September 2026

Building Thoroughly Tested Semantic Rule Engines in the Age of GenAI

Towards realizing Interoperable, Legally Compliant, and Trustworthy Data Flows

Patrick HOCHSTENBACH[iD] [a,b,1], Jos DE ROO[iD] [a], Wout SLABBINCK[iD] [a], Beatriz ESTEVES[iD] [a] and Pieter COLPAERT[iD] [a]

[a] *Ghent University, IMEC-IDLab, Ghent, Belgium*
[b] *Ghent University, Ghent University Library, Ghent, Belgium*

Abstract. Data sovereignty starts before data leaves a user's device. As data processing moves toward the edge, future Web applications will need to evaluate rules, policies, and explanations locally, rather than delegating every decision to server-side infrastructure. At the same time, data will increasingly flow across organizations, applications, and regulatory contexts. We argue that such flows must be interoperable across ecosystems, legally compliant with applicable policies, and trustworthy to end-users. This requires a rule language that independent engines can implement consistently. In this paper, we introduce 1k+ Notation3 compliance tests for validating implementations against the Notation3 specification, developed in parallel with Eyeling: a JavaScript, browser-deployable reference implementation for interoperable in-browser trust engines. Applied to cwm, EYE, and jen3, the suite reveals conformance gaps with the published specification, while Eyeling reaches full conformance and shows that LLM-driven synthesis can bootstrap spec-compliant rule engines from a specification and conformance suite. These results suggest that in-browser semantic rule engines can provide a practical foundation for evaluating interoperability, legal compliance, and trust where data is accessed and used. While some may see Semantic Web tooling as deprecated in the era of genAI, our work can serve as an example of a symbiosis leveraging the best from both worlds.

Keywords. Notation3, rule engine, Javascript, LLM code generation, Neurosymbolic AI

1. Introduction

The future Web will not be organized around a single global data model, a single integration platform, or a single trusted authority. Data will flow across organizations, applications, sectors, and regulatory contexts, while remaining shaped by the local needs of the actors that produce, manage, and consume it. In such an environment, data sovereignty starts before data leaves a user's device: users and applications must be able to understand, evaluate, and act on data locally, rather than delegating every decision to remote

[1] Corresponding Author: Patrick Hochstenbach, Patrick.Hochstenbach@UGent.be.

infrastructure. This requires building Web applications in which interoperability, legal compliance, and end-user trust are taken into account [5].

Interoperability cannot depend solely on prior agreement. Organizations will continue to use different vocabularies, schemas, policies, and operational assumptions. Rather than forcing all actors into a common model from the start, interoperability needs to be achieved progressively, when concrete integration needs arise. This idea of eventual interoperability shifts the problem from universal standardization to explicit interpretation: mappings, alignments, constraints, and transformations must be expressed in a form that other systems can inspect, reuse, and execute. Semantic rules are a natural fit for this role. They make translation assumptions explicit, provide a formal account of how one representation is interpreted in terms of another, and allow heterogeneous data to be aligned without erasing local semantics.

The same need for explicit, executable interpretation appears in **legal compliance** [29]. Data flows are constrained not only by technical formats, but also by permissions, prohibitions, obligations, purposes, legal bases, and contextual conditions. Policy languages such as the Open Digital Rights Language (ODRL) [19] and vocabularies such as the Data Privacy Vocabulary (DPV) [24] make it possible to describe aspects of these constraints in machine-interpretable form. However, descriptions alone are not enough. Applications must be able to evaluate whether a given data use is allowed, whether additional obligations apply, or whether access should be denied under a specific policy. For such evaluations to be interoperable across implementations, the underlying rule semantics must be shared, testable, and independent of a single proprietary engine.

Trust then becomes a question of execution as much as representation. If interoperability mappings and compliance policies are evaluated only by remote services, users and applications must trust those services before they can trust the outcome. Client-side reasoning changes this relation. Rules, policies, input data, and derivations can be executed and inspected close to the user, for instance inside a browser, a data wallet, or an edge application. Yet this only helps if the local engine itself can be trusted. A client-side rule engine must therefore be validated against a shared specification and through implementation-independent compliance tests.

Contributions. This paper addresses that need by leveraging the decades of work done creating Notation3 (a first team submission was published in 2008[2], that was later adopted by the Notation3 W3C Community Group). We present a Notation3 conformance test suite of more than thousend cases, each traceable to a specific normative clause of the N3 specification, publicly released for community use. Applied to the established reasoners cwm [2], EYE [6], and jen3 [30], the suite quantifies the conformance gap between current implementations and the published specification. We further present Eyeling, a Notation3 reasoner constructed via human-in-the-loop LLM synthesis directly from the specification text, supporting both forward and backward chaining with complete coverage of the N3 built-in functions and achieving 100% on the suite. Beyond the system itself, our work illustrates a broader methodological pattern: well-scoped Semantic Web specifications, paired with compliance tests, can serve as a viable basis for AI agents to implement core interoperable infrastructure.

[2] `https://www.w3.org/TeamSubmission/2008/SUBM-n3-20080114/`

2. Symbolic rules for trust and interoperability in the age of AI

Assertions and alignments Interoperability and trust are closely related. Every assertion is made relative to a vocabulary, a context, and an intended interpretation. When an organization states that something is true, it implicitly commits to the meanings of the terms used, the modelling choices behind them, and the conditions under which others may rely on the statement. Across ecosystems, different actors may use different identifiers for similar concepts, reuse the same term with different assumptions, or publish data at different levels of abstraction. By using a rule language and engine, one enables eventual interoperability[3]. Alignments, mappings, constraints, and rules express how one actor's assertions may be understood in another actor's context, and under which assumptions additional knowledge may be inferred. A rule can state that, for a specific purpose, a locally used class or property can be treated as corresponding to another one; that a value must be normalized before comparison; or that a conclusion only follows if a particular provenance, permission, or policy condition is present. For these resulted alignments and assertions to be trusted, one certainly also needs to trust the rule engine used, which thus needs heavy testing.

Agent spaces In an idea coined by Polleres et al. [26], data agents are a further evolution of data spaces, where not only data assets, metadata, policies, contracts, and credentials are first-class citizens, but also knowledge graphs, services, tools, and agentic language models. Their central observation is that simply allowing LLM-based agents to operate over data spaces does not by itself preserve sovereignty or trust: agents may discover data, call tools, compose services, and act on behalf of humans or organizations, but their non-deterministic behavior also makes it harder to guarantee which data was used, for which purpose, under which policy, and with which justification. The proposed answer is a neuro-symbolic architecture in which agentic models are governed by **symbolic policy guardrails**. Such guardrails make policies, constraints, contracts, and process rules explicit and executable, so that an agent may propose an action, formulate a query, or mediate between heterogeneous sources, while a deterministic symbolic component checks whether the action is permitted, whether the relevant constraints hold, and which derivation justifies the result. In this view, symbolic rules are the mechanism through which agent spaces can become controllable, explainable, and policy-compliant: they define the boundary between what an agent can suggest and what the surrounding space can accept as valid, allowed, or trustworthy.

Executable Legal Compliance Deterministic rule engines in the form of programming-language-based formalisms have recently gained renewed interest as foundations for policy and rights languages such as ODRL [10,25,8], eFLINT [3], and LegalRuleML [13, 11,27]. Their expressive power enables the modeling of complex conditions found in law, regulations, and data-sharing agreements: conditions that cannot be captured by OWL alone. While OWL offers a starting point for interoperability, no single-logic implementation has emerged that can express all legal requirements; existing approaches therefore resort to cumbersome combinations of OWL with SPARQL, SHACL, and ASP. What is needed instead is a unified logic that can express both the policies and their consequences within a single formalism [18].

[3]Eventual interoperability was coined by Pieter Colpaert in a blog post published in January 2026 `https://pietercolpaert.be/interoperability/2026/01/08/eventual-interoperability`

Notation3 (N3), as a Semantic Web language, can express rules over existing legal and policy vocabularies such as ODRL [9] and DPV [24]. Combined with semantic annotations of Terms of Service, N3 rules can infer the presence, absence, or inconsistency of legally relevant clauses, such as missing safeguards for cross-border transfers or missing transparency provisions for profiling and automated decision-making [21]. Its support for rule chaining, date-time arithmetic, string parsing, cryptography, and (scoped) negation-as-failure (SNAF) makes it possible to express policies and consequences within one deterministic execution environment, rather than relying on combinations of OWL, SPARQL, SHACL, and ASP. Recent attempts to apply Notation3 in legal compliance use-cases can be found in [28,29,18,33,22].

Thus, whether it's about scaling up trust in data sources expressed using heterogeneous data models, about providing safeguards for agents in agent spaces, or for providing legal compliance, the basis for trust anywhere is a well-tested rule engine.

3. Notation3 Conformance Test Suite

The Notation3 specification is accompanied by an official conformance test suite [4], which provides a valuable foundation for any compliance evaluation of N3 implementations. The suite was, however, assembled during the drafting phase and has not been revised alongside subsequent editorial and normative changes. As a consequence, some tests reflect intermediate drafts of the specification, deprecated language features, or the behavior of early reference implementations such as cwm [2], rather than the normative content of the team submission published in July 2023 [31]. Conversely, several grammar and built-in requirements introduced, or refined in the later stages of standardization remain unexercised. In addition the original test suite contains tests for built-ins and language features that did not make it into the team submission. This motivates the revised conformance test suite presented in this article.

To test the adherence of emerging N3 reasoners against the published N3 specification, we developed a new handcrafted test suite, `notation3tests` [15] (two of us are part of the Notation3 Community Group). The suite provides extensive coverage of N3 grammar, semantics, and adherence to the official list of built-in functions [32] together with their argument modes and datatype casting. Each test case exercises a single, atomic grammar feature, semantic feature, built-in, argument mode or datatype cast, and cites the normative section of the specification it covers. Test outcomes are partitioned into four categories:

- *conformant* : the reasoner produces the expected output;
- *non-conformant* : the reasoner produces an output where none is expected;
- *incomplete* : the expected output is absent from the reasoner's response;
- *crash* : the reasoner terminates during execution.

Within the crash category, the suite further distinguishes *expected* crashes, those arising from tests that deliberately introduce grammar errors or contradictions in the input theory, from *unexpected* crashes, which indicate implementation defects.

As of April 29, 2026, `notation3tests` comprises 972 test cases (today this is over 1000). The results of running the compliance suite against three principal existing N3

[4] `https://w3c.github.io/N3/tests/`

Table 1. Overview of compliance tests results for the cwm, EYE and jen3 N3 reasoners (as of April 29,2026).

Test (N=972)	**cwm**	**EYE**	**jen3**
conformant	681 (70%)	786 (81%)	735 (76%)
non-conformant	20 (2%)	22 (2%)	48 (5%)
incomplete	220 (23%)	147 (15%)	146 (15%)
crashed	51 (5%)	17 (2%)	43 (4%)

implementations (cwm commit `7a71cfd` [2], EYE `v11.23.10` [6], and jen3 commit `3e6c11d` [30]) are presented in Table 1.

These three scores indicate a conformance gap between the existing implementations and the final published N3 specification, but the gap is not a measure of implementation quality. Each reasoner completed its main development cycle before the final Notation3 community group report was available, and each project validates against its own project-specific test suite or against the original W3C conformance suite mentioned at the start of this section.

In order to bridge the conformance gap and provide a fully compliant implementation, we set out to create a new N3 reasoner using LLM-driven synthesis, with our test suite made available to the LLM from the outset. The resulting Eyeling reasoner (described in detail in Section 4) reached a near-100% pass rate by version 1.6.13 using the live version of `notation3test` that existed at that time (the 73rd version in chronological order). At that point, 440 of the eventual 972 tests had been written, so this milestone reflects compliance against roughly 45% of the final suite.

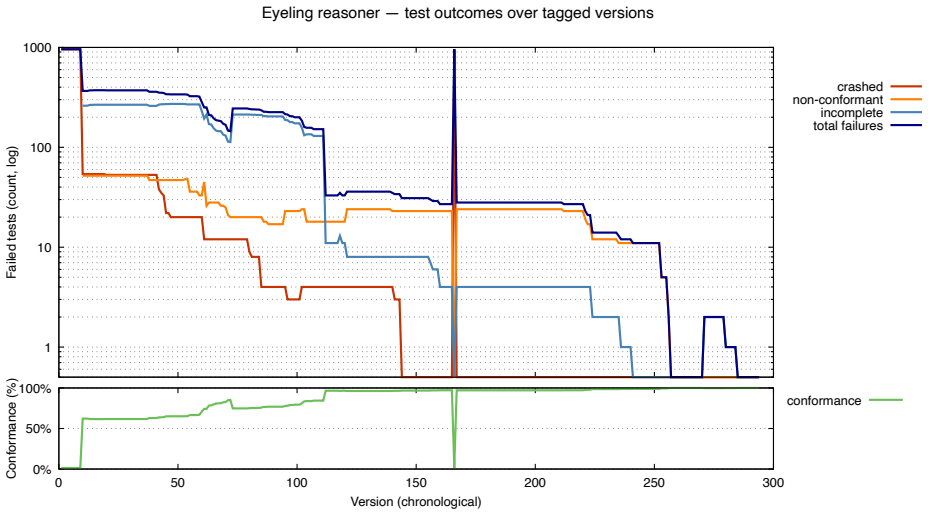

Figure 1. Eyeling conformance over tagged versions. Top: share of failing tests broken down by failure mode (crashed, non-conformant, incomplete) on a logarithmic axis. Bottom: percentage of conforming tests over the same versions.

This headline figure, however, overstates true compliance: the suite was incrementally extended by the testing team throughout the project, so each historical version of the Eyeling reasoner was scored only against the tests that existed at the time. A more

faithful measure can be obtained *post factum*, by running the 972-test suite against every historical version of the Eyeling reasoner. The results of this retrospective evaluation of `notation3tests` are shown in Figure 1 (up to April 29, 2026). The single outlier visible in the figure is version 1.11.10 (the 167th version, chronologically), for which the LLM synthesis run did not yield an executable reasoner.[5]

Eyeling's seemingly "100%" score is a consequence of its development being carried out in lockstep with the compliance suite. This co-development is an obvious limitation of our compliance claims. To provide confidence in our results, two authors not involved in the prompting sessions reviewed the parser, rule normalizer, and built-in dispatch module for case-specific branches and test-string matching (such as the names of test cases, names of predicates used in the test cases). No such patterns were found, which suggest a genuine reasoning behavior rather than overfitting to the test results. As an additional external check, on May 20, 2026 we ran the latest Eyeling version at that time, v1.24.27, against the official W3C N3 Turtle and syntax tests; it passed 308 of 339 (91%). Some of the remaining failures correspond to language features the team submission excluded, as discussed in Section 3.

We hope that publishing our compliance suite will encourage the development of further comparable implementations and provide a common benchmark against which their conformance can be measured.

4. Eyeling

Eyeling is an LLM-driven synthesized N3 reasoner produced by the project's reasoning team using *GPT-5 Extended Thinking* (using versions 5.1, 5.2, 5.4 and 5.5) throughout the course of the project. The inputs used to develop the Eyeling reasoner comprised the three N3 specifications [31,32,1] and our `notation3tests` compliance test suite. Beyond compliance, the synthesis had to produce a mature implementation whose performance was on par with the EYE reasoner. This required thousands of prompts guided by human feedback. When the LLM was stuck, selected fragments of the EYE prolog source were provided as hints. Eyeling is therefore not a translation of EYE: the underlying V8 engines (Eyeling) and SWI-Prolog (EYE) differ in ways that materially affect implementation choices, clause indexing as the most consequential. As of April 29, 2026, this process had produced 294 chronological versions.[6]

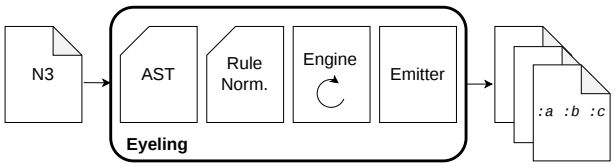

Figure 2. High-level architecture of the Eyeling reasoner.

[5]The LLM-driven synthesis run produced source code that could not be executed, because the resulting file lacked the required executable permission bits. The testing team chose not to correct the synthesized reasoner even for such simple cases.

[6]This corresponds to approximately five months of human-assisted development time.

Figure 2 provides a high-level architectural overview of the Eyeling reasoner. Eyeling accepts one or more N3 documents as input, each parsed individually into an internal AST representation. A *Rule Normalization* step then applies blank node normalization to the AST, promoting blank nodes appearing in rule premises to universally quantified variables, as required by the N3 specification. This step also handles RDF collections, ensuring that both `rdf:first`/`rdf:rest` chains and native N3 list notation are treated uniformly throughout the reasoning pipeline. The normalized rules and initial facts are then passed to the reasoning engine, which iteratively applies forward and backward chaining until a fixpoint is reached. Derived solutions are emitted as N3 output in a streaming fashion.

Both forward and backward chaining are supported within a single N3 input document, in accordance with the N3Logic semantics as specified in [1]. In addition, the synthesized reasoner implements complete coverage of the N3 Built-in Functions specification.

Despite its LLM-driven synthesis, Eyeling has no runtime dependencies on any language model: every execution consists entirely of deterministic symbolic reasoning steps. Once generated, Eyeling operates as a fully self-contained deterministic symbolic N3 reasoner. The implementation is written in pure JavaScript and has no external dependencies and no runtime requirements (e.g., no WebAssembly runtime). It can be executed as a command-line tool, consumed via an API, or embedded directly in a web browser.[7]

In addition to its score on the compliance suite (Section 3), Eyeling was also the fastest of the four N3 implementations (eye, eyeling, cwm and jen3) on the compliance-suite throughput. These timings should be read with caution: the compliance suite is correctness-oriented and does not exercise the math-heavy or large-scale workloads that engines like EYE are typically used for.

We compared the latest Eyeling version (v2.35.9 at the time of writing) against four benchmark families: Deep Taxonomy [23], LUBM(1) [14], OpenRuleBench (join1/tc/sg) [20], and eight Gabriel compute kernels [12]. On LUBM(1), Eyeling answers all fourteen queries correctly at its parse floor and is the fastest of the N3 reasoners tested (EYE, jen3, cwm), though still about $10\times$ slower than the fastest Datalog engines. On join-heavy Datalog workloads it is 6–$18\times$ slower than the best Datalog engine.

5. Discussion: AI-Assisted Semantic Infrastructure

Eyeling is a symbolic component that can be embedded next to AI agents. In an agent space, an LLM may communicate with users and services, but a N3 reasoner such as Eyeling can act as a meaning boundary: it decides which conclusions follow from the formalized knowledge available to the system. The LLM can help author, refactor, or explain N3 rules, but the rule engine remains responsible for deterministic derivation and validation. This separation is particularly important for trust, because the outcome of a policy or interoperability decision should not depend solely on a probabilistic model invocation.

A browser-native rule engine further strengthens this trust relation. If a remote server is the only place where policies are evaluated, then the user must first trust that server before trusting the decision it returns. By contrast, an in-browser reasoner allows facts,

[7] `https://eyereasoner.github.io/eyeling/playground`

policies, checks, and explanations to be evaluated close to the user and close to the data. For example, an agent may ask whether it may use a dataset for a given purpose, whether a planned action is compatible with a contract, or which facts justify a recommendation. Eyeling can represent the relevant data, rules, checks, and answers in one N3 artifact, enabling governed and inspectable decisions without requiring the user to expose all raw data to a remote policy engine.

This also suggests an alternative to unconstrained data sharing in agent spaces. Instead of giving agents broad access to raw data, a local rule engine can derive narrow, purpose-limited, and auditable insights. The Answer–Reason Why–Check pattern supported by Eyeling follows this idea: the system can compute an answer, provide the derivation that explains why the answer follows, and verify constraints that must not be violated. Such a pattern is well suited to agent spaces, where agents should operate over governed insights rather than unrestricted data dumps. It allows symbolic policies to determine not only whether an action is allowed, but also which explanation can be exposed to the user, another agent, or an auditor.

Finally, agent spaces require portability. Symbolic guardrails are only useful if they can be deployed where agents actually run: in Web applications, data-space connectors, policy editors, sandboxed runtimes, personal data stores, and client-side agent environments. Eyeling targets this requirement by providing a JavaScript implementation of N3 reasoning that can run in browsers and standard Node.js environments.

A natural question is whether this synthesis methodology generalizes beyond Eyeling's JavaScript target. As a check, we used Claude Opus 4.7 to synthesize a C++ N3 reasoner from the W3C specifications, the `notation3tests` suite, and Eyeling's source, with minimal human input [17]. After 29 commits and about six hours we had a spec-compliant reasoner: it matches Eyeling on *compliance*, but not on *maturity* or *performance*. One caveat: the N3 specification leaves room for interpretation, and the C++ implementation resolved the ambiguous cases the same way Eyeling does, so agreement on the test suite is not independent conformance. Still, the experiment suggests that once a reference implementation exists, porting to new platforms via LLM synthesis is feasible and cheap. Independent LLM-generated N3 reasoners have since appeared in Rust [7], Python [4] and Go [16], each with different features and performance.

6. Conclusion

We position the role of interoperable rule engines, like Eyeling, as a solution for symbolic guardrails in agent spaces, as a solution for automating legal compliance, and as a solution for scaling up assertions across data sources using heterogeneous data models. This requires a rule language with extensive tests, for which in this case we've chosen the RDF-native N3 language, and made sure running those rules becomes inspectable, portable, and testable across implementations. The results presented in this paper indicate that such rule engines can be built for the browser, validated against a shared conformance suite, and even synthesized with substantial assistance from LLMs when the underlying specification is precise enough. The blue-sky vision emerging from this research is that GenAI does not deprecate Semantic Web infrastructure, but become a means to accelerate the construction of the deterministic components needed for building trustworthy data ecosystems.

References

[1] Arndt, D., Champin, P.A.: Notation3 semantics. https://w3c.github.io/N3/reports/20230703/semantics.html (2024), accessed: 2024-12-20

[2] Berners-Lee, T.: cwm – a general purpose data processor for the semantic web. https://github.com/linkeddata/swap, accessed: 2026-04-29

[3] van Binsbergen, L.T., Kebede, M.G., Baugh, J., van Engers, T., van Vuurden, D.G.: Dynamic generation of access control policies from social policies. Procedia Computer Science (Jan 2022).

[4] Colpaert, P.: pyling-n3 (Jul 2026), https://pypi.org/project/pyling-n3/

[5] De Meester, B., Rojas, J.A., Ongenae, F., Colpaert, P., Verborgh, R.: Tackling the write-to-read web of data with trustflows (2025)

[6] De Roo, J.: eyereasoner/eye: Euler yet another proof engine. https://github.com/eyereasoner/eye, accessed: 2024-05-25

[7] De Roo, J.: eyereasoner/eyeron (Aug 2026), https://github.com/eyereasoner/eyeron, original-date: 2026-06-30T21:51:57Z

[8] De Vos, M., Kirrane, S., Padget, J., Satoh, K.: ODRL policy modelling and compliance checking. In: Fodor, P., Montali, M., Calvanese, D., Roman, D. (eds.) Rules and Reasoning, pp. 36–51. Springer International Publishing, Cham (2019).

[9] Esteves, B., Rodríguez-Doncel, V.: Analysis of ontologies and policy languages to represent information flows in GDPR. Semantic Web **15**(3), 709–743 (2024).

[10] Fornara, N., Chiappa, A., Colombetti, M.: Using Semantic Web Technologies and Production Rules for Reasoning on Obligations and Permissions. In: Lujak, M. (ed.) Agreement Technologies, pp. 49–63. Springer International Publishing, Cham (2019).

[11] Francesconi, E., Governatori, G.: Patterns for legal compliance checking in a decidable framework of linked open data. Artificial Intelligence and Law **31**(3), 445–464 (Sep 2023).

[12] Gabriel, R.P.: Performance and Evaluation of LISP Systems. The MIT Press (Aug 1985). , https://direct.mit.edu/books/book/3870/Performance-and-Evaluation-of-LISP-Systems

[13] Gandon, F., Governatori, G., Villata, S.: Normative requirements as linked data. In: Legal Knowledge and Information Systems, pp. 1–10. IOS Press (2017).

[14] Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics **3**(2), 158–182 (Oct 2005). , https://www.sciencedirect.com/science/article/pii/S1570826805000132

[15] Hochstenbach, P.: notation3tests. https://codeberg.org/phochste/notation3tests, accessed: 2026-04-29

[16] Hochstenbach, P.: romeo (2026), https://patrickhochstenbach.net/git/phochste/romeo

[17] Hochstenbach, P.: Tango: a C++20 spec compliant Notation3 reasoner (Aug 2026). , https://doi.org/10.5281/zenodo.22140562

[18] Hochstenbach, P., Esteves, B., Verborgh, R.: Automated policy negotiation: A four-course meal. In: Proceedings of the OPAL Workshop. CEUR Workshop Proceedings, vol. 3977. Portorož, Slovenia (2025), https://ceur-ws.org/Vol-3977/OPAL2025-2.pdf

[19] Iannella, R., Villata, S.: ODRL Information Model 2.2 (2018), https://www.w3.org/TR/odrl-model/

[20] Liang, S., Fodor, P., Wan, H., Kifer, M.: OpenRuleBench: an analysis of the performance of rule engines. In: Proceedings of the 18th international conference on World wide web. pp. 601–610. WWW '09, Association for Computing Machinery, New York, NY, USA (Apr 2009). , https://dl.acm.org/doi/10.1145/1526709.1526790

[21] Molino-Peña, E., De Roo, J., Esteves, B., García, J.M., Ruiz-Cortés, A.: Detecting Contractual Risk in ODRL Policies Using DPV. In: Accepted to be Published in the Proceedings of the 2nd ODRL And Beyond: Practical Applications And Challenges For Policy-Based Access And Usage Control (OPAL 2026) workshop, co-located with the Extended Semantic Web Conference (ESWC 2026) (2026)

[22] Molino-Peña, E., Slabbinck, W., García, J.M., Ruiz-Cortés, A., Esteves, B.: Automated Validation of ODRL Policies for Usage Control and Data Spaces | OpenReview (2026), https://openreview.net/forum?id=IKYFFh0qab#discussion

[23] Osmun, T.M.: WellnessRules: N3 Implementation Through the Euler Engine (Nov 2009), https://web.archive.org/web/20220119222608/http://responder.ruleml.org/WellnessRules/files/WellnessRulesN3-2009-11-10.pdf

[24] Pandit, H.J., Esteves, B., P. Krog, G., Ryan, P., Golpayegani, D., Flake, J.: Data Privacy Vocabulary (DPV) – Version 2.0. In: Demartini, G., Hose, K., Acosta, M., Palmonari, M., Cheng, G., Skaf-Molli, H., Ferranti, N., Hernández, D., Hogan, A. (eds.) The Semantic Web – ISWC 2024. pp. 171–193. Springer Nature Switzerland, Cham (2024).

[25] Pellegrini, T., Mireles, V., Steyskal, S., Panasiuk, O., Fensel, A., Kirrane, S.: Automated rights clearance using semantic web technologies: The DALICC framework. In: Hoppe, T., Humm, B., Reibold, A. (eds.) Semantic Applications: Methodology, Technology, Corporate Use, pp. 203–218. Springer, Berlin, Heidelberg (2018).

[26] Polleres, A., Rincon-Yanez, D., Anjomshoaa, A., Dobriy, D., Sallinger, E.: From Data Spaces to Agent Spaces (2026)

[27] Robaldo, L., Pacenza, F., Zangari, J., Calegari, R., Calimeri, F., Siragusa, G.: Efficient compliance checking of RDF data. Journal of Logic and Computation **33**(8), 1753–1776 (Dec 2023).

[28] Slabbinck, W., Rojas, J., Esteves, B., Verborgh, R., Colpaert, P.: May the FORCE be with you? A Framework for ODRL Rule Compliance through Evaluation. In: NeXt-generation Data Governance workshop 2025 (2025)

[29] Slabbinck, W., Rojas Meléndez, J., Esteves, B., Colpaert, P., Verborgh, R.: Interoperable Interpretation and Evaluation of ODRL Policies. In: Curry, E., Acosta, M., Poveda-Villalón, M., van Erp, M., Ojo, A., Hose, K., Shimizu, C., Lisena, P. (eds.) The Semantic Web. pp. 192–209. Springer Nature Switzerland, Cham (2025).

[30] Van Woensel, W.: william-vw/jen3. `https://github.com/william-vw/jen3` (2025), java. Accessed: 2026-04-29

[31] Van Woensel, W., Arndt, D., Champin, P.A., Tomaszuk, D., Kellogg, G.: Notation3 language. `https://w3c.github.io/N3/spec/`, accessed: 2025-09-29

[32] Van Woensel, W., Hochstenbach, P.: Notation3 builtin functions. `https://w3c-cg.github.io/n3Builtins/`, accessed: 2025-09-29

[33] Zhao, R., Zhao, J.: Perennial Semantic Data Terms of Use for Decentralized Web. In: Proceedings of the ACM Web Conference 2024. pp. 2238–2249. WWW '24, Association for Computing Machinery, New York, NY, USA (May 2024). , `https://dl.acm.org/doi/10.1145/3589334.3645631`