; This produces Fig. 7 above.
ask fuzzy-set-with-label "about 3" [show fuzzy-evaluation 2.5]
                                                   ;; my-set shows 0.75
show [fuzzy-evaluation 0] of fuzzy-set-with-label "about 3"  ;; 0 is shown
show my-set = fuzzy-set-with-label "about 3"                 ;; true is shown
```

**fuzzy-trapezoidal-set** [a b c d e f height]

Creates and reports a new piecewise linear fuzzy set with the shape of trapezoid. The parameter must be a list of numbers of the form: [*a b c d e f height*]. The membership function of the reported fuzzy set is a piecewise linear function that equals 0 in the range *a* to *b*, increases linearly from 0 to *height* in the range *b* to *c*, is equal to *height* in the range *c* to *d*, decreases linearly from *height* to 0 in the range *d* to *e*, and equals 0 in the range *e* to *f*. The universe is [*a*, *f*]. For values outside the universe, the membership function is not defined. Note that the membership function at the extreme values *a* and *f* is 0. Thus, do not use this procedure to create membership functions that are meant to equal a value other than 0 at the extremes. In that case, use **fuzzy-piecewise-linear-set** instead.

```
let my-set fuzzy-trapezoidal-set [1 2 6 7 9 10 0.7]
set-plot-x-range 1 10
fuzzy-plot my-set                              ;; This produces Fig. 8 below.
show [fuzzy-evaluation 3] of my-set            ;; 0.175 is shown
show [fuzzy-evaluation 6.5] of my-set          ;; 0.7 is shown
show [fuzzy-evaluation 11] of my-set           ;; "undefined" is shown
```

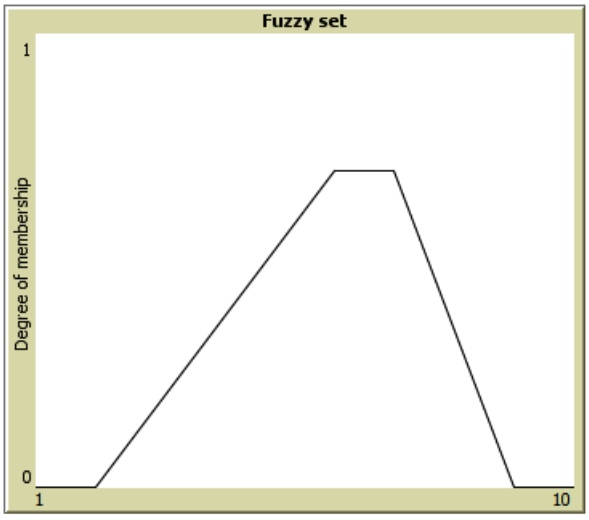


**Fig. 8. Membership function of a trapezoidal fuzzy set created running the code: fuzzy-plot fuzzy-trapezoidal-set [1 2 6 7 9 10 0.7]**

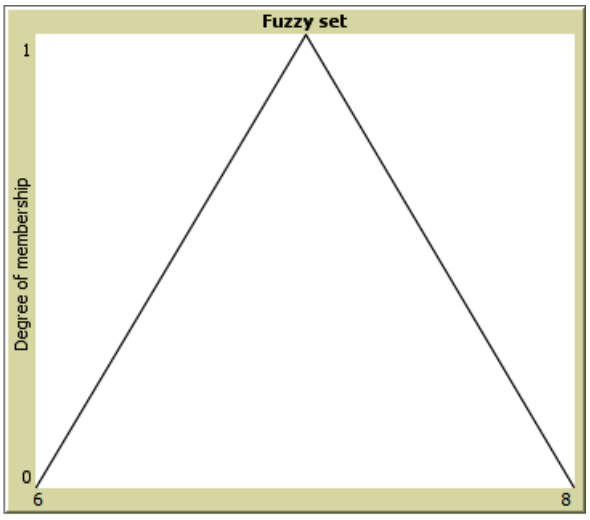


**Fig. 9. Membership function of a trapezoidal fuzzy set created running the code: fuzzy-plot fuzzy-trapezoidal-set [6 6 7 7 8 8 1]**

**fuzzy-trapezoidal-set-with-label** *label* [a b c d e f height]

Creates and reports a new trapezoidal fuzzy set with label *label*. The parameter label can be used to retrieve the fuzzy set at a later stage by running the procedure **fuzzy-set-with-label**. See **fuzzy-trapezoidal-set**.

```
let my-set fuzzy-trapezoidal-set-with-label "close to 7" [6 6 7 7 8 8 1]
set-plot-x-range 6 8
fuzzy-plot fuzzy-set-with-label "close to 7"          ;; This produces Fig. 9
ask fuzzy-set-with-label "close to 7" [show fuzzy-evaluation 6.25]
                                                      ;; my-set shows 0.25
show [fuzzy-evaluation 7.25] of fuzzy-set-with-label "close to 7"
                                                      ;; 0.75 is shown
show my-set = fuzzy-set-with-label "close to 7"       ;; true is shown
```

**fuzzy-logistic-set** [$x_0$ a b [lower-limit upper-limit]]

Creates and reports a new logistic fuzzy set. The parameter must be a list of the form: [$x_0$ *a b* [*lower-limit upper-limit*]]. The membership function of the reported fuzzy set is the logistic function:

$$MF(x) = \frac{1}{1 + a \cdot e^{-b(x-x_0)}}$$

within the range [*lower-limit upper-limit*] for x. For values of *x* outside the range (universe), the membership function is not defined.

```
let my-set fuzzy-logistic-set [2 1 1 [-6 10]]
fuzzy-plot my-set                              ;; This produces Fig. 10 below.
show [fuzzy-evaluation 0] of my-set            ;; 0.11920292202211755 is shown
show [fuzzy-evaluation 2] of my-set            ;; 0.5 is shown
show [fuzzy-evaluation 4] of my-set            ;; 0.8807970779778823 is shown
show [fuzzy-evaluation 8] of my-set            ;; 0.9975273768433653 is shown
show [fuzzy-evaluation 11] of my-set           ;; "undefined" is shown
```

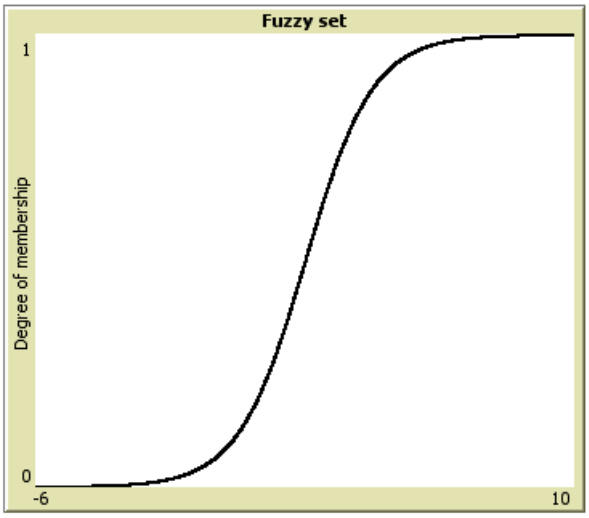


**Fig. 10. Membership function of a logistic fuzzy set created running the code: fuzzy-plot fuzzy-logistic-set [2 1 1 [-6 10]]**

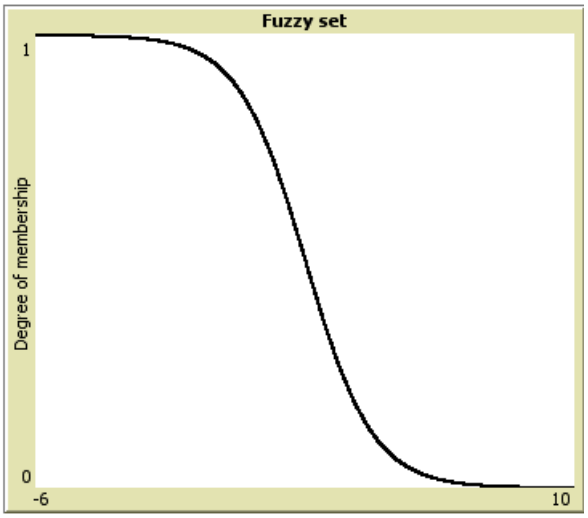


**Fig. 11. Membership function of a logistic fuzzy set created running the code: fuzzy-plot fuzzy-logistic-set [2 1 -1 [-6 10]]**

**fuzzy-logistic-set-with-label** *label* [$x_0$ a b [lower-limit upper-limit]]

Creates and reports a new logistic fuzzy set with label *label*. The parameter `label` can be used to retrieve the fuzzy set at a later stage by running the procedure **fuzzy-set-with-label**. See **fuzzy-logistic-set**.

```
let my-set fuzzy-logistic-set-with-label "low" [2 1 -1 [-6 10]]
fuzzy-plot fuzzy-set-with-label "low"          ;; This produces Fig. 11 above.
ask fuzzy-set-with-label "low" [show fuzzy-evaluation 4]
                                               ;; my-set shows 0.119202
show [fuzzy-evaluation 2] of fuzzy-set-with-label "low"  ;; 0.5 is shown
show [fuzzy-evaluation 0] of fuzzy-set-with-label "low"  ;; 0.880797 is shown
show [fuzzy-evaluation -4] of fuzzy-set-with-label "low" ;; 0.997527 is shown
show [fuzzy-evaluation -7] of fuzzy-set-with-label "low" ;; "undefined" is shown
show my-set = fuzzy-set-with-label "low"               ;; true is shown
```

**fuzzy-gaussian-set** [m s [lower-limit upper-limit]]

Creates and reports a new Gaussian (bell-shaped) fuzzy set. The parameter must be a list of the form: [*m s* [*lower-limit upper-limit*]], where *m* plays a role similar to the mean and *s* to the standard deviation.[2] Specifically, the membership function of the reported fuzzy set is the Gaussian function:

$$MF(x) = e^{-\frac{(x-m)^2}{2s^2}}$$

within the range [*lower-limit upper-limit*] for x. Note that the value of the function at *x* = *m* is always 1, so it is a scaled normal distribution. For values of *x* outside the range (universe), the membership function is not defined.

```
let my-set fuzzy-gaussian-set [5 2 [0 10]]
fuzzy-plot my-set                              ;; This produces Fig. 12 below.
show [fuzzy-evaluation 0] of my-set            ;; 0.04393693362340742 is shown
show [fuzzy-evaluation 2] of my-set            ;; 0.32465246735834974 is shown
show [fuzzy-evaluation 5] of my-set            ;; 1 is shown
```

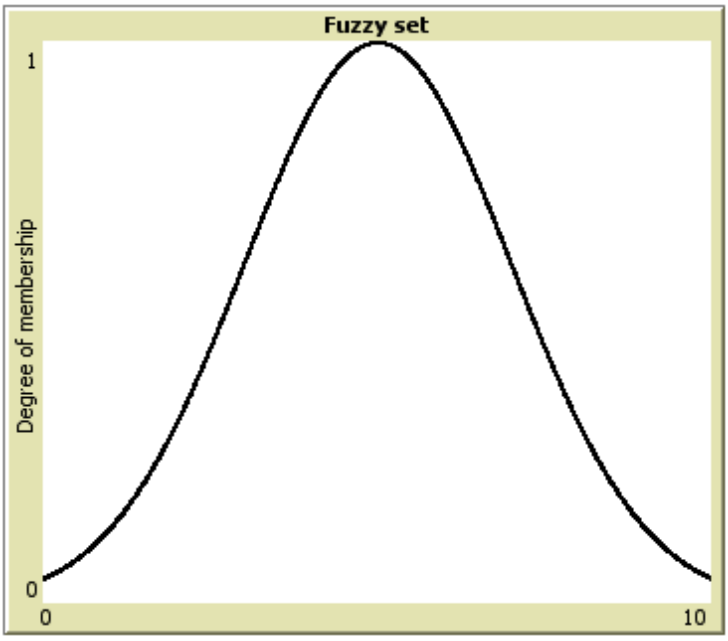


**Fig. 12. Membership function of a gaussian fuzzy set created running the code: fuzzy-plot fuzzy-gaussian-set [5 2 [0 10]]**

**fuzzy-gaussian-set-with-label** *label* [$x_0$ a b [lower-limit upper-limit]]

Creates and reports a new gaussian fuzzy set with label *label*. The parameter label can be used to retrieve the fuzzy set at a later stage by running the procedure **fuzzy-set-with-label**. See **fuzzy-gaussian-set**.

```
let my-set fuzzy-gaussian-set-with-label "normal" [5 2 [0 10]]
fuzzy-plot fuzzy-set-with-label "normal"              ;; This produces Fig. 12
ask fuzzy-set-with-label "normal" [show fuzzy-evaluation 3]
                                                      ;; my-set shows 0.60653
show [fuzzy-evaluation 7] of fuzzy-set-with-label "normal"   ;; 0.60653 is shown
show my-set = fuzzy-set-with-label "normal"                  ;; true is shown
```

[2] Note that the membership function is only defined in the range [*lower-limit upper-limit*] and *MF*(*m*) = 1.

**fuzzy-exponential-set** [a b c [lower-limit upper-limit]]

Creates and reports a new exponential fuzzy set. The parameter must be a list of the form: [*a b c* [*lower-limit upper-limit*]]. The membership function of the reported fuzzy set is the exponential function:

$$MF(x) = a \cdot e^{b(x-c)}$$

within the range [*lower-limit upper-limit*] for *x*; *MF*(*x*) is clipped so its value is always contained in the interval [0,1]. Note that the value of the function at *x* = *c* is equal to *a* (assuming $a \in [0,1]$). For values of *x* outside the range (universe), the membership function is not defined.

```
let my-set fuzzy-exponential-set [0.5 1 5 [0 10]]
fuzzy-plot my-set                              ;; This produces Fig. 13 below.
show [fuzzy-evaluation 0] of my-set            ;; 0.0033689734995427335 is shown
show [fuzzy-evaluation 4] of my-set            ;; 0.18393972058572117 is shown
show [fuzzy-evaluation 5] of my-set            ;; 0.5 is shown
show [fuzzy-evaluation 6] of my-set            ;; 1 is shown
show [fuzzy-evaluation 11] of my-set           ;; "undefined" is shown
```

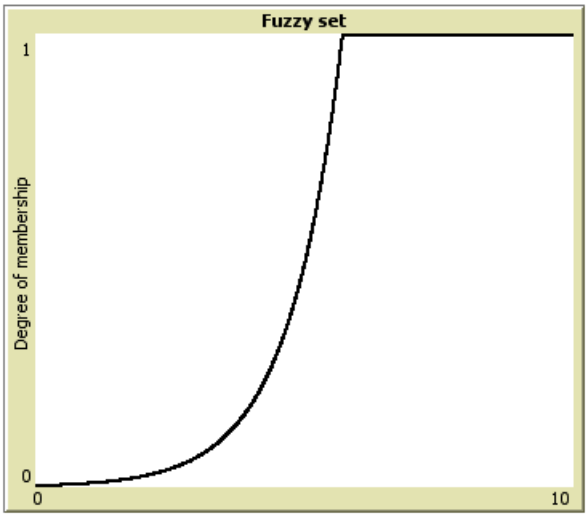


**Fig. 13. Membership function of a logistic fuzzy set created running the code: fuzzy-plot fuzzy-exponential-set [0.5 1 5 [0 10]]**

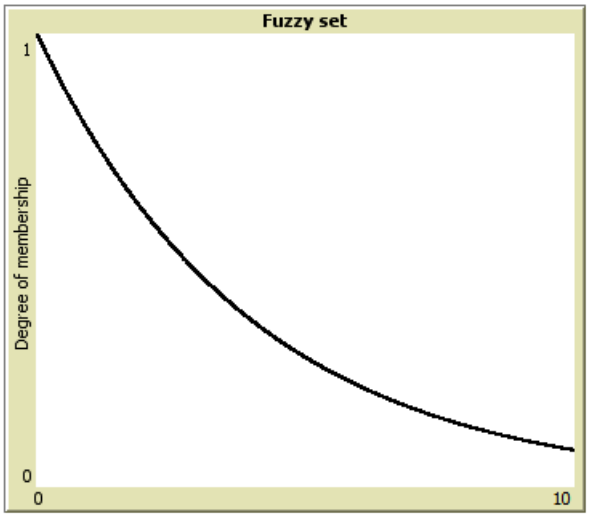


**Fig. 14. Membership function of a logistic fuzzy set created running the code: fuzzy-plot fuzzy-exponential-set [1 -0.25 0 [0 10]]**

**fuzzy-exponential-set-with-label** *label* [a b c [lower-limit upper-limit]]

Creates and reports a new exponential fuzzy set with label *label*. The parameter label can be used to retrieve the fuzzy set at a later stage by running the procedure **fuzzy-set-with-label**. See **fuzzy-exponential-set**.

```
let my-set fuzzy-exponential-set-with-label "low" [1 -0.25 0 [0 10]]
fuzzy-plot fuzzy-set-with-label "low"          ;; This produces Fig. 14 above
ask fuzzy-set-with-label "low" [show fuzzy-evaluation 0]     ;; my-set shows 1
show [fuzzy-evaluation 4] of fuzzy-set-with-label "low"      ;; 0.368 is shown
show [fuzzy-evaluation 5] of fuzzy-set-with-label "low"      ;; 0.287 is shown
show [fuzzy-evaluation 6] of fuzzy-set-with-label "low"      ;; 0.223 is shown
show [fuzzy-evaluation 11] of fuzzy-set-with-label "low"
                                                  ;; "undefined" is shown
show my-set = fuzzy-set-with-label "low"          ;; true is shown
```

**fuzzy-interval-with-points-set** [ [[a b] v] [[$x_1$ $y_1$] [$x_2$ $y_2$] ... [$x_n$ $y_n$]] ]

Creates and reports a new fuzzy set with (continuous) universe equal to the interval [*a b*]. The parameter must be a list of numbers of the form: [ [[*a b*] *v*] [[$x_1$ $y_1$] [$x_2$ $y_2$] ... [$x_n$ $y_n$]] ]. The membership function of the reported fuzzy set is a function that assigns the degree of membership v to every input in the interval [*a*,*b*], except for the inputs $x_i$ in the list [[$x_1$ $y_1$] [$x_2$ $y_2$] ... [$x_n$ $y_n$]]. The function assigns the degree of membership $y_i$ to input $x_i$. For values outside the universe, the membership function is not defined.

```
let tmp fuzzy-interval-with-points-set [ [[0 10] 0.25] [[0 0.5][2 1]] ]
show [fuzzy-evaluation 0] of tmp              ;; 0.5 is shown
show [fuzzy-evaluation 1] of tmp              ;; 0.25 is shown
show [fuzzy-evaluation 2] of tmp              ;; 1 is shown
show [fuzzy-evaluation 10.5] of tmp           ;; "undefined" is shown
```


**Fig. 15. Membership function of a fuzzy set created running the code:**
**fuzzy-plot fuzzy-interval-with-points-set list [[0 100] 0] (n-values 50 [list (2 * ? + 1) 1])**

**fuzzy-interval-with-points-set-with-label** *lbl* [[[a b] v] [[$x_1$ $y_1$]... [$x_n$ $y_n$]]]

Creates and reports a new interval-with-points fuzzy set with label *lbl*. The parameter `lbl` can be used to retrieve the fuzzy set at a later stage by running the procedure **fuzzy-set-with-label**. See **fuzzy-interval-with-points-set**. The example code below shows the creation of a conventional (crisp) set.

```
let my-new-set fuzzy-interval-with-points-set-with-label
  "odd naturals in [0,100]" list [[0 100] 0] (n-values 50 [list (2 * ? + 1) 1])
set-plot-x-range 0 100
fuzzy-plot my-new-set                         ;; this produces Fig. 15 above
ask fuzzy-set-with-label "odd naturals in [0,100]" [show fuzzy-evaluation 19]
                                                ;; my-new-set shows 1
show [fuzzy-evaluation 54] of fuzzy-set-with-label "odd naturals in [0,100]"
                                                ;; 0 is shown
show [fuzzy-evaluation 101] of fuzzy-set-with-label "odd naturals in [0,100]"
                                              ;; "undefined" is shown
show my-new-set = fuzzy-set-with-label "odd naturals in [0,100]"
                                              ;; true is shown
```

## Defuzzification

**fuzzy-FOM**

Reports the first value for which the membership function is maximal. (FOM = First Of Maxima)[3]

**fuzzy-LOM**

Reports the last value for which the membership function is maximal. (LOM = Last Of Maxima)[4]

**fuzzy-MOM**

Reports the average of the first and the last values for which the membership function is maximal, even if the membership function takes lower values in between. (MOM = Middle Of Maxima)

**fuzzy-MeOM**

Reports the mean of the values for which the membership function is maximal.[5] (MeOM = Mean Of Maxima).

**fuzzy-COG**

Reports the projection (on the horizontal axis) of the centre of gravity of the area under the membership function of the fuzzy set. (COG = Center Of Gravity).

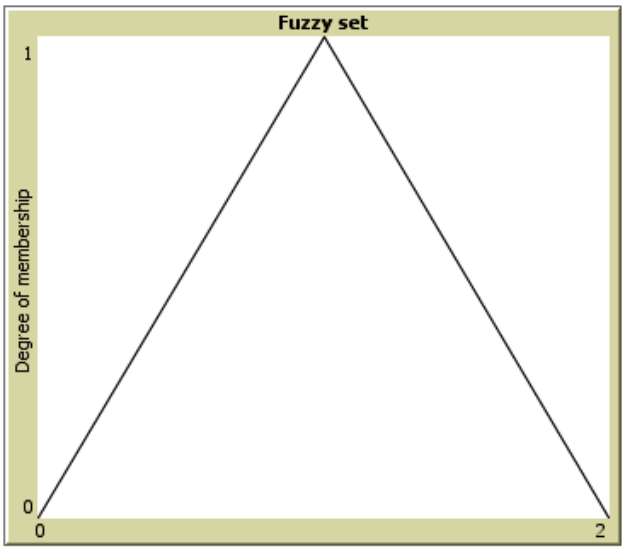


**Fig. 16. Membership function of a fuzzy set created running the code:**
**fuzzy-plot fuzzy-piecewise-linear-set [[0 0][1 1][2 0]]**

```
let my-set fuzzy-piecewise-linear-set [[0 0][1 1][2 0]]
fuzzy-plot my-set                     ;; This produces Fig. 16 above.
show [fuzzy-FOM] of my-set            ;; 1 is shown
show [fuzzy-LOM] of my-set            ;; 1 is shown
show [fuzzy-MOM] of my-set            ;; 1 is shown
show [fuzzy-MeOM] of my-set           ;; 1 is shown
show [fuzzy-COG] of my-set            ;; 1 is shown
```

[3] FOM is also known as the Smallest of Maxima.
[4] LOM is also known as the Largest of Maxima.
[5] If there is a continuum of values for which the membership function is maximal, isolated points (which would be intervals of null length) are ignored.

```
let my-set fuzzy-piecewise-linear-set [[0 1][1 0][2 1]]
fuzzy-plot my-set                    ;; This produces Fig. 17 below.
show [fuzzy-FOM] of my-set             ;; 0 is shown
show [fuzzy-LOM] of my-set             ;; 2 is shown
show [fuzzy-MOM] of my-set             ;; 1 is shown
show [fuzzy-MeOM] of my-set            ;; 1 is shown
show [fuzzy-COG] of my-set             ;; 1 is shown
```

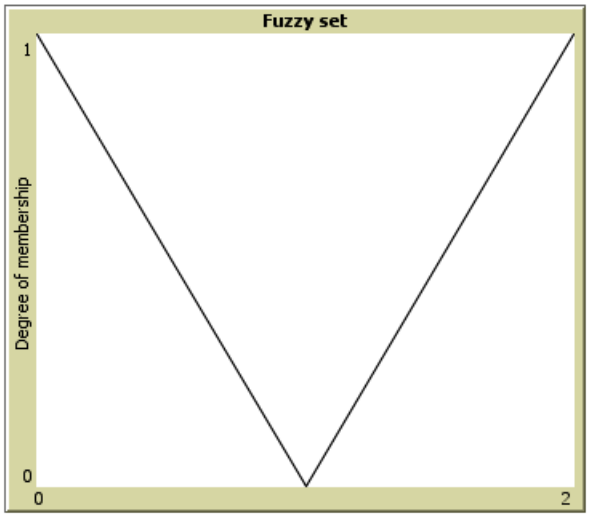


**Fig. 17. Membership function of a fuzzy set created running the code:**
**fuzzy-plot fuzzy-piecewise-linear-set [[0 1][1 0][2 1]]**

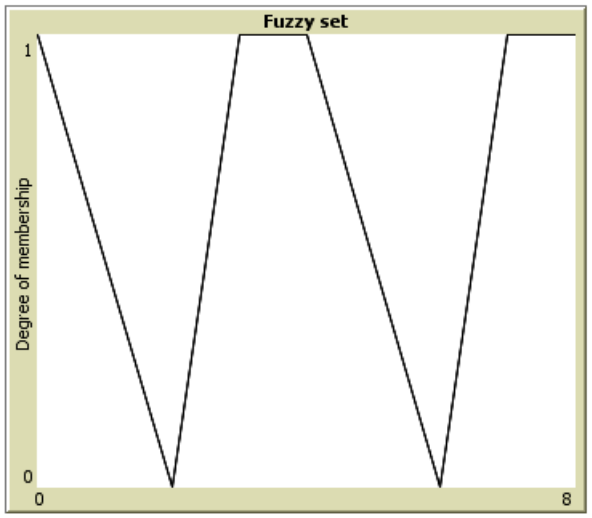


**Fig. 18. Membership function of a fuzzy set created running the code:**
**fuzzy-plot fuzzy-piecewise-linear-set**
**[[0 1][2 0][3 1][4 1][6 0][7 1][8 1]]**

```
let my-set fuzzy-piecewise-linear-set [[0 1][2 0][3 1][4 1][6 0][7 1][8 1]]
fuzzy-plot my-set                    ;; This produces Fig. 18 above.
show [fuzzy-FOM] of my-set             ;; 0 is shown
show [fuzzy-LOM] of my-set             ;; 8 is shown
show [fuzzy-MOM] of my-set             ;; 4 is shown
show [fuzzy-MeOM] of my-set            ;; 5.5 is shown
show [fuzzy-COG] of my-set             ;; 4.2 is shown
```

## Operations with fuzzy sets

```
fuzzy-min list-of-sets          (Another name: fuzzy-and)
```

Creates and reports a new fuzzy set whose membership function is the minimum of the membership functions of all the sets in the input list *list-of-sets*. The universe of the reported set is the intersection of the universes of the sets in the input list *list-of-sets*. For values outside the universe, the membership function is not defined.

```
let low fuzzy-piecewise-linear-set [[0 1][7 0][10 0]]
let high fuzzy-piecewise-linear-set [[0 0][3 0][10 1]]
set-plot-x-range 0 10
set-plot-pen-color green        ;; changes the color of the pen we use to plot
fuzzy-plot low                  ;; See Fig. 19 below.
set-plot-pen-color red          ;; changes the color of the pen we use to plot
fuzzy-plot high                 ;; See Fig. 19 below.
let high-and-low fuzzy-min (list high low)
set-plot-pen-color black        ;; changes the color of the pen we use to plot
fuzzy-plot high-and-low         ;; See Fig. 19 below.
```

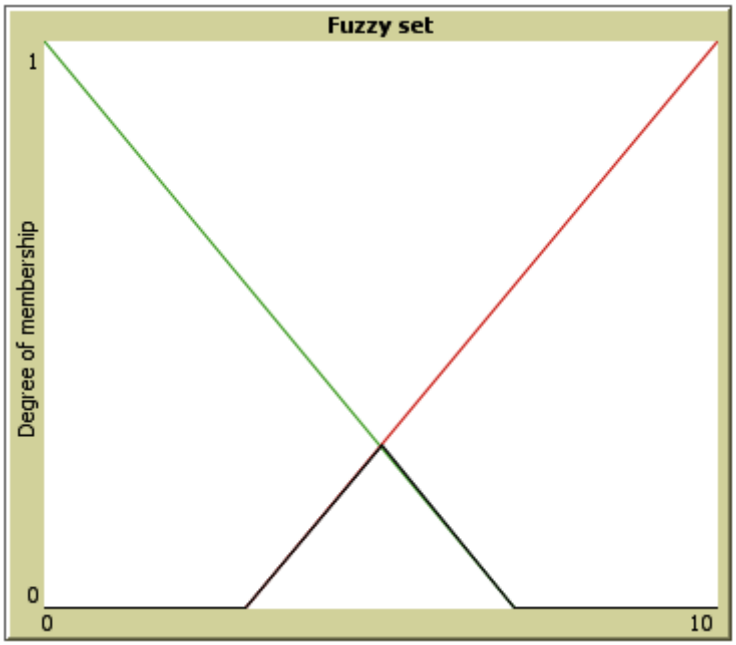


**Fig. 19. Membership function of two sets (green and red) and its fuzzy-min (black).**

```
show [fuzzy-evaluation [3 4 5 6 7]] of high-and-low
   ;; [0 0.14285714285714285 0.2857142857142857 0.14285714285714285 0] is shown
show [fuzzy-evaluation -1] of high-and-low            ;; "undefined" is shown
show [fuzzy-evaluation 11] of high-and-low            ;; "undefined" is shown
show [fuzzy-MOM] of high-and-low                      ;; 5 is shown
show [fuzzy-MeOM] of high-and-low                     ;; 5 is shown
show [fuzzy-COG] of high-and-low                      ;; 5 is shown
```

```
fuzzy-max list-of-sets          (Another name: fuzzy-or)
```

Creates and reports a new fuzzy set whose membership function is the maximum of the membership functions of all the sets in the input list *list-of-sets*. The universe of the reported set is the intersection of the universes of the sets in the input list *list-of-sets*. For values outside the universe, the membership function is not defined.

```
let low fuzzy-piecewise-linear-set [[0 1][7 0][10 0]]
let high fuzzy-piecewise-linear-set [[0 0][3 0][10 1]]
set-plot-x-range 0 10
set-plot-pen-color green        ;; changes the color of the pen we use to plot
fuzzy-plot low                  ;; See Fig. 20 below.
set-plot-pen-color red          ;; changes the color of the pen we use to plot
fuzzy-plot high                 ;; See Fig. 20 below.
let high-or-low fuzzy-max (list high low)
set-plot-pen-color black        ;; changes the color of the pen we use to plot
fuzzy-plot high-or-low          ;; See Fig. 20 below.
```

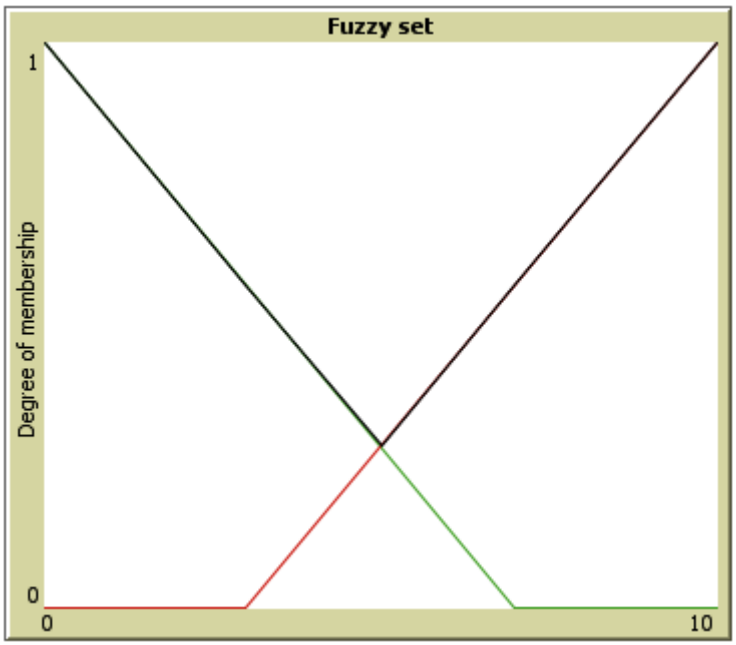


**Fig. 20. Membership function of two sets (green and red) and its fuzzy-max (black).**

```
show [fuzzy-evaluation [3 4 5 6 7]] of high-or-low
                                ;; [0.571 0.429 0.286 0.429 0.571] is shown
show [fuzzy-evaluation -1] of high-or-low            ;; "undefined" is shown
show [fuzzy-evaluation 11] of high-or-low            ;; "undefined" is shown
show [fuzzy-MOM] of high-or-low                      ;; 5 is shown
show [fuzzy-MeOM] of high-or-low                     ;; 5 is shown
show [fuzzy-COG] of high-or-low                      ;; 5 is shown
```

```
fuzzy-sum list-of-sets
```

Creates and reports a new fuzzy set whose membership function is the (clipped) sum of the membership functions of all the sets in the input list *list-of-sets*. If the sum is greater than 1, then the value is clipped to 1. The universe of the reported set is the intersection of the universes of the sets in the input list *list-of-sets*. For values outside the universe, the membership function is not defined.

```
let low fuzzy-piecewise-linear-set [[0 1][7 0][10 0]]
let high fuzzy-piecewise-linear-set [[0 0][3 0][10 1]]
set-plot-x-range 0 10
set-plot-pen-color green       ;; changes the color of the pen we use to plot
fuzzy-plot low                 ;; See Fig. 21 below.
set-plot-pen-color red         ;; changes the color of the pen we use to plot
fuzzy-plot high                ;; See Fig. 21 below.
let high-plus-low fuzzy-sum (list high low)
set-plot-pen-color black       ;; changes the color of the pen we use to plot
fuzzy-plot high-plus-low       ;; See Fig. 21 below.
```

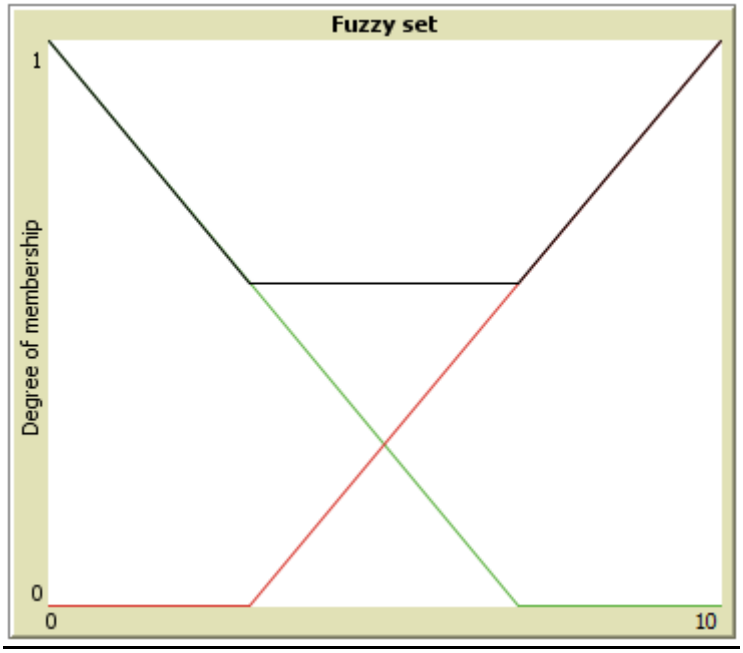


**Fig. 21. Membership function of two sets (green and red) and its fuzzy-sum (black).**

```
show [fuzzy-evaluation [2 4 5 6 8]] of high-plus-low
                              ;; [0.714 0.571 0.571 0.571 0.714] is shown
show [fuzzy-evaluation -1] of high-plus-low          ;; "undefined" is shown
show [fuzzy-evaluation 11] of high-plus-low          ;; "undefined" is shown
show [fuzzy-MOM] of high-plus-low                    ;; 5 is shown
show [fuzzy-MeOM] of high-plus-low                   ;; 5 is shown
show [fuzzy-COG] of high-plus-low                    ;; 5 is shown
show [fuzzy-FOM] of high-plus-low                    ;; 0 is shown
show [fuzzy-LOM] of high-plus-low                    ;; 10 is shown
```

```
fuzzy-prob-or list-of-sets
```

Creates and reports a new fuzzy set whose membership function is the probabilistic OR of the membership functions of all the sets in the input list *list-of-sets*. For two values *x y*, the probabilistic OR is (*x* + *y* – *x***y*). For more than two values the result can be computed recursively, given that the function is associative. The universe of the reported set is the intersection of the universes of the sets in the input list *list-of-sets*. For values outside the universe, the membership function is not defined.

```
let low fuzzy-gaussian-set [2 1 [0 10]]
let high fuzzy-gaussian-set [7 3 [0 10]]
set-plot-x-range 0 10
set-plot-pen-color green       ;; changes the color of the pen we use to plot
fuzzy-plot low                 ;; See Fig. 22 below.
set-plot-pen-color red         ;; changes the color of the pen we use to plot
fuzzy-plot high                ;; See Fig. 22 below.
let high-prob-or-low fuzzy-prob-or (list high low)
set-plot-pen-color black       ;; changes the color of the pen we use to plot
fuzzy-plot high-prob-or-low    ;; See Fig. 22 below.
```

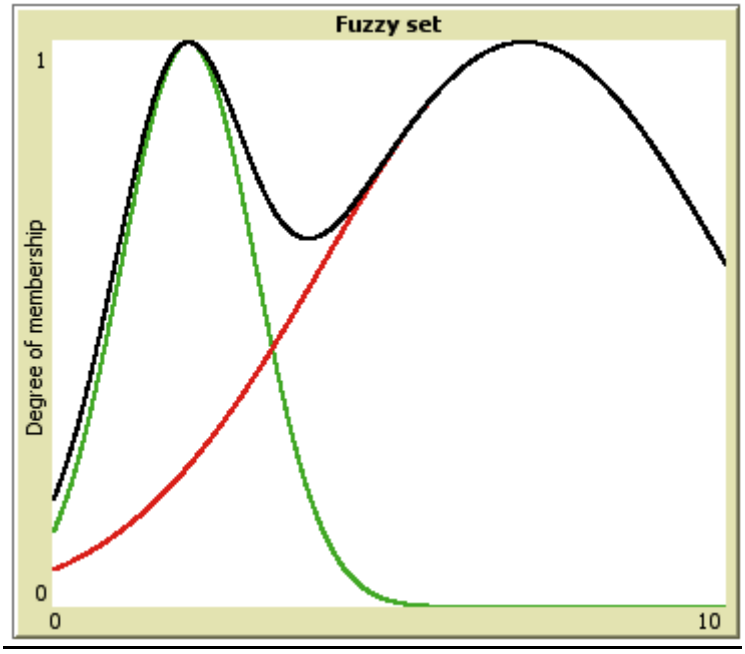


**Fig. 22. Membership function of two sets (green and red) and its fuzzy-prob-or (black).**

```
show [fuzzy-evaluation [2 4 5 6 7]] of high-prob-or-low
                               ;; [1 0.65978 0.80295 0.94598 1] is shown
show [fuzzy-evaluation -1] of high-prob-or-low       ;; "undefined" is shown
show [fuzzy-evaluation 11] of high-prob-or-low       ;; "undefined" is shown
show [fuzzy-MOM] of high-prob-or-low                 ;; 4.5 is shown
show [fuzzy-MeOM] of high-prob-or-low                ;; 4.5 is shown
show [fuzzy-COG] of high-prob-or-low                 ;; 5.26338 is shown
show [fuzzy-FOM] of high-prob-or-low                 ;; 2 is shown
show [fuzzy-LOM] of high-prob-or-low                 ;; 7 is shown
```

```
fuzzy-not A
```

*A* is a fuzzy set. `fuzzy-not` *A* creates and reports a new fuzzy set whose membership function is $(1 - MF_A(x))$, i.e. the complement of set *A*. The universe of the reported set is the universe of *A*. For values outside the universe, the membership function is not defined.

```
let low fuzzy-piecewise-linear-set [[0 1][7 0][10 0]]
let not-low fuzzy-not low
set-plot-x-range 0 10
set-plot-pen-color green       ;; changes the color of the pen we use to plot
fuzzy-plot low                 ;; See Fig. 23 below.
set-plot-pen-color black       ;; changes the color of the pen we use to plot
fuzzy-plot not-low             ;; See Fig. 23 below.
```

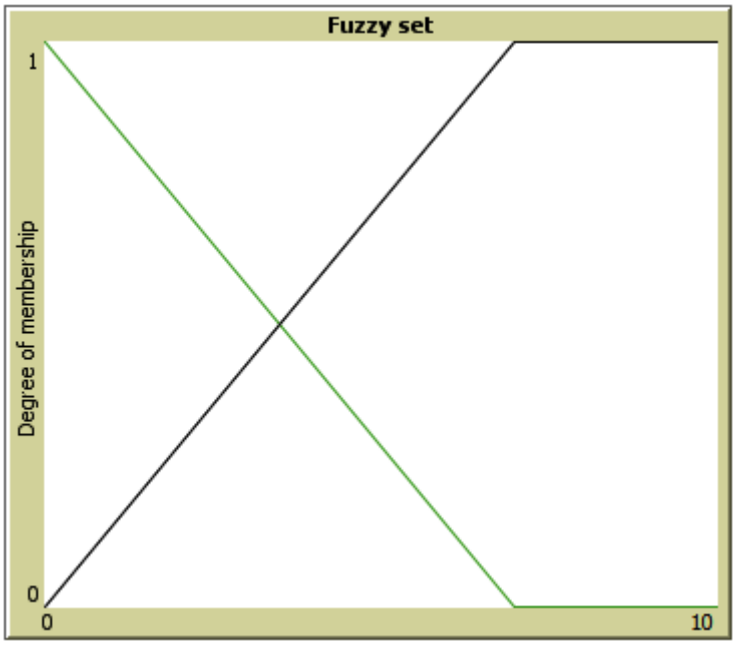


**Fig. 23. Membership function of one set (in green) and its fuzzy-not (black).**

```
let low-and-not-low fuzzy-and (list low not-low)
show [fuzzy-evaluation 3.5] of low-and-not-low       ;; 0.5 is shown
       ;; The contradiction axiom, which dictates that the intersection of a
       ;; set and its complement must be the empty set (Ross, 2010) is not
       ;; necessarily satisfied in fuzzy logic.

let low-or-not-low fuzzy-or (list low not-low)
show [fuzzy-evaluation 3.5] of low-or-not-low        ;; 0.5 is shown
       ;; The excluded middle axiom, which dictates that the union of a
       ;; set and its complement must contain all the elements (Ross, 2010)
       ;; is not necessarily satisfied in fuzzy logic.
```

```
fuzzy-truncate A c
```

*A* is a fuzzy set and *c* is a number in the interval [0,1]. This procedure creates and reports a new fuzzy set whose membership function is the same as *A*'s, but limited by the upper bound *c*. The universe of the reported set is the universe of *A*. For values outside the universe, the membership function is not defined.

```
let my-set fuzzy-gaussian-set [7 3 [0 10]]
set-plot-x-range 0 10
set-plot-pen-color red          ;; changes the color of the pen we use to plot
fuzzy-plot my-set               ;; See Fig. 24 below.
let my-set-truncated fuzzy-truncate my-set 0.5
set-plot-pen-color black        ;; changes the color of the pen we use to plot
fuzzy-plot my-set-truncated     ;; See Fig. 24 below.
```

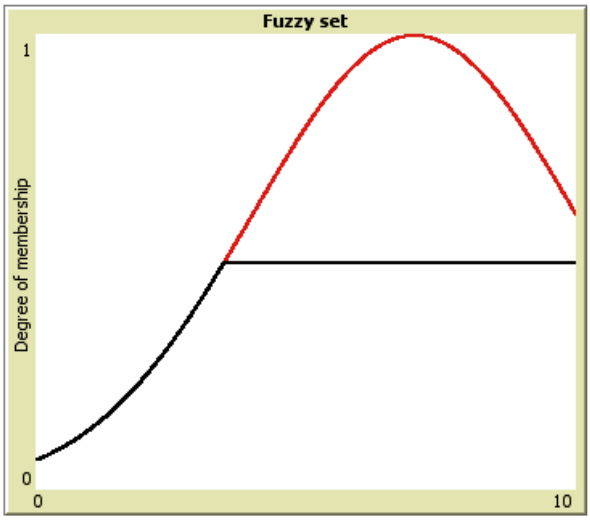


**Fig. 24. In black: Membership function of a fuzzy set created running the code: fuzzy-plot fuzzy-truncate fuzzy-gaussian-set [7 3 [0 10]] 0.5**

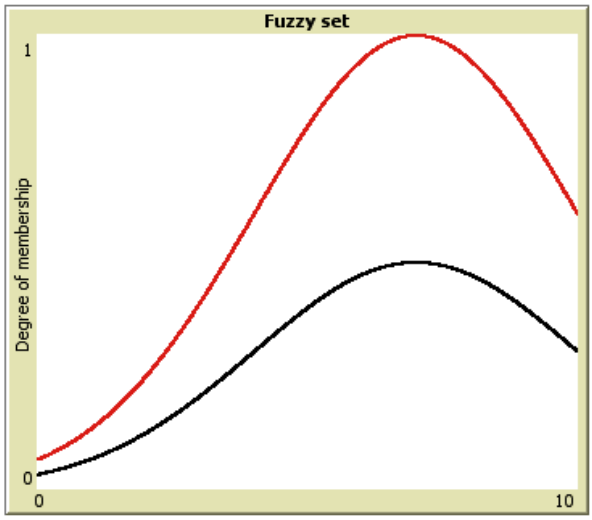


**Fig. 25. In black Membership function of a fuzzy set created running the code: fuzzy-plot fuzzy-prod fuzzy-gaussian-set [7 3 [0 10]] 0.5**

```
fuzzy-prod A f
```

*A* is a fuzzy set and *f* is a number in the interval [0,1]. This procedure creates and reports a new fuzzy set whose membership function is *A*'s membership function multiplied by factor *f*. Effectively, this operation scales down the original membership function to 100·*f* % its original size. The universe of the reported set is the universe of *A*. For values outside the universe, the membership function is not defined.

```
let my-set fuzzy-gaussian-set [7 3 [0 10]]
set-plot-x-range 0 10
set-plot-pen-color red          ;; changes the color of the pen we use to plot
fuzzy-plot my-set               ;; See Fig. 25 above.
let my-set-prod fuzzy-prod my-set 0.5
set-plot-pen-color black        ;; changes the color of the pen we use to plot
fuzzy-plot my-set-prod          ;; See Fig. 25 above.
```

## Hedges

```
fuzzy-power A exp
```

*A* is a fuzzy set and *exp* is a positive number. This procedure creates and reports a new fuzzy set whose membership function is the same as *A*'s, but raised to the exponent *exp* (i.e. $MF_A(x)^{exp}$), as long as the value is in the interval [0,1]; otherwise, it is clipped. The universe of the reported set is the universe of *A*. For values outside the universe, the membership function is not defined.

```
let hot fuzzy-logistic-set [20 1 .125 [-30 80]]
set-plot-x-range -30 80
set-plot-pen-color green       ;; changes the color of the pen we use to plot
fuzzy-plot hot                 ;; See Fig. 26 below.

let very-hot fuzzy-power hot 2
set-plot-pen-color black       ;; changes the color of the pen we use to plot
fuzzy-plot very-hot            ;; See Fig. 26 below.

let really-hot fuzzy-power hot 6
set-plot-pen-color red         ;; changes the color of the pen we use to plot
fuzzy-plot really-hot          ;; See Fig. 26 below.
```

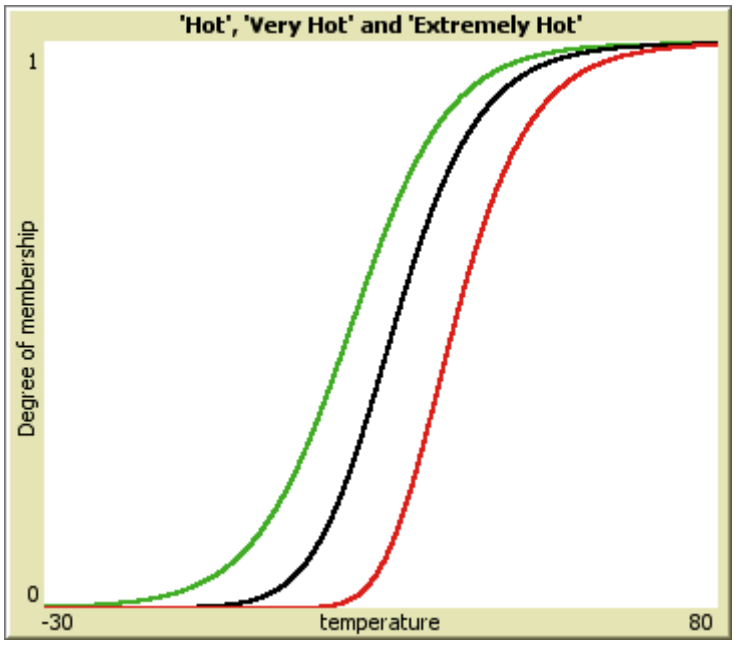


**Fig. 26. Membership function of one set (in green), its square (in black), and the green set powered to the 6th (in red).**

## Rules

```
fuzzy-truncate-rule list-of-2-items consequent-fuzzy-set
(Another name: fuzzy-rule)
```

This procedure can be used to implement simple rules such as “If temperature is hot, set the air-conditioning on high-power”, if one wants to use **fuzzy-truncate** to modify the consequent fuzzy set. The rule would be written as:

```
fuzzy-rule (list temperature hot) high-power
```

The first parameter *list-of-2-items* is a list of 2 items. The first item in the list (which can be a number or a fuzzy set) will be evaluated in the second item (which is a fuzzy-set), to obtain *evaluation*. The procedure creates and reports a new fuzzy set whose membership function is the same as that of *consequent-fuzzy-set* truncated by the calculated number *evaluation*. The universe of the reported set is the universe of *consequent-fuzzy-set*. For values outside the universe, the membership function is not defined.

```
;; Implementation of the rule:
;; "If temperature is hot, set the air-conditioning on high-power",
;; using fuzzy-truncate to modify the consequent fuzzy set.

let hot fuzzy-piecewise-linear-set [[0 0][27 0][37 1][60 1]]
                                                ;; See Fig. 27 below.
let high-power fuzzy-piecewise-linear-set [[0 0][5 0][10 1]]
                                                ;; See Fig. 27 below.
let temperature 35
    ;; note that temperature could have any value previously computed

let air-conditioning-fuzzy fuzzy-rule (list temperature hot) high-power
    ;; air-conditioning-fuzzy would be the fuzzy set painted in red on Fig. 27.

let air-conditioning-crisp [fuzzy-FOM] of air-conditioning-fuzzy
    ;; air-conditioning-crisp would equal 9 in this case
```

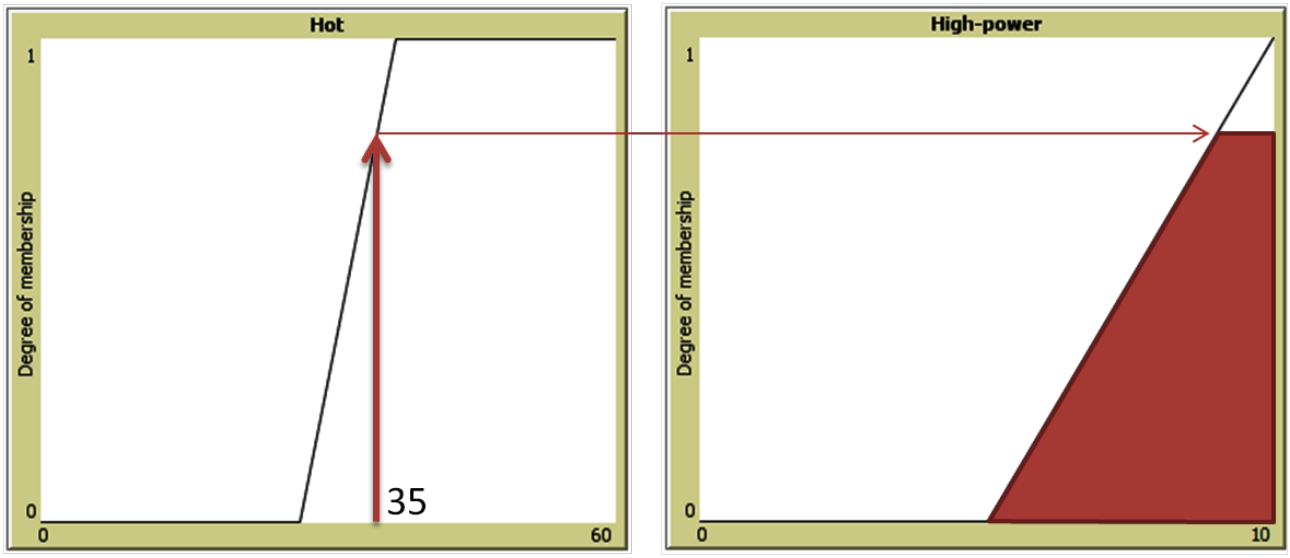


**Fig. 27. Illustration of the fuzzy-rule “If temperature is hot, set the air-conditioning on high-power”, evaluated at temperature = 35, using `fuzzy-truncate` to modify the consequent fuzzy set.**

```
fuzzy-prod-rule list-of-2-items consequent-fuzzy-set
```

This procedure can be used to implement simple rules such as “If temperature is hot, set the air-conditioning on high-power”, if one wants to use `fuzzy-prod` to modify the consequent fuzzy set. The rule would be written as:

```
fuzzy-prod-rule (list temperature hot) high-power
```

The first parameter *list-of-2-items* is a list of 2 items. The first item in the list (which can be a number or a fuzzy set) will be evaluated in the second item (which is a fuzzy-set), to obtain *evaluation*. The procedure creates and reports a new fuzzy set whose membership function is *A*'s membership function multiplied by factor *evaluation*. In effect, this operation scales down the original membership function to 100·*evaluation* % its original size. The universe of the reported set is the universe of *consequent-fuzzy-set*. For values outside the universe, the membership function is not defined.

```
;; Implementation of the rule:
;; "If temperature is hot, set the air-conditioning on high-power"
;; using fuzzy-prod to modify the consequent fuzzy set.

let hot fuzzy-piecewise-linear-set [[0 0][27 0][37 1][60 1]]
                                                ;; See Fig. 28 below.

let high-power fuzzy-piecewise-linear-set [[0 0][5 0][10 1]]
                                                ;; See Fig. 28 below.

let temperature 35
    ;; note that temperature could have any value previously computed

let air-conditioning-fuzzy fuzzy-prod-rule (list temperature hot) high-power
    ;; air-conditioning-fuzzy would be the fuzzy set painted in red on Fig. 28.

let air-conditioning-crisp [fuzzy-FOM] of air-conditioning-fuzzy
    ;; air-conditioning-crisp would equal 10 in this case
```

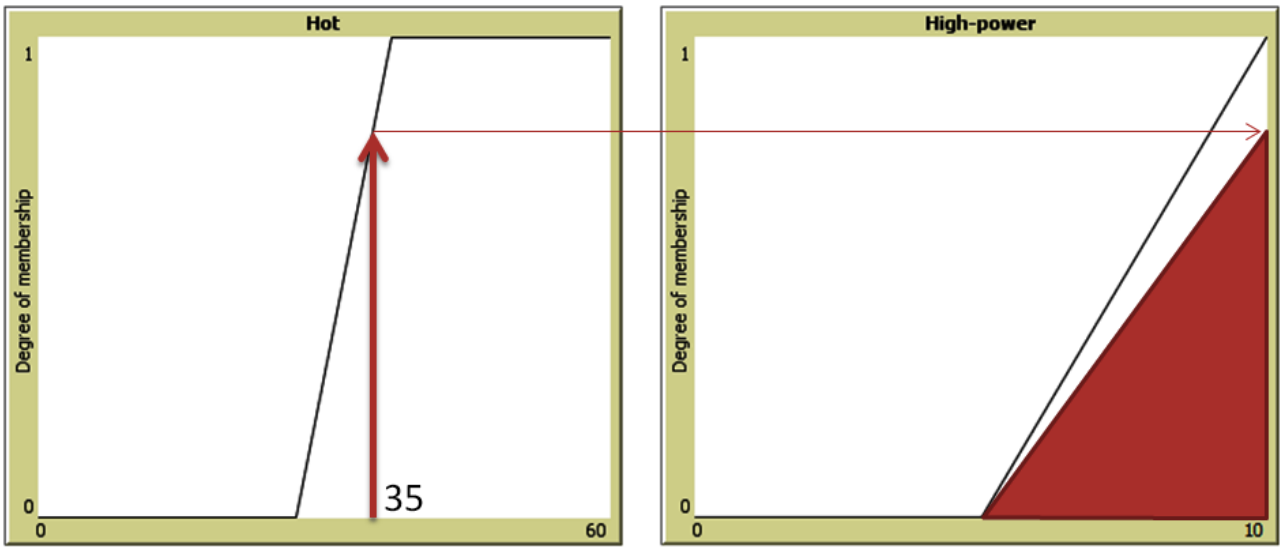


**Fig. 28. Illustration of the fuzzy-rule “If temperature is hot, set the air-conditioning on high-power”, evaluated at temperature = 35, using `fuzzy-prod` to modify the consequent fuzzy set.**

```
fuzzy-min-truncate-rule list-of-2-item-lists consequent-fuzzy-set
(Another name: fuzzy-and-rule)
```

This procedure can be used to implement rules such as "If temperature is around 20 AND rainfall is low, THEN weather is nice", if one wants to use **fuzzy-min** as AND method and **fuzzy-truncate** to modify the consequent fuzzy set. The rule would be written as:

```
fuzzy-and-rule (list (list temperature around-20) (list rainfall low)) nice
```

The first parameter *list-of-2-item-lists* is a list of any number of 2-item lists. The first item in each of the 2-item lists (which can be a number or a fuzzy set) will be evaluated in the second item (which is a fuzzy-set) of the 2-item list. This gives a list of *evaluations*. The procedure creates and reports a new fuzzy set whose membership function is the same as that of *consequent-fuzzy-set* truncated by the minimum of all the numbers in the list *evaluations*. The universe of the reported set is the universe of *consequent-fuzzy-set*. For values outside the universe, the membership function is not defined.

```
;; Implementation of the rule:
;; "If temperature is around 20 and rainfall is low, weather is nice"
;; using fuzzy-min as AND method and fuzzy-truncate to modify the consequent
;; fuzzy set.

let around-20 fuzzy-gaussian-set [20 3 [-10 50]]
                                                    ;; See Fig. 29 below.
let low fuzzy-piecewise-linear-set [[0 1][2 0][10 0]]
                                                    ;; See Fig. 29 below.
let temperature 22
        ;; note that temperature could have any value previously computed
let rainfall 1.5
        ;; note that rainfall could have any value previously computed

let nice fuzzy-piecewise-linear-set [[0 0][10 1]]
                                                    ;; See Fig. 29 below.
let weather-rating-fuzzy fuzzy-and-rule
        (list (list temperature around-20) (list rainfall low) ) nice

    ;; weather-fuzzy would be the fuzzy set painted in red on Fig. 29 below.

let weather-rating-crisp [fuzzy-MeOM] of weather-rating-fuzzy
    ;; weather-rating-crisp would equal 6.25 in this case
```

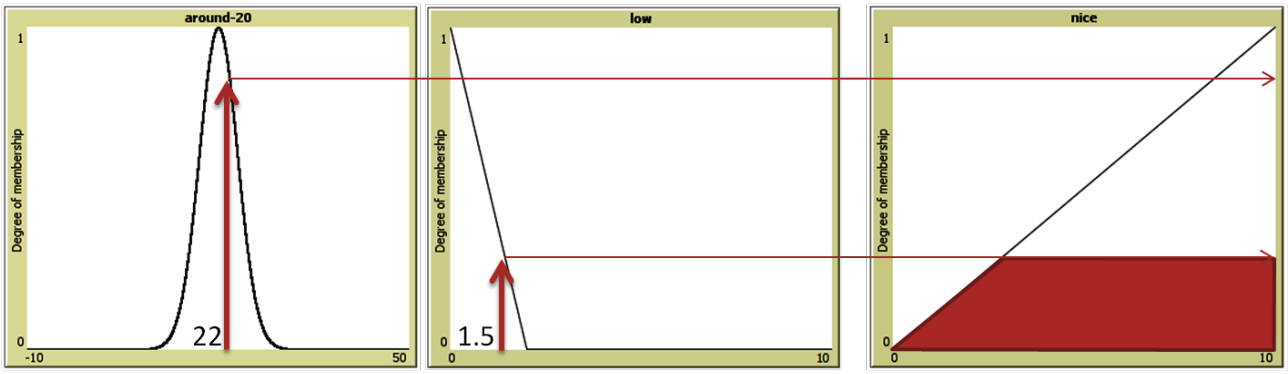


**Fig. 29. Illustration of the fuzzy-and-rule "If temperature is around 20 and rainfall is low, weather is nice", evaluated at temperature = 22 and rainfall = 1.5, using `fuzzy-min` as AND method and `fuzzy-truncate` to modify the consequent fuzzy set.**

**fuzzy-min-prod-rule** *list-of-2-item-lists consequent-fuzzy-set*

This procedure can be used to implement rules such as "If temperature is around 20 AND rainfall is low, THEN weather is nice", if one wants to use **fuzzy-min** as AND method and **fuzzy-prod** to modify the consequent fuzzy set. The rule would be written as:

```
fuzzy-min-prod-rule (list (list temperature around-20) (list rainfall low)) nice
```

The first parameter *list-of-2-item-lists* is a list of any number of 2-item lists. The first item in each of the 2-item lists (which can be a number or a fuzzy set) will be evaluated in the second item (which is a fuzzy-set) of the 2-item list. This gives a list of *evaluations*. The procedure creates and reports a new fuzzy set whose membership function is the same as that of *consequent-fuzzy-set* multiplied by the minimum of all the numbers in the list *evaluations*. The universe of the reported set is the universe of *consequent-fuzzy-set*. For values outside the universe, the membership function is not defined.

```
;; Implementation of the rule:
;; "If temperature is around 20 and rainfall is low, weather is nice"
;; using fuzzy-min as AND method and fuzzy-prod to modify the consequent fuzzy
;; set.

let around-20 fuzzy-gaussian-set [20 3 [-10 50]]
                                                    ;; See Fig. 30 below.
let low fuzzy-piecewise-linear-set [[0 1][2 0][10 0]]
                                                    ;; See Fig. 30 below.
let temperature 22
        ;; note that temperature could have any value previously computed
let rainfall 1.5
        ;; note that rainfall could have any value previously computed

let nice fuzzy-piecewise-linear-set [[0 0][10 1]]
                                                    ;; See Fig. 30 below.
let weather-rating-fuzzy fuzzy-min-prod-rule
        (list (list temperature around-20) (list rainfall low) ) nice

    ;; weather-fuzzy would be the fuzzy set painted in red on Fig. 30 below.

let weather-rating-crisp [fuzzy-MeOM] of weather-rating-fuzzy
    ;; weather-rating-crisp would equal 10 in this case
```

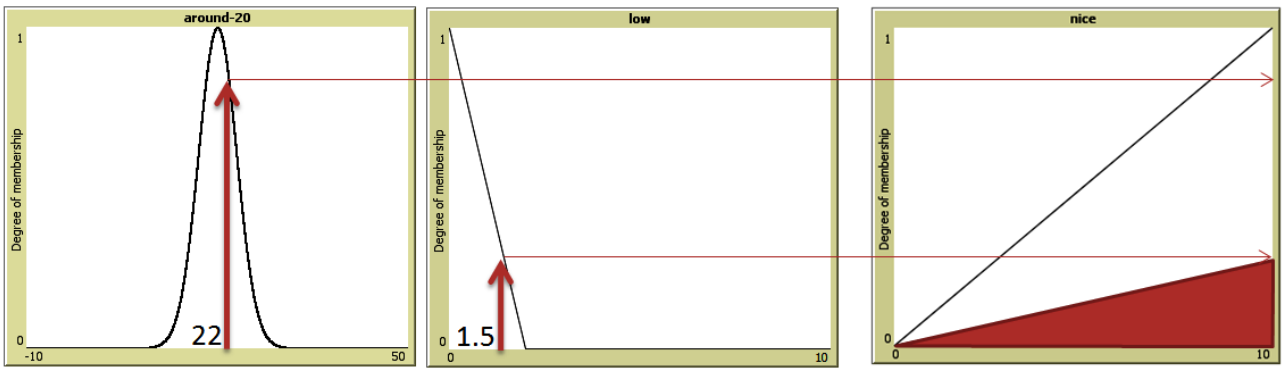


**Fig. 30. Illustration of the fuzzy-and-rule "If temperature is around 20 and rainfall is low, weather is nice", evaluated at temperature = 22 and rainfall = 1.5, using `fuzzy-min` as AND method and `fuzzy-prod` to modify the consequent fuzzy set.**

```
fuzzy-max-truncate-rule list-of-2-item-lists consequent-fuzzy-set
(Another name: fuzzy-or-rule)
```

This procedure can be used to implement rules such as "If temperature is high OR temperature is low, THEN my comfort is low", if one wants to use **fuzzy-max** as OR method and **fuzzy-truncate** to modify the consequent fuzzy set. The rule would be written as:

```
fuzzy-or-rule (list (list temperature high) (list temperature low)) low-comfort
```

The first parameter *list-of-2-item-lists* is a list of any number of 2-item lists. The first item in each of the 2-item lists (which can be a number or a fuzzy set) will be evaluated in the second item (which is a fuzzy-set) of the 2-item list. This gives a list of *evaluations*. The procedure creates and reports a new fuzzy set whose membership function is the same as that of *consequent-fuzzy-set* truncated by the maximum of all the numbers in the list *evaluations*. The universe of the reported set is the universe of *consequent-fuzzy-set*. For values outside the universe, the membership function is not defined.

```
;; Implementation of the rule:
;; "If temperature is high or temperature is low, my comfort is low"
;; using fuzzy-max as OR method and fuzzy-truncate to modify the consequent
;; fuzzy set.

let high fuzzy-piecewise-linear-set [[-10 0][25 0][35 1][50 1]]
                                                ;; See Fig. 31 below.
let low fuzzy-piecewise-linear-set [[-10 1][5 1][15 0][60 0]]
                                                ;; See Fig. 31 below.
let low-comfort fuzzy-piecewise-linear-set [[0 1][7 0][10 0]]
                                                ;; See Fig. 31 below.
let temperature 30
       ;; note that temperature could have any value previously computed

let comfort-fuzzy fuzzy-or-rule
       (list (list temperature high) (list temperature low) ) low-comfort

 ;; comfort-fuzzy would be the fuzzy set painted in red on Fig. 31 below.

let comfort-crisp [fuzzy-MOM] of comfort-fuzzy
    ;; comfort-crisp would equal 1.75 in this case
```

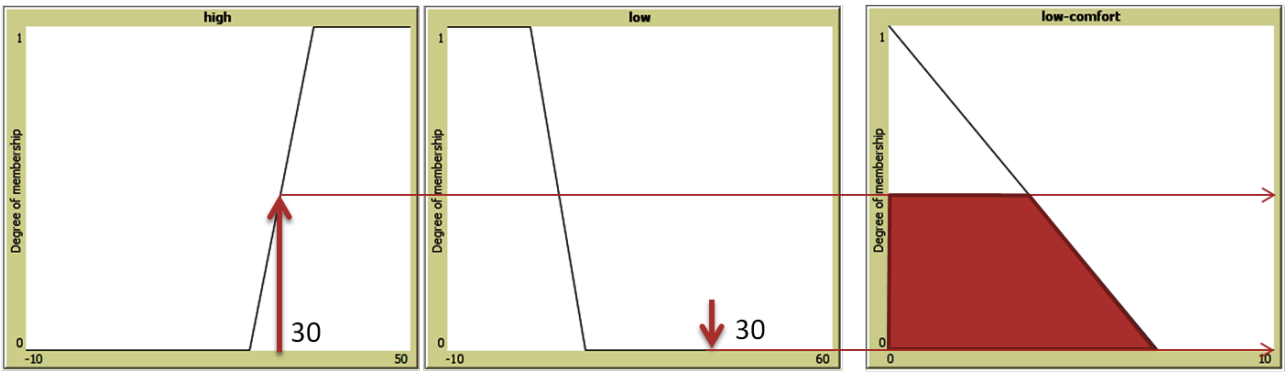


**Fig. 31. Illustration of the fuzzy-or-rule "If temperature is high or temperature is low, my comfort is low", evaluated at temperature = 30, using `fuzzy-max` as OR method and `fuzzy-truncate` to modify the consequent fuzzy set.**

**fuzzy-max-prod-rule** *list-of-2-item-lists consequent-fuzzy-set*

This procedure can be used to implement rules such as "If temperature is high OR temperature is low, THEN my comfort is low", if one wants to use **fuzzy-prod** to modify the consequent fuzzy set. The rule would be written as:

```
fuzzy-max-prod-rule
        (list (list temperature high) (list temperature low)) low-comfort
```

The first parameter *list-of-2-item-lists* is a list of any number of 2-item lists. The first item in each of the 2-item lists (which can be a number or a fuzzy set) will be evaluated in the second item (which is a fuzzy-set) of the 2-item list. This gives a list of *evaluations*. The procedure creates and reports a new fuzzy set whose membership function is the same as that of *consequent-fuzzy-set* multiplied by the maximum of all the numbers in the list *evaluations*. The universe of the reported set is the universe of *consequent-fuzzy-set*. For values outside the universe, the membership function is not defined.

```
;; Implementation of the rule:
;; "If temperature is high or temperature is low, my comfort is low"
;; using fuzzy-max as OR method and fuzzy-prod to modify the consequent fuzzy
set.

let high fuzzy-piecewise-linear-set [[-10 0][25 0][35 1][50 1]]
                                                ;; See Fig. 32 below.
let low fuzzy-piecewise-linear-set [[-10 1][5 1][15 0][60 0]]
                                                ;; See Fig. 32 below.
let low-comfort fuzzy-piecewise-linear-set [[0 1][7 0][10 0]]
                                                ;; See Fig. 32 below.
let temperature 30
        ;; note that temperature could have any value previously computed

let comfort-fuzzy fuzzy-max-prod-rule
        (list (list temperature high) (list temperature low) ) low-comfort

 ;; comfort-fuzzy would be the fuzzy set painted in red on Fig. 32 below.

let comfort-crisp [fuzzy-MOM] of comfort-fuzzy
    ;; comfort-crisp would equal 0 in this case
```

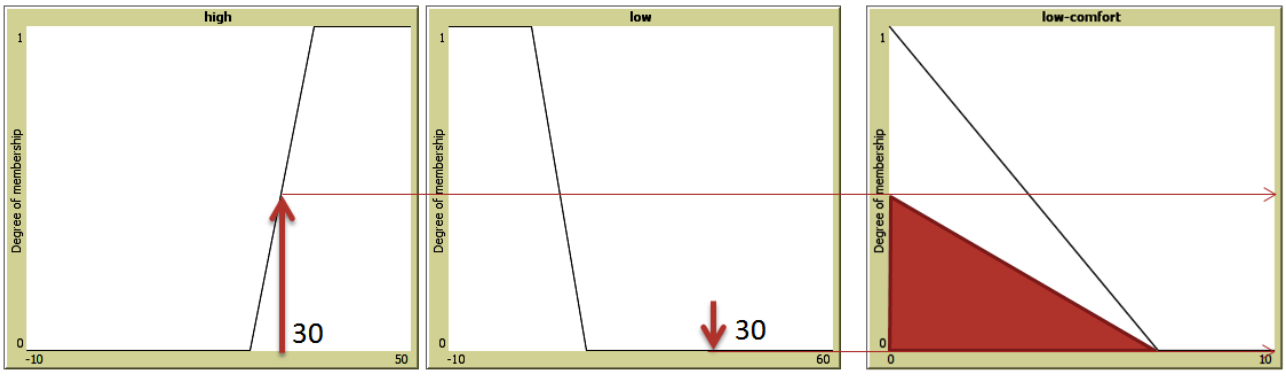


**Fig. 32. Illustration of the fuzzy-or-rule "If temperature is high or temperature is low, my comfort is low", evaluated at temperature = 30, using `fuzzy-max` as OR method and `fuzzy-prod` to modify the consequent fuzzy set.**

## Rule evaluation and aggregation

Rules can be easily combined using so-called aggregation operators. Common choices are **fuzzy-max** (= **fuzzy-or**), **fuzzy-sum**, or **fuzzy-prob-or**. As an example, we include the implementation of the **fuzzy-or** aggregation of the following two rules:

"If temperature is high OR temperature is low, THEN my comfort is low"
"If temperature is average, THEN my comfort is high".

We create a procedure that takes temperature as an input and produces a crisp value for comfort in the interval [0, 10]. (Remember to invoke **fuzzy-start** before calling this procedure.)

```
to-report compute-comfort [temperature]
        ;; Implementation of the fuzzy-or aggregation of the rules:
        ;; "If temperature is high or temperature is low, my comfort is low"
        ;; "If temperature is average, my comfort is high"

 ;; FUZZY SETS
  let high fuzzy-piecewise-linear-set [[-10 0][25 0][35 1][60 1]]
  let low fuzzy-piecewise-linear-set [[-10 1][5 1][15 0][60 0]]
  let avg fuzzy-piecewise-linear-set [[-10 0][10 0][20 1][30 0][60 0]]
  let low-comfort fuzzy-piecewise-linear-set [[0 1][7 0][10 0]]
  let high-comfort fuzzy-piecewise-linear-set [[0 0][3 0][10 1]]

 ;; RULES
  let R1 fuzzy-or-rule
        (list (list temperature high) (list temperature low) ) low-comfort
  let R2 fuzzy-rule
        (list temperature avg) high-comfort

 ;; AGGREGATION
  let comfort-fuzzy fuzzy-or (list R1 R2)

 ;; DEFUZZIFICATION
  let comfort-crisp [fuzzy-MOM] of comfort-fuzzy

 ;; CLEAR MEMORY
  ask (turtle-set high low avg low-comfort high-comfort R1 R2 comfort-fuzzy)
                                                                    [fuzzy-die]
        ;; fuzzy sets are turtles, so it's best if we kill them once we no
        ;; longer use them

  report comfort-crisp
end
```

To see a plot of temperature against comfort, you can use the following code

```
fuzzy-start
let temperatures n-values 70 [? - 10]
let comforts map [compute-comfort ?] temperatures
(foreach temperatures comforts [plotxy ?1 ?2])
```

We advise you to try out different methods of defuzzification and see how different the results are.

# Optimizing the use of memory

Fuzzy sets are implemented as turtles, so it's best if you kill them once you no longer use them. You can do that nicely (i.e. killing also any dependent fuzzy sets) using the procedure **fuzzy-die**. See section "Rule evaluation and aggregation" for an example on how to do this.

# Implementation of the library (advanced material - optional)

## Global variables

The code in the file "fuzzy-logic.nls" declares the following global variables:

```
globals [
  $fuzzy-creator
  $fuzzy-resolution
  $undefined-degree-of-membership
  $undefined-degree-of-fulfilment
]
```

These global variables are initialised in the procedure **fuzzy-start**. The user is welcome to modify the value of the variables **$fuzzy-resolution**, **$undefined-degree-of-membership** and **$undefined-degree-of-fulfilment**, but the value of **$fuzzy-creator** should not be changed. The following describes each of the global variables in turn.

- **$fuzzy-creator** is a special fuzzy set which is used to create all fuzzy sets. This is not a variable meant to be modified by the user.
- **$fuzzy-resolution** is an integer that measures the precision in any process that involves discretization (e.g. plotting a membership function). The value denotes the number of parts in which an interval of length 1 is divided. The default value is 32. The user can change the value of this variable to e.g. 8 by typing: set **$fuzzy-resolution** 8. It is highly recommended to use numbers that are powers of 2 (i.e. 2, 4, 6, 8, 16, 32, …).
- **$undefined-degree-of-membership** is the value returned whenever the membership function is evaluated with an argument for which the function is not defined. The default value is the string "undefined". The user can change the value of this variable to e.g. 0 by typing: set **$undefined-degree-of-membership** 0.
- **$undefined-degree-of-fulfilment** is the value returned whenever the membership function is evaluated with a fuzzy set such that there is no number at which the two sets are defined. The default value is the string "undefined". The user can change the value of this variable to e.g. 0 by typing: set **$undefined-degree-of-fulfilment** 0.

## Built-in variables of fuzzy sets

Fuzzy sets are implemented as a breed of turtles with the following built-in variables:

```
fuzzy-sets-own [
  $description
  $membership-function
  $membership-function-parameters
  $universe
  $continuous?
  $dependent-temporary-sets
]
```

The following describes each of the fuzzy-set built-in variables in turn. The user is not meant to modify the value of any built-in variables.

- **$description** is a string that specifies the type of fuzzy set (i.e. its type of membership function). It can be "discrete", "continuous", "interval with points", "piecewise-linear", "logistic", "gaussian", "exponential", "power", "truncate", "min", "max", "sum", "prob-or", "not", "empty fuzzy set" or "creator".
- **$membership-function** is the actual membership function (implemented as a task).
- **$membership-function-parameters** is a list with all the parameters of the membership function. The structure of the list varies depending on the type of fuzzy set, and is explained in detail in the description of the different methods that can be used to create fuzzy sets from scratch.

  As an example, the structure of the **$membership-function-parameters** list for piecewise-linear fuzzy sets (such as the one shown in Fig. 1) is a list of points of the form: $[[x_1\ y_1]\ [x_2\ y_2]\ ...\ [x_n\ y_n]]$. The membership function is a piecewise linear function that joins all such points $[x_i\ y_i]$.
- **$universe** is a list that specifies the elements of the universe *X*, i.e. the domain of the membership function. For values outside the universe *X*, the membership function is not defined. For fuzzy sets with continuous domain, the **$universe** is a 2-number list denoting a real interval.1 For fuzzy sets with discrete domain, the **$universe** is a non-empty list of numbers (of any length).
- **$continuous?** is a Boolean variable that equals true if the universe *X* of the fuzzy set is continuous and false otherwise. As an example, **$continuous?** is true in fuzzy sets created with the procedure **fuzzy-piecewise-linear-set** and false in fuzzy sets created with the procedure **fuzzy-discrete-numeric-set**.
- **$dependent-temporary-sets** is a list of fuzzy sets, which is potentially empty. It stores fuzzy sets on which the current fuzzy sets depends. These are copies that can be killed safely if the current fuzzy set is not going to be used anymore.

# Bibliography


Bezdek, J. C. (1993). "Fuzzy Models - What Are They, and Why?", *IEEE Transactions on Fuzzy Systems*, 1:1, pp. 1-6.

Klir, G. J., & Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Upper Saddle River, New Jersey: Prentice Hall PTR.

Kantrowitz, M., Horstkotte E., and Joslyn, C. (1993), "Answers to Frequently Asked Questions about Fuzzy Logic and Fuzzy Expert Systems", *comp.ai.fuzzy*, April, 1993, http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/fuzzy/part1/faq.html, mkant+fuzzy-faq@cs.cmu.edu.

The MathWorks, Inc. (2014). *Fuzzy Logic Toolbox™ User's Guide. R2014b*. http://www.mathworks.com.au/help/fuzzy/index.html

Ross, T. J. (2010). *Fuzzy Logic with Engineering Applications*, Wiley. 3rd edition. ISBN-13: 978-0470743768.


# Acknowledgments


This research was supported under Australian Research Council's *Discovery Projects* funding scheme (project number DP130100570, "Modelling Network Innovation Performance Capability: A Multidisciplinary Approach"). The authors are also very grateful to Forrest Stonedahl for several useful comments and suggestions.