A Many-Sorted Intuitionistic Logic for Asserting and Querying Ontologies

Federico Donato

State University of New York at Buffalo, 197 Founders Promenade, Buffalo, NY 14260-4150, USA

Abstract

The present PhD project aims to create a new logical language designed for both defining and querying ontologies, in competition with the current standards represented by OWL and SPARQL. The underlying idea is to augment the expressivity now present in OWL, while keeping the new language decidable and open-world by deploying a tailored version of many-sorted intuitionistic logic interpreted via a special form of Kripke Frame semantics. This approach will also allow us to harness the power of a semantic type-checker in parallel to the traditional logical reasoner. Proofs of the theorems concerning this new logic will constitute the main effort of the project. Lastly, we will discuss a possible implementation of the new logic in an extended Datalog.

Keywords

intuitionistic logic, type theory, OWL, SPARQL, Datalog

1. Motivation

The dominant ontology languages in the last decade have been the Ontology Web Language (OWL) [1] and the SPARQL Protocol and RDF Query Language (SPARQL) [2]. Both were published by the World Wide Web Consortium (W3C) and were designed to meet the needs of the Semantic Web community in its early stages. OWL was a major breakthrough in knowledge representation because it brought the rigorous foundations of Description Logics into the RDF language and enabled interoperable ontology engineering on the Web while retaining decidable reasoning, making tasks such as classification, consistency checking, and subsumption practically feasible at large scale. SPARQL provided the Semantic Web with a standard query language analogous to SQL for relational databases. It is a declarative language that does not itself provide derivation rules. It enabled uniform access to RDF graphs across heterogeneous sources and became the cornerstone of practical Semantic Web applications and linked data infrastructures. These two languages have been successful in their tasks and are used both by the academic community and industry, having their own code editors and engines.

On the other hand, there are limitations in expressivity, semantics and implementation choices with both OWL and SPARQL. Can these limitations be overcome by a new logic for ontologies and knowledge graphs, which would inherit the successful features of both? In recent years, researchers have attempted to extend or modify certain features of OWL and SPARQL. Our project builds on this literature while also seeking to reconsider the foundations of these standards [3] [4].

We can consider more precisely here the features of OWL and SPARQL that motivate us to depart from them. In the following, we will understand "ontology" as the general rules of a certain domain of study, and "knowledge graph" as the instance data that can exemplify those rules.

A first issue of OWL and SPARQL is that there is a lack of precision in certain query results that would be desirable. For example, given an agricultural goods knowledge graph and its ontology, if we asked whether this apple was an asparagus, we would easily get a FALSE result by traversing the graph. Instead, if we asked if the same apple was planned after harvesting, we would still get a FALSE result from SPARQL, knowing that, according to our ontology, only events are scheduled at certain times, and apples cannot possibly be planned. This second answer, unfortunately, should not merely

Proceedings of FOIS 2026 Satellite events co-located with the 16th International Conference on Formal Ontology in Information Systems (FOIS 2026), September 21–25, 2026, Vitória, Brazil

fdonato@buffalo.edu (F. Donato)

0009-0001-6600-240X (F. Donato)

© 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 2026-11-27

CEUR-WS.org/Vol-4342/paper14.pdf

be classified as FALSE; it should not be assigned a truth value at all. To be precise, the second query was ontologically malformed, since the ex:planned relation cannot have apples in the domain, and it would be a waste of computational resources to do pattern matching on the graph and figure it out. Additionally, the user who is not familiar with all the nuances of the ontology at hand, cannot be informed by the query results that only the first query was properly false, and the second semantically malformed and therefore neither true nor false. SPARQL has already something similar, which is the ERROR result for syntactically malformed queries, or those that make forbidden datatypes operations, like asking which OWL class is heavier than 100kg. This is still not enough, because it does not consider ontological rules, and would not distinguish between FALSE and semantically MALFORMED. To obtain this level of precision in query answers, we may need an ontological checker that validates a query before it is executed, but there is nothing of this kind that OWL or SPARQL can presently offer.

A second issue to consider pertains again to a mismatch between the answers SPARQL can give and what OWL can actually store in its semantics. The query layer does not clearly distinguish between FALSE and the absence of an answer to a legitimate query. Consider an agricultural ontology that asserts in its axioms that all apples must have a color, and that in the current knowledge graph we never said that this apple has a specific color. Now, we can run two queries, the former asking if this apple has some unspecified color, the latter asking if this apple has the color ex:yellow33. SPARQL will return to both queries FALSE as an answer, while at least to the former OWL would have entailed that there is some unknown color that this apple must have. Fortunately, SPARQL can be enhanced with the OWL entailment regimes [5], which allows us to make some inferences given the ontological rules, and the first query would return TRUE, even if the color remains unknown.

Still, something is missing, since nowhere in the graph we asserted that this apple does not have color ex:yellow33, and it might be possible that an update will exactly add that information. The second query should distinguish between what is currently held to be false, and what is still unknown, and this is another lack of precision in the query answering that not even the enhancement with OWL entailment regimes can muster.

A last issue is the fragmentation between OWL and SPARQL, therefore between asserting and querying. The two issues discussed above highlight aspects of this fragmentation, which affects both the proof-theoretic and semantic foundations of the two languages. First of all, SPARQL has no proof theory whatsoever, since it is not a logical calculus, and that is why some query answers are not as informative as the ontology rules would like to allow. Moreover, the type of semantics OWL and SPARQL have is of a totally different kind. The former has a Tarskian model-theoretic semantics that admits the Open-World Assumption (OWA), while the other has an operational semantics and exhibits behavior associated with the Closed-World Assumption (CWA), meaning that what is not asserted in the knowledge graph is treated as false rather than simply unknown, as in OWL.[1] It would be a design error to constrain OWL to the Closed-World Assumption, of course, since it is a language that must tackle the diversity typical of the web, with various uncontrolled sources of data and the ability to increase the knowledge in unexpected ways, breaking away from the overly restrictive schemas of relational databases. Some might say that this fragmentation of the semantics is a feature, and not a bug, and that historically it was also more practical to build the two languages this way, maybe leveraging already existing studies on SQL and transporting them into SPARQL, for example, without reinventing the wheel [6]. Another major motivation for making the query language less expressive than the language used for assertions was presumably computational complexity: as it is, and without entailment regimes, evaluating general SPARQL patterns is PSPACE-complete [7].

Still, lacking unity between asserting and querying might have unexpected consequences, and does not give us the credentials needed to embed ontologies into larger software applications. If we also consider SWRL and SHACL, commonly used ontology languages that add further capabilities to RDF and OWL, there is no known single semantics that takes all four languages into account, even though they

[1] Technically, SPARQL does not follow the OWA, since only logical languages can do so. SPARQL performs pattern matching on a graph without deriving consequences from formulas. Nevertheless, its answers exhibit behavior similar to that associated with the CWA.

are used in combination in practice [8] [9]. Proving properties without a common semantic foundation becomes difficult, which is undesirable in high-stakes application domains such as medicine and finance.

Some solutions to the issues listed above have already been explored in areas beyond applied ontology, including intuitionistic logic and type theory. Old and new ideas can inform a new logic that might address the issues with OWL and SPARQL identified above without discarding the achievements of these languages. Other research areas—such as type theory, database theory, and rule-based systems—provide well-developed mechanisms for negation, closed-world reasoning, and integrated query answering, suggesting that a unified logical framework could overcome the current fragmentation in ontology engineering. However, designing a suitable new language without implementing it would offer little practical benefit to end users, so an implementation must also be considered. This motivates us to consider not only a theoretical solution but also a practical language, preferably implemented by extending an existing programming language, so as to avoid building a new language and compiler from scratch.

2. Research Questions

1. Can we design a single language for assertion and querying of knowledge graph and ontologies?
2. Can an intuitionistic logic with Kripke Frame semantics help us capture the Open World Assumption better than the alternatives?
3. Can the same logic tackle the undecidability problems typical of its semantics?
4. Can type systems represent ontological taxonomies as well as set theory does in OWL?
5. Can type systems help us to distinguish between merely false and ontologically malformed expressions?
6. Can our new logic for ontologies with the features above be efficiently implemented in a computer language?

3. Research Objectives

This PhD project has several objectives, all of which contribute to the development of a new logical language for knowledge graphs and ontologies. The language is intended to adopt the open-world assumption fully and to provide more informative and precise query answers than current approaches. The features that we want this new logic to provide are the following: unity of semantics between the assertion and query languages; a distinction between ontologically malformed and false formulas; open-world semantics that is preserved during querying; and an implementation of the new logic as an extension of Datalog.

First of all, the unity can be achieved by design. The main idea is that querying and asserting can be treated within a single logical framework [10]. If we already have the syntax and the semantics of our logic, queries are simply either formulas with free variables for SELECT queries or existentially quantified formulas for ASK queries, that then might be grounded into affirmative answers if the knowledge graph allows it. Now, the question is whether we want achieve unity by a monolithic single language, with one proof theory for all use cases, or if we can be content with one common semantics between the assertion layer and the query layer, that might still have a few documented differences. Whatever the case, the system that we will build will gain more control for the software verification we might do with it, and it will be easier to prove theorems about it compared to having multiple semantics at play. To keep track of the design choices and desiderata behind OWL and SPARQL—which are possible, but not the only, ways of encoding those desiderata—we will create a synoptic table comparing their encodings in OWL and SPARQL with their encoding in our new logic. This will help us track our progress and identify what the new design achieves. Along the way, we might add a research objective of incorporating into our new logic selected features of SWRL and SHACL that OWL cannot easily express, particularly in light of what our chosen implementation language, Datalog, will allow us to do.

Second, regarding the more precise answers to queries, type theory can help us in distinguishing between FALSE and MALFORMED queries, especially in this form: many-sorted logic. A many-sorted logic is a predicate logic that adds static type judgements on the side of logical formulas, but which confines such judgements to terms only, so it is not a fully-typed language [11]. These type judgments are called "sort judgments". In this logic, the taxonomical structures typical of ontologies won't be bestowed by set-theory-like operations, like in OWL, but by sub-sorting operations [12]. This is not merely a new formulation of familiar concepts, since writing our ontological rules in this way can have two major upshots: a type-checker doing the taxonomical reasoning instead of the usual logical reasoner; a large pool of literature and theorems for present and future purposes. Imagine we ask again when a certain apple is planned within our agricultural goods knowledge graph: the query result would be MALFORMED instead of FALSE. Moreover, this check will be performed at compile time, as type-checkers do, instead of at run time, saving us computational resources and avoiding inspecting a graph when it is useless, gaining precision and speed. It is also important to notice that the literature on type theory in general, and on many- sorted logic in particular, far exceeds that of description logics, which are mainly deployed for ontologies, while theorems, techniques and well-maintained libraries can be more easily found and adapted from the first area of research. This aspect of type theory concerns the research ecosystem more than the theory itself, but it matters when making sure that our new logic can have a long-lasting future and have members that can maintain it up to date.

Third, if the assertion language will be truly open-world, so it will be the query language, for obvious reasons. OWL, in its classical proof theory, implements the Open World Assumption via classical Tarskian semantics, where an ontology admits multiple models and missing information is interpreted as underspecification rather than falsity [13]. A natural alternative would be intuitionistic logic rather than the classical logic underlying OWL. In intuitionistic logic, the law of excluded middle is not generally derivable, thus it is not true that all sentences are true or false, leaving space for sentences without a truth-value.

The chosen semantics for intuitionistic logic is Kripke-frame semantics. A model consists of multiple worlds related by accessibility relations; in our case, the Kripke worlds would be knowledge graphs composed of instance data [14]. However, under the traditional Kripkean interpretation of negation, evaluating negative formulas requires considering accessible worlds beyond the current world. This may render the resulting system undecidable. To address this issue and make negation more local to the world at hand, we aim to modify traditional Kripke semantics so that negative queries remain decidable by bounding the search to the current knowledge graph rather than all the worlds it can access. The idea is to represent negative relations explicitly within the model, alongside the positive relations used in traditional predicate logic. Members of the positive and negative relations cannot overlap in the model, but the union of a positive relation and its negated version do not necessarily recreate the universe of discourse, in compliance with the open-world assumption, since we might not know the full extent of our domain. This would also distinguish our special negative relations from OWL complementary classes, among other features. This changes the truth condition for negation: for example, if we know that this apple is not yellow, this information can be represented by a binary relation such as notColored in the model. Beyond this, Kripke semantics still leads to undecidability in the cases of implications and of universally quantified sentences, and further techniques must be found to deal with them. Interestingly, when considering universal sentences in an expanding model as the Kripke semantics allow, we will also tackle the issue of the monotonic versioning of ontologies, since not only instance data, but also rules can change.

Finally, after the design and theorem proving phases of this project have been successfully completed, we aim to implement our new logic in an effective computer language, so that it can be used in practice rather than remaining purely theoretical. The language of choice is Datalog. Although standard Datalog may not be sufficiently expressive for our purposes, it has a thriving developer community that has extended it in various ways while preserving efficient evaluation. These extensions are valued in industry as well as academia [15]. The new logic sketched above and its semantics will be encoded in such extension. Ordered-sorted Datalog can be considered as a way to embed our many-sorted logic, for example [16]. It is also compelling to us that Datalog works directly with Horn clauses, so it seems

practical to express some SWRL rules inaccessible to OWL, like rule-chains. Datalog with negation might also be considered as an implementation of the validation constraints typical of SHACL [17]. Besides this, we might also consider that Datalog is natively apt to query relational databases, that are far from being replaced by the RDF data model, and still require some ad hoc solution to combine the two, like the Ontology-Based Data Access (OBDA) [18]. Integration with relational databases is achieved via mediating frameworks such as Ontology-Based Data Access (OBDA), where SPARQL queries are rewritten into SQL through mappings. While effective in practice, this approach introduces an additional semantic layer and does not provide a fully unified logical framework across ontological and relational representations. The envisioned Datalog extension could therefore support queries over knowledge graphs while remaining compatible with relational databases, provided it is designed appropriately. This would be a major advantage in bridging the gap between these two data models.

4. Research Methodology

The research methodology begins with an in-depth literature review of OWL and SPARQL to establish the full capabilities of these languages and provide a benchmark against which to evaluate our new logic. The W3C documentation will also help us understand the motivations behind OWL and SPARQL, and whether these existing languages meet the goals that should be considered desiderata for our project. There is also literature on alternatives to OWL, such as RIF [19] and frame logic [20]. These were considered viable ontology languages before OWL became popular and may offer further insight into what users want from an ontology language.

The design phase consists of following the principles discussed in the research objectives, such as using a single language for assertion and querying, and formulating conjectures about the capabilities of the new logic and whether it is as expressive as the benchmark languages. Typical conjectures might concern the soundness of the new logic, whether its derivability relation is conservative with respect to OWL, or whether a particular aspect of the new logic is decidable.

The testing phase consists of attempting to prove our conjectures. Proofs by induction can be done by hand and documented with natural language annotations. A stronger approach would be to formalize the new logic in a proof assistant such as Rocq, enabling machine-checked proofs and providing assistance during proof development. This would strengthen the methodology beyond what unaided human verification can achieve.

Regarding the implementation in an extended Datalog, we would need to write a Datalog library in Rust or OCaml, and then do some unit testing for the correctness of the rules, as well as measuring the performances via regression testing and via benchmark suites.

Specifically, since we will add negation, it is relevant to test stratification, that the semantics is still well-founded and the behavior of the stable models.

5. Results to Date

The results to date, representing approximately five months of work, include the design of the signature and semantics of our logic for knowledge graphs, together with proofs of several theorems. In particular, this is the signature of the new logic from knowledge graphs:

$$\langle \Sigma, <:, F = \bigcup_{\vec{\sigma} \in \Sigma^*, \sigma \in \Sigma} F^{\vec{\sigma} \to \sigma}, R = \bigcup_{\vec{\sigma} \in \Sigma^*} R^{\vec{\sigma}} \rangle$$

where Σ^* is the set of finite lists of elements of Σ. Here, Σ is a set of sorts, F is a set of function symbols, and R is a set of relation symbols. The relation $<:$ between sorts is subsorting, which determines a partial order on sorts. A sort judgement in a sequent, which presents a sortal context but not a logical formula context, has this shape: $\Gamma \vdash t : \sigma$

This is the template for the models in our semantics:

$$\langle (U_\sigma^M)_{\sigma\in\Sigma}, (f^M)_{f\in F^{\vec{\sigma}\to\sigma}}, (r^M)_{r\in R^{\vec{\sigma}}}, (\check{r}^M)_{r\in R^{\vec{\sigma}}}\rangle$$

U_σ^M is the universe of sort σ

$U^M = \bigcup_{\sigma\in\Sigma} U_\sigma^M$ is the universe of discourse

f^M is the interpretation of the function symbol f. We require $f^M : U_{\sigma_1}^M \times \cdots \times U_{\sigma_n}^M \to U_\sigma^M$ whenever $f \in F^{\sigma_1,\ldots,\sigma_n\to\sigma}$

r^M is the interpretation of the relation symbol r. We require $r^M \subseteq U_{\sigma_1}^M \times \cdots \times U_{\sigma_n}^M$ whenever $r \in R^{\sigma_1,\ldots,\sigma_n}$. If S is a multi-sorted first-order signature with subsorting, then for every $\sigma, \sigma' \in \Sigma$ such that $\sigma <: \sigma'$, we have $U_\sigma^M \subseteq U_{\sigma'}^M$.

$\check{r}^M$ is the interpretation of the negated relation symbol r. Similarly, we require $\check{r}^M \subseteq U_{\sigma_1}^M \times \cdots \times U_{\sigma_n}^M$ whenever $\check{r} \in R^{\sigma_1,\ldots,\sigma_n}$.

So far, we have proven various theorems, and I confine myself to mention some for lack of space: all true sentences are well-sorted; entailment in a knowledge graph without our novel negated relation implies entailment in a knowledge graph with negated relations; entailment in a knowledge graph with our novel negated relation implies entailment in the closed-world version of our knowledge graph without the negated relation; and partial soundness of our calculus. Regarding this latter result, the calculus is sound except in the case of implication formulas. We are currently investigating the reason for this limitation.

Regarding the sortal system, so far, we just have given a glimpse of the taxonomical structure, since we focused more on instance data than types. In fact, our sortal judgements allow sub-sorting, but not all the other operations typical of set-theory yet, even if there are easy ways to overcome this with more ideas from type theory.

6. Acknowledgments

This work would not have reached this point without the fundamental intuitions and guidance of my supervisor, Dr. Andrew Hirsch, and the collaboration of Halle Bryant, with whom I proved various theorems and discussed the theory. I also thank Dr. John Beverley for additional comments.

Declaration on Generative AI

During the preparation of this work, the author used ChatGPT and Claude to help ensure that the paper adhered to the formatting guidelines required for the proceedings. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the published work.

References

[1] F. Baader, I. Horrocks, U. Sattler, Description Logics as Ontology Languages for the Semantic Web, in: Mechanizing Mathematical Reasoning, 2005.

[2] B. DuCharme, Learning SPARQL: Querying and Updating with SPARQL 1.1, "O'Reilly Media, Inc.", 2013.

[3] S. Borgo, D. Porello, N. Troquard, Logical Operators for Ontological Modeling, Formal Ontology in Information Systems (2014).

[4] V. Doing, R. Wisnesky, Towards A More Reasonable Semantic Web, 2024. doi:10.48550/ARXIV.2407.19095.

[5] R. Kontchakov, M. Rezk, M. Rodriguez-Muro, G. Xiao, M. Zakharyaschev, Answering sparql queries over databases under owl 2 ql entailment regime, in: International Workshop on the Semantic Web, 2014. URL: https://api.semanticscholar.org/CorpusID:15640998.

[6] I. Kollia, B. Glimm, I. Horrocks, SPARQL Query Answering over OWL Ontologies, in: G. Antoniou, M. Grobelnik, E. Simperl, B. Parsia, D. Plexousakis, P. De Leenheer, J. Pan (Eds.), The Semantic

Web: Research and Applications, volume 6643, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 382–396. doi:10.1007/978-3-642-21034-1_26.
[7] J. Pérez, M. Arenas, C. Gutierrez, Semantics and complexity of SPARQL, ACM Transactions on Database Systems 34 (2009) 1–45. doi:10.1145/1567274.1567278.
[8] B. Bogaerts, M. Jakubowski, J. Van Den Bussche, SHACL: A Description Logic in Disguise, in: G. Gottlob, D. Inclezan, M. Maratea (Eds.), Logic Programming and Nonmonotonic Reasoning, volume 13416, Springer International Publishing, Cham, 2022, pp. 75–88. doi:10.1007/978-3-031-15707-3_7.
[9] Q. Wang, X. Yu, Reasoning over OWL/SWRL Ontologies under CWA and UNA for Industrial Applications, in: D. Wang, M. Reynolds (Eds.), AI 2011: Advances in Artificial Intelligence, Springer, Berlin, Heidelberg, 2011, pp. 789–798. doi:10.1007/978-3-642-25832-9_80.
[10] G. Cima, G. D. Giacomo, M. Lenzerini, A. Poggi, Querying OWL 2 QL ontologies under the SPARQL Metamodeling Semantics Entailment Regime, in: Sistemi Evoluti per Basi Di Dati, 2017.
[11] M. Manzano, V. Aranda, Many-Sorted Logic, The Stanford Encyclopedia of Philosophy (2022).
[12] M. Cerioli, A. Haxthausen, B. Krieg-Brückner, T. Mossakowski, Permissive subsorted partial logic in CASL, in: M. Johnson (Ed.), Algebraic Methodology and Software Technology, Springer, Berlin, Heidelberg, 1997, pp. 91–107. doi:10.1007/BFb0000465.
[13] I. Horrocks, B. Parsia, U. Sattler, OWL 2 Web Ontology Language Direct Semantics, 2009.
[14] J. Moschovakis, Intuitionistic Logic, The Stanford Encyclopedia of Philosophy (2024).
[15] M. Krötzsch, Modern Datalog: Concepts, Methods, Applications (Invited Paper), in: A. Artale, M. Bienvenu, Y. I. García, F. Murlak (Eds.), OASIcs, Volume 138, RW 2024 & RW 2025, volume 138, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, pp. 7:1–7:41. doi:10.4230/OASICS.RW.2024/2025.7.
[16] J. A. Goguen, J. Meseguer, Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations, Theoretical Computer Science 105 (1992) 217–273. doi:10.1016/0304-3975(92)90302-V.
[17] P. Pareti, G. Konstantinidis, T. J. Norman, M. Şensoy, SHACL Constraints with Inference Rules, in: C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, F. Gandon (Eds.), The Semantic Web – ISWC 2019, Springer International Publishing, Cham, 2019, pp. 539–557. doi:10.1007/978-3-030-30793-6_31.
[18] D. Baura, D. Calvanese, Assessing Privacy Requirements for Controlled Query Evaluation in OBDA, in: V. Torra, Y. Narukawa, J. Domingo-Ferrer (Eds.), Modeling Decisions for Artificial Intelligence, Springer Nature Switzerland, Cham, 2026, pp. 183–197. doi:10.1007/978-3-032-00891-6_15.
[19] M. Kifer, Rule Interchange Format: The Framework, in: N. Bassiliades, G. Governatori, A. Paschke (Eds.), Rule Representation, Interchange and Reasoning on the Web, volume 5321, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 1–2. doi:10.1007/978-3-540-88808-6_1.
[20] J. Angele, M. Kifer, G. Lausen, Ontologies in F-Logic, in: S. Staab, R. Studer (Eds.), Handbook on Ontologies, Springer, Berlin, Heidelberg, 2009, pp. 45–70. doi:10.1007/978-3-540-92673-3_2.