Communication Papers of the 21st Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 111–118
DOI: 10.15439/2026F6359
ISSN 2300-5963 ACSIS, Vol. 49

Iterative Inductive Learning for Generalized Modeling of Technical Systems

Balázs Ádám Toldi, András Földvári, and András Pataricza
0009-0007-4751-5043, 0000-0002-4559-6990, 0000-0002-6516-129X
Budapest University of Technology and Economics
Budapest, Hungary
Email: balazs.toldi@edu.bme.hu, {foldvari.andras, patericza.andras}@vik.bme.hu

Zsigmond Pap
0009-0004-4980-8075
Ericsson Hungary
Budapest, Hungary
Email: zsigmond.pap@ericsson.com

***Abstract*—Designing critical IT infrastructures to meet service-level agreements requires evaluating a combinatorial space of deployment parameters and workload scenarios that cannot be exhaustively tested. We present an integrated framework that operates on qualitative abstractions of measurement data to produce deterministic, interpretable classification rules transferable across deployments with different hardware dimensioning. The framework combines Rough Set Theory (RST) with Inductive Logic Programming (ILP): RST separates certain from ambiguous measurement outcomes and directly constrains the ILP rule learner through an asymmetric cost function reflecting safety requirements. A monotonicity check validates resource-scaling assumptions and classifies violations, and when violations indicate architectural bottlenecks, a structural system model with causal annotations traces root causes and recommends targeted measurement extensions. We evaluate the framework on a 5G UPF scaling testbed.**

I. Introduction

MODERN IT infrastructures must meet user needs while efficiently using system resources. To this end, modular and scalable construction is prevalent, with functional and performance compliance increasingly obtained through modular composition at installation time. This allows an extremely large number of final systems to be built from a single basic set, but the resulting combinatorial explosion prevents comprehensive testing of the entire system.

For instance, a containerized network service, such as a 5G Core deployment, is dimensioned using numerous parameters (e.g., the number and capacity of processing instances, CPU and memory provisioning, network configuration) that must meet strict service-level agreements (SLAs) under diverse designated workloads (varying traffic types and client counts). Even a modest number of parameters with a few qualitative levels yields hundreds or thousands of distinct configurations.

Since exhaustive testing is infeasible, the central problem is *prediction* by *extrapolation from limited benchmark measurements*: from a small number of tested configurations, reliably predict whether an untested combination of deployment parameters will satisfy SLAs under a given workload. In safety-critical infrastructures, the answer must be conservative: given the severe impact of service failures, it is preferable to flag a viable configuration as potentially failing than to declare a failing configuration healthy.

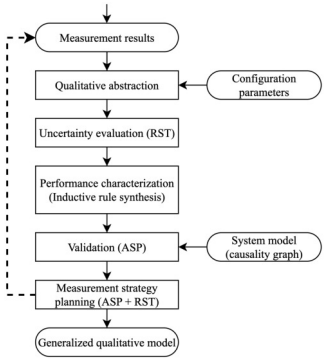


Fig. 1. Workflow

Standard machine-learning approaches fall short on three counts. They are probabilistic classifiers producing confidence scores rather than the deterministic pass/fail verdicts that strict SLAs require; their black-box nature prevents operators from inspecting the reason behind a classification, hindering root-cause diagnosis; and models trained on absolute metric values (e.g., CPU utilization) are tied to a specific hardware dimensioning and are not portable across deployments.

We propose an integrated framework that relies on *qualitative abstraction* to derive a discrete model that preserves the main phenomena influencing the SLA while eliminating the scalability and portability problems of the continuous domain. On this representation it combines *Rough Set Theory* (RST), which reasons about the uncertainty and inconsistency introduced by abstraction and noise without distributional assumptions, with *Inductive Logic Programming* (ILP), which produces human-readable deterministic rules, to extract a generalized, interpretable, and transferable model for configurable systems (Section II gives the necessary background).

A. Workflow

The framework operates as a closed-loop pipeline (Fig. 1): *measurement campaigns* produce benchmark data; RST-based *uncertainty evaluation* separates certain from ambiguous outcomes; *performance characterization* synthesizes deterministic rules under RST-derived constraints; a *validation* stage checks physical assumptions (e.g., monotonicity under re-

©2026, PTI

Topical area: Intelligent Software, System, and Service Engineering

source scaling) and, where available, a structural system model to trace inconsistencies; and *measurement strategy planning* produces a prioritized list of configurations for the next round. By expressing the entire pipeline in *Answer Set Programming* (ASP), uncertainty evaluation, rule synthesis, and measurement planning reside in a single declarative framework, enabling formal verification and seamless integration of domain knowledge.

The main contributions are:

1) An *integrated framework* in which RST analysis directly constrains and parameterizes inductive rule learning: lower approximations define hard coverage/exclusion constraints, while the boundary region and outcome frequencies drive an asymmetric cost function for conservative classification under noise.
2) An *ASP-encoded checking module* that detects physical constraints (dominance violations under the same workload) and classifies their confidence via RST boundary membership (distinguishing genuine non-monotonicity from noise-induced inconsistency).
3) A targeted *measurement strategy* derived from these analyses, replacing exhaustive or random testing with a prioritized selection that maximally reduces uncertainty and validates the assumptions behind the learned rules.

The remainder of the paper reviews background (Section II), presents the framework and data preparation (Section III), formalizes the RST uncertainty evaluation (Section IV) and the one-sided rule learning with an asymmetric cost function (Section V), details validation (monotonicity checking, RST integration, system-model root-cause analysis, and the measurement strategy) (Section VI) and reports a case study (Section VII).

B. Running Example

Throughout the paper we use, as a running example, the scaling of a distributed packet-processing microservice deployed in a Kubernetes cluster, treating it as a representative data-intensive service rather than dwelling on protocol details. A custom traffic generator drives a configurable number of client devices through one or more service instances and measures end-to-end performance, forming a closed measurement loop. Each experiment is defined by a *workload* (a traffic profile and a client count) and a *deployment configuration* (CPU allocation and instance count), and is summarized by application-level metrics (CPU and memory utilization, packet rates, throughput, packet drops) from which the HEALTHY/UNHEALTHY outcome is derived. The concrete testbed and parameter ranges are detailed in the case study (Section VII).

C. Related work

a) Performance modeling of configurable systems: Performance-influence models [1] learn the effect of configuration options via stepwise linear regression, with later work reducing sample size through sampling strategies [2] or reusing models across environments via transfer learning [3]. These approaches produce continuous regression outputs (confidence scores or numerical estimates) rather than deterministic pass/fail verdicts, are opaque about *why* a configuration fails, and are tied to a specific hardware dimensioning because they are trained on absolute metric values. Our framework addresses all three: qualitative abstraction maps measurements to relative levels (LOW/MEDIUM/HIGH), making rules hardware-independent by construction; ILP produces deterministic, human-readable rules; and RST-based uncertainty quantification drives an asymmetric cost function that guarantees conservative classification.

b) Rough set methods in cloud and IT systems: Recent work applies rough sets and their derivatives to operational cloud decisions: three-way decision fusion for elastic scaling [4] (whose learnability is formally established [5]), decision-theoretic rough sets for task classification [6], and RST discernibility [7] for QoS dimensionality reduction before metaheuristic service composition [8]. These use RST as a one-shot classifier or a pre-processing reducer. We instead treat the boundary region as the central analytical object that (i) drives an asymmetric cost function for conservative rule learning, (ii) classifies the confidence of monotonicity violations, and (iii) iteratively guides the measurement strategy. Moreover, we couple RST with ILP to produce deterministic, transferable rules, which none of the above do.

II. Background

A. Qualitative Modeling

Qualitative modeling [9] represents continuous aspects of systems symbolically: instead of precise numerical values, it abstracts quantities into discrete categories capturing the operationally relevant distinctions. For example, CPU allocations of 500 and 1000 millicore can be represented as LOW and HIGH relative to the workload's demand, without committing to absolute thresholds that differ across hardware. The abstraction is governed by three principles [9]: *discretization* partitions a continuous domain into qualitative values defined by context-meaningful thresholds; *relevance* requires the chosen categories to capture the distinctions that matter (here, whether a configuration meets its SLA); and *ambiguity* is the inherent cost of abstraction, since values near a threshold may behave differently despite sharing a qualitative label, producing inconsistent observations.

In our framework, qualitative modeling serves two purposes. It enables *scale-independent* rules that transfer across deployments with different hardware dimensioning, provided thresholds are defined relative to capacity; and the discretization yields a finite attribute space directly compatible with RST's indiscernibility relation, where the abstraction's ambiguity is precisely what the boundary region captures.

B. Rough Set Theory

Rough Set Theory (RST) [10] reasons about uncertainty and inconsistency in data without requiring prior assumptions about data distributions, relying solely on the observed data to separate certain knowledge from ambiguous cases.

An *information system* (U, A) consists of a finite set of objects U and attributes A. A *decision table* $(U, A \cup \{d\})$ adds a decision attribute; in our setting, $d \in \{\text{HEALTHY}, \text{UNHEALTHY}\}$. Any subset $B \subseteq A$ induces an *indiscernibility relation* that groups objects sharing the same values on all attributes in B into equivalence classes.

Given a decision class $X \subseteq U$, the *lower approximation* $\underline{B}(X)$ contains equivalence classes entirely within X (certain membership), the *upper approximation* $\overline{B}(X)$ contains classes with any overlap with X (possible membership), and the *boundary region* $\text{BND}_B(X) = \overline{B}(X) \setminus \underline{B}(X)$ contains classes that cannot be deterministically classified. The *approximation quality* $\gamma_B(d)$ measures the fraction of objects outside the boundary region; $\gamma = 1$ means fully deterministic classification.

C. Answer Set Programming

Answer Set Programming (ASP) [11], [12] is a declarative formalism for modeling and solving search and optimization problems, based on the stable model semantics [13].

An ASP program consists of rules of the form:

```
h :- a_1, ..., a_n, not b_1, ..., not b_m.
```

where h is the *head* and the conjunction of literals forms the *body*: the head is derived when all positive body literals can be proven and none of the negated ones can. The operator `not` implements *negation as failure* (NAF): `not` b holds when there is no derivation for b, expressing absence of evidence rather than explicit falsity and giving ASP a non-monotonic, *closed-world* semantics. Solutions are *answer sets*: self-consistent atom sets satisfying all rules. Solvers additionally support aggregates (`#count`, `#sum`) and optimization (`#minimize`, `#maximize`), making ASP well-suited as the unifying formalism across our framework's uncertainty analysis, rule learning, and measurement strategy.

D. Inductive learning

Inductive Logic Programming (ILP) [14] aims to find a set of logical rules, called a *hypothesis*, that together with existing background knowledge explains a set of examples. In this work, we employ ILP in the context of Answer Set Programming, using the ILASP framework [15] and its scalable variant FastLAS [16].

A learning task is a tuple $T = \langle B, M, E \rangle$, where B is background knowledge (an ASP program), M is a *mode bias* defining the rule space, and E is a set of examples. Each example carries a *penalty* (the cost of not covering it), a partial interpretation (inclusions/exclusions), and a per-instance ASP *context*; examples with infinite penalty must be covered, while finite-penalty ones may be left uncovered at their cost. The learner minimizes a score combining hypothesis length with the penalties of uncovered examples. FastLAS [16] generalizes this with *domain-specific scoring functions* that decompose over individual rules, improving scalability. This fits our setting in two ways: finite penalties tolerate noisy examples proportionally to their frequency, and customizable scoring lets us encode domain-specific preferences (e.g., biasing toward conservative classifications) directly into the optimization criterion.

III. Framework Overview

This section defines the variable groups arising in benchmark campaigns, describes how raw data is prepared, and outlines how the pipeline stages interact; each stage is then detailed in its own section.

A. Variable Structure

The variables of a benchmark campaign fall into three groups with distinct roles. *(1) Measured metrics* (CPU and memory utilization, throughput, packets per second, dropped packets, etc.) serve exclusively to derive each run's binary HEALTHY/UNHEALTHY outcome against predefined service-level thresholds. *(2) Workload requirements* (traffic type, user equipment count, and further demand-side parameters) characterize the service demand. *(3) Deployment configuration* (UPF count, core strength, memory capacity, etc.) describes the provisioned resources. The learning task operates on groups (2) and (3): given a workload requirement and a candidate deployment configuration, the learned rules predict whether the configuration can support the workload. Group (1) is consumed during data preparation but does not appear as an attribute in the learned rules.

B. Data Preparation

Each campaign run produces time-series telemetry from which the measured metrics (and hence the binary outcome) are derived, while the workload and deployment variables are represented as qualitative levels. The same configuration-workload pair may be measured multiple times; repeated runs are aggregated per equivalence class while retaining the frequency of each outcome, which feeds both the uncertainty evaluation and the cost function of the learning stage (Section IV details the construction).

C. Pipeline Stages

The pipeline comprises four stages (Fig. 1), each detailed later: *uncertainty evaluation* (Sec. IV) applies RST to the aggregated decision table, separating the lower approximations from the boundary region and reporting γ; *performance characterization* (Sec. V) synthesizes deterministic rules for one decision class under the RST-derived constraints and cost function, with the complementary class as the NAF default; *validation* (Sec. VI) runs an ASP-encoded monotonicity check and, when a system model is available, traces persistent inconsistencies to candidate unmeasured variables; and *measurement strategy planning* prioritizes the next campaign (boundary resolution, monotonicity testing, coverage gaps), closing the loop with analysis-driven selection.

IV. Uncertainty Evaluation

A. Decision Table Construction

We now detail how the three variable groups enter the decision table. The measured metrics derive the binary decision

attribute only. Let $M = \{m_1, \ldots, m_k\}$ be the metrics of a run, each with a service-level threshold τ_i and compliance condition (e.g., $m_i \leq \tau_i$ for drop rates, $m_i \geq \tau_i$ for throughput). A run is UNHEALTHY if *any* metric violates its requirement:

$$d = \begin{cases} \text{HEALTHY} & \text{if all metrics satisfy their thresholds} \\ \text{UNHEALTHY} & \text{otherwise} \end{cases} \tag{1}$$

The workload and deployment variables form the attribute set. Categorical variables (e.g., traffic type) enter directly; numerical ones (e.g., UPF count, core strength) are mapped to qualitative levels for scale independence. For a numerical variable v with value x and thresholds $\theta_{\text{low}}, \theta_{\text{high}}$:

$$q(v, x) = \begin{cases} \text{LOW} & \text{if } x \leq \theta_{\text{low}} \\ \text{MEDIUM} & \text{if } \theta_{\text{low}} < x \leq \theta_{\text{high}} \\ \text{HIGH} & \text{if } x > \theta_{\text{high}} \end{cases} \tag{2}$$

with thresholds defined relative to the deployment's capacity range.

The placement of these landmarks directly governs the boundary region: wider levels merge numerically distinct configurations into the same class, enlarging the boundary and lowering γ, while tighter levels shrink it but leave fewer repeated observations per class. The thresholds are therefore an *engineering modeling decision* (whether derived from a capacity model or from automated clustering) and must reflect the underlying system model and analysis goal for the abstraction to remain meaningful.

Each run r_i produces a row in the decision table $\mathcal{T} = (U, A \cup \{d\})$, where $U = \{r_1, \ldots, r_n\}$ is the set of runs, A the qualitative attributes, and d the decision. The same configuration–workload pair may be executed *multiple times*, yielding rows with identical attribute vectors but potentially different decisions; this is the source of inconsistency that RST captures through the boundary region.

Consider two deployment variables, Core Strength and UPF Count $\in \{\text{L}, \text{M}, \text{H}\}$, and one workload variable, Traffic $\in \{\text{mIoT}, \text{eMBB}, \text{std}\}$. Suppose 8 runs cover 6 distinct configurations, one of which (Core = L, Count = M, eMBB) is measured twice with contradictory outcomes.

B. Lower Approximations and Boundary Region

Applying the indiscernibility relation $\text{IND}(A)$ groups repeated runs of the same configuration into shared equivalence classes. Let $X_{\text{U}} = \{r \in U \mid d(r) = \text{UNHEALTHY}\}$; the resulting classes and their classification (with UH:H outcome counts) are shown in Table I.

The lower approximation of UNHEALTHY contains classes where *all* runs are unhealthy: $\underline{A}(X_{\text{U}}) = E_1 \cup E_4$ (*certainly failing*). Symmetrically, $\underline{A}(X_{\text{H}}) = E_3 \cup E_5 \cup E_6$ are certainly healthy, and the boundary region $\text{BND} = \{E_2\}$ holds the single mixed-outcome configuration. The approximation quality at the class level is

$$\gamma_A(d) = \frac{|\{E_1, E_3, E_4, E_5, E_6\}|}{|\{E_1, \ldots, E_6\}|} = \frac{5}{6} \approx 0.833, \tag{3}$$

TABLE I
EQUIVALENCE CLASSES AND THEIR RST CLASSIFICATION

Class	Core	Count	Traf.	UH:H	Region
E_1	L	L	mIoT	2:0	$\underline{A}(X_{\text{U}})$
E_2	L	M	eMBB	1:1	BND
E_3	M	M	std	0:1	$\underline{A}(X_{\text{H}})$
E_4	M	L	mIoT	1:0	$\underline{A}(X_{\text{U}})$
E_5	H	H	eMBB	0:1	$\underline{A}(X_{\text{H}})$
E_6	H	L	std	0:1	$\underline{A}(X_{\text{H}})$

i.e., one of six configurations yields inconsistent results across repeated runs.

These results directly constrain and parameterize rule learning. *Coverage:* rules for UNHEALTHY must fire on all configurations in $\underline{A}(X_{\text{U}})$; failing to do so is a *false negative*, a critical error in safety-relevant systems. *Exclusion:* rules must *not* fire on $\underline{A}(X_{\text{H}})$; firing here is a *false positive* (overly conservative but not dangerous). *Cost-driven decision:* configurations in BND are classified by an asymmetric cost function penalizing false negatives (weight α) more than false positives (weight β), with the outcome frequencies (Table I) letting the cost reflect empirical evidence, where a 3:1 UH:H boundary class warrants a different penalty than a 1:1 split.

C. Interpreting the Boundary Region

The boundary region is not merely a nuisance but an informative diagnostic signal with three possible causes: (i) *environmental noise*: behavior is non-deterministic at the chosen abstraction level, an irreducible inconsistency the cost function must tolerate; (ii) *too coarse quantization*: the thresholds of Eq. 2 group meaningfully different values, so refining them may resolve the inconsistency; and (iii) *missing variable*: an unobserved variable drives the differing outcomes, in which case a structural system model can identify candidate variables architecturally connected to the inconsistent component (Section VI). A large boundary region with low γ thus guides the evolution of the measurement campaign.

V. PERFORMANCE CHARACTERIZATION

This section describes how inductive learning in ASP synthesizes deterministic classification rules from the decision table.

A. One-Sided Learning with NAF Default

The framework learns rules for one decision class while the complementary class serves as the default via NAF: any configuration not explicitly classified by a learned rule is assigned the default class. The choice depends on safety requirements. The *conservative* (safety-first) option learns rules for HEALTHY with default UNHEALTHY, so a configuration is acceptable only if a rule explicitly supports it (unclassified configurations are treated as potentially failing). The *permissive* option is symmetric (learning UNHEALTHY with default HEALTHY), appropriate when over-provisioning costs outweigh under-provisioning risk. In both cases ASP's closed-world assumption makes the classification deterministic and complete. We present the conservative formulation (learning

HEALTHY), matching our 5G case study; the permissive case is symmetric.

B. Mapping to an ILP Task

The decision table is mapped to an ILP task $T = \langle B, M, E \rangle$. The background knowledge B encodes the domain: the observed equivalence classes (as node identifiers) and any structural information constraining the rule space. The mode bias M defines the hypothesis space: the head mode declares the target predicate (e.g., `healthy(V)`) and the body modes declare the qualitative attribute predicates, both positive and negated, allowing rules of the form:

```
healthy(ID) :- attr_1(ID, val_1),
               not attr_2(ID, val_2),
               ... .
```

Negated body literals are essential: they let rules express conditions such as "healthy if core strength is *not* low," which are more compact and general than enumerating positive combinations.

Each equivalence class produces weighted examples E. For a class E_i with n_h healthy and n_u unhealthy runs, the conservative formulation (learning HEALTHY) generates: if $n_h > 0$, a positive example with inclusion $\{\texttt{healthy}(i)\}$ and penalty $\propto n_h \cdot \alpha$ (α = weight of false negatives, i.e. unnecessary rejection); if $n_u > 0$, a negative example with exclusion $\{\texttt{healthy}(i)\}$ and penalty $\propto n_u \cdot \beta$ (β = weight of false positives, i.e. wrongly accepting a failing configuration). The context encodes the class's qualitative attribute values as ASP facts. The asymmetry $\beta > \alpha$ encodes the safety requirement (wrongly accepting a failing configuration is penalized more heavily than rejecting a healthy one), and the permissive formulation uses the opposite ratio. Because penalties scale with observation counts, a class observed 100 times carries proportionally more weight than one observed twice.

For lower-approximation classes the asymmetry is irrelevant: a class in $\underline{A}(X_{\text{H}})$ has $n_u = 0$ (only a positive example, which must be covered) and one in $\underline{A}(X_{\text{U}})$ has $n_h = 0$ (only a negative example, which must not be violated). The asymmetric cost therefore operates primarily on the *boundary region*, where both outcomes coexist and the learner decides by the balance of evidence and the safety-driven penalty ratio. Beyond example penalties, the scoring function can add structural preferences via FastLAS rule-level penalties: a fixed penalty per rule (discouraging complex hypotheses), a per-literal penalty (preferring shorter rules), and custom domain-specific penalties.

C. Connecting RST Constraints to the ILP Task

The RST constraints are translated through the penalty structure. Lower-approximation classes become hard constraints via infinite penalties (∞) on their positive or negative examples, forcing coverage or exclusion; boundary-region classes produce both example types with finite, frequency- and α/β-derived penalties, letting the learner resolve the ambiguity by minimizing total cost. In practice the hard constraints can instead use very high finite penalties that dominate the boundary ones, trading a strict guarantee for robustness under extreme noise. The approximation quality γ is a global diagnostic: high γ means a tightly constrained task with accurate rules, while low γ shifts weight onto the cost function, making rule quality depend more on the α/β ratio and observation counts.

VI. Validation and Measurement Strategy

The NAF default implicitly encodes a *monotonicity hypothesis*: under the same workload requirements, increasing deployment resources should never degrade system health, so if a weaker configuration passes, every stronger one should too. This can be violated for two reasons: *environmental noise* (background load fluctuations or scheduling jitter causing a stronger configuration to occasionally fail) or a *missing variable* that interacts with the resource dimensions and produces genuinely non-monotone behavior. The validation stage distinguishes the two by integrating the violation analysis with the RST boundary region classification.

A. Monotonicity Checking

Let $\mathcal{W}$ be the *workload requirement* variables (e.g., traffic type, UE count) and $\mathcal{R}$ the *deployment configuration* variables representing provisionable resources (e.g., core strength, UPF count), whose qualitative levels are ordered (LOW$\prec$MEDIUM$\prec$HIGH). Two configurations share the same workload profile, $c_a \sim_{\mathcal{W}} c_b$, iff they agree on every workload variable:

$$c_a \sim_{\mathcal{W}} c_b \iff \forall w \in \mathcal{W}\colon c_a(w) = c_b(w) \tag{4}$$

Given $c_a \sim_{\mathcal{W}} c_b$, c_a *resource-dominates* c_b ($c_a \succ_{\mathcal{R}} c_b$) if every resource variable is at least as high and at least one is strictly higher:

$$\begin{aligned} c_a \succ_{\mathcal{R}} c_b \iff \quad & \big(\forall r \in \mathcal{R}\colon c_b(r) \preceq c_a(r)\big) \\ & \wedge \big(\exists r \in \mathcal{R}\colon c_b(r) \prec c_a(r)\big) \end{aligned} \tag{5}$$

A pair with $c_a \succ_{\mathcal{R}} c_b$ is a *monotonicity violation* if c_b is HEALTHY and c_a UNHEALTHY: a stronger configuration fails where a weaker one passes under identical workload.

The check is encoded declaratively in ASP in four layers. **(1) Stratification:** measured configurations are partitioned by workload profile and a componentwise partial order (Eq. 5) is established over resources, so monotonicity is never evaluated across incomparable workloads. **(2) Violation detection:** within each stratum, the encoding finds violating pairs and records the differing resource dimensions; a single differing dimension is a strong candidate for investigation. **(3) Intermediate generation:** for a violation between a stronger c_s and weaker c_w, it generates the unmeasured lattice points $\{c_x \mid \forall r\colon c_w(r) \preceq c_x(r) \preceq c_s(r)\} \setminus \{c_w, c_s\}$ as *suggested measurements* that narrow the gap between known HEALTHY and UNHEALTHY regions. **(4) Boundary classification:** a *direct boundary* separates an UNHEALTHY c_a and HEALTHY c_b with $c_a \succ_{\mathcal{R}} c_b$ and no measured configuration between them; a *minimal boundary* further requires differing in exactly one dimension by one level, the finest granularity at which the transition can be localized.

B. Integration with Rough Set Analysis

A violation's confidence depends on the RST membership of the two configurations. If both lie in the lower approximation of their decision classes, it is *confirmed non-monotonicity*: every repeated run agreed, so the contradiction cannot be measurement variability and the NAF assumption is invalid in that region. If either lies in the boundary region, the violation is *possibly noise-induced* and needs more measurements before concluding monotonicity fails. In the ASP encoding, the boundary classification predicate from the uncertainty stage is imported and each violation annotated accordingly; boundary-involving violations are flagged and their Layer 3 suggested measurements receive elevated priority.

C. System Model and Root Cause Analysis

Violations and the boundary region identify *where* the data is inconsistent, not *why*. When violations cluster around a resource variable, the framework can incorporate a structural *system model* encoded in ASP to find the unmeasured factor causally linked to it. The model represents the architecture as ASP facts: *components and containment* (network functions, bridges, routing daemons and their topology); *connections and data paths* (the user-plane path, distinguished from control-plane connections because bottlenecks typically occur on it); and *observable variables* (configurable parameters and metrics per component, annotated with whether they are currently measured; the gap defines the candidate missing variables). Beyond connectivity, the model encodes directed *causal influence* relations recording that a resource/workload variable increases the load on a component, with the mechanism (e.g., more UPF instances cause routing overhead on a BGP pod maintaining iBGP sessions, and more traffic on a shared bridge). These are verifiable engineering assertions, not statistically inferred causal claims. The model marks *shared resources* and derives *scaling-sensitive* components and *potential bottlenecks* (scaling-sensitive, shared, and lacking a measured capacity variable).

The root cause analysis combines violations with the causal model in three steps: *pattern analysis* groups violations by differing resource variable and identifies the *dominant violation variable*; *causal trace* traverses influence relations from that variable to downstream components, flagging scaling-sensitive shared components without a measured capacity variable as high-confidence root causes (medium if not shared); and *aggregate diagnosis* scores each component by the number of violations tracing to it, yielding the top-ranked root cause. This produces two recommendation types: *extend the learning task*, adding a configurable parameter on the bottleneck as a new attribute may separate previously indistinguishable boundary configurations and improve γ; or *refine the decision attribute*, adding a metric (e.g., routing convergence time alongside packet drops) that distinguishes failure modes yields cleaner labels and may shrink the boundary. The operator decides which interventions are feasible.

D. Measurement Strategy

The uncertainty evaluation and monotonicity analysis jointly drive a targeted strategy with three priority tiers: *boundary resolution* (additional runs for violation endpoints in the boundary region, with extended telemetry when the system model identifies candidate variables, simultaneously testing the violation and reducing the boundary); *monotonicity testing* (intermediate lattice configurations between known HEALTHY/UNHEALTHY points, prioritizing those near a direct or minimal boundary); and *coverage gaps* (configurations classified solely by NAF, the lowest priority). This replaces random or exhaustive testing with a minimal, analysis-driven selection, closing the loop to the next campaign round.

VII. Case Study

We evaluate the framework on measurement data from a 5G Core testbed, showing how it guides the expansion of a measurement campaign and diagnoses the sources of uncertainty that emerge as the configuration space grows.

A. Dataset preparation

The dataset was built from custom traffic-generator logs and Prometheus time-series data. Preprocessing normalized resource measurements and annotated them with traffic and deployment metadata, then categorized observations into qualitative states via K-means clustering combined with rule-based heuristics (e.g., UNHEALTHY when the packet drop count exceeds a threshold).

B. Baseline Measurements

The initial campaign covers 9 configurations with core strength fixed at LOW, varying UPF count, UE count, and traffic type across all three levels. After aggregation, the dataset has 10 equivalence classes (one configuration appears with both outcomes) over 1 490 runs. The approximation quality is $\gamma = 0.500$: five classes in $\underline{A}(X_{\mathrm{U}})$, three in $\underline{A}(X_{\mathrm{H}})$, and two in the boundary region (a single configuration, UPF = HIGH, UE = LOW, eMBB, observed healthy in 102 runs and unhealthy in 47). No monotonicity violations are detected: within each workload stratum, the NAF assumption holds.

The framework produces stable results: eleven of twelve tested penalty configurations yield the same rules:

```
healthy(V) :- traffic_type(V, embb),
              upf_count(V, high).
healthy(V) :- traffic_type(V, standard).
```

These state that with LOW core strength, the system is healthy when the traffic is eMBB with high UPF count, or when the traffic is standard; all mIoT and low-count eMBB configurations default to UNHEALTHY via NAF. The framework identifies 18 unmeasured combinations (Priority 3), including all MEDIUM/HIGH core strength configurations; no boundary-resolution or monotonicity-testing measurements are needed. The clear recommendation is to extend the campaign to stronger deployments.

C. Expanded Campaign

Following this recommendation, the campaign is extended with 10 additional configurations covering MEDIUM and HIGH core strength. Combined with the baseline, the full dataset has 22 equivalence classes over 3 724 runs. The approximation quality *drops* to $\gamma = 0.455$. This is not a regression but a diagnostic signal: the new configurations introduce inconsistencies invisible in the LOW-only baseline. Ten classes are now in $\underline{A}(X_{\mathrm{U}})$ and four in $\underline{A}(X_{\mathrm{H}})$ (up from three), but eight land in the boundary region (up from two). Although the boundary grows from 20% to 36%, the number of certainly classified configurations increases in both classes, enabling more expressive rule learning.

Six monotonicity violations are detected, all classified as possibly noise-induced (at least one endpoint in the boundary region). The dominant violation variable is UPF count (4 of 6); core strength accounts for 3. For example, (medium core, medium count, medium UE, standard) is UNHEALTHY while the weaker (medium core, LOW count, medium UE, standard) is HEALTHY. The learned rules now capture interactions across all four variables:

```
healthy(V) :- core_strength(V, medium),
              upf_count(V, high),
              not traffic_type(V, miot).
healthy(V) :- not core_strength(V, low),
              ue_count(V, medium).
healthy(V) :- core_strength(V, low),
              not ue_count(V, high),
              not traffic_type(V, embb).
```

The third rule is notable: LOW core strength is not inherently UNHEALTHY, it suffices when UE count is not high and traffic is not eMBB, a nuance invisible in the first campaign where core strength was fixed.

The root cause analysis traces the dominant violation variable (UPF count) through the system model, surfacing two high-confidence potential bottlenecks: the *BGP routing pod*, a shared resource servicing all UPF instances whose route-advertisement overhead grows with UPF count (fixed capacity, not in the attribute set); and the *shared OVS bridge* (N6 interface), through which all instances communicate, with fixed PMD core and queue allocation. The framework thus recommends (1) adding OVS bridge capacity or BGP session indicators as new attributes to separate currently indistinguishable boundary configurations, and (2) measuring BGP convergence time and OVS drop rates alongside packet drops to distinguish routing-induced from capacity-induced failures. This diagnosis was independently corroborated by domain-expert analysis, which identified BGP routing overhead as a likely contributor to the non-monotone behavior.

The framework produces a prioritized plan for the next round: (1) re-measure the 4 boundary-region configurations involved in violations with extended BGP/OVS telemetry; (2) no intermediate configurations are needed (all lattice points between violation endpoints are already measured); (3) 36 unmeasured combinations, primarily LOW core strength with uncovered workloads and MEDIUM core strength with mIoT traffic.

TABLE II
COMPARISON ACROSS THE THREE ANALYSES

	Baseline	**Expanded**	**Insufficient**
Equivalence classes	10	22	28
Approx. quality γ	0.500	0.455	0.286
Certainly healthy	3	4	0
Boundary region	2	8	20
Violations	0	6	22
Rule stability	stable	stable	unstable
System model	n/a	BGP, OVS	BGP, OVS

D. Contrast: Insufficient Variable Selection

To illustrate behavior when the attribute set is fundamentally insufficient, we analyze a third variable selection from the same data: core strength (LOW/MEDIUM), UPF count (all three levels), and traffic type, with UE count fixed at 1 024. This yields 28 equivalence classes over 10 727 runs. The approximation quality drops to $\gamma = 0.286$: no class is in $\underline{A}(X_{\mathrm{H}})$ (every healthy configuration was also observed unhealthy), 20 of 28 classes lie in the boundary, and 22 monotonicity violations are detected (all noise-induced, UPF count dominant in 18). The learned rules are highly unstable across penalty settings. With balanced penalties the learner produces a single rule covering one class:

```
healthy(V) :- core_strength(V, medium),
              upf_count(V, high),
              not traffic_type(V, miot).
```

With aggressive settings (α:β = 3:1), broader rules emerge that cover most boundary-region classes:

```
healthy(V) :- not core_strength(V, low),
              traffic_type(V, embb).
healthy(V) :- not core_strength(V, low),
              not upf_count(V, low),
              traffic_type(V, standard).
```

Several configurations produce an empty hypothesis: the NAF default of UNHEALTHY is optimal because healthy runs constitute only $\sim$10% of observations. The framework correctly diagnoses this: the low γ signals that the current variables cannot explain the outcomes, and the system model points to the same bottlenecks (BGP pod, OVS bridge). Without UE count as a differentiating variable, configurations behaving differently under different loads are merged into one class, inflating the boundary.

E. Discussion

The *iteration* from baseline to expanded campaign shows the measurement strategy driving productive data collection: the baseline's Priority 3 list motivated adding MEDIUM/HIGH core strength, revealing new interactions. The *drop in* γ from 0.500 to 0.455 illustrates that more data does not automatically mean better models: the expansion surfaced inconsistencies, and the root-cause analysis (BGP stack, OVS bridge) gives a concrete path forward. The *contrast* ($\gamma = 0.286$) shows graceful degradation: rather than silently producing unreliable rules, the framework flags the problem and guides the operator toward a better variable selection. Table II summarizes the three analyses.

VIII. Conclusions and future work

We presented an integrated framework for deterministic, interpretable diagnostics of highly configurable systems. Operating on qualitative abstractions, the framework combines RST with ILP in a unified ASP encoding. RST quantifies measurement uncertainty and constrains the rule learner, a monotonicity check validates resource-scaling assumptions with RST-based confidence classification, and a structural system model traces root causes of violations to architectural bottlenecks. The case study on a 5G testbed demonstrated that the framework produces stable rules when the data is informative, diagnoses the sources of uncertainty when it is not, and provides actionable recommendations for the next measurement campaign.

A. Implications

For theory, the framework shows that the rough-set boundary region can serve not only as an uncertainty indicator but as a first-class parameter shaping learning constraints, confidence classification, and experimental design within a single formal loop. For practice, it offers operators a deterministic, auditable classification that remains valid across hardware changes without retraining, replacing exhaustive testing with a targeted strategy that reduces campaign cost while maintaining conservative safety guarantees.

B. Limitations and Threats to Validity

The framework is evaluated on a single 5G UPF testbed, so the reported rules and γ values illustrate the methodology rather than constitute general findings. We address this by construction: qualitative landmarks are defined relative to each deployment's capacity, making rules hardware-independent, and the pipeline applies to any configurable system with repeatable benchmarks, though cross-environment transfer remains to be shown empirically. The results depend on the placement of the quantization landmarks (Eq. 2) and the SLA thresholds defining the decision attribute, which is precisely why the framework represents residual noise explicitly rather than assuming deterministic outcomes. The few equivalence classes limit independent train/test evaluation, so we report diagnostic behavior (rule stability, γ, boundary growth) rather than point accuracy.

C. Future Work

Alternative quantization methods (entropy-based, data- or expert-driven) could improve discretization quality where a large boundary region stems from poorly calibrated thresholds. The system model's causal annotations could be strengthened toward interventional semantics, enabling predictive reasoning about adding variables before measuring them. Finally, a systematic comparison with alternative interpretable classifiers (decision trees, Boolean Rule Column Generation) would further establish the framework's positioning.

Acknowledgements

Project no. 2025-2.1.2-EKÖP-KDP-2025-00005 has been implemented with the support provided by the Ministry of Cultvia Innovation of Hungary from tthat reflectsl Research, Development and Innovation Fund, financed under the EKÖP_KDP-25-1-BME-27 funding scheme.

During the preparation of this work, the authors used Claude to assist with drafting and refining manuscript text. Gemini and Claude were also used for generating and optimizing Python scripts used in the experimental simulations. The authors take full responsibility for the content of this publication.

References

[1] N. Siegmund, A. Grebhahn, S. Apel, and C. Kästner, "Performance-influence models for highly configurable systems," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 284–294. [Online]. Available: https://doi.org/10.1145/2786805.2786845

[2] J. Guo, D. Yang, N. Siegmund, S. Apel, A. Sarkar, P. Valov, K. Czarnecki, A. Wasowski, and H. Yu, "Data-efficient performance learning for configurable systems," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1826–1867, 2018. [Online]. Available: https://doi.org/10.1007/s10664-017-9573-6

[3] P. Jamshidi, M. Velez, C. Kästner, and N. Siegmund, "Transfer learning for performance modeling of configurable systems: An exploratory analysis," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, 2017, pp. 497–508. [Online]. Available: https://doi.org/10.1109/ASE.2017.8115661

[4] C. Jiang and Y. Duan, "Elasticity unleashed: Fine-grained cloud scaling through distributed three-way decision fusion with multi-head attention," *Information Sciences*, vol. 660, p. 120127, 2024.

[5] A. Campagner and D. Ciucci, "Three-way learnability: A learning theoretic perspective on three-way decision," in *Proceedings of the 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*, ser. Annals of Computer Science and Information Systems, vol. 30, 2022, pp. 243–246.

[6] S. Liu, X. Ma, Y. Jia, and Y. Liu, "An energy-saving task scheduling model via greedy strategy under cloud environment," *Wireless Communications and Mobile Computing*, vol. 2022, p. 8769674, 2022.

[7] S. Widz and S. Stawicki, "Generalized majority decision reducts," in *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS)*, ser. Annals of Computer Science and Information Systems, vol. 8, 2016, pp. 165–174.

[8] H. Naghavipour, M. Y. I. B. Idris, T. K. Soon, R. B. Salleh, and A. Gani, "Hybrid metaheuristics using rough sets for QoS-aware service composition," *IEEE Access*, vol. 10, pp. 112 609–112 626, 2022.

[9] K. D. Forbus, "Chapter 9 qualitative modeling," in *Handbook of Knowledge Representation*, ser. Foundations of Artificial Intelligence, F. van Harmelen, V. Lifschitz, and B. Porter, Eds. Elsevier, 2008, vol. 3, pp. 361–393. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S157465260703009X

[10] Z. Pawlak, "Rough sets," *International journal of computer & information sciences*, vol. 11, no. 5, pp. 341–356, 1982.

[11] V. Lifschitz, *Answer set programming*. Springer Berlin, 2019.

[12] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, M. Maratea, F. Ricca, and T. Schaub, "ASP-core-2 input language format," *Theor. Practice Logic*, vol. 20, no. 2, pp. 294–309, 2019.

[13] M. Gelfond and V. Lifschitz, "The stable model semantics for logic programming," in *Proceedings of the Fifth International Conference and Symposium on Logic Programming*. MIT Press, 1988, pp. 1070–1080.

[14] S. Muggleton, "Inductive logic programming," *New generation computing*, vol. 8, no. 4, pp. 295–318, 1991.

[15] M. Law, A. Russo, and K. Broda, "The ilasp system for inductive learning of answer set programs," *arXiv preprint arXiv:2005.00904*, 2020.

[16] M. Law, A. Russo, E. Bertino, K. Broda, and J. Lobo, "Fastlas: Scalable inductive logic programming incorporating domain-specific optimisation criteria," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 03, 2020, pp. 2877–2885.