The Violation Situation Pattern: A Knowledge-Graph Pattern for Compliance Violations

Nima Kamali Lassem, Fuqi Song, Seyid Amjad Ali

The Violation Situation Pattern: A Knowledge-Graph Pattern for Compliance Violations*

Nima Kamali Lassem[1][0009−0008−4326−9639], Fuqi Song[1][0009−0001−4900−5369], and Seyid Amjad Ali[2][0000−0001−9250−9020]

[1] DiliTrust, Paris, France
{nima.kamali, fuqi.song}@dilitrust.com
[2] Department of Information Systems and Technologies, Bilkent University, Türkiye
syedali@bilkent.edu.tr

Abstract. Compliance pipelines detect violations as transient query results and do not keep the violation itself as a persistent graph object with review state, affected entities, or audit history. The Violation Situation Pattern (VSP) closes this gap. Building on the Situation pattern of Gangemi and Mika, VSP reifies each detected violation as a graph node with a rule identifier, a temporal validity interval, a lifecycle state, and evidence links to the entities involved. Lifecycle transitions are stored as immutable, PROV-O-aligned events, so audit history is a graph traversal. We instantiate VSP in a legal entity and contract lifecycle property graph and operationalize four deontic rules (V1 unauthorized signature, V2 expired mandate, V3 missing confidentiality clause, V4 missing breach-notification clause) through an FCL→Cypher→MERGE pipeline. We check V1 and V2 against BODACC corporate-officer publications, evaluate V4 on 73 GDPRhub enforcement decisions, and run a SHACL cross-formalism check on V3 and V4. The central finding is rule-body independence: extending V4 from clause-presence to deadline checking raises F1 from 0.312 to 0.602, while the pattern's identity, lifecycle, and evidence semantics stay the same. This separates a pattern contribution from a detector contribution, so detection logic can evolve without invalidating accumulated audit history.

Keywords: ontology design patterns · knowledge graph · legal compliance · violation lifecycle · provenance · audit · deontic logic · SHACL

1 Introduction

Compliance pipelines typically execute detection queries, return a set of flagged results, and then discard them. Once the query finishes, the violation no longer exists as a persistent entity in the graph. Information such as the original detection time and whether the issue was reviewed, rejected, or resolved is typically

* Preprint. Under review at EKAW 2026.

stored in external ticketing or workflow systems separate from the underlying graph data. If the same query is executed again at a later date, the system produces the same violation without retaining any record of earlier detections. As a result, the graph can represent contracts, clauses, and signatories, but it cannot describe its own compliance history.

Established approaches based on Shapes Constraint Language (SHACL) validation [1] and deontic logic over reified General Data Protection Regulation (GDPR) representations [17] are effective at identifying violations when the required information is available, but both treat violations as transient query results rather than durable graph entities. Knowledge graphs more generally represent domain entities as persistent semantic objects [8], and legal knowledge graphs apply this approach to contracts, clauses, persons, and signatures [4,13]. The violations generated by those same entities are usually not modeled the same way, so the graph cannot answer whether a particular violation has been previously detected, reviewed, or remediated.

The Violation Situation Pattern (VSP) resolves this problem by representing each detected violation as a persistent graph node. Each violation node contains a rule identifier, a temporal validity interval governed by a staleness constraint, a finite-state lifecycle for review and remediation, and evidence relations linking the violation to the associated contracts and persons. The pattern builds on the Situation ODP proposed by Aldo Gangemi and Peter Mika [5], while extending it with lifecycle management, staleness handling, and explicit identity conditions required for compliance auditing. Every lifecycle transition is stored as an immutable graph event, enabling audit history to be reconstructed via graph traversal.

Representing violations as persistent graph entities changes what the compliance graph can answer: repeated detections attach to an existing incident rather than spawning duplicate records, review decisions become provenance-aware graph facts, and audit queries run against the same schema that already holds contracts and entities. The contributions are:

C1. The Violation Situation Pattern (VSP), a graph pattern for persistent compliance violation representation, is defined as a tuple (r, T, S, E) and is grounded in the Situation ODP.

C2. A lifecycle model with explicit transition provenance and audit-trail log: every state change is a LifecycleTransition node that includes actor, reason, and timestamp.

C3. Cross-domain validation on corporate authority compliance (V1 unauthorized signature, V2 expired mandate) against BODACC corporate-officer publications, and on GDPR compliance (V4 missing DPA breach-notification clause) against 73 GDPRhub enforcement decisions, along with contractual control detection (V3 missing confidentiality clause) verified through a SHACL cross-formalism check.

C4. Portability evidence: a SHACL cross-formalism check on V3/V4 with vocabulary extensions.

The novelty of this work is the composition of persistent identity, finite-state lifecycle, immutable transition provenance, and multi-referent evidence relations into a single graph pattern; existing ontology design patterns cover these properties individually, but none combine them for compliance-violation representation.

2 Background and Problem Setting

2.1 Compliance in Legal-Tech Systems

Most operational legal-tech compliance systems are built around an alert-driven workflow. Rules are executed as queries on contracts, governance datasets, or process records, and the resulting alerts are passed to compliance personnel for further action. Follow-up activities such as review, escalation, dismissal, or remediation are usually managed through external ticketing and workflow systems rather than within the graph itself. As a result, the graph stores entities such as contracts, organizations, clauses, and signatories, while the compliance violations they generate are not maintained as persistent objects.

Governance demands more than merely executing rules repeatedly. During regulatory review or audit, organizations must demonstrate when a violation was first detected, who assessed it, and how it was resolved. That history rarely sits inside the graph; it is scattered across emails, spreadsheets, and case-management tools, and the lifecycle of a compliance incident cannot be reconstructed from detection outputs alone.

2.2 Properties of Compliance-Grade Representation

Compliance-oriented representation requires three core properties: stable identity, lifecycle management, and immutable provenance. Stable identity makes certain that when a detection rule is executed again, the system recognizes an existing violation rather than creating a duplicate instance. Lifecycle management records the current stage of the violation, such as detected, under review, confirmed, remediated, or dismissed, together with the actions associated with each state. Immutable provenance ensures that every transition or decision becomes part of the audit history and can be reconstructed directly from the stored data. Established approaches, including SHACL `sh:ValidationReport` graphs, relational alert repositories, and event-driven ticketing systems, support some of these capabilities individually. However, none combines all three within a single governing representation framework.

2.3 Legal Knowledge Graphs (LEM and CLM)

DiliTrust's Legal Entity Management (LEM) captures corporate governance facts: companies, governance bodies, mandates with appointment and cessation dates, and signing authorities. Contract Lifecycle Management (CLM) captures contract-level facts, including contract metadata, contracting parties, signatures,

and clauses by typology. Clause-boundary and typology extraction that feeds CLM ingestion is inspired by contract NLP benchmarks such as CUAD [7] and ContractNLI [10]. Graphs of this shape support entity-level queries (who serves on which board, which contracts a company has signed), but compliance checks are queries over those entities rather than facts within the graph. A query that returns a contract signed without an active mandate surfaces a fact; that fact does not enter the data as a persistent object.

2.4 The Representational Gap

In the framework proposed by Aldo Gangemi and Peter Mika [5], a situation is a reified context linking entities, properties, and a temporal scope. A detected compliance violation fits this structure because it connects a rule, a contract, a clause or a person, a detection time, and a review state that evolves over time. Situation reification is already well established in ontology design. However, existing catalogue patterns do not include compliance-specific requirements such as a lifecycle automaton, immutable transition history, multi-referent evidence relations, and a staleness constraint.

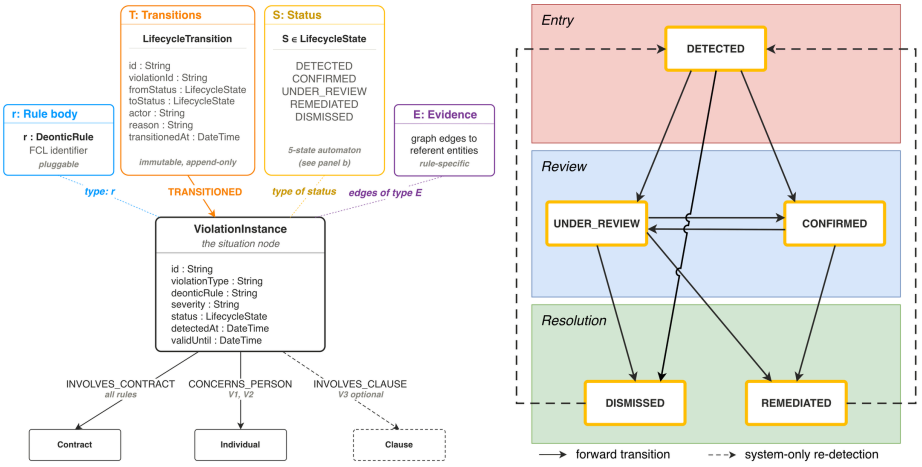
Table 1 compares the closest catalogue patterns throughout four criteria: stable identity under repeated evaluations, finite-state lifecycle support, immutable transition tracking, and multi-referent evidence cardinality. None satisfies all four simultaneously. The Violation Situation Pattern (VSP) fills this gap by extending the Situation pattern with a lifecycle automaton, a staleness constraint, and an explicit identity mechanism across re-evaluations.

Table 1. ODP catalogue gap analysis. Columns: ID = stable identity across re-evaluations; LC = finite-state lifecycle with named transitions; AT = immutable audit/transition log; EV = multi-referent evidence cardinality. ✓ = directly provided; ∼ = partially provided; ✗ = not provided.

Pattern (source)	**ID**	**LC**	**AT**	**EV**	**Gap for VSP**
Situation / Description+Situation [5]	∼	✗	✗	✓	No identity, lifecycle, or transition log
EventCore [11]	✗	✗	✗	∼	Point-in-time only; no lifecycle or AT log
TimeIndexedSituation (ODP catalogue)	∼	✗	✗	✓	Adds T; no S or AT log
TimeIndexedPersonRole (ODP catalogue)	✓	✗	✗	∼	Identity per role-period; no lifecycle
Participation / AgentRole (ODP catalogue)	✗	✗	✗	∼	Binary participation; no state-bearing object
N-ary Relations [14]	✓	✗	✗	✓	Closest structural sibling; no LC or AT log
PROV-O `prov:Activity` [12]	∼	∼	✓	∼	Provenance trail; no native LC states or transition table per situation

We reviewed the WOP 2023, 2024 (LKM proceedings), and 2025 collections to recognize patterns addressing violation reification, situation lifecycle automata, or audit-oriented detected events. No such pattern was identified. A parallel pattern-design effort from WOP 2025, ARGOS [15], targets governance patterns for LLM-driven applications and data operations, which is a different problem from persistent violation modeling. The combination of these extensions is essential for the audit capabilities targeted by VSP. Reconstructing a violation's review history, including cases that were dismissed and later re-detected, requires the joint presence of persistent identity, lifecycle state management, and an immutable transition history.

3 The Violation Situation Pattern



(a) Pattern components **(b) Lifecycle automaton**

Fig. 1. The Violation Situation Pattern. (a) Pattern components: rule body (r), transitions (T) with the LifecycleTransition log, status (S) automaton, and evidence (E). Evidence cardinality is rule-specific; per-rule details are given in Section 4.3. (b) The five-state lifecycle automaton. Solid arrows denote forward transitions (human reviewer or system revalidation); dashed arrows denote system-only re-detection from REMEDIATED or DISMISSED back to DETECTED.

3.1 Pattern Intuition

Treat each detected violation as a graph node. The node has stable identity (so re-evaluation can recognize it), evidence edges to the entities the rule concerns (so the violation is traceable), a lifecycle state (so review and remediation can be expressed), and a transition log that makes audit history immutable. Detection creates a violation object; review and remediation transition it; auto-revalidation re-confirms or releases it.

3.2 Formal Definition

We define the Violation Situation Pattern as the tuple

$$\text{VSP} = (r, T, S, E) \tag{1}$$

where r is the Formal Contract Logic (FCL) rule identifier the violation instantiates, $T = [\texttt{detectedAt}, \texttt{validUntil}]$ is the temporal validity interval, $S \in \{\text{DETECTED, CONFIRMED, UNDER\_REVIEW, REMEDIATED, DISMISSED}\}$ is the lifecycle state, and E is a set of evidence edges binding the violation to its referent entities.

Following the ODP catalogue style of Presutti and Gangemi [16]:

Context. A knowledge graph in which a recurring rule-driven check produces detection events that consumers need to query historically without re-executing the rule.
Solution. Reify each detection as a ViolationInstance node carrying (r, T, S, E). Pair it with an immutable LifecycleTransition class so the projection (current ViolationInstance state) and the event log (LifecycleTransition history) are both graph objects. Enforce two invariants at the application layer: only transitions in the lifecycle automaton are accepted, and `detectedAt` is immutable.
Consequences. The tradeoffs are application-layer (not schema-level) enforcement and a property-graph realization that does not directly support OWL entailment.
Related patterns. Situation [5] (ancestor); TimeIndexedPersonRole, Participation+AgentRole (referent-node patterns); PROV-O `prov:Activity` [12] (LifecycleTransition is a property-graph counterpart).

Competency queries. The composition of (r, T, S, E) supports audit questions that stateless detection cannot answer in a single traversal. For example: *Was violation v previously dismissed and later re-detected?* is a path query over the LifecycleTransition log with *fromStatus* = DISMISSED and *toStatus* = DETECTED; *Which violations were active at time t?* is a temporal-window query over ViolationInstance restricted to $\texttt{detectedAt} \leq t \leq \texttt{validUntil}$; *Which entities are jointly implicated in two or more open violations?* is a multi-hop join across rule-specific evidence edges. Under stateless detection these require correlating an external ticketing or workflow system with the underlying graph; under VSP each is a single Cypher traversal.

Properties under re-evaluation.
Proposition 1 (Re-evaluation idempotence). Let $\kappa(r, x) = (r.\text{id}, \text{evidence\_key}(x))$ be the composite identity function instantiated by VSP. For any sequence of detections $\langle d_1, \ldots, d_n \rangle$ produced by re-evaluating r over time, if $\kappa(r, d_i) = \kappa(r, d_j)$ for some $i \neq j$, then d_i and d_j map to the same ViolationInstance node and contribute a unioned validity interval rather than distinct nodes.
Proposition 2 (Audit-history reconstructibility). For any ViolationInstance v, the complete sequence of lifecycle states v has occupied is recoverable by a single traversal of the LifecycleTransition nodes connected to v, or-

dered by `transitionedAt`. This holds independently of how many re-evaluations have occurred.

Proposition 1 follows from the `MERGE` semantics of the identity key (Section 3.3); Proposition 2 from the append-only invariant on LifecycleTransition (Section 3.5). Neither holds for any single ancestor pattern in Table 1: Situation lacks the identity key, TimeIndexedPersonRole lacks the lifecycle log, and PROV-O `prov:Activity` lacks a per-situation projection node. The pattern contribution is the joint realization of both properties on a single situation node.

3.3 Graph Realization

The tuple defines the pattern abstractly; the property-graph realization below is one instantiation, and Section 5.3 shows the same pattern carries to RDF stacks via a SHACL+PROV-O retrofit. VSP realizes in a property graph as four classes — ViolationInstance (the situation node), LifecycleTransition (the immutable event-log node), `TRANSITIONED` edge, and evidence edges (domain-specific; in the legal domain, `INVOLVES_CONTRACT`, `CONCERNS_PERSON`, `INVOLVES_CLAUSE`).

Identity is composite: $(\text{violationType}, \text{contract\_id}, \text{person\_id})$ for V1/V2 and $(\text{violationType}, \text{contract\_id})$ for V3/V4. Re-evaluation matching an existing key extends `validUntil` rather than creating a duplicate.

3.4 Lifecycle Automaton

The state machine has five states and ten arcs, shown in Fig. 1b. Actor type (human reviewer vs. system revalidation) is recorded on each LifecycleTransition node rather than constrained per arc; in practice DETECTED→CONFIRMED and CONFIRMED→REMEDIATED can be triggered by either a reviewer's confirmation or the auto-revalidation cycle, while DISMISSED→DETECTED and REMEDIATED→DETECTED are taken exclusively by the system revalidator.

DISMISSED→DETECTED records re-detection on a later evaluation cycle of a violation an earlier review had judged not actionable. REMEDIATED→DETECTED records re-emergence of a previously resolved violation when the underlying condition reappears. Each accepted transition writes a LifecycleTransition node with `TRANSITIONED` edge to the ViolationInstance.

3.5 Enforcement and Threat Model

Two invariants are required: only arcs in the lifecycle automaton (Fig. 1b) are committed, and `detectedAt` is immutable across an instance's lifetime. Property-graph stores generally lack column-level immutability or transition-table constraints, so both invariants are enforced at the application layer through

a single `transition()` mediator. The threat model is explicit: a caller bypassing this mediator with a direct write can violate either invariant. Schema-level enforcement is achievable on RDF-star + SHACL stacks [9,18] (Section 6).

4 Rule Operationalization

We operationalize four deontic rules through a three-layer pipeline: FCL deontic formula → Cypher detection pattern → MERGE step instantiating a persistent ViolationInstance. Table 2 lists the rules.

Table 2. Four implemented rules. V1, V2, V4 are statutory; V3 is policy-derived (see Section 4.3).

Rule	Type	Severity	Source
V1 Unauthorized Signature	statutory	HIGH	Ultra vires; Dir. 2009/101/EC Art. 9; CdC L225-35
V2 Expired Signatory Mandate	statutory	HIGH	CdC L225-35 (corporate authority expiry)
V3 Missing Confidentiality Clause	policy	MEDIUM	Contractual control (commercial practice)
V4 Missing DPA Breach-Notification	statutory	HIGH	GDPR 28(3)(f), 33; SCC 2021/915 [3] Art. 9

CdC: French Code de Commerce. DPA: Data Processing Agreement. SCC: Standard Contractual Clauses.

V1 and V2 use the full FCL form; V3 and V4 use a clause-presence operationalization of the FCL obligation, because the data layer represents clauses by `clauseType` rather than by structured obligation flags. We discuss this rule-correctness boundary explicitly in Section 4.3.

4.1 V1: Unauthorized Signature

FCL form (informal).

$$\begin{aligned}&\text{OBLIGATION}(p, sig, c) \Rightarrow \exists\, m : \\ &\quad \text{holds}(p, m) \wedge \text{at}(m, c.party) \\ &\quad \wedge\ \text{scope}(m) \supseteq \{\text{SIGNATURE\_AUTHORITY}\} \\ &\quad \wedge\ \text{valid}(m, c.signedAt)\end{aligned}$$

A violation is the negation of the consequent for any non-Other signatory role.

Detection identifies signatures by individuals whose `AUTHORIZED_BY` mandates either lack Signature_Authority scope or are out of temporal validity at signing date. Matching cases trigger a `MERGE` instantiating a ViolationInstance with `INVOLVES_CONTRACT` and `CONCERNS_PERSON` evidence edges and the appropriate temporal-validity defaults.

4.2 V2: Expired Signatory Mandate

V2 differs from V1 in that the signer once held a SIGNATURE_AUTHORITY mandate. Detection finds an `AUTHORIZED_BY` mandate whose $endDate < c.signedAt$ and confirms no replacement mandate covers the signing date. The `MERGE` step is structurally identical to V1, with $violationType$ = `'V2_EXPIRED_MANDATE'` and the same evidence edges. The legal distinction is substantive: V1 is ultra vires from inception; V2 is action beyond expiry. Under French corporate law the remedies and ratification options differ.

4.3 V3 (Missing Confidentiality) and V4 (Missing Breach Notification)

V3 flags commercial contracts (excluding NDAs and DPAs) lacking a CONFIDENTIALITY clause. V3 is policy-derived: confidentiality is a standard contractual control rather than a statutory obligation and is labeled a violation for uniformity with V1/V2/V4 rather than normative parity. V3 demonstrates that VSP applies identically to policy-derived rule bodies: the pattern's identity, lifecycle, and evidence semantics are independent of the rule's legal status. V4 flags DPAs lacking a BREACH_NOTIFICATION clause per SCC 2021/915 Art. 9. V4 detects clause presence rather than clause compliance. A DPA with a 90-day notification window passes V4 but violates GDPR Art. 33's 72-hour requirement. This is a rule-correctness boundary, not a pattern boundary; Section 5.2 quantifies the recall this operationalization achieves on regulator-issued labels. Evidence cardinality is rule-specific: V1 and V2 attach `INVOLVES_CONTRACT` and `CONCERNS_PERSON` edges to CONTRACT and INDIVIDUAL nodes respectively; V3 attaches `INVOLVES_CONTRACT` and, when the contract carries clause nodes, an `INVOLVES_CLAUSE` edge to a CLAUSE node; V4 attaches `INVOLVES_CONTRACT` only.

5 Evaluation

The evaluation supports the pattern claim through four pieces of evidence of distinct types: a gazette-grounded consistency check for V1/V2 (BODACC, Section 5.1), a scoped operational evaluation of V4 against regulator decisions (GDPRhub, Section 5.2), a cross-formalism portability check on V3/V4 (SHACL, Section 5.3), and a schema-transfer sanity check on a second domain (GDPR consent, Section 5.4). The graph carries 271 COMPANY, 249 INDIVIDUAL, 178 MANDATE, 215 CONTRACT, 311 CONTRACTPARTY, 100 CLAUSE, and 140 VIOLATIONINSTANCE nodes, plus 19 LIFECYCLETRANSITION nodes from a deterministic lifecycle exercise reaching all five states.

5.1 Consistency Check on BODACC Corporate Authority

This is a gazette-grounded consistency check on V1/V2, not a full external accuracy validation: labels and rule condition both reference the same temporal predicate over the gazette record.

The original V1/V2 controlled dataset was generated from signatory titles, meaning that positive rule instances were partially linked to title-derived synthetic mandates. To reduce this dependency using real-world evidence, we retrieved data from BODACC [2], the French *Bulletin officiel des annonces civiles et commerciales*, through the OpenDataSoft public API. The collection process produced 281 modification records covering 40 French joint-stock companies (SA/SAS) and yielded 243 mandate periods, of which 177 contained both start and end dates. For a given SIREN identifier (the French business registration number), two consecutive modification entries define a gazette-certified mandate interval that exists independently of the detection logic.

From the 177 complete mandate periods, two evaluation datasets with different evidentiary characteristics were constructed. The first was a gazette-grounded V2 dataset containing 354 cases: 177 CLEAN cases where signing occurred within the mandate period and 177 V2 cases where signing took place 30 days after the mandate had expired. Labels in this dataset were derived directly from gazette-published start and end dates. The second dataset, referred to as V1-proxy, consisted of 40 cases where signing occurred 180 days before the earliest available mandate record for the officer. These labels were inferential rather than regulator-certified. Specifically, the absence of an earlier gazette record was treated as evidence that no SIGNATURE_AUTHORITY mandate existed at the signing date. While this assumption is reasonable, it should not be interpreted as an official regulatory finding. To further challenge the detection mechanism, a separate 20-case adversarial dataset was created using deliberately generic titles and mandate scopes defined independently from the signatory title.

On the gazette-grounded V2 dataset, the rule separated all 177 CLEAN cases (signing within mandate) from all 177 expired-mandate cases (signing 30 days post-expiry) without error. The V1-proxy dataset and the 20-case adversarial-titles dataset produced the same clean separation. We report this as a separation outcome rather than as precision/recall numbers by design: the labels and the rule both reference the same temporal predicate over the gazette record ($signedAt > endDate$ for V2, missing-record for V1-proxy), so a metric-style report would overstate what the check evidences. What the result does show is that the rule applies real temporal information correctly and that, on the adversarial dataset, detection does not depend on signatory titles. Independent external validation would require a publicly accessible corpus pairing real contracts with regulator-confirmed signing-authority outcomes, which is not currently available.

5.2 Scoped Operational Evaluation: V4 on GDPRhub Decisions

Finding. On 73 GDPRhub Article 33 enforcement decisions, extending V4's rule body from clause-presence (Rule A) to clause-presence-plus-72h-deadline (Rule B) raises F_1 from 0.312 to 0.602 ($\Delta F_1 = +0.290$; Fig. 2). The detector improvement is real but secondary; the pattern-level claim is that audit continuity survives the rule revision. Every DPA flagged under Rule A retains its VIOLATIONINSTANCE identity under Rule B via the composite key (Propo-

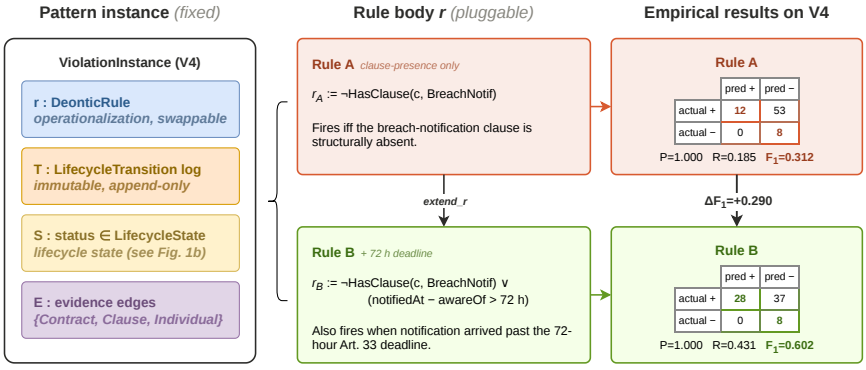


Fig. 2. Rule-body independence on V4. The pattern instance (left, fixed) is unchanged across two rule bodies (center, pluggable). Rule A checks clause presence only; Rule B extends r to also check the 72-hour notification deadline. Empirical results on 73 GDPRhub enforcement decisions (right) show $\Delta F_1 = +0.290$ from extending r, with the pattern's identity, lifecycle, and evidence semantics unchanged.

sition 1), the LifecycleTransition history accumulated under Rule A remains valid under Rule B without replay or migration (Proposition 2), and dismissed-then-redetected traversals span both rule-body regimes. Compliance teams need exactly this property when regulator interpretation forces detection logic to evolve, and no individual ODP in Table 1 delivers it. The 37 *other-Art. 33* findings sit outside both rule bodies by design (Section 6); covering them is a detector task, outside the pattern's scope.

Ground-truth construction. The synthetic V4 set is circular by construction: violated cases are constructed to lack the breach-notification clause. To externalize V4, we query the GDPRhub [6] MediaWiki API for all 92 non-subcategory cases tagged `Category:Article_33_GDPR`, fetching the structured outcome field and the English summary; 73 yield parseable outcomes and article citations. Cases are labeled through a transparent outcome→label mapping (*Violation Found*, *Upheld*, *Partly Upheld* → violation; *No Violation Found*, *Rejected*, *No further action* → non-violation; three parser-glitch outcomes hand-resolved against the case fine and substance), yielding 65 violations and 8 non-violations. Subtype assignment combined keyword classification with manual verification of all 65 violations against the regulator's case summary and Holding section, producing 12 missing-notification, 16 late-notification, and 37 other-Art. 33 violations. The residual *other* category collects content-incompleteness, documentation-only Art. 33(5) findings, procedural violations, and content-related findings outside both rule bodies. Labels span 18 supervisory authorities across 10 EU member states.

Results. V4 clause-presence detection (Rule A — fires iff the BreachNotification clause is absent) achieves $P = 1.000$, $R = 0.185$, $F_1 = 0.312$ ($TP = 12$, $FN = 53$, $FP = 0$, $TN = 8$; Table 3). Precision of 1.0 holds

across all 8 regulator-confirmed non-violations and across all 37 other-Art. 33 findings outside the rule body's scope: the rule fires only on cases where the breach-notification clause is genuinely absent, including under adversarial-direction inputs (regulator decisions where Art. 33 was found violated for reasons other than clause absence, plus regulator decisions where no Art. 33 violation was found at all). Extending r to a deadline-checking rule body (Rule B — fires iff clause absent OR notification past 72 h) achieves $P = 1.000$, $R = 0.431$, $F_1 = 0.602$ ($TP = 28$, $FN = 37$, $FP = 0$, $TN = 8$). For 8 of the 16 verified late-notification cases the regulator's case summary confirmed the delay exceeded 72 h without stating an exact value; we treat the regulator's finding as ground truth for rule-firing purposes, recording these as `above_threshold_unspecified=true` in the verification artifact with the numeric hour field left null. The remaining 8 cases have specific hour values extracted from the case summaries using GPT-4.1 as a structured-extraction tool, with all extractions manually verified by the authors against the source document.

Table 3. V4 detection on 73 real DPA enforcement decisions from 18 supervisory authorities across 10 EU member states. Rule A = clause-presence only; Rule B = clause-presence + deadline-check.

Finding type	n	**Rule A fires**	**Rule B fires**	**Reason**
Missing notification	12	12	12	Clause absent → both rules fire correctly
Late notification	16	0	16	Clause present, deadline >72 h → only Rule B fires
Other Art. 33	37	0	0	Outside both rule bodies' scope
No violation	8	0	0	Regulator confirmed no Art. 33 violation
Total	**73**	**12**	**28**	Rule A: F_1=0.312 (R=0.185); Rule B: F_1=0.602 (R=0.431)

5.3 Portability: SHACL Cross-Formalism Check

This section evaluates portability across formalisms (SHACL+SPARQL versus property-graph Cypher), not external detection performance. Both implementations operationalize the same clause-presence condition; the success criterion is equal firing on shared scope (V4), with expected divergence where scoping conventions differ (V3). We export the Contract+Clause subgraph to RDF/Turtle (945 triples) and run pyshacl 0.31 against shapes encoding V3 and V4 with SHACL-Core [9] SPARQL targets. V1 and V2 are not encoded as SHACL shapes: their reasoning over mandate scope and signing-date validity spans subgraphs that a single `sh:NodeShape` cannot capture without external SPARQL plumbing. Table 4 reports the firing comparison.

Table 4. Cross-formalism firing comparison. V4 shows full parity (10/10 overlap). V3 differs in scoping (VSP uses the manifest eval set; SHACL operates over the full contract-type taxonomy); each fires correctly under its own scoping convention.

Rule	VSP fires	SHACL fires	Overlap	Interpretation
V4	10	10	10	Both formalizations flag the same 10 violated DPAs on the SCC-template set.
V3	50	57	18	VSP scopes via the manifest eval set; SHACL operates on the full data-graph contract-type taxonomy. Both are correct under their respective scoping conventions.

The middle column of Table 5 makes the portability claim explicit. Most VSP capabilities are reachable in a SHACL+SPARQL+PROV-O retrofit, but only by adding a custom violation-IRI mint policy, a custom lifecycle vocabulary, an external revalidation scheduler, and a persisted report store on top of the standard. VSP collects these capabilities into one pattern with the projection-plus-log discipline encoded in the graph.

Table 5. Capability comparison: SHACL Core, SHACL+SPARQL+PROV-O, VSP. ✓ = directly supported; ∼ = supported with the additional vocabulary or orchestration named in the row; ✗ = not supported.

Capability	SHACL Core	SHACL+SPARQL + PROV-O	VSP
Detect clause-absence violations (V3, V4)	✓	✓	✓
Cross-domain governance violations (V1/V2)	✗	∼ (SPARQL over both subgraphs + temporal)	✓
Persistent identity + multi-referent evidence	✗	∼ (custom IRI mint + auxiliary triples)	✓
Lifecycle state per violation	✗	∼ (custom vocab; PROV transition activities)	✓
First-detection vs. latest-verification timestamps	✗	∼ (PROV `atTime` on generation activity)	✓
Staleness invariant + revalidation cycle	✗	∼ (external scheduler)	✓
Audit traversals (e.g., dismissed-then-redetected)	✗	∼ (persisted report store + cross-report correlation)	✓

5.4 Schema-Transfer Sanity Check: GDPR Consent

We re-instantiate VSP over a GDPR consent schema as a sanity check that the pattern's identity, lifecycle, and evidence machinery carry across to a different

domain vocabulary. The example is illustrative, not a quantitative cross-domain evaluation; a full transfer study is future work. The schema is:

```
DataSubject -[:HAS_CONSENT]-> ConsentRecord
            -[:FOR_PURPOSE]-> ProcessingPurpose
ProcessingActivity -[:PROCESSES]-> DataSubject
ProcessingActivity -[:SERVES_PURPOSE]-> ProcessingPurpose
```

The `V_CONSENT_MISSING` rule is triggered when a processing activity involves a data subject and purpose combination without valid active consent. In this setting, the core VSP structure remains unchanged. The same identity mechanism, lifecycle model, staleness handling, and evidence-cardinality semantics are reused, with the identity criterion defined as (violationType, activity_id). The five-state lifecycle automaton and the staleness invariant are also preserved. Only the rule body and the evidence relationships are adapted, using the vocabulary {`INVOLVES_PROCESSING`, `CONCERNS_DATA_SUBJECT`, `INVOLVES_PURPOSE`}.

A small `consent_demo` database (5 DataSubjects, 4 ProcessingPurposes, 6 ProcessingActivitys with 2 violations) exercised the rule; both violations were detected correctly and one was transitioned from DETECTED to REMEDIATED. The pattern carried over without changes to identity, lifecycle, or evidence mechanics; only the rule body and edge vocabulary changed.

6 Limitations and Future Work

The limitations of this work fall into two groups: rule-level boundaries and system-level properties of the property-graph realization. We discuss each below alongside its natural extension.

6.1 Rule-Level Limitations

- **V4 rule-correctness boundary (Sections 4.3, 5.2).** V4 detects clause presence, not clause compliance: a DPA with a 90-day notification window passes V4 but violates Art. 33's 72-hour requirement. Closing this gap requires clause-text Natural Language Inference (NLI), a detector extension that keeps the pattern unchanged.
- **V3 normative status (Section 4.3).** V3 is a policy-derived contractual control, not a statutory obligation; "violation" is used for uniformity, not normative parity. A `deonticForce` field on the rule would distinguish the two without altering the pattern.
- **V4 recall ceiling (Section 5.2).** Of the 37 other-Art. 33 cases unrecovered by Rule B, 25 are outside V4's scope by design (procedural fines, Art. 34 notifications, Art. 33-cited but operatively elsewhere) and 12 are within the rule body's natural extension trajectory (mis-tagged missing-notification and Art. 33(3) content-completeness). The 12 are recoverable through clause-text NLI and structured Art. 33(5) predicates; neither extension alters the pattern.

6.2 System-Level Limitations

- **Upstream extraction quality.** Precision depends on upstream extraction (contract-signatory, clause-boundary, clause-type), which we do not separately benchmark. A full system evaluation would compose extraction-stage and pattern-stage error rates.
- **Application-layer enforcement (Section 3.5).** Lifecycle invariants are enforced through the `transition()` mediator; a caller bypassing it can violate either invariant. The natural port is an RDF-star + SHACL realization that lifts the mediator into schema-level shapes.
- **Concurrency.** Concurrent transitions on the same ViolationInstance are not evaluated. An optimistic check on `lastVerifiedAt` or a serialized transition queue would address this without modifying the pattern.
- **Large-scale performance.** Production-scale (10^5–10^7 ViolationInstances) traversal performance is not evaluated. The append-only LifecycleTransition log grows linearly with re-evaluation frequency; indexing `transitionedAt` and `violationId` is the expected scaling lever.

7 Conclusion

This paper presented the Violation Situation Pattern, a graph pattern that reifies compliance violations as persistent nodes with composite identity, a five-state lifecycle, an immutable LifecycleTransition log aligned with PROV-O, and rule-specific evidence edges. Four deontic rules (V1–V4) were operationalized through an FCL→Cypher→MERGE pipeline. The strongest empirical takeaway is rule-body independence: extending V4 from clause-presence to a 72-hour deadline check raised F_1 by 0.290 on 73 GDPRhub decisions while leaving the pattern's identity, lifecycle, and evidence semantics intact, separating a pattern contribution from a detector contribution. The remaining evidence covers complementary aspects: V1 and V2 separate gazette-certified mandate periods without error on the BODACC consistency check; the SHACL cross-formalism check confirms portability of V3/V4 (not external accuracy); the GDPR-consent sanity check confirms the same identity, lifecycle, and evidence machinery applies to an unrelated schema.

Acknowledgments. The authors used LLM assistance during the preparation of this manuscript. GPT-4.1 was used as a structured-extraction tool to extract deadline values from regulator case summaries in Section 5.2, with all extracted values manually verified by the authors against the source documents. Anthropic's Claude was used for editorial assistance during manuscript preparation, including copyediting, identifying inconsistencies, and suggesting rephrasings; all suggestions were reviewed and accepted or rejected by the authors. All research contributions, methodological decisions, claims, and analyses are the authors' own.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. David, R., Ahmeti, A., Bushati, G., Tauqeer, A., Fensel, A.: SHACL-based contract compliance verification. In: ESWC 2025 Workshops (2025)
2. Direction de l'information légale et administrative: BODACC: Bulletin officiel des annonces civiles et commerciales. https://www.bodacc.fr/ (2025), accessed via OpenDataSoft public API
3. European Commission: Commission implementing decision (EU) 2021/915 — standard contractual clauses. Official Journal of the European Union (2021)
4. Filtz, E., Kirrane, S., Polleres, A.: The linked legal data landscape: Linking legal data across different countries. Artificial Intelligence and Law **29**(4), 485–539 (2021)
5. Gangemi, A., Mika, P.: Understanding the semantic web through descriptions and situations. In: On The Move to Meaningful Internet Systems 2003 (OTM 2003). LNCS, vol. 2888. Springer (2003)
6. GDPRhub: GDPRhub: A wiki of GDPR enforcement decisions. https://gdprhub.eu/ (2025), operated by noyb – European Center for Digital Rights
7. Hendrycks, D., Burns, C., Chen, A., Ball, S.: CUAD: An expert-annotated NLP dataset for legal contract review. In: Advances in Neural Information Processing Systems (NeurIPS): Datasets and Benchmarks Track (2021)
8. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Kirrane, S., Labra Gayo, J.E., Navigli, R., Neumaier, S., Ngonga Ngomo, A.C., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge graphs. ACM Computing Surveys **54**(4) (2021)
9. Knublauch, H., Kontokostas, D.: Shapes constraint language (SHACL). W3C Recommendation (2017), https://www.w3.org/TR/shacl/
10. Koreeda, Y., Manning, C.D.: ContractNLI: A dataset for document-level natural language inference for contracts. In: Findings of the Association for Computational Linguistics (EMNLP 2021) (2021)
11. Krisnadhi, A., Hitzler, P.: A core pattern for events. In: Advances in Ontology Design and Patterns, Studies on the Semantic Web, vol. 32, pp. 29–37. IOS Press (2017)
12. Lebo, T., Sahoo, S., McGuinness, D.: PROV-O: The PROV ontology. W3C Recommendation (2013), https://www.w3.org/TR/prov-o/
13. Leone, V., Di Caro, L., Villata, S.: Taking stock of legal ontologies: A feature-based comparative analysis. Artificial Intelligence and Law (2020)
14. Noy, N., Rector, A.: Defining N-ary relations on the semantic web. W3C Working Group Note (2006)
15. Pathirage, N.D., Seneviratne, O., McGuinness, D.L.: ARGOS: Ontology design patterns for governing dynamic data operations in LLM-powered applications. In: Joint Proceedings of the 16th Workshop on Ontology Design and Patterns (WOP 2025) and the 1st Workshop on Bridging Hybrid Intelligence and the Semantic Web (HAIBRIDGE 2025), co-located with ISWC 2025. CEUR Workshop Proceedings, vol. 4093, pp. 45–56. CEUR-WS.org, Nara, Japan (2025), https://ceur-ws.org/Vol-4093/paper4.pdf
16. Presutti, V., Gangemi, A.: Content ontology design patterns. In: Conceptual Modeling - ER 2008. LNCS, vol. 5231. Springer (2008), see also http://ontologydesignpatterns.org
17. Robaldo, L., Bartolini, C., Palmirani, M., Rossi, A., Martoni, M., Lenzini, G.: Formalizing GDPR provisions in reified I/O logic: The DAPRECO knowledge base. Journal of Logic, Language and Information (2020)

18. W3C RDF-star Working Group: RDF 1.2 concepts and abstract syntax. W3C Working Draft (2024)