Applying Answer Set Programming with Fuzzy Membership Functions: a Case Study

Luca Ferragina[1], Ilenia Galati[1,*], Lorena Gullone[1] and Francesco Scarcello[1]

[1]DIMES Dept., University of Calabria, 87036 Rende (CS), Italy.

Abstract

Human reasoning often operates through qualitative concepts expressed by linguistic labels such as *high*, *low*, *expensive*, or *cheap*, whose interpretation depends on context and is usually vague, despite being rooted in numerical data. This paper explores a novel fuzzy-logic-based qualitative extension of Answer Set Programming (ASP) to bridge numerical information and qualitative reasoning. The underlying language, formally introduced in a separate work, provides a principled framework that avoids rigid thresholds and supports robust reasoning under vagueness.

Focusing on a representative use case, we illustrate how the framework integrates numerically grounded inputs (such as outputs of machine learning models) with symbolic reasoning over qualitative labels. Key features, including learning-based membership functions and semantically enriched predicates, enable the combination of expert knowledge, contextual factors, and subjective interpretations within a unified declarative setting.

Keywords

Fuzzy Logic, Answer Set Programming, Neuro-symbolic AI, Membership Functions

1. Introduction

Recent advances in machine learning have significantly enhanced the ability to extract patterns from unstructured data. However, despite their predictive accuracy, such systems often remain difficult to interpret and audit, particularly in applications where decisions must be traceable and grounded in explicit assumptions. In contrast, symbolic reasoning frameworks offer transparency and formal rigor, but they typically struggle to capture the continuous, noisy, and context-dependent nature of real-world information. This gap becomes especially evident when expert knowledge is expressed through qualitative notions (e.g., *high*, *low*, *severe*, *acceptable*) that are inherently vague, even when derived from numerical measurements. Traditional approaches often rely on rigid thresholds to map numerical values to qualitative categories, leading to brittle behavior in which small variations in input may cause abrupt and unintuitive changes in outcomes.

Fuzzy logic offers a natural foundation to address this issue by enabling reasoning with graded truth values. In this work, we build on a novel qualitative, label-based fuzzy extension of Answer Set Programming (ASP), introduced in a companion paper, which explicitly couples linguistic labels with numerically grounded information through membership functions. In this framework, predicates are enriched with sets of qualitative labels, and their truth degrees are constrained both by data-driven membership functions (for input predicates) and by semantic compatibility conditions (for derived predicates), expressed through admissible truth profiles. Let us illustrate this point through a rule extracted from a program in a travel recommendation scenario:

$$\texttt{recommend}(X, C; \textit{affordable}) \leftarrow \texttt{client}(C) \odot \texttt{price}(F, C; \textit{cheap}) \odot \texttt{flight}(X, F) \odot \texttt{hotel}(X, H) \odot \texttt{price}(H, C; \textit{cheap})$$

24th International Workshop on Nonmonotonic Reasoning, July 17–19, 2026, Lisbon, Portugal

*Corresponding author.

luca.ferragina@unical.it (L. Ferragina); ilenia.galati@unical.it (I. Galati); lorena.gullone@unical.it (L. Gullone); francesco.scarcello@unical.it (F. Scarcello)

0000-0003-3184-4639 (L. Ferragina); 0009-0006-3566-7296 (I. Galati); 0009-0004-8142-0513 (L. Gullone); 0000-0001-7765-1563 (F. Scarcello)

The predicate `recommend` has labels *infeasible* and *affordable*, while `price` is described by *cheap*, *medium*, and *expensive*. The rule assigns to *affordable* a degree bounded by the aggregation (via a t-norm $\odot$, e.g., $\odot_{\min}$) of the body, thus reflecting that affordability depends on both flight and hotel being sufficiently cheap.

Note this notion is vague, and the mapping from actual money values to the degree of *cheapness* is provided by suitable membership functions associated with the predicate `price`. Such mapping can be determined by those studies that relate the actual perceived value of money for different individuals, typically depending on their annual income. That is, the notion of *cheap* is not universal, but depends on contextual and subjective factors: \$200 is a medium price for a rich customer, but it is very expensive for a low-income customer. Thus, in this fuzzy program, economic status is treated not as a passive filter but as a semantic dimension that adapts the concept definition. In particular, the same travel package can therefore receive different graded valuations of *affordability* from different users.

This example reflects a general setting where numerical information (often produced by statistical or learning methods) is integrated into a logical framework, allowing experts to express rules naturally while retaining control over graded outcomes.

It is worthwhile noting that our language is a direct extension of classical, two-valued ASP, and it naturally permits the definition and manipulation of crisp predicates and rules. Whenever there are only crisp predicates, all fuzzy operators collapse to their classical counterparts, namely conjunction, disjunction, and negation, and we recover precisely Answer Sets semantics.

The focus of this paper is on illustrating the practical effectiveness of this approach in a concrete setting, namely damage detection in the production line of screws, arising from a collaboration with an industrial partner. In such environments, automated visual inspection systems produce numerical features (such as defect size, shape irregularity, or color intensity) derived from image processing and machine learning components. These features must then be interpreted in qualitative terms that align with domain expertise, such as assessing whether a defect is *small*, *moderate*, or *severe*, and whether a screw is *repairable* or should be *discarded*.

Our framework enables this transition by associating each qualitative label with a membership function that maps numerical observations to degrees of membership, thus preserving the inherent vagueness of the domain. On top of these input predicates, higher-level concepts are defined through declarative rules, while admissible truth profiles ensure that the resulting qualitative assessments remain semantically coherent. This allows the system to detect inconsistencies when conflicting evidence arises, and to propagate partial information across related qualitative dimensions.

We illustrate how this approach supports a full reasoning pipeline for quality control: from low-level perception, through qualitative abstraction, to decision-making actions such as *discard*, *rework*, or *accept*. The example highlights how numerically grounded data and symbolic rules can be seamlessly integrated, yielding outcomes that are both flexible and aligned with domain-specific interpretations. While the formal properties of the language are proved in the companion work, this paper emphasizes its role in enabling a robust and expressive modeling of real-world decision processes in a concrete use-case.

The main contributions of this paper are: (i) the instantiation of the proposed qualitative fuzzy ASP framework in a realistic industrial damage-detection scenario, and (ii) the design and prototyping of a complete reasoning pipeline integrating data-driven membership functions with rule-based qualitative inference.

2. Related Works

The integration of uncertainty in rule-based reasoning has evolved from quantitative extensions of logic programming to neuro-symbolic approaches. Early works [1] introduced attenuation factors, while Fuzzy Logic Programming (FLP) [2, 3, 4, 5] adopted a truth-functional view. These ideas were later extended to structured formalisms such as Description Logics and Answer Set Programming (ASP).

Handling uncertainty with non-monotonic negation has been studied in generalized logic programs. Parametric Deductive Databases were extended in [6, 7] with approximate well-founded semantics and lattice-based frameworks. In parallel, multi-adjoint logic programming was enriched with non-monotonic reasoning [8, 9], providing expressive semantics and translations to simpler fragments.

In fuzzy Description Logics [10, 11, 12], vagueness is modeled via membership functions while preserving decidability. In ASP, fuzzy extensions [13, 14] generalize stable models using $[0, 1]$ truth degrees. CFASP [15] showed that many constructs can be reduced to a core language. Further studies addressed negation and inconsistency [16, 17], introducing notions such as unfounded-freeness. Fuzzy Linguistic Logic Programming [18] instead performs reasoning directly over linguistic terms. More recently, fuzzy Datalog with existential rules [19] achieved high expressivity while retaining tractable data complexity. Neuro-symbolic AI complements these developments by combining learning and reasoning. *DeepProbLog* [20] and *NeurASP* [21], along with *SLASH* [22], integrate neural outputs into probabilistic or ASP-based reasoning, while Logic Tensor Networks [23] embed logical constraints into differentiable frameworks.

Note that all these approaches clarify, from different perspectives, how non-monotonic reasoning, uncertainty management, and graded truth can coexist. However, they do not consider membership functions and qualitative labels as main components of the semantics of programs, which includes dealing with the implicit constraints based on how membership functions of qualitative labels may co-vary.

3. Preliminaries

In this section, we introduce the core elements of our language, including its syntax, the fundamental components of its semantics, and the main algorithm underlying the inference process.

3.1. Syntax of f-Datalog

A program Π in f-Datalog includes a set $\mathcal{C}$ of constants, a set of $\mathcal{V}$ variables, and a set $\mathcal{P}$ of predicate symbols. Constants and variables play the same role as in standard logic programming. Predicates, however, are enriched with additional structure to capture linguistic terms and fuzzy information.

As usual in Datalog programming, we distinguish two classes of predicate symbols in $\mathcal{P}$: i) *Input predicates* (EDB), whose truth degrees are given as inputs to the program; ii) *Derived (or intentional) predicates* (IDB), whose truth degrees are obtained by rule-based inference within the program.

Each predicate symbol $p^{(k)} \in \mathcal{P}$ of arity k is associated with a triple $\langle \Lambda_p, \mathrm{M}_p, \Sigma_p \rangle$, where Λ_p is a finite set of *linguistic labels* describing the linguistic domain of p, M_p is a family of *membership functions*, and Σ_p is a collection of *S-norms*.

For each label $\ell \in \Lambda_p$, the set Σ_p provides an associated S-norm $\mathcal{S}_\ell : [0,1] \times [0,1] \to [0,1]$, which satisfies commutativity, associativity, monotonicity, and the boundary condition $\mathcal{S}_\ell(x, 0) = x$. Typical examples include the Gödel S-norm, defined as $\mathcal{S}_{\max}(x, y) = \max(x, y)$, and the probabilistic sum, defined as $\mathcal{S}_{\mathrm{prob}}(x, y) = x + y - x \cdot y$. The S-norm $\mathcal{S}_\ell$ aggregates the contributions to the truth/membership value for label ℓ from the rules defining atoms over predicate p, while membership functions are used to determine the degree to which a given value satisfies a linguistic label.

For an EDB predicate p, the family M_p contains, for each label $\ell \in \Lambda_p$, a membership function $\mu_\ell : \mathbf{D} \to [0, 1]$, where $\mathbf{D}$ denotes the cartesian product of the domains of the underlying variables. Each function μ_ℓ assigns to a tuple $\mathbf{x} \in \mathbf{D}$ a degree of membership in the fuzzy set associated with the label ℓ. Membership functions are fixed *a priori*, for instance on the basis of domain expertise or by means of machine learning techniques applied to data [24]. The resulting membership degree $\mu_\ell(\mathbf{x})$ is interpreted in the standard fuzzy-set-theoretic sense: $\mu_\ell(\mathbf{x}) = 0$ denotes non-membership, $\mu_\ell(\mathbf{x}) = 1$ denotes full membership, and intermediate values express partial membership (see Figure 1).

We can view M_p as a unique function $\gamma_p : \mathbf{D} \to [0, 1]^{\Lambda_p}$, where each membership function component provides the result over the dimension associated with its specific label. The relation containing all

compatible values for those membership functions, which we denote by $\gamma_p \subseteq [0,1]^{\Lambda_p}$, is precisely the image of M_p.

For a derived predicate IDB p, we would like to compute membership values by means of program rules, according to the intended semantics of the program. However, not every combination of truth values for the labels are logically meaningful, thus for IDB predicates we are interested in the membership-values compatibility set γ_p. With this respect, it is convenient to represent the membership functions in a parametric way, with respect to some parameter t, so that γ_p can be conveniently represented as a mapping $\gamma_p : [a_p, b_p] \to [0,1]^{\Lambda_p}$, where the *closed interval* $[a_p, b_p]$ is the parametrization domain, and $\forall t \in [0,1]$, $\gamma_p(t) = (\gamma_p(t)_\ell)_{\ell \in \Lambda_p}$ represents a coherent assignment of truth degrees to all labels of p. Intuitively, γ_p describes the intended semantic structure of the predicate p: each label $\ell \in \Lambda_p$ corresponds to one dimension of the truth space, and the curve γ_p encodes how these labels may consistently co-vary.

A *term* is either a constant $c \in \mathcal{C}$ or a variable $x \in \mathcal{V}$. A *fuzzy literal* is an expression of the form $p(\mathbf{x}; \ell)$ or $\neg_{\mathcal{N}} p(\mathbf{x}; \ell)$, where $p \in \mathcal{P}$ is a predicate symbol of arity k, $\mathbf{x} = (x_1, \ldots, x_k)$ is a tuple of terms, and $\ell \in \Lambda_p$ is a linguistic label associated with p.[1] The former is referred to as a (positive or *directed*) atom, while the latter as a *negated* atom. Formally, a negated atom is denoted as $\mathcal{N}(p)$ where $\mathcal{N} : [0,1] \to [0,1]$ is a decreasing function that satisfies $\mathcal{N}(0) = 1$ and $\mathcal{N}(1) = 0$. Typical negator examples include the *Łukasiewicz Negator* $\mathcal{N}_L(x) = 1 - x$ and the *Gödel Negator* $\mathcal{N}_G(x) = 1$ if $x = 0$ or $\mathcal{N}_G(x) = 0$ otherwise.

A *fuzzy Datalog rule* is an expression of the form

$$h \leftarrow b_1 \odot_{\mathcal{T}} \cdots \odot_{\mathcal{T}} b_m,$$

where the head h is a (positive) fuzzy atom and the body consists of a (possibly empty) conjunction of fuzzy literals $b_1, \ldots, b_m$. The operator $\odot_{\mathcal{T}}$ denotes conjunction interpreted via some T-norm $\mathcal{T}$. Intuitively, the body specifies a set of conditions whose satisfaction contributes to the truth value of the head.

For a rule r, denote by $H(r)$ its head atom and by $B(r)$ the set of literals in its body where $B^+(r)$ and $B^-(r)$ denote, respectively, the sets of positive and of negated atoms occurring in the body of r. Then, consider a (ground) atom h, and denote by R_h the set of rules having h as their head.

We assume the existence of a reserved predicate $const(x; \ell_c)$ where x is some constant term. It is used to encode constant truth values within the rules and it is equipped with a specific membership function $\ell_c(x) = 1$ assigning truth value 1 to any given term x.

The logical conjunction AND is modeled by a T-norm $\mathcal{T} : [0,1] \times [0,1] \to [0,1]$, which satisfies commutativity, associativity, monotonicity, and the boundary condition $\mathcal{T}(x, 1) = x$. Commonly used examples include the Gödel T-norm $\mathcal{T}_{\min}(x, y) = \min(x, y)$, the product T-norm $\mathcal{T}_{\mathrm{prod}}(x, y) = x \cdot y$, and the Łukasiewicz T-norm $\mathcal{T}_L(x, y) = \max(0, x + y - 1)$.

A *fuzzy (Datalog) program* Π is a finite set of rules. A *fact* is a rule with empty body. The *grounding* of a program Π is obtained by replacing all variables occurring in Π with constants from the domain. A program is said to be *ground* if it contains no variables. The *Herbrand domain* H_C of a program Π is the set of all constants appearing in Π. The *Herbrand base* $\mathcal{B}_\Pi$ is the set of all ground fuzzy atoms that can be constructed from the predicate symbols in $\mathcal{P}$ and the constants in H_C.

3.2. Semantics

A *fuzzy interpretation* $\mathcal{I}$ is a function $\mathcal{I} : \mathcal{B}_\Pi \to [0,1]$ that assigns a truth degree to each ground atom $p(\mathbf{C}; \ell)$ in the Herbrand base $\mathcal{B}_\Pi$. The value $\mathcal{I}\big(p(\mathbf{C}; \ell)\big)$ is interpreted as a truth degree of the atom with respect to the label ℓ, or as a degree of membership to the fuzzy set ℓ. As we discuss below, classical (crisp) atoms can be modeled as atoms with a membership function having binary values only.

Denote by $p(\mathbf{C}; \cdot)$ the basic unlabeled version of any ground atom with over predicate p and with terms $\mathbf{C}$. By abusing notation, we simply refer to it as an atom.

[1] In the examples, we omit the label if there is only one label and it is irrelevant, for instance for crisp EDB predicates.

Membership functions. An interpretation $\mathcal{I}$ is said to be *consistent* if, for every atom $p(\mathbf{C};\cdot)$, the truth values over its linguistic labels are compatible according to γ, that is,

$$\big(\, \mathcal{I}\big(p(\mathbf{C};\ell)\big)\,\big)_{\ell\in\Lambda_p} \in \gamma_p \tag{1}$$

Equation (1) therefore requires that truth degrees of $p(\mathbf{C},\cdot)$ according to $\mathcal{I}$ belong to the admissible region γ_p (or they are all zero—recall that $\mathbf{0}^{\Lambda_p} \in \gamma$, if this atom is false at all). Note that we do not allow arbitrary and independent truth values to individual labels, because membership functions carry out semantic information. Typically, e.g., it is impossible that an atom corresponding to some physical measure has a high membership value for both the label "fast" and the label "slow".

In our language, however, the membership functions encoded in γ play a much more central role than mere consistency checking: they are directly involved in the inference process, since once a truth value for a predicate p is derived for a label, the profile γ_p can propagate consistent truth values to other labels (line 6 of Algorithm 1). More precisely, if $\mathcal{I}(p(\mathbf{C};\ell)) = v$, then the profile γ_p determines a set of admissible tuples in $[0,1]^{\Lambda_p}$ containing v as the component for ℓ, and the inference step selects (e.g., minimally) a tuple $\mathbf{t} \in \gamma_p$ such that $\mathbf{t}(\ell) \geq v$, assigning $\mathcal{I}(p(\mathbf{C};\ell')) \geq \mathbf{t}(\ell')$ for every other label $\ell' \in \Lambda_p$. By this mechanism, partial evidence derived for a single label can be extended to a semantically coherent interpretation over all labels, preserving the intended qualitative relationships encoded in γ_p and avoiding arbitrary or inconsistent assignments. This feature can also be exploited to allow a single predicate p to exhibit both a fuzzy and a crisp behavior. For instance, p can be equipped with two labels, e.g., *fuzzy* and *crisp*, each associated with a different membership function: a continuous increasing function (e.g., linear or even the identity on $[0,1]$) for *fuzzy*, and a 0-1 step function at a fixed cutoff value for *crisp* (see Figure 2). In this setting, the label *fuzzy* is used in the head of fuzzy rules to derive graded truth values, while the γ-propagation mechanism automatically induces a corresponding Boolean value for $p(\cdot;\textit{crisp})$, which will be either 1 if the threshold degree exceeds the given cutoff, or 0 otherwise. By construction, the label *crisp* thus exhibits a crisp behavior, which can be seamlessly used in subsequent (standard ASP) rule layers requiring Boolean semantics. This pattern is indeed exploited in the program of Figure 3, where predicates such as `discard`, `discount`, and `repair` are first derived fuzzily and then used within a purely crisp decision component, by means of their crisp (binary valued) labels.

Models and Answer Sets. For a rule r, $\mathcal{I}(B(r))$ is obtained by applying the T-norm $\mathcal{T}_r$ to the interpretation $\mathcal{I}(b_i)$ of all literals $b_i \in B(r)$. Recall that $\mathcal{T}_r$ is associative and commutative. Then, $\mathcal{I}(R_h)$ denotes the application of the co-norm associated with h to the interpretation of all body rules defining h, that is, $\mathcal{I}(R_h) = \mathcal{S}_h\{\, \mathcal{I}(B(r) \mid r \in R_h \,\}$.

A *model* of a fuzzy program Π is a consistent interpretation $\mathcal{I}$ (in the sense of Equation (1)) such that all rules in Π are satisfied. That is, for each head atom h, it holds that

$$\mathcal{I}(h) \geq \mathcal{I}(R_h) \tag{2}$$

We write $\mathcal{I} \models \Pi$ to denote that $\mathcal{I}$ is a model of Π.

Definition 1 (Minimal model). *An interpretation $\mathcal{I}$ for a program Π is said to be* minimal *w.r.t. a given set of interpretations W, if there exists no other interpretation $\mathcal{I}' \in W$ ($\mathcal{I}' \neq \mathcal{I}$), such that $\mathcal{I}'(\bar{p}) \leq \mathcal{I}(\bar{p})$, for every ground atom $\bar{p}$.*

An interpretation $\mathcal{I}$ is called a minimal model of Π, if it is minimal w.r.t. to the set of all models of Π.

It is easy to see that a program may have many minimal models, even infinitely many, or no model at all. However, we are interested in those models that are maximally supported by the program rules.

Definition 2 (Tight Interpretations). *Let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations of a fuzzy program Π. We say that $\mathcal{I}$ is* tight *w.r.t. $\mathcal{I}'$ if, for every (unlabeled) atom $p(\mathbf{C};\cdot)$, there is a maximal set of critical labels, whose membership values are the lowest possible to satisfy their defining rules w.r.t. the given $\mathcal{I}'$. That is, for every such a label ℓ^*, said $\bar{p} = p(\mathbf{C};\ell^*)$, we have $\mathcal{I}(\bar{p}) = \mathcal{I}'(R_{\bar{p}})$. If $\mathcal{I} = \mathcal{I}'$, we just say it is a tight interpretation.*

Tight minimal models contain no superfluous truth information: decreasing the truth (membership) value of a label necessarily destroys model-hood (either by violating consistency or by failing to satisfy some rule). They are maximally informative solutions compatible with both the rules of the program and the semantic constraints imposed by γ. If a program has a unique tight minimal model, that model is called the *least model* of the program.

Given an interpretation $\mathcal{I}$ of a fuzzy program Π, define the *reduct* $\Pi^{\mathcal{I}}$ of Π with respect to $\mathcal{I}$ as the set of rules $Pi^{\mathcal{I}} = \{\, r^{\mathcal{I}} \mid r \in \Pi \,\}$, where each rule $r^{\mathcal{I}}$ is obtained from r by replacing every negated body atom $\neg_{\mathcal{N}} b$ with a (positive) atom $const(\mathcal{N}(\mathcal{I}(b)); \ell_c)$. That is, we assume the truth values of negative literals are correct. By construction, $\Pi^{\mathcal{I}}$ is a positive fuzzy program.

Definition 3 (Fuzzy answer set). *An interpretation $\mathcal{I}$ is an* answer set *of Π if $\mathcal{I}$ is a* tight minimal model *of the reduct $\Pi^{\mathcal{I}}$.*

Algorithm 1: Fixpoint iteration.

Input: Positive γ-monotonic fuzzy program
Output: Least model $\mathcal{M}^*$ (or `NoModel`)

Construct the Herbrand base $\mathcal{B}_\Pi$;
Initialize $\mathcal{I} \leftarrow \bot$;
repeat
  `// Deterministic one-step operator` $\widehat{T}$
  foreach *grounded atom* $p(\mathbf{C}; \cdot)$ **do**
    `// rule-induced lower bounds`
    foreach $\ell \in \Lambda_p$ **do**
      $b_\ell \leftarrow \mathcal{I}(R_{p(\mathbf{C};\ell)})$;
    `// unique tight propagation via` γ_p
    $A \leftarrow \{\, t \in [a_p, b_p] \mid \gamma_p(t)_\ell \geq b_\ell \; \forall \ell \in \Lambda_p \,\}$;
    if $A = \emptyset$ **then**
      return *NoModel*
    $\alpha \leftarrow \min A$;
    foreach $\ell \in \Lambda_p$ **do**
      $\mathcal{I}'(p(\mathbf{C}; \ell)) \leftarrow \gamma_p(\alpha)_\ell$;
  $\mathcal{I} \leftarrow \mathcal{I}'$;
until $\mathcal{I}' = \mathcal{I}$;
return $\mathcal{M}^* \leftarrow \mathcal{I}$;

Let $\mathcal{I}$ be an interpretation for a program Π. Let $W_{\mathcal{I}}$ be the set of interpretations $\mathcal{I}'$ that satisfy all rules according to $\mathcal{I}$, that is, for every atom $\bar{h}$, $\mathcal{I}'(\bar{h}) \geq \mathcal{I}(R_{\bar{h}})$.

We define the *immediate consequence operator* as the mapping $T_\Pi : [0,1]^{\mathcal{B}_\Pi} \to \mathcal{P}\big([0,1]^{\mathcal{B}_\Pi}\big)$ such that $T_\Pi(\mathcal{I})$ is the set of the tight interpretations in $W_{\mathcal{I}}$ that are minimal w.r.t. $W_{\mathcal{I}}$. Moreover, for a set of interpretations $\mathbf{I}$, define $T_\Pi(\mathbf{I}) = \{T_\Pi(I) \mid \mathcal{I} \in \mathcal{I}\}$.

Definition 4 (γ-monotonic programs). *A program Π is called* γ-monotonic *if, for each rule $r \in \Pi$ the component function $\gamma_p(t)_\ell : [a_p, b_p] \to [0,1]$ relative to the predicate p and the linguistic label ℓ that appears in $H(r)$ is continuous and (strictly) increasing on $[a_p, b_p]$.*

It can be shown that the γ-monotonicity property of a program Π guarantees the existence of a least model, which can be computed via Algorithm 1. In contrast, non-monotonic membership profiles may induce nondeterministic behavior, giving rise to multiple tight minimal models.

Furthermore, as observed in other fuzzy logic-based frameworks [15] even in the monotonic setting, the inference procedure may fail to terminate in a finite number of steps, due to a lack of convergence

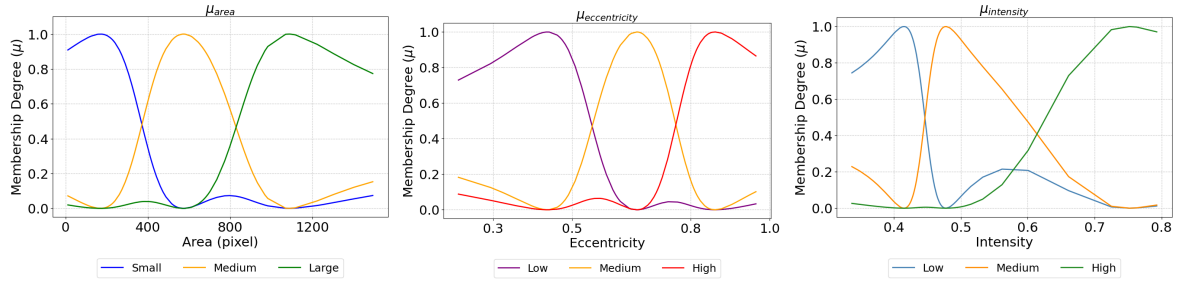


Figure 1: Membership functions of the predicates `area_dimension`, `eccentricity`, and `color_intensity`, obtained from sub-simbolic components.

to the least model. In the framework considered in this paper, however, this is not an issue because we restrict ourselves to a fragment that guarantees polynomial time evaluation (for programs with stratified negation).

4. Case Study: Industrial Quality Control

In this section, we present a program $\Pi_{control}$ for a case study arising from a collaboration with an industrial partner, aimed at analyzing images acquired from a screws production line. The goal is threefold: (i) to detect the presence of damages, (ii) to assess their severity, and (iii) to determine the appropriate action for defective items. As a benchmark, we rely on the well-known MVTec dataset [25], which contains images of objects categorized as normal or defective, with various types of damages.

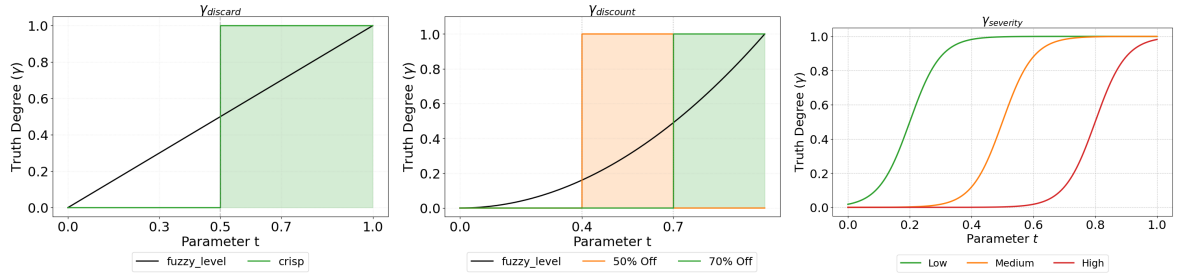
The implemented system can be conceptually divided into two layers: the first, sub-symbolic, is based on machine learning algorithms that provide the values for the membership functions of the input (EDB) predicates of the symbolic layer. The latter, based on fuzzy reasoning, logically combines this information through the program $\Pi_{control}$, shown in Figure 3. to obtain the desired results according to the (vague) indications of domain experts.

Note that $\Pi_{control}$ has two modules: the first one, which defines the fuzzy predicates linked to the sub-symbolic methods that process input images, identifies anomalous products and extracts relevant linguistic features describing the detected damages. The second one consists of crisp rules that, based on the inferred qualitative descriptions, support decision-making about the handling of defective items. In fact, the proposed language allows us to seamlessly combine rules and predicates with fuzzy values with rules having crisp truth values, which behave exactly as in standard ASP programs.

Each item is defined by a tuple (X, I), where each instance of X is a unique identifier associated with an image tensor I. The information is enriched by a neural-generated anomaly heatmap H linked to the image I. The anomaly heatmap H highlights pixels corresponding to potential surface defects. This is described by the two facts $\texttt{object}(X, I)$ and $\texttt{anomaly}(I, H)$. Note that these facts are encoded by crisp predicates, indeed we omit linguistic labels in these predicates (formally, we should use a dummy label, say *true*, with a binary membership function).

The sub-symbolic layer combines: (i) AE-XAD [26], a recent explainable anomaly detection method based on autoencoders, and (ii) three instances of fuzzy k-means clustering [27]. The adoption of AE-XAD is primarily motivated by its ability to produce interpretable anomaly heatmaps that highlight specific pixels corresponding to potential surface defects, providing localized and explainable visual evidence. Furthermore, we rely on fuzzy k-means to characterize each detected damage in terms of area dimension, shape eccentricity, color intensity, distance from the center of the image, and solidity. The choice of a fuzzy clustering approach is crucial to manage the ambiguity of irregular shapes and physical defects. Instead of forcing a rigid threshold, it provides graded truth degrees that naturally define the membership functions of the input EDB predicates (`area_dimension`, `eccentricity`, `color_intensity`, `distance`, and `solidity`), preserving the vagueness of the domain. The linguistic labels of these predicates are reported in the first block of Table 1, and the corresponding membership functions are displayed in Figure 1.

It is important to note that the raw truth degrees produced by sub-symbolic components (e.g., fuzzy k-means) may occasionally exhibit semantically unintuitive behaviors, such as assigning non-zero degrees to both *small* and *large* without a clearly dominant *medium* value, as observed for the predicates `area_dimension`, `eccentricity`, and `color_intensity` in the program $\Pi_{control}$ (see Figure 1). Such phenomena are intrinsic to membership functions derived from machine learning systems. However, as we show later, the symbolic fuzzy component of our framework is robust to these irregularities: by enforcing semantic constraints and leveraging well-designed rules, it mitigates the imprecision of sub-symbolic outputs, yielding more coherent, precise, and interpretable results.



(a) Predicates `discard` and `repair`.

(b) Predicate `discount`.

(c) Predicates `severity` and `defect`.

Figure 2: Membership functions for the predicates `discard`, `discount`, `repair`, and `severity`. The predicates `discard` and `repair` are equipped with a simple fuzzy-to-crisp pattern, with a continuous label (*fuzzy*) and a corresponding Boolean label (*crisp*) obtained via a single cutoff. The predicate `discount` instead adopts a multi-threshold scheme, where three distinct step functions map the fuzzy degree to crisp decisions corresponding to different discount levels (50% and 70%). The predicates `severity` and `defect` are entirely fuzzy (no crisp labels) with γ-monotonic truth profiles.

$\Pi_{control}$ Rules. The symbolic layer of $\Pi_{control}$ employs a set of rules to evaluate anomaly severity, classify defects, and determine the optimal handling strategy. Initially, the framework assesses the overall severity of detected anomalies by evaluating their geometric and visual features, assigning, by means of the rules (P1) and (P2), high severity to anomalies with significant area, high eccentricity, or elevated color intensity, while classifying small, faint anomalies as low severity, through rule (P3). At the same level, the rules (P4), (P5), and (P6) semantically distinguish between functional and aesthetic defects based on the physical traits of the screws. Functional defects are typically characterized by low solidity or specific combinations of small size, low color intensity, and high distance from the center, whereas aesthetic damages are identified through high solidity, low distance, and medium color intensity. In particular, note that the parts farthest from the center are precisely the head and the tip of the screw, whose possible absence in the image is clearly associated with defects that compromise the functionality of the screw under examination.

Based on these inferred classifications and their corresponding truth degrees, the subsequent block of rules infers the appropriate handling actions. Objects are flagged for discarding if they exhibit severe functional defects (rules (P7) and (P8)), or multiple critical aesthetic anomalies (rule (P9)). On the other hand, rule (P10) provides a recommendation for discounting of objects with high-severity aesthetic damages. Repair strategies, encoded in rules (P11) and (P12), are assigned based on the defect profile: internal repairs target minor aesthetic flaws lacking severe functional implications, while external repairs are reserved for minor functional defects requiring specialized processing. That is, minor damages can be handled internally by the company, whereas more severe but still repairable damages must be delegated to specialized external companies. Finally, items designated for external repair that do not meet the crisp discard criteria are automatically routed to an available repair company (rule (P13)) and the price of the items designated for discount is updated (rules (P14), (P15), (P16)).

We point out that these final rules behave precisely as in standard ASP, because all their body predicates use only binary (crisp) step membership functions, and thus will be either (entirely) true

or (entirely) false in any interpretation of the program. To see how this works, consider for instance the predicate `discard`: the compatibility γ of membership functions (see Figure 2) implies that any *fuzzy_level* value exceeding a certain threshold automatically sets the value 1 (entirely true) for the step membership function we have called *crisp*, which is then used in the subsequent rules (e.g., rule (P13)). The same happens for predicates `repair` and `discount`.

Moreover, it is important to note that some rules of $\Pi_{control}$ exhibit peculiar labels containing the symbols $\geq$ and $\leq$. For example, in the rule (P5), the predicate `color_intensity` is used with the label $\leq medium$. Intuitively, this expression extends the linguistic notion of *at least medium*. Formally, these labels are in fact additional membership functions that are usually obtained by combining other membership functions of the same predicate. In the considered program, this combination is performed by means of the probabilistic sum operator, e.g., $p(;\geq medium) = p(;medium)+p(;high)-p(;medium)\cdot p(;high)$.

Predicate	Linguistic Labels
`area_dimension`	$\{small, medium, large\}$
`eccentricity`	$\{low, medium, high\}$
`color_intensity`	$\{low, medium, high\}$
`solidity`	$\{low, medium, high\}$
`distance`	$\{low, medium, high\}$
`severity`	$\{low, medium, high\}$
`defect`	$\{low, medium, high\}$
`discard`	$\{fuzzy\_level, crisp\}$
`discount`	$\{fuzzy\_level, disc50, disc70\}$
`repair`	$\{fuzzy\_level, crisp\}$

Table 1
Corresponding set of labels for each predicate.

4.1. Examples of Evaluation

In this section, we present the derivation for two specific instances (displayed in Figure 4). In this application we explicitly evaluate the t-norm and s-norm using the min and max operators respectively, for every clause in the extended $\Pi_{control}$ program. The min operator ensures that when aggregating multiple conditions in a rule body (e.g., determining if a defect is highly severe), the resulting truth degree is strictly bounded by the weakest piece of evidence. Conversely, the max operator (s-norm) guarantees that when multiple, independent rules derive the same concept for an item, the strongest available evidence dictates the final classification. This prevents severe defects from being smoothed out or averaged down by less critical features.

In real-world scenarios, the transition between qualitative concepts—such as moving from a *(at least) medium* to a *(at least) high* severity—is rarely abrupt. We adopt a *smooth approach*, as shown in Figure 2c, which provides continuous transitions for such notions.

We define our admissible monotonic truth profiles γ by mapping each ordered label ℓ_i to a shifted logistic sigmoid [28]:

$$f(t; k, c_i) = \sigma\big(k(t - c_i)\big) = \frac{1}{1 + e^{-k(t-c_i)}}$$

where $k > 0$ governs the steepness and $c_i \in (0, 1)$ determines the threshold center. We fix $k = 20$, allowing the closed-form inverse derivation of the parameter value $t = c_i + \frac{1}{20} \ln\left(\frac{y}{1-y}\right)$.

For a predicate like `severity` with three ordinal labels (low, $medium$, $high$), the profile maps a curve to each threshold using monotonically increasing centers $c_1 = 0.20$, $c_2 = 0.50$, and $c_3 = 0.80$:

$$\gamma_{severity}(t) = \big(f(t; 20, 0.20),\ f(t; 20, 0.50),\ f(t; 20, 0.80)\big)$$

The final decision stage of the pipeline is modeled through the predicates `discard`, `discount`, and `repair`. As explained above, these predicates are equipped with linguistic labels that enable a transition

$\Pi_{control}$: Industrial Quality Control Program

$$\texttt{severity}(X, H; high) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{anomaly}(I, H) \odot_{\min} \texttt{area\_dimension}(I, H; large) \odot_{\min} \texttt{eccentricity}(I, H; high) \quad \text{(P1)}$$

$$\texttt{severity}(X, H; high) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{anomaly}(I, H) \odot_{\min} \texttt{color\_intensity}(I, H; \geq medium) \odot_{\min} \texttt{eccentricity}(I, H; high) \quad \text{(P2)}$$

$$\texttt{severity}(X, H; low) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{anomaly}(I, H) \odot_{\min} \texttt{color\_intensity}(I, H; \leq medium) \odot_{\min} \texttt{area\_dimension}(I, H; low) \quad \text{(P3)}$$

$$\texttt{defect}(I, H, functional; high) \leftarrow \texttt{anomaly}(I, H) \odot_{\min} \texttt{area\_dimension}(I, H; small) \odot_{\min} \texttt{distance}(I, H; high) \odot_{\min} \texttt{color\_intensity}(I, H; low) \quad \text{(P4)}$$

$$\texttt{defect}(I, H, functional; high) \leftarrow \texttt{anomaly}(I, H) \odot_{\min} \texttt{color\_intensity}(I, H; \leq medium) \odot_{\min} \texttt{solidity}(I, H; low) \odot_{\min} \texttt{distance}(I, H; medium) \quad \text{(P5)}$$

$$\texttt{defect}(I, H, aesthetic; high) \leftarrow \texttt{anomaly}(I, H) \odot_{\min} \texttt{color\_intensity}(I, H; medium) \odot_{\min} \texttt{solidity}(I, H; high) \odot_{\min} \texttt{distance}(I, H; low) \quad \text{(P6)}$$

$$\texttt{discard}(X; fuzzy\_level) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{anomaly}(I, H) \odot_{\min} \texttt{defect}(I, H, functional; high) \odot_{\min} \texttt{severity}(X, H; medium) \quad \text{(P7)}$$

$$\texttt{discard}(X; fuzzy\_level) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{anomaly}(I, H) \odot_{\min} \texttt{defect}(I, H, functional; medium) \odot_{\min} \texttt{severity}(X, H; high) \quad \text{(P8)}$$

$$\texttt{discard}(X; fuzzy\_level) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{anomaly}(I, H_1) \odot_{\min} \texttt{anomaly}(I, H_2) \odot_{\min} \texttt{defect}(I, H_1, aesthetic; high) \odot_{\min} \texttt{defect}(I, H_2, aesthetic; high) \odot_{\min} H_1 \neq H_2 \quad \text{(P9)}$$

$$\texttt{discount}(X; fuzzy\_level) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{anomaly}(I, H) \odot_{\min} \texttt{defect}(I, H, aesthetic; high) \odot_{\min} \texttt{severity}(X, H; high) \quad \text{(P10)}$$

$$\texttt{repair}(X, self; fuzzy\_level) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{anomaly}(I, H) \odot_{\min} \texttt{defect}(I, H, aesthetic; \leq medium) \odot_{\min} \neg_{\iota}\texttt{defect}(I, H, functional; high) \quad \text{(P11)}$$

$$\texttt{repair}(X, external; fuzzy\_level) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{anomaly}(I, H) \odot_{\min} \texttt{defect}(I, H, functional; low) \odot_{\min} \neg_{\iota}\texttt{defect}(I, H, aesthetic; low) \quad \text{(P12)}$$

$$\texttt{send}(X, A) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{repair}(X, external; crisp) \odot_{\min} \neg_{\iota}\texttt{discard}(X; crisp) \odot_{\min} \texttt{available\_company}(A) \quad \text{(P13)}$$

$$\texttt{sell}(X, P) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{original\_price}(X, P_0) \odot_{\min} \texttt{discount}(X; disc50) \odot_{\min} P = P_0 * 0.5 \odot_{\min} \neg_{\iota}\texttt{discard}(X; crisp) \quad \text{(P14)}$$

$$\texttt{sell}(X, P) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{original\_price}(X, P_0) \odot_{\min} \texttt{discount}(X; disc70) \odot_{\min} P = P_0 * 0.7 \odot_{\min} \neg_{\iota}\texttt{discard}(X; crisp) \quad \text{(P15)}$$

$$\texttt{sell}(X, P) \leftarrow \texttt{object}(X, I) \odot_{\min} \texttt{original\_price}(X, P) \odot_{\min} \neg_{\iota}\texttt{discard}(X; crisp) \odot_{\min} \neg_{\iota}\texttt{discount}(X; disc50) \odot_{\min} \neg_{\iota}\texttt{discount}(X; disc70) \quad \text{(P16)}$$

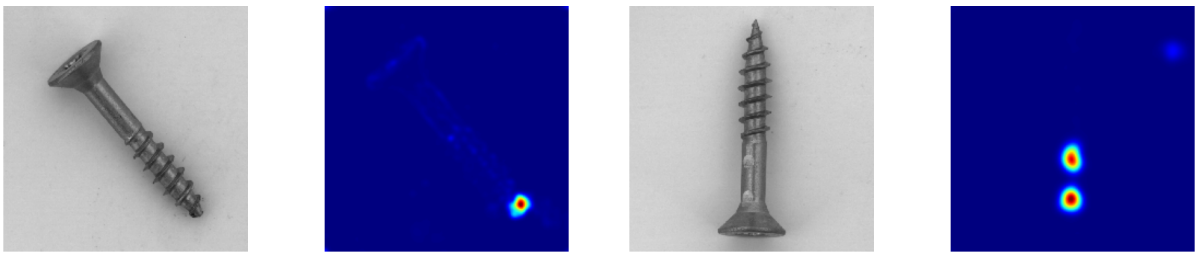
Figure 3: The Fuzzy Datalog program $\Pi_{control}$.

from fuzzy evaluations to crisp behaviors by means of their associated membership functions depicted in Figure 2. In more detail, the predicates `discard` and `repair` are defined with two labels, *fuzzy_level* and *crisp*: the former is associated with a continuous increasing membership function (in our case, linear in $[0, 1]$), while the latter is defined via a step function with a cutoff, fixed in this example at 0.5. This means that a fuzzy degree assigned to $\texttt{discard}(X; \textit{fuzzy\_level})$ (respectively $\texttt{repair}(X; \textit{fuzzy\_level})$) is turned into a crisp value for $\texttt{discard}(X; \textit{crisp})$ (respectively $\texttt{repair}(X; \textit{crisp})$), which is 1 if the degree is at least 0.5, and 0 otherwise.

The predicate `discount` follows a similar approach, but it is associated with two crisp labels corresponding to different discount levels, namely *disc50* and *disc70*. Each of these labels is defined by a step membership function with thresholds set to 0.4 and 0.7, respectively. As a result, the fuzzy degree

computed for $\texttt{discount}(X; \textit{fuzzy\_level})$ is mapped, via the γ-propagation mechanism, to either crisp discount decision depending on which thresholds are exceeded.

Note that the threshold values associated with the fuzzy-to-crisp predicates are meant to capture a specific decision policy and are application-dependent design parameters. Future work will investigate data-driven approaches for automatically learning these thresholds, in addition to the shape and parameters of the underlying membership functions.



(a) Manipulated Front Anomaly.

(b) Two Scratch Anomalies.

Figure 4: Two input examples with corresponding AE-XAD anomaly explanations.

Instance 1: Manipulated Front Anomaly Let X_1 be the item featuring a single manipulated front anomaly H (as seen in Figure 4a). In this scenario, the geometric evaluation maps the defect to a functional anomaly.

We focus the explicit derivation on the `severity` predicate to showcase the continuous latent inference. First, we compute the required compound labels via probabilistic sum ($a \oplus b = a + b - a \cdot b$):

$$\texttt{color\_intensity}(X_1, H; \leq \textit{medium}) = 0.998 \oplus 0.002 = 0.998$$
$$\texttt{color\_intensity}(X_1, H; \geq \textit{medium}) = 0.002 \oplus 0.000 = 0.002$$

Next, we evaluate the lower bounds dictated by rules (P1–P2) using the min t-norm over the geometric memberships:

$$\texttt{severity}(H; \textit{high}) \leftarrow \max\big(\min(0.020, 0.074), \min(0.002, 0.074)\big) = 0.020 \quad \text{(P1, P2)}$$
$$\texttt{severity}(H; \textit{low}) \leftarrow \min(0.998, 0.909) = 0.909 \quad \text{(P3)}$$

With the constraints $\textit{high} \geq 0.020$ and $\textit{low} \geq 0.909$, we map the evaluations to the latent state t using the inverse sigmoid. The constraint on $\textit{high}$ ($c_3 = 0.80$) imposes $t \geq 0.80 + 0.05 \ln(0.020/0.980) = 0.605$. This is strictly greater than the requirement for $\textit{low}$ ($t \geq 0.315$). Thus, substituting the strictest bound $t = 0.605$ back into the $\gamma_{\textit{severity}}$ profile provides the fully coherent semantic state

$$\texttt{severity}(H; \cdot) \text{ with labels } \{\textit{low} = 0.999,\ \textit{medium} = 0.892,\ \textit{high} = 0.020\}.$$

The framework successfully infers a strong $\textit{medium}$ severity (interpreted as "at least medium´´ because these monotonic function encode a cumulative behavior), despite the absence of an explicit rule, naturally bridging the gap between the $\textit{low}$ and $\textit{high}$ evaluations.

Processing the morphological variables through rules (P4–P5) yields a confident classification for a functional defect ($\texttt{defect}(I, H, \textit{functional}; \textit{high}) = 0.909$), primarily driven by the anomaly's localized area and its position at the object's boundaries.

$$\texttt{discard}(X_1; \textit{fuzzy\_level})) \leftarrow \min(0.909, 0.892) = 0.892$$

Because of the threshold profile $\gamma_{\textit{discard}}$ (with cutoff 0.5), we get a definitive crisp action: $\text{discard}(X_1; \textit{crisp}) = 1$.

Instance 2: Two Scratch Anomalies (Image 4b) Let X_2 be an item presenting two candidate anomaly regions identified by the sub-symbolic layer, denoted as H_A (Component 1) and H_B (Component 2) (Figure 4b).

We evaluate the severity profiles independently for both components. For H_A, we compute the aggregated bounds via probabilistic sum ($\texttt{color\_intensity}(X_2, H_A; \leq medium) = 0.043 \oplus 0.953 = 0.955$) and apply rules (P1–P3):

$$\texttt{severity}(H_A; high) \leftarrow \max\big(\min(0.037, 0), \min(0.953, 0)\big) = 0.000$$
$$\texttt{severity}(H_A; low) \leftarrow \min(0.955, 0.264) = 0.264$$

With bounds $high \geq 0$ and $low \geq 0.264$, the strictest latent state is $t = 0.148$. Substituting this into $\gamma_{severity}$ yields the following profile for the first component $\texttt{severity}(H_A, \cdot)$: $\{low = 0.261,\ medium = 0.001,\ high = 0.000\}$. Conversely, for the region H_B, the probabilistic sum yields $\texttt{color\_intensity}(X_2, H_B; \leq medium) = 0.056 \oplus 0.938 = 0.941$ and $\texttt{color\_intensity}(X_2, H_B; \geq medium) = 0.938$. Then,

$$\texttt{severity}(H_B; high) \leftarrow \max\big(\min(0.038, 0.026), \min(0.938, 0.026)\big) = 0.026$$
$$\texttt{severity}(H_B; low) \leftarrow \min(0.941, 0.272) = 0.272$$

Here, the constraint on $high$ governs the latent inference, forcing $t \geq 0.80 + 0.05 \ln(0.026/0.974) = 0.619$. The semantic state naturally bridges the evaluations:

$$\texttt{severity}(H_B, \cdot) \text{ with labels } \{low = 0.999,\ medium = 0.915,\ high = 0.026\}.$$

Aggregating the truth degrees across the previous rule set, and applying step-function profile $\gamma_{discard}$ yields $\texttt{discard}(X_2; crisp) = 0$. The framework safely avoids a false-positive discard.

Instead, the presence of a verified single aesthetic anomaly (H_B) with a safely bounded severity enables the self-repair policy. Given the latent state of H_B, the cumulative label $\texttt{defect}(H_B, aesthetical; \leq medium)$ evaluates to 1.000. Coupled with the Łukasiewicz negation of functional damages ($1 - 0.023 = 0.977$), we obtain:

$$\texttt{repair}(X_2, self; fuzzy\_level) \leftarrow \min(1.000, 0.977) = 0.977$$

Through γ_{repair}, this yields $\texttt{repair}(X_2, self; crisp) = 1$.

The neuro-symbolic pipeline leverages declarative geometric semantics to route each item to the most appropriate and cost-effective recovery procedure. The examples are handled correctly: the first is discarded by the system due to a recognized functional defect, while the second is self-repaired, as expected for scratch anomalies.

The choice of t-norm and s-norm, however, strongly affects the final outcome. As an example, in Instance 1, the product or the Łukasiewicz t-norm would prevent the expected discard decision despite a clearly defective screw, as both degrade the truth of a conjunction as its conditions grow, unlike *min*. Analogously, *max* retains only the strongest supporting rule, whereas additive s-norms may turn several weak signals into a strong conclusion, which is undesirable in this quality control scenario.

The propagation by means of the γ function highlights a fundamental theoretical advantage of our framework: the shift from continuous fuzzy reasoning to crisp operational decisions comes easily. Instead of relying on external procedural scripts, ad-hoc defuzzification algorithms, or hardcoded post-processing rules, the binarization may be handled entirely within the logical semantics. Ambiguous neural outputs are gracefully absorbed by the continuous logic, naturally ending into deterministic boolean actions without ever breaking the end-to-end logical pipeline.

5. Conclusion

In this paper, we presented a case study of recently introduced qualitative fuzzy extension of Answer Set Programming, where predicates are equipped with finite linguistic labels and membership functions.

The prototype leverages the structure of this language to combine a machine learning-based component with a set of symbolic rules to detect and localize damages in images obtained from a production line of screws, and to provide decision-making pattern for damaged items. The illustrated examples reveal the capacity of our language to incorporate the extremely effective, although potentially noisy, predictions of machine learning models with the rigor of symbolic reasoning into a formal, declarative pipeline. The association of data-driven membership functions with linguistic labels allows domain experts to encode their knowledge naturally, capturing the inherent vagueness of qualitative concepts without sacrificing logical power.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M. H. van Emden, Quantitative deduction and its fixpoint theory, J. Log. Program. 3 (1986) 37–53. URL: https://doi.org/10.1016/0743-1066(86)90003-8. doi:10.1016/0743-1066(86)90003-8.

[2] P. Vojtás, Fuzzy logic programming, Fuzzy Sets Syst. 124 (2001) 361–370. URL: https://doi.org/10.1016/S0165-0114(01)00106-3. doi:10.1016/S0165-0114(01)00106-3.

[3] R. Ebrahim, Fuzzy logic programming, Fuzzy Sets Syst. 117 (2001) 215–230. URL: https://doi.org/10.1016/S0165-0114(98)00300-5. doi:10.1016/S0165-0114(98)00300-5.

[4] J. Medina, M. Ojeda-Aciego, P. Vojtáš, A procedural semantics for multi-adjoint logic programming, in: Portuguese Conference on Artificial Intelligence, Springer, 2001, pp. 290–297.

[5] Á. Achs, A. Kiss, Fuzzy extension of datalog, Acta Cybern. 12 (1995) 153–166. URL: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3454.

[6] Y. Loyer, U. Straccia, The approximate well-founded semantics for logic programs with uncertainty, in: B. Rovan, P. Vojtás (Eds.), Mathematical Foundations of Computer Science 2003, 28th International Symposium, MFCS 2003, Bratislava, Slovakia, August 25-29, 2003, Proceedings, Lecture Notes in Computer Science, Springer, 2003, pp. 541–550. URL: https://doi.org/10.1007/978-3-540-45138-9_48. doi:10.1007/978-3-540-45138-9_48.

[7] Y. Loyer, U. Straccia, Approximate well-founded semantics, query answering and generalized normal logic programs over lattices, Ann. Math. Artif. Intell. 55 (2009) 389–417. URL: https://doi.org/10.1007/s10472-008-9099-0. doi:10.1007/S10472-008-9099-0.

[8] M. E. Cornejo, D. Lobo, J. Medina, Syntax and semantics of multi-adjoint normal logic programming, Fuzzy Sets Syst. 345 (2018) 41–62. URL: https://doi.org/10.1016/j.fss.2017.12.009. doi:10.1016/J.FSS.2017.12.009.

[9] M. E. Cornejo, D. Lobo, J. Medina, Extended multi-adjoint logic programming, Fuzzy Sets Syst. 388 (2020) 124–145. URL: https://doi.org/10.1016/j.fss.2019.03.016. doi:10.1016/J.FSS.2019.03.016.

[10] U. Straccia, Reasoning within fuzzy description logics, Journal of artificial intelligence research 14 (2001) 137–166.

[11] F. Bobillo, U. Straccia, fuzzydl: An expressive fuzzy description logic reasoner, in: 2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 923–930.

[12] T. Lukasiewicz, U. Straccia, Managing uncertainty and vagueness in description logics for the semantic web, Journal of Web Semantics 6 (2008) 291–308.

[13] M. Alviano, R. Peñaloza, Fuzzy answer sets approximations, Theory Pract. Log. Program. 13 (2013) 753–767. URL: https://doi.org/10.1017/S1471068413000471. doi:10.1017/S1471068413000471.

[14] J. Janssen, S. Schockaert, D. Vermeir, M. De Cock, General fuzzy answer set programs, in: International Workshop on Fuzzy Logic and Applications, Springer, 2009, pp. 352–359.

[15] J. Janssen, S. Schockaert, D. Vermeir, M. De Cock, A core language for fuzzy answer set programming, Int. J. Approx. Reason. 53 (2012) 660–692. URL: https://doi.org/10.1016/j.ijar.2012.01.005. doi:10.1016/J.IJAR.2012.01.005.

[16] M. Blondeel, S. Schockaert, D. Vermeir, M. De Cock, Fuzzy answer set programming: An introduction, in: R. R. Yager, A. M. Abbasov, M. Z. Reformat, S. N. Shahbazova (Eds.), Soft Computing: State of the Art Theory and Novel Applications, volume 291 of *Studies in Fuzziness and Soft Computing*, Springer, 2013, pp. 209–222. URL: https://doi.org/10.1007/978-3-642-34922-5_15. doi:10.1007/978-3-642-34922-5_15.

[17] D. Van Nieuwenborgh, M. De Cock, D. Vermeir, Fuzzy answer set programming, in: M. Fisher, W. van der Hoek, B. Konev, A. Lisitsa (Eds.), Logics in Artificial Intelligence, 10th European Conference, JELIA, volume 4160 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 359–372. URL: https://doi.org/10.1007/11853886_30. doi:10.1007/11853886_30.

[18] V. H. Le, F. Liu, D. K. Tran, Fuzzy linguistic logic programming and its applications, Theory Pract. Log. Program. 9 (2009) 309–341. URL: https://doi.org/10.1017/S1471068409003779. doi:10.1017/S1471068409003779.

[19] M. Lanzinger, S. Sferrazza, P. A. Walega, G. Gottlob, Fuzzy datalog∃ over arbitrary t-norms, in: N. S. Bjørner, M. Heule, A. Voronkov (Eds.), LPAR 2024: Proceedings of 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning, Port Louis, Mauritius, May 26-31, 2024, volume 100 of *EPiC Series in Computing*, EasyChair, 2024, pp. 426–444. URL: https://doi.org/10.29007/cngw. doi:10.29007/CNGW.

[20] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. D. Raedt, Neural probabilistic logic programming in DeepProbLog, Artificial Intelligence 298 (2021) 103504. URL: https://doi.org/10.1016/j.artint.2021.103504. doi:10.1016/J.ARTINT.2021.103504.

[21] Z. Yang, A. Ishay, J. Lee, NeurASP: Embracing neural networks into answer set programming, in: C. Bessiere (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, ijcai.org, 2020, pp. 1755–1762. URL: https://doi.org/10.24963/ijcai.2020/243. doi:10.24963/IJCAI.2020/243.

[22] A. Skryagin, W. Stammer, D. Ochs, D. S. Dhami, K. Kersting, Neural-probabilistic answer set programming, in: G. Kern-Isberner, G. Lakemeyer, T. Meyer (Eds.), Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel, July 31 - August 5, 2022, 2022. URL: https://proceedings.kr.org/2022/48/.

[23] S. Badreddine, A. S. d'Avila Garcez, L. Serafini, M. Spranger, Logic tensor networks, Artif. Intell. 303 (2022) 103649. URL: https://doi.org/10.1016/j.artint.2021.103649. doi:10.1016/J.ARTINT.2021.103649.

[24] S. Medasani, J. Kim, R. Krishnapuram, An overview of membership function generation techniques for pattern recognition, Int. J. Approx. Reason. 19 (1998) 391–417. URL: https://doi.org/10.1016/S0888-613X(98)10017-8. doi:10.1016/S0888-613X(98)10017-8.

[25] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, C. Steger, The mvtec anomaly detection dataset: a comprehensive real-world dataset for unsupervised anomaly detection, International Journal of Computer Vision 129 (2021) 1038–1059.

[26] F. Angiulli, F. Fassetti, L. Ferragina, S. Nisticò, Explaining anomalies through semi-supervised autoencoders, Array 28 (2025) 100537. URL: https://doi.org/10.1016/j.array.2025.100537. doi:10.1016/J.ARRAY.2025.100537.

[27] S. Nascimento, B. Mirkin, F. Moura-Pires, A fuzzy clustering model of data and fuzzy c-means, in: Ninth IEEE International Conference on Fuzzy Systems. FUZZ-IEEE 2000 (Cat. No. 00CH37063), volume 1, IEEE, 2000, pp. 302–307.

[28] D. Huybrechs, L. N. Trefethen, Sigmoid functions, multiscale resolution of singularities, and hp-mesh refinement, Siam Review 66 (2024) 683–693.