# A Visual Programming Language for Qualitative Data

Marita Duecker**, Christian Geiger*, Georg Lehrenfeld*, Wolfgang Mueller**, C. Tahedl*

* Heinz Nixdorf Institut, ** C–LAB
Fuerstenallee 11, 33102 Paderborn, Germany

## 1 Introduction

Modeling of human knowledge and reasoning requires the formulation of uncertainty in its various forms. Fuzzy logic was introduced to directly support these applications [4]. Fuzzy Control (FC) which is based on fuzzy logic allows to control complex systems based on qualitative information like human knowledge [4, 2]. In fuzzy logic, fuzzy sets are usually defined and manipulated by the means of complex mathematics whereas the Fuzzy Control process is frequently outlined by visual sketches based on set diagrams in order to enhance the comprehension of the inference process. The rule–based execution of this process usually follows the lines of rule–based visual programming languages (VPLs), i.e., languages comparable to Agentsheets and ChemTrains. This strongly indicates that VPLs are thus well–applicable for this use.

In this article, we first outline the basic concepts of fuzzy logic and Fuzzy Control. Thereafter, we sketch a visual language which integrates fuzzy set diagrams in the visual representation of rules. The basic concepts are inherited from the complete visual programming language Pictorial Janus (PJ). However, we significantly simplify PJ's visual concepts in order to adapt it for our purpose.

## 2 Fuzzy Control

The individual computation steps of fuzzy control are based on fuzzy set theory. Let $x$ be an element of the universe $\Omega$ and $F \subseteq \Omega$. The characteristic function $\mu_F : \Omega \to \{0, 1\}$ denotes the membership $x \in F$. That is, $\mu_F$ is 1 if $x \in F$ and 0 otherwise. Zadeh [4] redefines the characteristic function in order to map it to an interval by $\mu_F : \Omega \to [0, 1]$ measuring the *grade of membership*. The closer $\mu_F(x)$ is to 1 the more $x$ becomes a member of $F$. $\mu_F$ is called a *fuzzy set* over $\Omega$. Based on crisp (non fuzzy) input data a fuzzy rule base is evaluated by combining all matching rules. A rule is composed of premises (or conditions) and conclusions which are described by fuzzy sets and model qualitative information. A fuzzy control process converts crisp I/O data into fuzzy data and back applying three basic steps: (i) fuzzification, (ii) inference, and (iii) defuzzification.

We briefly outline the basic principles by the behavior of a car when approaching a crossing. Consider the two rules given in Fig. 1 and the crisp input values SPEED=50km/h and DISTANCE=25m. In a first step a membership value is computed from the given input values and the given fuzzy set. Applying rule 1 the given SPEED yields to 0.8 with fuzzy set S_MEDIUM. DISTANCE yields to 0.6 with D_MEDIUM.

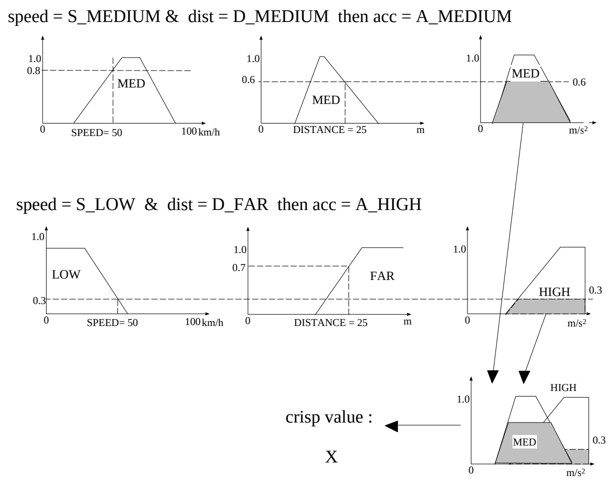


Figure 1: Fuzzy Control Process

The inference step combines the membership values to a degree of fulfillment (DoF) and applies it to the conclusion(s) given by output fuzzy set(s). There are various inference strategies. The widely used max–min inference takes the minimal membership value to determine the DoF in the output fuzzy set [4]. For rule 1, the cut for the DoF for fuzzy set A_MEDIUM is at 0.6 and for rule 2 the cut for A_HIGH is at 0.3. If more than one cut is greater than 0.0 (more than one rule fires) the resulting sets are combined by a union. In a final step the union is transformed into a crisp value. The center–of–area method is one possible approach for such a transformation.

## 3 Fuzzy Visual Language

We can identify a significant correspondence between the VPL Pictorial Janus [3] and fuzzy control processes as it is shown by Geiger's and Lehrenfeld's fuzzy extension for Concurrent Prolog [2]. In this article, we restrict our investigations on the behavior of an individual agent rather than on the network of communicating agents. Corresponding to a PJ agent a fuzzy control process is defined by a set of rules. Each PJ rule has preconditions and a behavior. A fuzzy rule has premises and a conclusion. The conclusion can be interpreted as a very simple form of a PJ agent behavior. However, whereas PJ's execution model of an individual agent is based on pattern matching with a possibly non–deterministic selection of one rule fuzzy control elaborates all possible alternatives and combines the results.

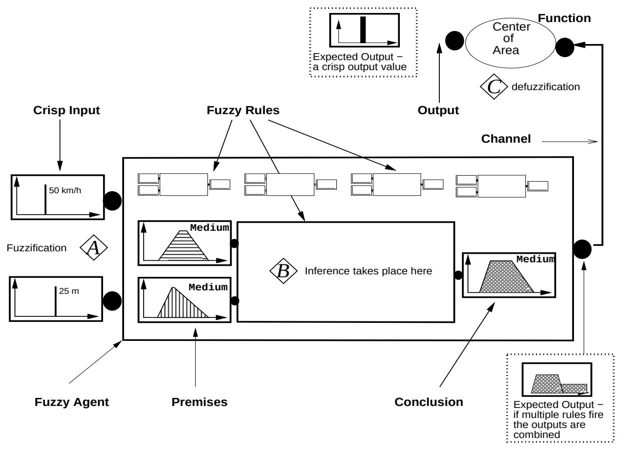
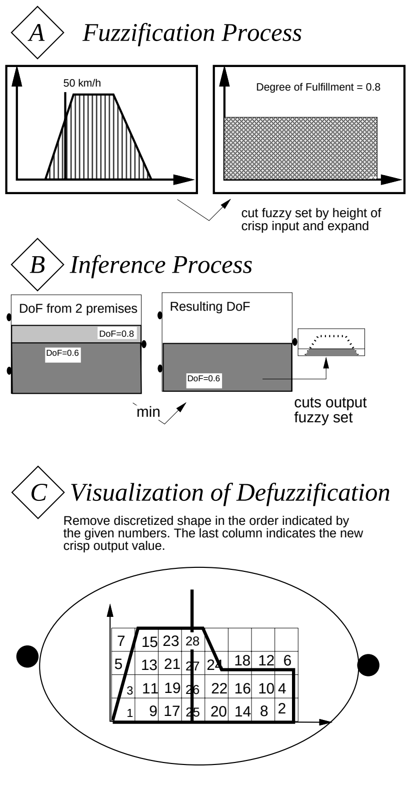


Figure 2: Agent with Fuzzy Rules

Figure 2 gives the definition of a fuzzy controller (agent) with one magnified rule inside. The rule's preconditions are at its left whereas its conclusion is at its right. The fuzzy sets are visually given inside the contour of the preconditions and the conclusion. *Fuzzification* is performed by visually matching the crisp input objects with the corresponding preconditions. When matching, a visual AND–operation is applied and the result is horizontally expanded (see Fig. 3A). The *inference* process combines the results of all input/condition pairs applying an AND–operation. The result then cuts the output fuzzy set (see Fig. 3B). If more than one rule fires (DoF $> 0.0$) all cut output sets are combined by a visual OR–operation (see Fig. 2). The *defuzzification* is visually realized using the algorithm outlined in Fig. 3C.

This specifies a visual language which correctly implements the fuzzy control process including commonly used fuzzy set operations (AND/OR/NOT) and defuzzification methods. Our work demonstrates PJ's flexibility for deriving domain–specific languages. This short article only sketches the static description of fuzzy controllers. We expect promising results when elaborating this approach in the direction of an advanced dynamic visual representation, i.e., animation of fuzzification, inference, and defuzzification. The implementation of our PJ environment JIM [1] has demonstrated that interactive visual animation environments greatly support the debugging of visual programs when visualizing the individual computation steps. This significantly improves the specification and validation of fuzzy rule bases which are currently implemented and checked applying textual–based trial–and–error methods.



A Fuzzification Process
50 km/h
Degree of Fulfillment = 0.8
cut fuzzy set by height of crisp input and expand
B Inference Process
DoF from 2 premises
DoF=0.8
DoF=0.6
Resulting DoF
DoF=0.6
min
cuts output fuzzy set
C Visualization of Defuzzification
Remove discretized shape in the order indicated by the given numbers. The last column indicates the new crisp output value.
7 15 23 28
5 13 21 27 24 18 12 6
3 11 19 26 22 16 10 4
1 9 17 25 20 14 8 2

Figure 3: Visual Fuzzy Inference

## References

[1] M. Duecker, G. Lehrenfeld, W. Mueller, and C. Tahedl. A Generic System for Interactive Real–Time Animation. In *Proc. of the IEEE Int. Conf. on Engineering of Computer–Based Systems (ECBS'97)*, Monterey, CA, March 1997.

[2] C. Geiger and G. Lehrenfeld. The Application of Concurrent Fuzzy Prolog in the Field of Modelling Flexible Manufacturing Systems. In L. Sterling, editor, *Int. Conf. on Practical Application of PROLOG*, pp. 233– 252, London, April 1994.

[3] K. Kahn and V.A. Saraswat. Complete Visualization of Concurrent Programs and their Execution. In *1990 IEEE Workshop on Visual Languages*, pp. 7–15, Skoje, IL, October 1990.

[4] Hans-Jürgen Zimmermann. *Fuzzy Set Theory — and Its Applications*. Kluwer Academic Publishers, 2nd, revised edition, 1991.