SHACL-in-the-Loop: Enhancing Structural Conformance in Agentic Knowledge Graph Construction

Vladislav Snytko[1], Rishabh Jakhar[1] and Remzi Celebi[1]

[1]Maastricht University, Maastricht, The Netherlands

Abstract

Large Language Models (LLMs) enable flexible Knowledge Graph Construction (KGC) but rarely guarantee adherence to complex ontological constraints, and current benchmarks lack the textual and logical complexity needed to test this. We introduce SHACL-in-the-Loop, an exploratory architecture that uses Shapes Constraint Language (SHACL) validation as iterative feedback during agentic graph generation, and evaluate it against a pure-LLM baseline on a custom dataset that stress-tests Terminological Box (TBox) constraints. The loop acts as an effective localized filter, cutting logically inconsistent assertion boxes from 24 to 5 across 499 documents; the residual failures delineate the boundary between SHACL validation and full OWL reasoning. Aggregate extraction quality is unchanged (0.88 F_1), but a targeted repair experiment concentrates the benefit where logical errors actually occur: on 97 such graphs the loop raises micro precision from 0.79 to 0.87 and resolves 87% of the inconsistencies, while consistent control graphs are mildly affected. We trace the flat aggregate to error rarity and the limited reach of the constructed SHACL shapes, analyse which constraints the loop can act on, and quantify its operational cost together with concrete ways to reduce it.

Keywords

Knowledge Graph Construction, Large Language Models, Ontology-driven extraction, SHACL validation, Structural conformance

1. Introduction

Knowledge Graphs (KGs) organize complex domain data into queryable, reusable structures, making them central to applications such as retrieval-augmented generation [1]. Building a high-quality, FAIR domain-specific KG requires strict adherence to an underlying ontology, or Terminological Box (TBox) [2]. LLMs have largely replaced the rigid, error-prone pipelines of traditional KG construction with flexible extraction [3], but still cannot guarantee logical conformance to that ontology. Current pipelines rely on prompt guidance or embedding-based schema alignment [4, 5, 6], neither of which guarantees that the extracted graph satisfies the schema's structural constraints. Ringwald et al. [7] use SHACL to fine-tune extraction models, applying it during training rather than as an active check during generation.

Existing benchmarks also fail to stress-test ontological conformance. As Table 1 shows, OSKGC, Text2KGBench and Text2KGBench-LettrIA [8, 9, 10] consist mostly of single-sentence snippets covering a handful of axiom types. This may explain why recent systems achieve under 1% schema hallucinations and over 97% domain/range adherence on Text2KGBench-LettrIA [10] without any error-correction loop or schema-enforcing decoding. The more expressive constraints [11] that a non-negligible share of real-world domain ontologies include therefore go untested, leaving an important case unevaluated.

We introduce **SHACL-in-the-Loop**, an LLM-based KGC architecture that treats formal ontology conformance as explicit feedback: intermediate graphs are checked against SHACL constraints, and violations are passed back to an agentic LLM loop for refinement before the final graph is produced. Since existing benchmarks are largely saturated on this front, we also built a dataset to test conformance directly. We present the architecture not as a finalized production system but as an exploratory baseline for further research on ontology conformance in LLM-driven KGC. We ask (RQ1) how SHACL validation within a KGC loop affects logical consistency and triple extraction quality (Precision, Recall, F_1) relative to a pure-LLM baseline, and (RQ2) at what cost in token consumption and processing time.

Proceedings of the 5th International Workshop on Natural Language Processing for Knowledge Graph Construction (NLP4KGC 2026), co-located with the 25th International Semantic Web Conference (ISWC 2026), Bari, Italy, October 26, 2026

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-19

CEUR-WS.org/Vol-4335/paper3.pdf

Table 1
Comparison of ontology-guided KGC datasets.

Dataset	Input Texts	Triples	Classes	Object Properties	Avg. Text Size (sentences)	Logical Axiom Types
This work	499	62,161	8	24	35.25	14[a]
OSKGC [8]	10,183	19,792	2,208	1,399	1.24	5
Text2KGBench [9]	6,076	13,491	266	685	1.21	2
Text2KGBench-LettrIA [10]	4,860	14,882	208	262	1.00	5

[a] DisjointClasses, DisjointObjectProperties, EquivalentClasses, EquivalentObjectProperties, FunctionalObjectProperty, InverseObjectProperties, ObjectPropertyDomain, ObjectPropertyRange, SubClassOf, SubObjectPropertyOf, SubPropertyChainOf, SymmetricObjectProperty, TransitiveObjectProperty, Rule.

Our contributions are: (1) a benchmark of 499 text–KG pairs over a Family ontology covering 14 logical axiom types (Table 1, note a); (2) an open-source agentic KGC pipeline incorporating SHACL validation as iterative feedback; (3) a cost analysis of this loop, including how much validation feedback is redundant; and (4) an analysis of which constraints shape-based validation can reach, with a controlled repair experiment isolating where the loop improves extraction quality.

2. Pipeline and Experimental Setup

This section details the SHACL-in-the-Loop pipeline[1], our dataset and shape generation, and the evaluation setup used to compare it against a pure-LLM extraction baseline.

2.1. The SHACL-in-the-Loop Architecture

At the core of the method is an agentic LLM that incrementally builds a KG from a source document and a target TBox. Both the main agent and the explanation model (below) use Gemini 3.1 Flash Lite at temperature 0.7; the full prompts are available in the repository[2]. Owing to API budget constraints, Table 2 reports a single run per condition, while the supplementary analyses in Sections 3.2 and 3.3 use five baseline runs. Guided by a system prompt and structured output schemas, the agent calls editing tools to add or remove object-property triples, assign classes to entities, and manage datatype literals. Tool arguments are restricted to valid TBox classes and properties, preventing schema hallucination (Figure 1).

To enforce strict structural conformance, the agent can call a ValidateSHACL tool at any iteration to check the intermediate graph against closed-world SHACL shapes derived from the TBox. Because standard validation reports are verbose and offer little concrete repair guidance, they are not passed directly to the main agent; a secondary explanation LLM parses the output, adds context from the local TBox definitions, and turns it into short natural-language repair instructions. This keeps noise out of the agent's context window and points it toward the actual structural issues.

The construction loop ends via a Finish command, gated by three checks: ValidateSHACL has been called at least once, the most recent validation returned zero violations, and every extracted entity has been assigned an ontology class. The pure-LLM baseline uses the same pipeline without the ValidateSHACL tool and without these final checks.

2.2. Dataset Construction and Shape Generation

Since existing benchmarks lack the expressive constraints needed to stress-test logical conformance (Table 1), we built a custom dataset from the Family ontology [12]. Seed facts and labels about individual people and their family relationships were collected from Wikidata and paired with the corresponding

[1] https://github.com/kurolesica0211/shacl-in-the-loop
[2] https://github.com/kurolesica0211/shacl-in-the-loop/blob/main/docs/prompts.md

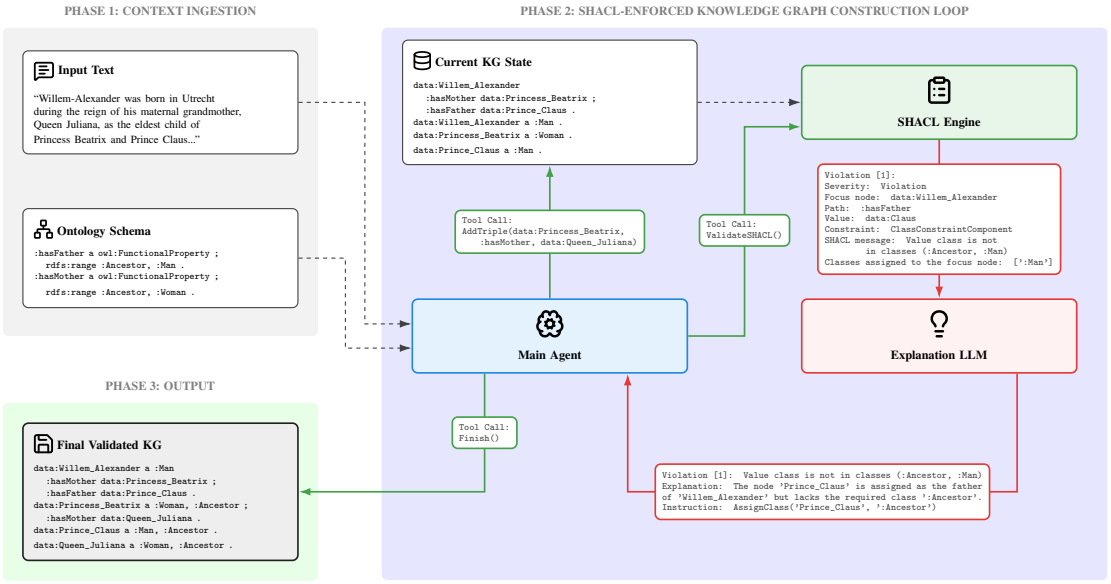


Figure 1: High-level overview of the SHACL-in-the-Loop construction pipeline.

biographical Wikipedia articles, which we denoised by fuzzy mention matching to keep only sentences describing the target people and their extended family networks. Running an OWL reasoner (`owlready2` [13]) over the localized graphs materializes all inferred kinship triples per family network, yielding 499 text–KG pairs with complex family ties. The full creation pipeline is documented in the repository[3].

The shapes document was generated in two stages. Standard constraints, property domains, ranges and cardinalities were derived from the ontology using the standard `owl2shacl`[4] conversion rules. Because real-world TBoxes use more expressive axioms that `owl2shacl` does not handle, a reusable script[5]automatically adds targeted constraints: whitelists of permitted properties for closed class shapes, and translations of property- and class-disjointness axioms into SHACL Core. We emphasize that the script can be reused across different ontologies. SHACL is not complete OWL reasoning, but it is a practical compromise: computationally lightweight, it gives the agent actionable signals for a broad class of structural errors, rather than an opaque "logically inconsistent" flag.

We group the resulting shapes into five families: *typing* (the value of `hasFather` must be a `:Man`), *cardinality* (nobody has two fathers), *class disjointness*, *property disjointness*, and *closed shapes* (a `:Man` node may not use `isSisterOf`).

2.3. Evaluation

We evaluate generated KGs along three dimensions: logical consistency, triple-level extraction quality, and operational performance. Consistency of final graphs is checked with an OWL reasoner.

For extraction quality (micro- and macro-averaged Precision, Recall, F_1) we compare generated and gold graphs semantically, materializing all OWL inferences in both before scoring. Because LLM outputs may contain inconsistencies that break standard materialization, we use brave semantics [14]: for inconsistent graphs we isolate the minimal inconsistent subsets, remove each statement in turn to materialize the inferences of the resulting consistent subgraphs, and take the union of all inferred triples. Extracted entities are then linked to gold entities via fuzzy label matching; triples containing unmatchable entities or relations absent from the gold graph are excluded from scoring, so the model is

[3]https://github.com/kurolesica0211/shacl-in-the-loop/blob/main/docs/dataset_collection.md
[4]https://github.com/sparna-git/owl2shacl
[5]https://github.com/kurolesica0211/shacl-in-the-loop/blob/main/experiments/shacl_inject_constraints.py

Table 2
Logical inconsistencies and triple-level extraction quality (Precision, Recall, F_1).

Condition	Inconsistent ABoxes	Micro-Averaged			Macro-Averaged		
		Precision	Recall	F_1	Precision	Recall	F_1
Baseline	24	0.9057	0.8551	0.8797	0.9003	0.8756	0.8833
SHACL-backed	5	0.9032	0.8610	0.8816	0.8936	0.8779	0.8814

not penalized for facts outside the localized target network.

Finally, we log total token usage, processing time, and the number of main-agent action steps (capped at 20 tool-use iterations per document; no run reached this cap).

2.4. Supplementary Experiments

Two further experiments support the analyses in Section 3. To characterize what the shapes detect, we ran the pure-LLM baseline five times over the dataset and validated every resulting graph twice, once as asserted and once after OWL materialization. To isolate the repair effect of the loop, we passed 146 finished baseline graphs, 97 logically inconsistent ones and 49 consistent controls, back into the SHACL-backed pipeline for up to 20 further iterations against the full shapes document, then re-materialized and re-scored them as above. Since every repair starts from an already finalized graph, this measures the change the loop makes to that graph rather than comparing two separately generated extractions, and the extra passes are not budget denied to the baseline, which is itself agentic and stopped because it judged the graph complete.

3. Results and Discussion

This section reports baseline and SHACL-backed results, following our two research questions on quality (RQ1) and cost (RQ2).

3.1. Logical Consistency

As Table 2 shows, the pure-LLM baseline produced 24 logically inconsistent Assertion Boxes (ABoxes), mostly basic class-disjointness and domain/range errors such as assigning female entities to male-restricted relations (`isBrotherOf`, `hasFather`), or making one person both parent and child of another. The SHACL-backed run eliminated these direct structural errors, cutting inconsistent graphs from 24 to 5.

All five residual graphs reached full SHACL conformance, so the loop did not fail to converge on them. The reasoner's minimal justifications trace every case to the same chain: a single wrong edge makes two people share a parent, most often a woman extracted as the mother of both a person and that person's own child when she is really the spouse; a property chain then derives `isSiblingOf` from the shared parent; a rule specializes it to `hasBrother` or `hasSister`; and the derived sibling relation collides with a correctly extracted parent-child edge through a disjointness axiom. Our validator evaluates neither property chains nor rules, so the derived sibling fact is never written down and the disjointness shape that would catch it has nothing to check. This is an expressivity gap between shape-based validation and OWL-DL reasoning rather than a gap in the coverage of our shapes.

3.2. What the Shapes Reach

Table 3 reports how often each shape family fires across the five baseline runs, on the asserted graph and after OWL materialization. The two halves diverge in opposite directions.

Typing dominates the asserted graph with 2,382 violations across 498 of the 499 documents, yet only 31 survive materialization. Almost every typing violation asks for a class that the TBox already entails from the assertion that triggered it: the range of `hasFather` derives `:Man` for its value, so an untyped

Table 3
SHACL violations across five pure-LLM baseline runs, before and after OWL materialization.

Shape family	Asserted Viol.	Asserted Docs	Materialized Viol.	Materialized Docs
Typing	2,382	498	31	29
Cardinality	80	59	116	79
Class disjointness	0	0	60	46
Property disjointness	3	3	32	29
Closed shapes	24	19	61	47

Table 4
Targeted repair experiment: extraction quality before and after re-entering the SHACL loop, for baseline graphs selected on logical inconsistency and for consistent controls.

Group		Micro-Averaged P	Micro-Averaged R	Micro-Averaged F_1	Macro-Averaged P	Macro-Averaged R	Macro-Averaged F_1
Inconsistent	before	0.7893	0.8134	0.8011	0.8335	0.8451	0.8321
($n = 97$)	after	0.8725	0.8323	0.8519	0.8748	0.8591	0.8618
Control	before	0.9240	0.8789	0.9009	0.8657	0.8543	0.8561
($n = 49$)	after	0.9037	0.9111	0.9073	0.8430	0.8660	0.8499

value node is not missing anything once inferences are applied. Since we materialize before scoring, repairing such a violation cannot improve the scored graph, and can only hurt it if the agent satisfies the constraint by deleting the triple instead of adding the type. This is where the loop spends most of its iterations.

Class and property disjointness behave in reverse, affecting 46 and 29 documents after materialization but almost none before. The shapes detect these contradictions reliably when present; the difficulty is that the contradiction is usually implicit. A node is rarely asserted to be both a `:Man` and a `:Woman`, and becomes both only once the ranges of `hasFather` and `hasMother` are applied, which SHACL does not do.[6] Cardinality and closed shapes work as intended, being reachable before materialization and growing only moderately after it.

Part of this reachability gap can be closed by writing shapes for the error rather than transliterating the axiom: "the value of `hasFather` must not be a `:Woman`" fires only on real conflicts, whereas "must be a `:Man`" also fires on thousands of merely untyped nodes. Our closed shapes already work this way, and their gender-restricted whitelists give the sharpest diagnosis in the run: the model chooses between `isBrotherOf` and `isSisterOf` by the gender of the other person rather than its own, as if they were their inverses `hasBrother` and `hasSister`.

3.3. Triple-Level Extraction Quality

We expected the loop to also improve extraction quality. As Table 2 shows, aggregate quality instead remained stable across both conditions, with micro and macro F_1 around 0.88.

Two things explain this. Logical errors are rare in these biographical texts, affecting roughly 20 of 499 documents per run, so even a large improvement on the affected graphs is diluted at corpus scale. The coverage analysis above supplies the second reason: the violations the loop can act on before materialization mostly carry no extraction signal, while the ones that do are mostly visible only after it.

The repair experiment (Section 2.4, Table 4) tests the dilution explanation by focusing on inconsistent graphs, whose rarity in the original runs can mask substantial repair gains: on 97 inconsistent graphs, the loop raised micro precision from 0.789 to 0.873 and micro F_1 from 0.801 to 0.852, resolving 87 inconsistencies. On the 49 consistent controls, both F_1 scores changed little, but micro and macro precision fell by 2.0 and 2.3 percentage points, while recall rose by 3.2 and 1.2 points, respectively. This

[6] Class disjointness does fire in the SHACL-backed run, from two situations: the model occasionally treats the sex class `:Male` and the person class `:Man` as synonyms, and it sometimes silences a typing violation by adding the demanded class to a node that already carries the opposite one.

trade-off arose from 180 added triples across the controls, only 100 of which were scored as correct. Inspection traced the remaining 80 to entity-matching and ground-truth errors rather than genuine semantic mistakes by the LLM, yet these measurement errors were sufficient to lower precision. This does not undermine the targeted repair result: our analysis attributes the gains on inconsistent graphs solely to the removal of 743 incorrect triples, supporting SHACL-backed correction rather than renewed extraction as the source of improvement.

3.4. Computational Cost

Structural conformance comes at a measurable cost. The validation-and-repair loop raised the median number of agent iterations from 1 to 5, mean processing time per document from 11.75 to 26.05 seconds, and mean total token usage from 16,855 to 79,684 tokens per task; neither condition approached the 20-iteration cap, peaking at 15 and 17 respectively. The initial document and ontology prompt is comparable in the two runs, at 12,368 and 12,500 tokens, so the entire overhead falls on two components that are negligible or absent in the baseline: the iterative conversation history, which grows from 474 to 17,031 input tokens, and the explanation LLM, which adds 15,653 tokens where the baseline uses none. Prompt caching, increasingly supported by LLM APIs, would avoid recomputing the static context each iteration. Reports also repeatedly flag the same defect: the 8,404 validation results from first validation calls correspond to only 2,273 distinct defects. Deduplicating violations before translation and having the explanation LLM summarize each whole report rather than issue one instruction per violation could reduce both explanation tokens and conversation history.

4. Conclusion and Future Work

On our custom Family-ontology dataset, SHACL-in-the-Loop worked as an effective localized filter: it eliminated the direct class-disjointness and domain/range violations and cut logically inconsistent ABoxes from 24 to 5, without shifting aggregate extraction quality. A targeted repair experiment localizes that quality effect to the graphs that actually contain logical errors, while affecting the control graphs notably less.

The negative findings are equally informative. The five residual inconsistencies all reach full SHACL conformance, because their contradictions live in property-chain and rule entailments that a shape validator does not compute, which marks a genuine expressivity boundary. The flat aggregate result follows from error rarity and from the fact that the violations reachable before materialization mostly carry no extraction signal. Together with the finding that roughly three quarters of raw violation reports restate an already-reported defect, this suggests that the largest gains available to shape-based KGC pipelines lie in how the shapes are written and where validation is applied, rather than in running more iterations.

Future work should first address the limitations of the current experiments: inaccuracies in matching extracted entities to ground truth, errors in the ground-truth graphs, and weaknesses in SHACL shape construction. Fairly assessing the added value of SHACL-backed repair also requires comparisons with prompt-based conformance instructions, fine-tuning, and self-critique. A dedicated KGC benchmark with expressive ontologies remains a pressing need for systematically evaluating these approaches under complex ontological constraints.

Acknowledgments

The first author thanks Remzi Celebi and Rishabh Jakhar for their mentorship during the bachelor thesis from which this paper originated, and for discussions that taught him what it means to think like a scientist.

Declaration on Generative AI

During the preparation of this work, the authors used Gemini, OpenAI Prism in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the authors reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] Z. Zhu, T. Huang, K. Wang, J. Ye, X. Chen, S. Luo, Graph-Based Approaches and Functionalities in Retrieval-Augmented Generation: A Comprehensive Survey, ACM Computing Surveys 58 (2026) 261:1–261:38. URL: https://dl.acm.org/doi/10.1145/3795880. doi:10.1145/3795880.

[2] B. Abu-Salih, Domain-specific knowledge graphs: A survey, Journal of Network and Computer Applications 185 (2021) 103076. URL: https://www.sciencedirect.com/science/article/pii/S1084804521000990. doi:10.1016/j.jnca.2021.103076.

[3] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying Large Language Models and Knowledge Graphs: A Roadmap, IEEE Transactions on Knowledge and Data Engineering 36 (2024) 3580–3599. URL: https://ieeexplore.ieee.org/abstract/document/10387715. doi:10.1109/TKDE.2024.3352100.

[4] S. Khorshidi, A. Nikfarjam, S. Shankar, Y. Sang, Y. Govind, H. Jang, A. Kasgari, A. McClimans, M. Soliman, V. Konda, A. Fakhry, X. Qi, ODKE+: Ontology-Guided Open-Domain Knowledge Extraction with LLMs, 2025. URL: http://arxiv.org/abs/2509.04696. doi:10.48550/arXiv.2509.04696, arXiv:2509.04696 [cs].

[5] X. Feng, X. Wu, H. Meng, Ontology-grounded Automatic Knowledge Graph Construction by LLM under Wikidata schema, CEUR Workshop Proceedings 3841 (2024) 117–135. URL: https://www.scopus.com/pages/publications/85210894055.

[6] B. Zhang, H. Soh, Extract, Define, Canonicalize: An LLM-based Framework for Knowledge Graph Construction, 2024. URL: http://arxiv.org/abs/2404.03868. doi:10.48550/arXiv.2404.03868, arXiv:2404.03868 [cs].

[7] C. Ringwald, F. Gandon, C. Faron, F. Michel, H. A. Akl, Kastor: Fine-tuned Small Language Models for Shape-based Active Relation Extraction, 2025. URL: https://hal.science/hal-05078493. doi:10.1007/978-3-031-94575-5_6.

[8] D. Wang, M. Iwaihara, OSKGC: A benchmark for Ontology Schema-based Knowledge Graph Construction from Text, in: S. Razniewski, J.-C. Kalo, D. Islakoğlu, T.-P. Nguyen, B. Zhang (Eds.), Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction (KBC-LM+LM-KBC 2025), volume 4041 of *CEUR Workshop Proceedings*, CEUR, Nara, Japan, 2025. URL: https://ceur-ws.org/Vol-4041/#paper1.

[9] N. Mihindukulasooriya, S. Tiwari, C. F. Enguix, K. Lata, Text2KGBench: A Benchmark for Ontology-Driven Knowledge Graph Generation from Text, 2023. URL: https://zenodo.org/records/7916716. doi:10.5281/zenodo.7916716.

[10] J. Plu, O. M. Escobar, E. Trouillez, A. Gapin, P. Lisena, T. Ehrhart, R. Troncy, Text2KGBench-LettrIA: A Refined Benchmark for Text2Graph Systems, in: S. Razniewski, J.-C. Kalo, D. Islakoğlu, T.-P. Nguyen, B. Zhang (Eds.), Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction (KBC-LM+LM-KBC 2025), volume 4041 of *CEUR Workshop Proceedings*, CEUR, Nara, Japan, 2025. URL: https://ceur-ws.org/Vol-4041/#paper3.

[11] R. Power, A. Third, Expressing OWL axioms by English sentences: dubious in theory, feasible in practice, in: C.-R. Huang, D. Jurafsky (Eds.), Coling 2010: Posters, Coling 2010 Organizing Committee, Beijing, China, 2010, pp. 1006–1013. URL: https://aclanthology.org/C10-2116/.

[12] R. Stevens, M. Stevens, A Family History Knowledge Base Using OWL 2, 2008. URL: https://www.

semanticscholar.org/paper/A-Family-History-Knowledge-Base-Using-OWL-2-Stevens-Stevens/37b1663e97e3aeaba23795003fc6671f6193d463.

[13] J.-B. Lamy, Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies, Artificial Intelligence in Medicine 80 (2017) 11–28. URL: https://www.sciencedirect.com/science/article/pii/S0933365717300271. doi:10.1016/j.artmed.2017.07.002.

[14] A. Nentidis, C. Akasiadis, A. Charalambidis, A. Artikis, Dealing with Inconsistency for Reasoning over Knowledge Graphs: A Survey, 2025. doi:10.48550/arXiv.2502.19023.