RDF Surfaces: Enabling Classical Negation and First-Order Expressivity on the Semantic Web

Patrick Hochstenbach [a,b,*], Mathijs van Noort [b], Rebekka Martens [c], Jos De Roo [b], Ruben Verborgh [b], Pieter Bonte [b,d] and Femke Ongenae [b]

[a] *Ghent University Library, Ghent University, Belgium*
E-mail: patrick.hochstenbach@ugent.be
[b] *IDLab, Ghent University - imec, Belgium*
E-mails: mathijs.vannoort@ugent.be, jos.deroo@ugent.be, ruben.verborgh@ugent.be, femke.ongenae@ugent.be
[c] *International Center for Computational Logic, Technische Universität Dresden, Germany*
E-mail: rebekka.martens@mailbox.tu-dresden.de
[d] *Department of Computer Science, KU Leuven Campus Kulak, Belgium*
E-mail: pieter.bonte@kuleuven.be

Abstract.
The Resource Description Framework (RDF) is a fundamental technology in the Semantic Web, enabling the representation and interchange of structured data. However, RDF lacks the capability to express negated statements in a generic way. As a result, exchanging negative information on a Web scale is thus far restricted to specific cases and predefined statements. The ability to negate (virtually) any RDF statement allows for a comprehensive way to refute, deny or otherwise invalidate claims on a Web scale. We present RDF Surfaces, an extension of RDF that integrates classic negation, thereby enabling first-order logic expressivity within RDF. RDF Surfaces offers a more general and widely applicable approach than previous attempts at incorporating negation. For RDF Surfaces, we define both an abstract and concrete syntax, along with the formal semantics for first-order logic. We demonstrate practical applications of RDF Surfaces with two use cases drawn from the domains of academic publishing and eHealth. We hope this vision paper attracts new implementers and opens the discussion for its specification as a W3C Community Group report.

Keywords: RDF Logic, Classical negation, BLOGIC, Existential graphs

1. Introduction

Reasoning with classical negation has attracted interest in the Semantic Web since its early days. Classical negation has properties similar to processing boolean values with the `NOT` operator in computer languages. Classical negation can take "true" to "false" and vice versa. It follows the law of the excluded middle: any statement P is either "true" or "false". If some P is "true", then a double negated P is also

*Corresponding author. E-mail: patrick.hochstenbach@ugent.be.

"true" (and vice versa for "false"). This form of negation is also explicit: "false" is not a default value; it cannot be assumed; it should be stated. All other forms of negation that do not follow these principles – there are many variants – are "weaker" forms of negation. The interest in "strong" classical negation can be attributed to a desire to fulfill one of its core design principles. Just as the Web is open-ended, allowing anyone to create a webpage on a server and link to any other webpage, the Semantic Web aspires to enable anyone to express any statement about any topic [38]. The Resource Description Framework (RDF) [41] is the W3C recommendation that defines the language to express statements about anything in the universe in the form of RDF triples. Using RDF, not only Web resources can be described, but physical objects, abstract concepts, and, in general, anything that can be given an identifier. However, there are also limitations. RDF lacks the ability to express classical "strong" negation, explicitly stating negative information that also follows the law of the excluded middle and all other classical negation properties. Additionally, RDF can express existential quantification (using blank nodes) , but lacks the expressivity to express universal quantification. As we see below, there are reasons to desire such features, but there are also some compelling reasons why classical negation has mostly been avoided.

1.1. Why is classical negation and universal quantification desirable?

Negation and universal quantification are applicable across various use cases. Straccia and Casini [51] describe how, in medicine, it is important to distinguish between the absence of biochemical reactions between substances and not knowing about their existence, which results in a need for explicitly stating negative information. Wagner [58] makes the case for a generalized Web logic based on classic negation and quantification to drive business processes. In an earlier paper, Wagner [57] demonstrated that two types of negation are necessary to interpret data effectively. The monotonic strong (classical) negation "Patient X did not take pill-B" expresses negative knowledge. The non-monotonic weak negation "Patient X is not registered at a hospital" expresses a negation as failure (NAF). Esteves [17] and Kebede [32] advocate for the use of negation in making policy information on permissions, prohibitions, and obligations enforceable through the Open Digital Rights Language (ODRL) [30]. They also suggest that negation can facilitate conflict resolution while creating policy documents.

In addition to practical reasons related to solving real-world use cases, there are also technical reasons why a classical negation and, in extension, first-order logic (FOL) have desirable properties.

A Web logic without negation is monotonic, a Web logic extended with classical negation preserves monotonicity. Adding classic negated RDF triples to a consistent knowledge base can not invalidate older conclusions made about that knowledge base. Making a Web logic non-monotonic is not achieved by simply adding a negation, but rather by incorporating NAF-style negation. Hayes (2001) calls this latter form of reasoning "unsafe" on the open Web, comparing it to the act of jumping to conclusions. Web reasoning is inherently open-ended, i.e., one can never assume that all the facts about a topic are available. By consulting additional resources, new information might arise. Non-monotonic reasoning, such as NAF, assumes that information about any topic is complete. Missing information is assumed to be "false". For instance, from a missing statement "Patient X has fever" in a knowledge base, it is unsafe to infer that "Patient X has fever" is "false" and make other conclusions based on this assumption. The implication "If it is NAF(not) the case that Patient X has fever, then provide pill-B" would entail "provide pill-B" by an NAF style of negation. This is not the case for classical negation, which is explicit in the triples that should be considered "false". The implication can only entail "provide pill-B" when information about the explicit negation of the statement "Patient X has fever" is available. This, of course, assumes that the knowledge base is consistent from the outset. The knowledge base must neither contain nor entail statements and their negations, as such contradictions would lead to *ex falso quodlibet*.[1]

There are also grounds to believe that contradictions on the Semantic Web are inherent, not detrimental as suggested by our previous point, but rather benign. Hayes refers to this as the 'diamond of confusion' [22]:

[1] The principle of explosion according to which any statement can be proven from a contradiction.

we may agree on a shared reality (whether it be absolute or relative truth) and a common method to create statements and express logical reasoning about this reality, yet still end up with contradictory conceptualizations of this reality. Hayes provides an example of how concepts can be conceptualized with and without a temporal dimension, which leads to contradictory results. For instance, a patient can be a fixed "thing" in one formalization with a name, address, and social security number; the same patient can be an "event in time" in another formalization where the patient with a fever is not the same patient without fever after giving a medication (a "thing" cannot both have a fever and not a fever). Understanding and identifying these contradictions among RDF resources is crucial for interpreting resources in querying and reasoning scenarios.

In some way, negation is already implicitly available in the Semantic Web when asserting triples. In logic, there are no alternative versions of "true". If, within a context, some RDF triple is regarded as being "true", this means that the negation of that triple is "false" in that context (regardless of the absolute truth or availability of other RDF triples on the open Web). Or, stated differently, within a context, any assertion negates the negated triple because $P \equiv \neg\neg P$.

1.2. Why was classical negation not added to RDF?

RDF natively supports conjunction ($\wedge$ or 'AND') by putting together two triples in one graph, as well as existential quantification ($\exists$) via blank nodes. Adding full negation $\neg$ (over arbitrary graphs, not only single triples) would give RDF access to disjunction $\vee$ and universal quantification $\forall$: $A \vee B$ is equivalent to $\neg(\neg A \wedge \neg B)$ and $\forall x : A$ is equivalent to $\neg(\exists x : \neg A)$. Since $\{\neg, \wedge, \exists\}$ is functionally complete and $\forall$ is definable from $\neg$ and $\exists$, the result has all the connectives and quantifiers of first-order logic, giving it the expressive power of a function-free FOL over a triple signature.

Considering all the reasons considered above, why can't we incorporate this form of classic negation into RDF?

While monotonicity was appreciated for not depending on the absence of information, the full expressivity of monotonic logic in the form of FOL was not. The designers of RDF deliberately chose to exclude these features, citing Lassila [40], who expressed concern that such complex features "might discourage the acceptance and adoption of RDF within the Web community."

A more compelling argument is that FOL itself, which is what RDF would become once full negation is added, has an undecidable entailment problem, proven independently by Church and Turing in 1936: given a set of FOL formulae $\boldsymbol{F}$ and an arbitrary formula f, is f a logical consequence of $\boldsymbol{F}$?For machines, undecidability results in a fundamental computability problem. It is impossible to simultaneously demand (1) an input language with the full reasoning capabilities of FOL, (2) a (finite) algorithm capable of solving any entailment problem, and (3) the execution of this algorithm in a finite number of steps. At most, two of these features can be achieved [47]. Description logics, as used in the Web Ontology Language (OWL2 DL), provides features (2) and (3), but requires compromises for (1).

1.3. Rationale for proposing the addition of classical negation and FOL expressivity to RDF

Our rationale for adding classic negation and FOL expressivity to RDF is not novel. They were articulated in Hayes' "BLOGIC" invited talk at ISWC 2009 [23]. In his argumentation, RDF is portable, i.e. , any RDF triple expresses the same underlying data model, regardless of the standardized syntax used (e.g., RDF/XML, Turtle, N-Triples) or the processing environment. A set of RDF triples on one end of a communication line is translated into an identical set of RDF triples on the other end of the communication line. Web logics such as RDFS and OWL2 DL extend the RDF data model with semantics that enable more complex reasoning. Given a set of RDF triples, new additional RDF triples can be inferred by applying semantics. However, combinations of RDFS and OWL2 DL (and even within OWL2 DL) come with semantics that do not always agree. Web logics that use only a fraction of FOL do not commute: different fragments can disagree on what can be concluded from an RDF knowledge base. These concerns for the portability of Web logic led Hayes to propose weB LOGIC (BLOGIC) as a new approach for portable logic

on the Web, rooted in the theory of existential graphs by Charles Sanders Peirce (1839-1914) [49]. Hayes' ideal of BLOGIC envisions a unified semantic layer for RDF, with a single notion of entailment rather than multiple competing notions across the different layers of the Semantic Web Stack. Hayes's BLOGIC in the form of existential graphs mitigates the limited expressivity of RDF and the limitations on the portability of logic on the Web. To incorporate existential graphs, RDF needs to introduce two additional concepts: a *surface* as a boundary for negated triples and for collections of *graffiti* (blank) nodes that act as existentially quantified variables. Together with the standard assertion of triples and their conjunction, the full expressivity of FOL can be achieved without losing the structure and semantics of existing RDF resources.

Achieving BLOGIC with FOL expressivity again introduces undecidability to the Semantic Web, which is precisely what the RDF designers aimed to avoid. We argue that any *portable* Web logic in RDF will likely not be decidable. Decidability requires a delicate choice in limiting the expressive power of Web logics. While many fragments of FOL are decidable, combining them with the full semantics of RDF and RDFS generally does not preserve decidability; see, for example, [28], where OWL-Full is defined as the syntactic and semantic extension of RDFS and is argued to be undecidable. Only certain restricted constructs yield a decidable fragment, and even then only when combined in specific ways. In practice, combinations of RDFS and OWL constructs are already in use across the Web, and we cannot generally determine in advance which combinations a given source uses. A portable logic that aims to accommodate such arbitrary combinations therefore cannot guarantee decidability in general. We take this to mean that, for transporting Web logic as RDF, undecidability is a property to accommodate rather than a limitation we could remove by adopting a less expressive logic.

Undecidability, in general, should not be a showstopper. Real-world use cases might not require solving the most extreme computability problems. Decidable fragments of FOL can and will still be part of the Web. However, any attempt to integrate these fragments into a unified logical framework inevitably result in undecidability, regardless of the approach taken.

In our paper, we will highlight two use cases that can be solved by implementing our translation of the BLOGIC vision in RDF, which we call *RDF Surfaces*. In summary, our RDF Surfaces program aims to meet the following requirements as fully as possible:

1. Expressing the classic negation of every possible statement.
2. Providing the expressive power of function-free (relational) FOL over a triple signature.
3. Adhere to the RDF program of portability.

One might question why we limit ourselves to FOL and exclude modal logics or non-classical logics. While evaluating modal and non-classical logic is undoubtedly valuable for analyzing diverse forms of discourse, our program focuses on the portability of Web logic. For pragmatic reasons, classical logic emerges as a reasonable choice, given its extensive body of literature and the long history of implementation efforts dating back to the dawn of modern computing. The world of automatic theorem proving is extensive, featuring many FOL provers and an active community, including a "World Championship for Automated Theorem Proving" (CASC).[2] Also, the primary frameworks on the Web are based on fragments FOL, which is where the majority of data resides.

1.4. Contribution

As far as we know, no attempt has been made to implement classical negation, as proposed by the BLOGIC vision, in a concrete RDF syntax and implementation since Hayes' talk in 2009. Our paper will introduce RDF Surfaces as an extension of RDF's simple interpretation, with the extended semantics of classical negation and the expressivity of FOL. In this vision paper, we aim to translate Hayes' BLOGIC

[2] https://tptp.org/CASC/

vision into a concrete RDF syntax, investigate the expressivity of its semantics, test the applicability in real-world use cases , and encourage further research in formalization and potential implementations.

Our main contributions in this paper are thus as follows:

1. **RDF Surfaces Syntax:** We apply existential graphs to RDF in the form of RDF Surfaces and demonstrate how Hayes' BLOGIC vision can be made concrete with a serialization using a subset of the Notation3 syntax.
2. **Investigation of expressivity:** We demonstrate how two additions to the RDF model under simple entailment semantics provides the expressive power of function-free (relational) FOL over a triple signature with explicit quantification, with additions of (a) surfaces with negated contents and (b) collections of graffiti (blank) nodes that act as existentially quantified variables.
3. **Showcase applicability:** We provide two use cases, one from scholarly communication and one from the healthcare domain, demonstrating the need for classic negation. These use cases demonstrate how positive and negative data and logic can be shared using the RDF syntax with extended semantics. Both use cases implement FOL features, such as quantification, disjunctions, and implications, to share logic rules.

1.5. Paper outline

In the remainder of this paper, Section 2 will highlight two scenarios that will be used to illustrate the application of FOL Web logic in scholarly communication and healthcare. In Section 3, we will review related work on implementing FOL and negation on the Web. In Section 4, we introduce RDF Surfaces and its syntax, which implements Hayes' BLOGIC vision. In Section 5, the RDF Surfaces will be applied to the two use cases presented in Section 2. In Section 6, we will discuss our study's main points and further work. Finally, our conclusions will be presented in Section 7.

2. Running examples

This article will use two running examples to highlight the potential for negation on the Web. Our use cases help explain this paper's abstract concepts and put their application in the context of real-world use cases.

2.1. Scholarly communication

For many researchers, choosing the right place to publish is the most contentious question in their pathway to pursue an academic degree. Researchers do not need to spend much time searching online to find numerous websites from universities, libraries, publishers, and individual researchers that present the trade-offs in various publication paths. This question can be more complex than it seems. A researcher might want to give a topic more visibility by targeting a wide (nonspecialist) academic community or the general public. Alternatively, a researcher could choose to publish new results as fast as possible to claim precedence. Some scholarly communities progress in their field by sharing these fast research results as preprints in subject repositories, such as arXiv and medRxiv. However, certain journals refuse to accept research articles previously disseminated in this manner or charge high article processing charges (APC). Institutions might prefer to accept only some types of publications for satisfactory completion of a degree, for instance, only publications from high-impact journals indexed in the Web of Science (WOS) database. The academic world could utilize library databases to share lists of journals that are explicitly stated to be part of the WOS database or explicitly state them as excluded or formally removed from coverage.

It would benefit all scholarly communication network actors to share their preferences using publicly accessible preference documents. These preferences could then provide input for smart agents to suggest new venues and provide the best advice on where to publish. An example of each actor's different types of policies is sketched below in Table 1 and Table 2.

Researcher X Preferences	Department Y Preferences
Prefers a subject repository, a journal that does not charge APC costs or an indexed journal in WOS.	The publication venue must be indexed in WOS.

Table 1
Examples of possible publication venue preferences for a researcher and a department.

Journal ABC Facts	Journal DEF Facts	Journal GHI Facts	Repository JKL Facts
Indexed in WOS. Requires APC.	Indexed in WOS.	Not indexed in WOS. Does not require APC.	A subject repository.

Table 2
Facts for hypothetical journals ABC, DEF, GHI, and a repository JKL.

Both the researcher and department preferences contain explicit and implicit negations. A journal that explicitly states that APC costs are not charged is a researcher's X preference. The department's Y demand for a venue in WOS excludes all venues that are explicitly not indexed in WOS. Journal facts make positive and negative facts clear. These facts can be published online by the journal or publisher's homepage or provided by library databases that track journal information. In our examples, journals ABC and DEF would be researcher and departmental preferences, but journal GHI and repository JKL would not simultaneously match the researcher and departmental preferences.

2.2. Medicine Prescription

For the second use case, we show the applicability of FOL to the healthcare domain. Determining a suitable set of medications is a complex process, taking into account a patient's symptoms, the effectiveness of medication, the patient profile, and other factors. Specifically, the presence of allergies towards one or more types of medication, as well as possible secondary afflictions inferencing with some medication, makes for a difficult process to prescribe each patient the best suitable medication. Combining a high level of required domain knowledge and the need to perform complex reasoning upon said knowledge makes this an overall hard process to capture and automate accurately.

For a given condition, multiple possible medicines are often available to provide treatment. Here, we consider the condition of an acute myocardial infarction, with possible treatments of a high dosage of aspirin or beta-blockers. Aspirin should not be prescribed if a patient is allergic to it or suffers from active peptic ulcer disease. Likewise, beta-blockers should only be prescribed when a patient does not suffer from severe asthma or chronic obstructive pulmonary disease. Furthermore, a low aspirin dosage is known to be an effective treatment for fever, with identical exclusion criteria compared to a high aspirin dosage. The information is summarized in Table 3.

To arrive at a suitable medicine prescription, it is necessary to reason upon negative information. A patient should only be prescribed a given drug if it does hold that said drug is an effective treatment of a patient's condition and if all the known exclusion criteria of the drug are assured to *not* hold, e.g., in the case of aspirin prescription, it must hold that a patient does not have an aspirin allergy nor a peptic ulcer disease.

Medicine	Treated affliction	Exclusion criteria
Low dosage of aspirin	Fever	Aspirin Allergy, Active peptic ulcer disease
High dosage of aspirin	Acute myocardial infarction	Aspirin allergy, Active peptic ulcer disease
Beta-blockers	Acute myocardial infarction	Severe asthma, Chronic obstructive pulmonary disease

Table 3
Overview of Medicine treatment

3. Related Work

A significant body of research on defining the requirements for Web logic with negation is available with possible extensions to FOL expressivity. Table 4 presents an overview of the three main requirements we seek for a Web logic, i.e., enabling classical negation, providing the full expressivity of FOL (including universal quantification), and building upon the RDF data & syntax to attain portability, and comparing them against the solutions discussed in the following paragraphs in more detail. It can be concluded from this table that none of the available solutions fit all the requirements, hence the motivation for RDF Surfaces.

Technology	Classic negation	FOL expressivity	Expressed as RDF
KIF	+	+	-
Common Logic	+	+	-
N3Logic	-	-	+
FIPA	-	-	+
OWL2 DL	+ (restricted)	-	+
RIF	+	+(restricted)	+
SWRL	-	-	+
SWSL	+	+	-
De Bruijn, Tsarkov, Tammet	+	+	-
TPTP	+	+	-
Datalog	-	-	-
ASP	+	+(subset)	-
SPARQL	+(limited to filters)	+(limited to filters)	+

Table 4

Overview of technologies and their support for classic negation with explicit quantification for processing RDF data using an RDF syntax.

In the early 1990s, the Defense Advanced Research Projects Agency (DARPA) and other funding agencies started the development of the Knowledge Interchange Format (KIF) as a machine-readable interchange format of knowledge among disparate programs with an expressivity near equivalent to FOL predicate calculus, including classical negation [18]. KIF's Lisp-based syntax predates the Semantic Web and was, in the 1990s, the de facto exchange format in the research community. Common Logic continued the work of KIF and has since been developed and published as the ISO standard "ISO/IEC 24707:2018" as a framework for a family of logic-based languages [31]. Common Logic can process RDF data but the syntax relies on Lisp-like S-expressions. Common Logic is highly relevant for the BLOGIC vision as it includes Piercian existential graphs in Appendix B of the ISO standard. Our paper applies this logic to the Semantic Web using an RDF syntax, ensuring compatibility with all existing RDF resources. With RDF Surfaces, we strive not to separate data and logic. No separate syntax and semantics are required to negate a set of RDF triples.

In 2000, Berners-Lee called for developing a unifying language for classical logic as an extension, or even the modification, of the RDF model. His SWeLL proposal was imaged to "allow any Web software to read and manipulate data published by any other Web software" [8]. For all logical relations to be expressed, the SWeLL project proposal advocated negation and explicit quantification as an extension to RDF. The work on SWeLL influenced the development of N3Logic [9] as an extension of RDF so that the same language can be used for transporting logic and data. N3Logic provides negation in the form of scoped negation as failure (SNAF), i.e., the monotonic version of NAF. Both NAF and SNAF are logical operations to reason about information missing from a knowledge graph, but cannot be used (or in a very limited form) to express negative information that classic negation requires explicitly. Both NAF and SNAF do not have all the desired properties of classical negation. In N3Logic, it is possible to create negated statements and negated graphs by setting the consequent of an implication "false" $G \rightarrow$ `false`, but this negated graph

does not have the properties of a classic negation. From the assertion that graph G is "true", it does not immediately follow that the double negated $(G \rightarrow \texttt{false}) \rightarrow \texttt{false}$ is also "true", nor does the reverse hold.

In 2001, the FIPA RDF Content Language Specification [61] was created to specify how RDF can be used as a message content language in the communication acts of FIPA-compliant agents. It proposes a method to express negated RDF facts by adding a belief or disbelief in the facts. The FIPA proposal adds a `fipa:Proposition` and relies on reification of RDF triples. A `fipa:believe` predicate can be added to the reified triple to express a boolean trust. Using this mechanism, a single triple can be interpreted as "false", but it is not a classical negation for the same reason as N3Logic "false" is not a classical negation.

Related developments were made in 2004 with the Web Ontology Language (OWL) as an extension of RDF. The newest version, OWL 2 [10], has several profiles, of which OWL 2 DL is based on fragments of FOL. In all OWL 2 DL profiles, negation is available in the form of `owl:complementOf` and `owl:NegativePropertyAssertion`, but these negations are restricted to specific cases in the profiles to prevent the halting problem and ensure the language remains decidable. As an example, using the `owl:NegativePropertyAssertion` it is possible to state that two individuals do not have the relation `:hasParent`,

```
:John a owl:NamedIndividual .
:Mary a owl:NamedIndividual .

[ a owl:NegativeObjectPropertyAssertion ;
  owl:sourceIndividual :John ;
  owl:assertionProperty :hasParent ;
  owl:targetIndividual :Mary ] .
```

From these statements it follows that `:John :hasParent :Mary` is "false", which makes it a negation of one triple. However, this negation does not have the properties of a full classical negation. The construction only works for a single triple (and not arbitrary collections of triples), and the double negation properties of classical negation are unavailable (which would lead to full FOL expressivity). As for N3Logic, a "false" is available but in a limited form. These types of "false" statements can be used as constraints in an ontology but do not have the properties of a proper classical negation. There is even a more subtle difference between this type of negation and classical negation. The John and Mary example does not state that it is impossible that John has a parent Mary only that it is not modeled in a particular ontology.

Universal quantification is available in OWL 2 DL, but in a restricted form. The `ObjectAllValuesFrom` and `DataAllValuesFrom` predicates require a set of all individuals to choose from in a quantification. This restricted quantification differs from classical universal quantification, which permits an arbitrary (unlimited) number of individuals.

The Rule Interchange Format (RIF) [35] was an activity started in 2005 within the W3C to develop Web standards for the interchange of rules among disparate systems, especially on the Semantic Web. RIF is a collection of extensible languages and dialects serialized as XML documents. Two types of languages are available: logic-based dialects and dialects for rules with actions. The logic-based dialects include languages based on FOL and various non-FOL semantics but do not allow negation in the rule head or body (the Horn subset). In 2009 a mapping from RIF to RDF was created [21].

Semantic Web Rule Language (SWRL) [29] is a W3C membership submission that extends the OWL Web Ontology Language with the Rule Interchange Format (RIF). The proposed language extends OWL axioms to include Horn clauses for OWL descriptions and properties and a limited set of built-in functions. This form supports neither the disjunction nor the classical negation of clauses. Also, to guarantee decidability, rules are restricted to only include named individuals (and not existentially introduced individuals). The language allows for explicit quantification, but introducing variables goes beyond RDF semantics [44].

The Semantic Web Services Language (SWSL) [6] is a language for specifying the formal characterizations of Web service concepts and descriptions of individual Web services. The language consists of two sublanguages: SWSL-FOL, a full FOL language, and SWSL-Rules, a rule-based sublanguage with non-monotonic semantics. However, the authors of SWSL did not envision the need for full FOL reasoners based on SWSL-FOL as its main use case is creating Web service ontologies. The syntax of SWL is inspired by F-Logic and is not based on RDF.

Several papers by De Bruijn, Tsarkov and Tammet demonstrate the embedding of RDF in FOL using frameworks, such as F-Logic[12], Vampire[54] and JSON-LD[52]. The advantage of FOL lies in its well-established nature and the ability to define its mappings, as demonstrated by these papers. However, our project aims to achieve the expressivity of FOL within the Web language itself, specifically in the RDF model, from the ground up. The papers start from the opposite direction and try incorporating RDF semantics into a framework with FOL semantics. For similar reasons, plain FOL, such as the TPTP language[3], is excluded.

Datalog knows some extensions that aim to incorporate negation into the rule language, most notably semi-positive Datalog and stratified Datalog [34]. Including negative atoms in rule bodies allows negation to be contained. This inclusion allows for introducing disjunctive and negative statements through a back door. Nonetheless, as is the case for SWRL considered above, Datalog's variants do not allow for existential variables, only to reason over existing ones [1]. Furthermore, many Datalog variants are restricted to *safe rules*, where all variables of a rule must occur in a positive atom of the rule body, which further restricts the free use of negative statements.

Answer Set Programming (ASP) [15] provides classical "strong" negation and NAF, and even first-order extensions can be written [42]. Eiter et al. [16] provide an extension of ASP with description logic for the Semantic Web. However, using Datalog, ASP, and even Prolog as Web logic creates a dichotomy between the world of data (RDF) and the world of logic (the computer program). In the rationale of RDF Surfaces, it should be possible to negate information in RDF and transport logic/reasoning in a portable way (using RDF). RDF Surfaces could be the language, the syntactic sugar, to transport RDF data with FOL expressivity to an ASP, Datalog, or Prolog program. We do not deny the expressivity of any of these programming languages. Our argument reverses the conventional perspective: Web logics that follow the requirements of the computing agent (decidable, tractable). RDF Surfaces advocates for Web logic for the human agent, emphasizing sharing information and logic in a portable way.

SPARQL can use classical negation and first-order expressivity in filters that can be used to query an RDF data set (that has no FOL expressivity). Three types of negation are supported: (a) the Boolean NOT operator that can be used in filters; (b) the negation as failure operators MINUS and NOT-EXISTS; and (c) a combination of OPTIONAL with the BOUND operator , which is a form of NAF [2]. Additionally, SPARQL provides an RDF serialization through SPARQL-SPIN [37]. However, we regard SPARQL primarily as a query language rather than a Web logic language.

4. RDF Surfaces

The core concept of RDF Surfaces build on two well-established principles from propositional and predicate logic: (1) all logical operators can be constructed using only two primitives – conjunction (AND) and negation (NOT), (2) De Morgan's duality allows existentially quantified variables to be transformed into universally quantified variables, and vice-versa. In RDF, an RDF graph represents the conjunction of its triples, with blank nodes interpreted as existentially quantified variables. If a method were devised to negate triples and define variable scope, then the full set of of FOL operators could be achieved. In our paper, we define *surfaces* on which RDF triples and existentially quantified variables can be written. By default this surface is "positive". By asserting a positive surface, the conjunction of RDF triples on such

[3] https://tptp.org/Proposals/TPILanguage.html

a surface is regarded "true". By asserting a "negative" surface, the conjunction of RDF triples on such surface is regarded "false". These surfaces also establish a scope for variables. Using De Morgan's duality we can create existentially and universally quantified variables by placing variables – referred to as *graffiti nodes* – on positive or negative surfaces, respectively. Blank nodes will act as coreferences to these graffiti nodes. Graffiti nodes are local to an RDF source and can not be shared. Merging two or more RDF sources requires creating new graffiti.

4.1. Definitions and semantics

To define RDF Surfaces and their semantics, we introduce in this section some definitions and a notational mechanism we will use in our paper for constructing FOL formulas that incorporate RDF triples. The model-theoretic semantics we present here is complemented by a formal reduction of RDF Surfaces to FOL, we developed in a companion paper [5], which proves that entailment under the resulting semantics coincides with RDF simple entailment.

For the concept of an RDF triple, we will follow Hogan [27] and assume the existence of a pairwise disjoint infinite sets $\boldsymbol{U}$ (URIs), $\boldsymbol{L}$ (literals), and $\boldsymbol{B}$ (blank nodes). An RDF triple is then defined as follows:

Definition 4.1. An *RDF triple* is a tuple $\langle \boldsymbol{s}\ \boldsymbol{p}\ \boldsymbol{o}\rangle \in (\boldsymbol{U} \cup \boldsymbol{B}) \times \boldsymbol{U} \times (\boldsymbol{U} \cup \boldsymbol{B} \cup \boldsymbol{L})$ where $\boldsymbol{s}$ is called the *subject*, $\boldsymbol{p}$ the *predicate*, and $\boldsymbol{o}$ the *object*.

An RDF predicate originates from the use of predicates in FOL, such as the $\boldsymbol{p}$ in Definition 4.1. RDF triples can be represented in FOL in the form $\boldsymbol{p}(\boldsymbol{s}, \boldsymbol{o})$. However, we prefer in this paper to consider the RDF predicate as a standard term and use a static *ternary predicate* $tr(\boldsymbol{s}, \boldsymbol{p}, \boldsymbol{o})$ to represent RDF triples [26]. Throughout this paper, we generally omit tr, use a triple notation and express a static ternary predicate in FOL in the form $\langle \boldsymbol{s}\ \boldsymbol{p}\ \boldsymbol{o}\rangle$, where $\boldsymbol{p}$ is a URI, as required by the RDF data model.

An RDF Surface can be introduced by extending the RDF data model with a new type of triple called the *Hayes triple*:

Definition 4.2. A *Hayes triple* is a tuple $\langle \boldsymbol{Gr}\ \boldsymbol{S}\ \boldsymbol{H}\rangle$ where:

- $\boldsymbol{Gr} \subseteq \boldsymbol{B}$ is a (possibly empty) set of blank nodes called *graffiti*, and each element of the set is a *graffiti node*.
- $\boldsymbol{S} \in \{\boldsymbol{+}, \boldsymbol{-}\} \subseteq \boldsymbol{U}$ is called the *surface type*. Each surface type is identified by a distinct URI.
- $\boldsymbol{H}$ is a set of RDF triples and Hayes triples, called the *Hayes graph*.

Definition 4.2 is recursive, allowing the nesting of Hayes triples. The first role of a Hayes triple is to influence the logical interpretation of the set of graffiti and triples, as will be explained later in this section. To offer a preview of its usage, when a Hayes triple with a positive surface ($\boldsymbol{+}$) is asserted, then the conjunction of the triples in the Hayes graph is considered "true". Conversely, when a Hayes triple with a negative surface ($\boldsymbol{-}$) is asserted, the conjunction of the triples in the Hayes graph is considered "false". That is, the Hayes triple

$$\langle \{\} + \{\langle \boldsymbol{a}\ \boldsymbol{b}\ \boldsymbol{c}\rangle, \langle \boldsymbol{d}\ \boldsymbol{e}\ \boldsymbol{f}\rangle\}\rangle \text{ with } \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}\} \subseteq \boldsymbol{U}$$

will be understood as a FOL conjunction of RDF triples $(\langle \boldsymbol{a}\ \boldsymbol{b}\ \boldsymbol{c}\rangle \wedge \langle \boldsymbol{d}\ \boldsymbol{e}\ \boldsymbol{f}\rangle)$, and the triple

$$\langle \{\} - \{\langle \boldsymbol{a}\ \boldsymbol{b}\ \boldsymbol{c}\rangle, \langle \boldsymbol{d}\ \boldsymbol{e}\ \boldsymbol{f}\rangle\}\rangle \text{ using the same } \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}\}$$

will be understood as a FOL negation of a conjunction of RDF triples $\neg(\langle \boldsymbol{a}\ \boldsymbol{b}\ \boldsymbol{c}\rangle \wedge \langle \boldsymbol{d}\ \boldsymbol{e}\ \boldsymbol{f}\rangle)$. If $(\langle \boldsymbol{a}\ \boldsymbol{b}\ \boldsymbol{c}\rangle \wedge \langle \boldsymbol{d}\ \boldsymbol{e}\ \boldsymbol{f}\rangle)$ is considered "true", then $\neg(\langle \boldsymbol{a}\ \boldsymbol{b}\ \boldsymbol{c}\rangle \wedge \langle \boldsymbol{d}\ \boldsymbol{e}\ \boldsymbol{f}\rangle)$ should be considered "false".

The examples above illustrate a second role of the Hayes triple in RDF Surfaces: it defines a boundary around triples. This boundary functions analogously to parentheses in FOL, grouping statements to enforce a precedence applying logical operations. In our negated example, the conjunction of triples taking precedence over the negation.

To explain the role of graffiti, we require a new definition.

Definition 4.3. Every occurrence of a blank node x in any (nested) RDF triple of $\boldsymbol{H}$ that is part of the set of graffiti nodes $\boldsymbol{Gr}$ of the Hayes triple is called a *bound* occurrence of x in $\boldsymbol{H}$. This blank node is a *coreference* to the graffiti node with the same label. Any blank node x that is not bound is called a *free* occurrence of x in $\boldsymbol{H}$. We denote the set of free blank nodes in $\boldsymbol{H}$ as $\mathtt{fr}(\boldsymbol{H})$.

Definition 4.3 provides a new role for blank nodes in RDF Surfaces, distinguished from their usage plain RDF. In RDF, blank nodes have the role of existentially quantified variables. In RDF Surfaces, the graffiti nodes will take the role of existentially quantified variables, and the blank nodes will become coreferences to graffiti nodes. That is, the triple

$$\langle \{\boldsymbol{x}\} + \{\langle \boldsymbol{x}\ \boldsymbol{b}\ \boldsymbol{c}\rangle\}\rangle \text{ with } \boldsymbol{x} \in \boldsymbol{B} \text{ and } \{\boldsymbol{b},\boldsymbol{c}\} \subseteq \boldsymbol{U}$$

will be understood as the FOL formula $(\exists x : \langle x\ \boldsymbol{b}\ \boldsymbol{c}\rangle)$, and the triple

$$\langle \{\boldsymbol{x}\} - \{\langle \boldsymbol{x}\ \boldsymbol{b}\ \boldsymbol{c}\rangle\}\rangle \text{ with } \boldsymbol{x} \in \boldsymbol{B} \text{ and } \{\boldsymbol{b},\boldsymbol{c}\} \subseteq \boldsymbol{U}$$

will be understood as the FOL formula $\neg(\exists x : \langle x\ \boldsymbol{b}\ \boldsymbol{c}\rangle)$. In both examples, the blank node x in the RDF triple is bound to the graffiti node with the same label. The negated example highlights again the role of the Hayes triple as a form of precedence parentheses: the quantification taking precedence over the negation.

Using these definitions, we can define an RDF Surface:

Definition 4.4. An *RDF Surface* is a Hayes triple with a *positive* surface type, a finite set of triples as Hayes graph, and such that every Hayes triple within the Hayes graph is of a *negative* surface type (regardless of the depth of their nesting).

The set of *terms* that are part of the Hayes graph $\boldsymbol{H}$, denoted by $\mathtt{terms}(\boldsymbol{H})$, is the set of elements of $\boldsymbol{U} \cup \boldsymbol{B} \cup \boldsymbol{L}$ that occur in the RDF triples of $\boldsymbol{H}$ (possibly deeply nested). The *vocabulary* of $\boldsymbol{H}$, denoted by $\mathtt{voc}(\boldsymbol{H})$, is defined by $\mathtt{terms}(\boldsymbol{H}) \cap (\boldsymbol{U} \cup \boldsymbol{L})$, that is all the terms without the blank nodes.

To define the semantics of RDF Surfaces, we rely on an extension of the simple interpretation of RDF that does not rely on predefined vocabularies.

Definition 4.5. An *interpretation* is a tuple

$$\mathcal{I} = \langle Res, Prop, \rho, \pi, \beta, Ext\rangle$$

such that:

- Res is a non-empty set of resources called the *domain of discourse*. We consider the set of literals L to be part of Res, i.e., $L \subseteq Res$;
- $Prop$ is a set of property names (not necessarily disjoint from Res);
- $\rho : U \to Res$ that maps each URI to its representation in the domain of discourse;
- $\pi : U \to Prop$ that maps each URI to a property name;
- $Ext : Prop \to 2^{Res \times Res}$ is a mapping that assigns binary relation to each property name;
- $\beta : X \subseteq B \to U$ a partial mapping of blank nodes to URIs.

The above definition is highly similar to an RDF simple interpretation (cfr. [24]). *Res* represents a set of resources, *Prop* a set of properties, and *Ext* a mapping which maps properties to pairs of resources, each of which has a counterpart in RDF simple interpretations. Mapping β similarly serves to map blank nodes ('variables') to specific URI's. ρ and π serve to connect the URI to the actual resource ('entity') or property. They have no counterpart in RDF simple interpretations, as resource and URI are assumed to be the same in said theory.

Note that π can be applied to any URI, thus also to URIs that do not coincide with a property (e.g. `:JournalA`). It suffices to define $Ext(\pi(\texttt{:JournalA})) = \varnothing$.

The 'meaning' of a URI $u \in U$ within interpretation $\mathcal{I}$ is given by its mapping to the domain of discourse *Res*, i.e. by $\rho(u)$. We denote this meaning of u as $\mathcal{I}(u) := \rho(u)$. Similarly, the meaning of a blank node $b \in B$ in $\mathcal{I}$ is equal to the meaning of the URI it is assigned to, $\mathcal{I}(b) := \mathcal{I}(\beta(b)) = \rho(\beta(b))$, in case it is assigned by β. If not, b retains its meaning as a blank node in $\mathcal{I}$: $\mathcal{I}(b) := b$. Lastly, the meaning of each literal l within interpretation $\mathcal{I}$ is equal to the literal itself: $\mathcal{I}(l) = l$.

Given any *ground* RDF triple $\langle \boldsymbol{s}\ \boldsymbol{p}\ \boldsymbol{o}\rangle$ in $\boldsymbol{H}$, that is an RDF triple without blank nodes, it is "true" (also denoted by $\top$) under the interpretation $\mathcal{I}$, if $\big(\mathcal{I}(\boldsymbol{s}), \mathcal{I}(\boldsymbol{o})\big) \in Ext\big(\pi(\boldsymbol{p})\big)$. In short, we say $\mathcal{I}(\langle \boldsymbol{s}\ \boldsymbol{p}\ \boldsymbol{o}\rangle)$ is "true". In case of a blank node in $\langle \boldsymbol{s}\ \boldsymbol{p}\ \boldsymbol{o}\rangle$, the blank node will be assigned according to β, if applicable, e.g. $\mathcal{I}(\langle b\ \boldsymbol{p}\ \boldsymbol{o}\rangle)$ is true iff. $\big(\rho(\beta(b)), \mathcal{I}(\boldsymbol{o})\big) \in Ext\big(\pi(\boldsymbol{p})\big)$.

The truth value of a Hayes triple $\langle \boldsymbol{Gr}\ \boldsymbol{S}\ \boldsymbol{H}\rangle$ is defined as follows: For a Hayes triple $\langle \boldsymbol{Gr}\ \boldsymbol{S}\ \boldsymbol{H}\rangle$, we determine whether $\langle \boldsymbol{Gr}\ \boldsymbol{S}\ \boldsymbol{H}\rangle$ is "true" under an interpretation $\mathcal{I}$ in the following manner: firstly, suppose that $\texttt{fr}(\boldsymbol{H}) = \varnothing$ and $S = +$. $\mathcal{I}(\langle \boldsymbol{Gr}\ +\ \boldsymbol{H}\rangle)$ is considered "true" iff. there exists a partial mapping $\beta^* : \boldsymbol{Gr} \subseteq B \to U$ for which, for every triple t in $\boldsymbol{H}$, it holds that $\mathcal{I}^*(t)$, where $\mathcal{I}^*$ is the interpretation $\langle Res, Prop, \rho, \pi, \beta + \beta^*, Ext\rangle$. $\beta + \beta^*$ is thereby the combination of the two partial mappings, i.e. for $a \in X$ we have $(\beta + \beta^*)(a) = \beta(a)$, and for $a \in \boldsymbol{Gr}$ we have $(\beta + \beta^*)(a) = \beta^*(a)$. This implicitly assumes β and β^* do not contradict each other; for every $a \in X \cap \boldsymbol{Gr} : \beta(a) = \beta^*(a)$.

For the interpretation of a Hayes triple $\langle \boldsymbol{Gr}\ +\ \boldsymbol{H}\rangle$ with $\texttt{fr}(\boldsymbol{H}) \neq \varnothing$, we consider the existential closure of said Hayes triple; $\mathcal{I}(\langle \boldsymbol{Gr}\ +\ \boldsymbol{H}\rangle)$ is considered "true" iff. there exists a partial mapping $\beta^* : \texttt{fr}(\boldsymbol{H}) \to U$ for which $\mathcal{I}^*(\langle \boldsymbol{Gr}\ +\ \boldsymbol{H}\rangle)$ is "true". This is equivalent to encapsulating the original triple in a positive surface that contains all of the remaining free variables as graffiti: $\langle \texttt{fr}(\boldsymbol{H}) + \{\langle \boldsymbol{Gr}\ +\ \boldsymbol{H}\rangle\}\rangle$. Lastly, for $S = -$, we say that $\mathcal{I}(\langle \boldsymbol{Gr}\ -\ \boldsymbol{H}\rangle)$ is considered "false" iff. $\mathcal{I}(\langle \boldsymbol{Gr}\ +\ \boldsymbol{H}\rangle)$ is "true" and vice versa.

An interpretation $\mathcal{I}$ is called a *model* for a Hayes triple $\langle \boldsymbol{Gr}\ \boldsymbol{S}\ \boldsymbol{H}\rangle$ iff. $\mathcal{I}(\langle \boldsymbol{Gr}\ \boldsymbol{S}\ \boldsymbol{H}\rangle)$ is "true". In accordance with the W3C recommendation on RDF [24], an interpretation is infinite, as the set of all URIs – and therefore ρ and π – is infinite.

In Table 5, we provide an overview of how the elementary FOL constructs are translated into Hayes triples. Conjunction $\wedge$ follows easily by placing the triples together in a single Hayes graph and embedding said Hayes graph in a positive (graffiti-free) Hayes triple. For other operators such as the implication $\to$, we rely on an equivalent representation of implication, i.e., $\varphi \to \psi = \neg\varphi \vee \psi = \neg\big(\varphi \wedge \neg\psi\big)$. Working outwards, we deduce that ψ must be embedded in a negative surface $\langle\{\} - \{\psi\}\rangle$, subsequently put together with φ embedded in another negative surface, resulting in $\langle\{\} - \{\varphi, \langle\{\} - \{\psi\}\rangle\}\rangle$.

4.2. Concrete N3S syntax

Hayes introduced an annotated Turtle syntax, using comments, in his ISWC BLOGIC presentation to express negated RDF statements. With RDF Surfaces, we opted to use a subset of the Notation3 (N3) [59] syntax to provide a more structured serialization format in RDF itself. N3 already has parsers, such as n3 [56], RDF::N3 [33], RDFLib [39], and EYE [13], that provide syntactical support for N3 graph terms and N3 list terms , which provide a possibility for a direct translation of RDF Surfaces into N3. The subset of N3, referred to as "RDF Surfaces in N3" (N3S), is based on the Turtle syntax [7] with an additional an additional production rule to define a Hayes triple. The N3S syntax is defined as follows.

Definition 4.6. The EBNF notation of N3S is that of Turtle with the additional production rule for a Hayes triple and incorporating such a Hayes triple into the production rule for triples.

FOL	Equivalent expression	Hayes triple
$\top$		$\langle\{\} + \{\}\rangle$
$\bot$		$\langle\{\} - \{\}\rangle$
$\langle \boldsymbol{s}\ \boldsymbol{p}\ \boldsymbol{o}\rangle$		$\langle\{\} + \{\langle \boldsymbol{s}\ \boldsymbol{p}\ \boldsymbol{o}\rangle\}\rangle$
$\neg(\langle \boldsymbol{s}\ \boldsymbol{p}\ \boldsymbol{o}\rangle)$		$\langle\{\} - \{\langle \boldsymbol{s}\ \boldsymbol{p}\ \boldsymbol{o}\rangle\}\rangle$
$\langle \boldsymbol{s}_1\ \boldsymbol{p}_1\ \boldsymbol{o}_1\rangle \wedge \langle \boldsymbol{s}_2\ \boldsymbol{p}_2\ \boldsymbol{o}_2\rangle$		$\langle\{\} + \{\langle \boldsymbol{s}_1\ \boldsymbol{p}_1\ \boldsymbol{o}_1\rangle, \langle \boldsymbol{s}_2\ \boldsymbol{p}_2\ \boldsymbol{o}_2\rangle\}\rangle$
$\langle \boldsymbol{s}_1\ \boldsymbol{p}_1\ \boldsymbol{o}_1\rangle \vee \langle \boldsymbol{s}_2\ \boldsymbol{p}_2\ \boldsymbol{o}_2\rangle$	$\neg\big(\neg\langle \boldsymbol{s}_1\ \boldsymbol{p}_1\ \boldsymbol{o}_1\rangle \wedge \neg\langle \boldsymbol{s}_2\ \boldsymbol{p}_2\ \boldsymbol{o}_2\rangle\big)$	$\langle\{\} - \{\langle\{\} - \{\langle \boldsymbol{s}_1\ \boldsymbol{p}_1\ \boldsymbol{o}_1\rangle\}\rangle, \langle\{\} - \{\langle \boldsymbol{s}_2\ \boldsymbol{p}_2\ \boldsymbol{o}_2\rangle\}\rangle\}\rangle$
$\langle \boldsymbol{s}_1\ \boldsymbol{p}_1\ \boldsymbol{o}_1\rangle \rightarrow \langle \boldsymbol{s}_2\ \boldsymbol{p}_2\ \boldsymbol{o}_2\rangle$	$\neg\big(\langle \boldsymbol{s}_1\ \boldsymbol{p}_1\ \boldsymbol{o}_1\rangle \wedge \neg\langle \boldsymbol{s}_2\ \boldsymbol{p}_2\ \boldsymbol{o}_2\rangle\big)$	$\langle\{\} - \{\langle \boldsymbol{s}_1\ \boldsymbol{p}_1\ \boldsymbol{o}_1\rangle, \langle\{\} - \{\langle \boldsymbol{s}_2\ \boldsymbol{p}_2\ \boldsymbol{o}_2\rangle\}\rangle\}\rangle$
$\exists x : \langle x\ \boldsymbol{p}\ \boldsymbol{o}\rangle$		$\langle\{x\} + \{\langle x\ \boldsymbol{p}\ \boldsymbol{o}\rangle\}\rangle$
$\forall x : \langle x\ \boldsymbol{p}\ \boldsymbol{o}\rangle$	$\neg\big(\exists x : \neg\langle x\ \boldsymbol{p}\ \boldsymbol{o}\rangle\big)$	$\langle\{x\} - \{\langle\{\} - \{\langle x\ \boldsymbol{p}\ \boldsymbol{o}\rangle\}\rangle\}\rangle$

Table 5
Representation of FOL constructs in Hayes triples

```
hayestriple ::= graffiti surfacetype hayesgraph
graffiti ::= '(' BlankNode* ')'
surfacetype ::= predicate
hayesgraph ::= formula
formula ::= '{' ( triples ( '.' triples )* '.'? )? '}'
triples ::= subject predicateObjectList\\
          | blankNodePropertyList predicateObjectList?
          | hayestriple
```

In Definition Definition 4.6, `graffiti` is an N3 list term to represent the set of graffiti nodes $\boldsymbol{Gr}$. The `surfacetype` is a predicate that defines a surface type – currently we only define the negative surface type with the IRI `log:onNegativeSurface`.[4] The RDF source that contains the Hayes triples implicitly defines the default positive surface of the RDF Surface. The `hayesgraph` is an N3 graph term (`formula` as defined by N3) that represents the Hayes graph $\boldsymbol{H}$ which can contain triples and thus also deeper nested Hayes triples (by the extension of the production rule for `triples` with the `hayestriple`). The full N3S grammar is provided in Appendix A.

Using this N3S syntax, every well-formed Turtle document is an N3S document, and every well-formed N3S document is a well-formed N3 document. Syntactically, Turtle $\subset$ N3S $\subset$ N3.

To illustrate the negation of RDF statements, we start with a simple example in Listing 1.

```
@prefix : <https://example.org/ns#> .

_:x :indexed :MyArticle .
```

Listing 1: A simple triple in N3S syntax.

Listing 1 has the FOL translation $\exists x : \langle x$ `:indexed :MyArticle`$\rangle$, where the blank node `_:x` is interpreted as an existential quantified variable, as is required by standard RDF semantics. This FOL translation can be loosely read as "It is the case that someone indexed MyArticle."

When we negate the triple of Listing 1, we get Listing 2.

```
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix : <https://example.org/ns#> .

( ) log:onNegativeSurface {
    _:x :indexed :MyArticle .
```

[4] `log` is the prefix for the *Semantic Web Area for Play* namespace http://www.w3.org/2000/10/swap/log#.

```
} .
```

Listing 2: The negated version of Listing 1.

The FOL translation of Listing 2 is $\exists x : \neg\langle x$ `:indexed :MyArticle`$\rangle$ and can be read as "There exists an X such that it is not the case that X indexed MyArticle". A simplified reading would be: "Someone did not index MyArticle".

Listing 1 and Listing 2 highlight the implicit positive surface with possible graffiti nodes that is assumed on the boundary of every RDF source. Every 'free' blank node in an RDF graph is assumed to be implicit existential closed in the set of implicit graffiti nodes of this (default) positive surface. These implicit graffiti nodes cannot be shared between RDF sources. When combining two or more RDF sources, a new set of graffiti nodes must be created – "engraved" in Hayes terminology – by relabeling blank nodes in the resulting new RDF source.

Adding graffiti nodes to the N3 list in the subject position of a negative surface should be interpreted as *embedding* these nodes in the negative surface. These nodes are local to the enclosed surface, thereby defining their scope. We use the term embedding to highlight the role of the Hayes graph interpreted as precedence parentheses, with the quantification taking precedence over the negation of the triples in the Hayes graph. If the Hayes graph in the object position contains blank nodes with the same label as a graffiti node, then these blank nodes act as coreferences to that graffiti node. The effect of embedding is demonstrated in Listing 3.

```
1 @prefix log: <http://www.w3.org/2000/10/swap/log#> .
2 @prefix : <https://example.org/ns#> .
3
4 (_:x) log:onNegativeSurface {
5     :MyArticle :title _:x .
6 } .
```

Listing 3: A negative surface with a graffiti node labeled `_:x` made local to the negative surface and a blank node on the surface with the same label coreferring to the graffiti node.

The graffiti node on line 4 in Listing 3 with label `_:x` should be interpreted local to the Hayes graph defined on lines 4-6. The blank node with label `_:x` on line 5 is a coreference to the graffiti node defined on line 4. We translated the set of graffiti nodes in our formalism to an N3 collection of graffiti node for pragmatic reasons of staying close to the N3 abstract model. Listing 3 has the FOL translation:

$$\neg\left(\exists x : \langle \texttt{:MyArticle :title } x\rangle\right) \equiv \forall x : \neg\langle \texttt{:MyArticle :title } x\rangle$$

and can be read as "It is not the case that MyArticle has some title" which is the same as stating, "There is no title for MyArticle."

Depending on the nesting level of negative surfaces, and the embedding of graffiti nodes in these surfaces, any combination of existential or universal quantified variables can be created. Graffiti nodes embedded in a surface with an even nesting level are existentially quantified variables. Graffiti nodes embedded in a surface with an odd nesting level are universally quantified variables.

Listing 4 provides an example where the embedding of graffiti nodes in negative surfaces with different nesting levels is demonstrated.

```
1 @prefix log: <http://www.w3.org/2000/10/swap/log#> .
2 @prefix : <https://example.org/ns#> .
3
4 (_:x _:y) log:onNegativeSurface {
```

```
5     _:x a :Article .
6     _:y :reviewed _:x .
7     (_:z) log:onNegativeSurface {
8        _:z :publishes _:x .
9     } .
10 } .
```

Listing 4: A double nested negative surface. The outer surface has an odd nesting level. The inner surfaces has an even nesting level.

We determine the nesting level of surfaces, starting with level 1 for the outermost surfaces, level 2 for the next deeper layer, and so on. In Listing 4, the outermost negative surface, defined on lines 4-10, has an odd nesting level. The deeper nested negative surface, defined on lines 7-9, has an even nesting level. The FOL translation of this listing is:

$$\neg\left(\exists x, y : \langle x \texttt{ a :Article}\rangle \wedge \langle y \texttt{ :reviewed } x\rangle \wedge \neg\left(\exists z : \langle z \texttt{ :publishes } x\rangle\right)\right) \equiv$$
$$\forall x, y : \left(\langle x \texttt{ a :Article}\rangle \wedge \langle y \texttt{ :reviewed } x\rangle\right) \rightarrow \exists z : \langle z \texttt{ :publishes } x\rangle.$$

and can be loosely read as "If an article has been reviewed by someone, then there is someone who publishes it." This FOL translation illustrates how the graffiti nodes embedded in a negative surface with an odd nesting level can be interpreted as universal quantified variables and graffiti nodes embedded in a negative surface with an even nesting level as existential quantified variables.

The N3S syntax offers humans the expressivity to encode FOL formulas with binary predicates using only a limited amount of extra syntax. However, this comes at the cost of increased verbosity when expressing formulas, such as implications and, as will be shown later in this document, disjunctions. In practice, we foresee that the N3S syntax will mainly provide machine-level interoperability. Library documents can be envisioned which helps human editors subsuming the meaning of higher-level RDF concepts through RDF Surfaces rules. In the following subsection, an example of such a rule will be provided.

4.3. Simple entailment and the limitations on N3 list and graph terms

It is important to differentiate between the requirements for the RDF data model (and its semantics) to describe RDF Surfaces and the serialization format to transport that model.

Our first choice for using N3S as a serialization format is pragmatic, based on de facto approaches to make statements about sets of RDF triples. Alternatively, we could have considered named graphs. However, if the surfaces have IRIs, this approach risks self-referential surfaces and paradoxes. If the surfaces use blank node identifiers, it also leads to issues with the quantification of surfaces. We aimed to stay close to the Semantic Web ideal that everything can be expressed as triples, though with an extended data model and syntax that is compatible with N3.[5]

Our requirements for the RDF model focus on identifying the minimal additions needed to achieve the expressivity of FOL with explicit quantification. RDF Surfaces require two additions: a concept of a surface containing a (possibly empty) set of triples and a concept of a set of graffiti blank nodes that act as existentially quantified variables. RDF Surfaces relies on the simplest version of entailment: only simple entailment is required[6] and other entailments such as RDFS follow by applying RDF Surfaces formulas. That is, from:

$$\langle \texttt{:CelestialObject rdfs:subClassOf :Cheese}\rangle \wedge \langle \texttt{:Moon a :CelestialObject}\rangle \tag{1}$$

[5] Other extension to the RDF data model exists, such as RDF-star (https://www.w3.org/2021/12/rdf-star.html), which we did not consider due its limited expressivity for nesting multiple triples.

[6] https://www.w3.org/TR/rdf11-mt/#simpleentailment

one may conclude

$$\langle \texttt{:Moon a :CelestialObject} \rangle \tag{2}$$

but not

$$\langle \texttt{:Moon a :Cheese} \rangle, \tag{3}$$

as this relies on RDFS entailment. In RDF Surfaces, we do not rely on concepts such as classes. The `:Moon` resource is not an instance of the `:CelestialObject` class in our previous example. We build other RDF entailments from the ground up, creating a foundational layer to make entailments explicit based on FOL. To encode the meaning of the RDFS version[7], an extra formula could be added to the N3S source, such as the RDF Surfaces version of the symbolic form:

$$\forall x, y, z : ((\langle \texttt{x rdfs:subClassOf y} \rangle \wedge \langle \texttt{z a x} \rangle) \rightarrow \langle \texttt{z a y} \rangle)\,. \tag{4}$$

The N3S translation of Equation (4) is provided in Listing 5.

```
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <https://example.org/ns#> .

(_:x _:y _:z) log:onNegativeSurface {

    _:x rdfs:subClassOf _:y .
    _:z a _:x .

    ( ) log:onNegativeSurface {
        _:z a _:y .
    } .
} .
```

Listing 5: The N3S translation of Equation (4).

Using this method, RDFS concepts such as `rdfs:subClassOf` can become collections of syntactic sugar for RDF Surfaces abbreviations and restrictions [23]. The meaning of `rdfs:subClassOf` can be made explicit using RDF Surfaces. When applying Equation (4) to Equation (1), we can infer both Equation (2) and Equation (3) using the explicit meaning of `rdfs:subClassOf` without relying on any implicit notions of classes and instances.

The N3S syntax is based on a subset of the N3 syntax, incorporating one restriction and one extension to the interpretation of N3 list terms and N3 graph terms:

- N3 lists terms are first-class citizens in N3 (and N3S). We require this feature in N3S because "classical" RDF lists introduce blank nodes. This is problematic, as we need to know exactly how blank nodes coreference graffiti nodes. In N3, list terms can be deeply nested and may include elements such as IRI-s, blank nodes, literals, and graph terms. In contrast, N3S does not permit nesting of N3 list terms, and only blank nodes are allowed as list elements.
- N3 graph terms are a syntactical feature we borrow from N3 to provide a closed scope for negating triples. In N3, graph terms do not assert the contents of the enclosed RDF graph, ensuring referential opacity [4]. Blank nodes are existentially quantified variables scoped within a graph term. In N3S, a

[7] See also ***rdfs9*** https://www.w3.org/TR/rdf11-mt/#patterns-of-rdfs-entailment-informative

graph term in the object position of a negative surface negates the enclosed RDF graph. The graph term defines a syntactic scope for an RDF graph. Blank nodes are coreferences to graffiti nodes that are the existentially quantified variables.

The consequences of scoping on blank nodes in N3S will be the subject of the next subsection.

4.4. Blank nodes and explicit scoping of logical variables

Using the concept of scope a careful relabeling of graffiti nodes in Listing 4 without changing the meaning of the RDF Surfaces graph is possible.

```
1  @prefix log: <http://www.w3.org/2000/10/swap/log#> .
2  @prefix : <https://example.org/ns#> .
3
4  (_:x _:y) log:onNegativeSurface {
5      _:x a :Article .
6      _:y :reviewed _:x .
7      (_:y) log:onNegativeSurface {
8          _:y :publishes _:x .
9      } .
10 } .
```

Listing 6: A relabeled version of Listing 4 with the same meaning, using the scoping of graffiti nodes.

In Listing 6, a graffiti node with label `_:y` appears on lines 4 and 7 , with corresponding blank nodes on lines 6 and 8. The question remains: which graffiti nodes do the blank nodes on lines 4 and 7 refer to? RDF Surfaces uses the following convention:

Convention. Blank nodes refer to graffiti nodes with the same label on the closest parent surface.

In our example, this would imply that the blank node with label `_:y` on line 6 refers to the graffiti node with the same label declared on line 4, and the blank node with label `_:y` on line 8 refers to the graffiti node with the same label declared on line 7. This convention makes the FOL translation of Listing 6 equal (modulo renaming of variabes) to Listing 4.

The binding of blank nodes that are *not* mentioned in any graffiti is implicit and becomes 'existentially closed' on the top-level (default) positive surface. In this sense, RDF Surfaces mimic the behavior of plain RDF. Well-known ambiguities for quantification can be avoided [27]. For example, the design documents for N3Logic added syntactical features, such as `@forAll` and `@forSome` for quantification, but because graphs have no order in RDF, ambiguity exists in which N3 formulas are quantified by these quantifiers [3]. In RDF Surfaces, quantification is unambiguous using mandatory graffiti nodes with scope.

4.5. Explicit negation

RDF Surfaces provides a scope, a boundary, within positive or negative facts can be stated. For instance, to express that an RDF triple in an RDF Surfaces graph does not have a particular property, we can state (Listing 7):

```
1  @prefix log: <http://www.w3.org/2000/10/swap/log#> .
2  @prefix : <https://example.org/ns#> .
3
4  ( ) log:onNegativeSurface {
5      :WOS :indexed :JournalABC .
```

```
6  } .
```

Listing 7: An RDF Surfaces graph with an explicit negation with meaning "WOS did not index Journal-ABC."

The meaning of Listing 7 is: "It is not the case that `:WOS :indexed :JournalABC`". If this RDF Surfaces graph would contain any triples stating that `:WOS :indexed :JournalABC`, or if this could be inferred by deduction, it would be in conflict with the stated negation and thus create a contradiction. The RDF Surfaces data model enables a direct translation to classical negation without requiring competing notions of negation across different layers of the Semantic Web Stack (such as in OWL-DL ontologies and rules in Notation3).

Explicit negation is distinct from NAF and SNAF . The latter is available in Notation3 by the `log:notIncludes` builtin predicate[8]. Both NAF and SNAF styles of negation make statements about information that cannot be derived from a knowledge base. NAF requires assumptions about the Web in general, thus implying complete knowledge about missing information. SNAF requires assumptions about missing triples within the boundaries of an RDF source. In contrast, explicit negation does not require any assumptions about missing information. Instead, RDF Surfaces can explicitly state negative information or derive explicit negative information.

4.6. Disjunctions

A disjunction can be constructed using a combination of negations with conjunctions given that $A \vee B \equiv \neg(\neg A \wedge \neg B)$. Listing 8 provides a fictional example of such disjunction by adding nested negative surfaces in a negative surface.

```
1  @prefix log: <http://www.w3.org/2000/10/swap/log#> .
2  @prefix : <https://example.org/ns#> .
3
4  (_:x) log:onNegativeSurface {
5
6      ( ) log:onNegativeSurface {
7          _:x a :JournalArticle .
8      } .
9
10     (_:y) log:onNegativeSurface {
11        _:x a :Preprint .
12        _:y :reviewed _:x.
13     } .
14 } .
```

Listing 8: Adding nested negative surfaces in a negative surface creates a disjunction. The RDF Surfaces graph means, "Everything is a journal article, or a preprint that was also reviewed by someone."

Expressed in a symbolic form, this becomes:

$$\neg\left(\exists x : \neg\langle \texttt{x a :JournalArticle}\rangle \wedge \neg\left(\exists y : \langle \texttt{x a :Preprint}\rangle \wedge \langle \texttt{y :reviewed x}\rangle\right)\right)$$

which is equivalent to:

[8] https://w3c.github.io/N3/reports/20230703/builtins.html#log:notIncludes

$$\forall x : \langle \texttt{x a :JournalArticle} \rangle \vee (\langle \texttt{x a :Preprint} \rangle \wedge \exists y : \langle \texttt{y :reviewed x} \rangle) .$$

Listing 8 can be read as "Everything is a journal article, or a preprint that was also reviewed by someone."

Disjunctions can also be added to the antecedent and consequent of an implication. That is, both forms are possible:

$$\forall x : (\langle \texttt{x a :Journal} \rangle \vee \langle \texttt{x a :Book} \rangle) \rightarrow \langle \texttt{x a :Publication} \rangle$$

and

$$\forall x : \langle \texttt{x a :Publication} \rangle \rightarrow (\langle \texttt{x a :Journal} \rangle \vee \langle \texttt{x a :Book} \rangle) .$$

The latter implication could be written in RDF Surfaces using Listing 9.

```
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix : <https://example.org/ns#> .

( _:X ) log:onNegativeSurface {
    _:X a :Publication .

    ( ) log:onNegativeSurface {
        _:X a :Journal .
    } .

    ( ) log:onNegativeSurface {
        _:X a :Book .
    } .
} .
```

Listing 9: An RDF Surfaces graph with a disjunction in the conclusion with meaning: "Everything that is a publication is a journal or a book."

Disjunctions in the consequent of an implication is a feature of RDF Surfaces unavailable in the semantics of Notation3.

5. Examples

Using the definitions and syntax that were introduced in Section 4, we can apply RDF Surfaces to the use cases that were presented in Section 2.

5.1. Scholarly communication

For the scholarly communication use case, we envision a scholarly network of researchers who share their preferences when searching for publication venues. These preferences are preference documents that are serialized as N3S. These policies can contain information on what preference is regarded as positive, what certainly is not the case or preferences that are only "true" under some conditions. The latter takes the form of an implication. Section 2.1 provides an example of such a researcher preference:

– *Researcher X preferences*: Researcher X prefers preprints in a subject repository, a journal that does not charge APC costs, or a journal indexed in WOS.

One way to express this preference in FOL is by using disjunctions in the antecedent of an implication, as demonstrated in Equation (5).

$$\begin{aligned}\forall \mathtt{v} : \Big(& \langle \mathtt{v\ a\ :SubjectRepository} \rangle \vee \\ & (\langle \mathtt{v\ a\ :Journal} \rangle \wedge \neg \langle \mathtt{v\ :charges\ :APC} \rangle) \vee \\ & (\langle \mathtt{v\ a\ :Journal} \rangle \wedge \langle \mathtt{:WOS\ :indexed\ x} \rangle) \Big) \\ & \longrightarrow \langle \mathtt{:ResearcherX\ :prefers\ v} \rangle .\end{aligned} \tag{5}$$

In Equation (5), the universal quantified variable v stands for a publication venue. When the conditions hold for v, it is a preference for researcher X.

An alternative way to formulate this preference with the same meaning can be seen in Equation (6).

$$\begin{aligned}&\Big(\forall \mathtt{v} : \langle \mathtt{v\ a\ :SubjectRepository} \rangle \quad \rightarrow \langle \mathtt{:ResearcherX\ :prefers\ v} \rangle\Big) \wedge \\ &\Big(\forall \mathtt{v} : (\langle \mathtt{v\ a\ :Journal} \rangle \wedge \neg \langle \mathtt{v\ :charges\ :APC} \rangle) \quad \rightarrow \langle \mathtt{:ResearcherX\ :prefers\ v} \rangle\Big) \wedge \\ &\Big(\forall \mathtt{v} : (\langle \mathtt{v\ a\ :Journal} \rangle \wedge \langle \mathtt{:WOS\ :indexed\ v} \rangle) \quad \rightarrow \langle \mathtt{:ResearcherX\ :prefers\ v} \rangle\Big).\end{aligned} \tag{6}$$

We made here use of the fact that $(A \rightarrow C) \wedge (B \rightarrow C) \equiv (A \vee B) \rightarrow C$. This alternative formulation has the advantage that formula Equation (6) does not require the double nesting of all disjunction elements when serialized as N3S. New preference options can be added to a RDF Surfaces preferences document by appending a new implication pattern rather than inserting a new RDF Surfaces statement into an existing disjunction. The translation of the preferences of researcher X, as stated in Equation (6), into RDF Surfaces is provided in Appendix B.

The same procedure can be done for the departmental Y preferences that were stated as follows:

– *Department Y preferences*: All publication venues must be journals that are indexed in WOS.

A direct translation into FOL follows in Equation (7).

$$\forall \mathtt{v} : (\langle \mathtt{v\ a\ :Journal} \rangle \wedge \langle \mathtt{:WOS\ :indexed\ v} \rangle) \rightarrow \langle \mathtt{:DepartmentY\ :prefers\ v} \rangle) \tag{7}$$

The RDF Surfaces version of Equation (7) is available in Appendix C.

Publishing venue information is inherently decentralized. Sources that describe journal information can include information such as journal homepages, publisher's websites, library databases, and indexing services, such as Web of Science, to name a few. Establishing a common method for creating negative facts becomes essential in such an environment. For instance, it is quite common for journals to charge APC costs. That a particular journal does not charge APC costs is a negative fact that benefits the budgets of many researchers. Absence of information about APC charges should not be equated with implicit assumptions that no APC costs are charged. In Equation (8), we symbolize a summary of publishing venue facts that state:

– ABC, DEF, and GHI are journals.
– JKL is a subject repository.

– ABC charges APC costs, but GHI does not charge these costs.
– ABC and DEF are indexed in the WOS database, but GHI is not.

$$\begin{aligned}&\langle \texttt{:ABC a :Journal}\rangle \wedge \langle \texttt{:DEF a :Journal}\rangle \wedge \langle \texttt{:GHI a :Journal}\rangle \wedge \\ &\langle \texttt{:JKL a :SubjectRepository}\rangle \wedge \\ &\langle \texttt{:ABC :charges :APC}\rangle \wedge \neg\langle \texttt{:GHI :charges :APC}\rangle \wedge \\ &\langle \texttt{:WOS :indexed :ABC}\rangle \wedge \langle \texttt{:WOS :indexed :DEF}\rangle \wedge \neg\langle \texttt{:WOS :indexed :GHI}\rangle\end{aligned} \tag{8}$$

The translation of Equation (8) into RDF Surfaces is provided in Appendix D.

Facts about publishing venues can be publicly shared. This allows Web agents to harvest this data and match them against researcher and department policies. Consequently, this process can be used to make informed publishing recommendations as a service. For instance, when a Web agent gets hold of the policies expressed in Equation (5), Equation (6), Equation (7), and Equation (8), a logical query can be posed that asks if journal ABC is a researcher and department preference, by adding the negation of $\langle$`:ABC a :ResearcherPreference`$\rangle \wedge \langle$`:ABC a :DepartmentPreference`$\rangle$ to the knowledge base and test if this leads to a contradiction.

At https://w3c-cg.github.io/rdfsurfaces/demonstrator/, an experimental RDF Surfaces implementation is available to test the examples provided in this section. The RDF Surfaces from this section can be consulted as one resource at https://w3c-cg.github.io/rdfsurfaces/examples/scholary_publication.n3.

For illustration purposes, we assume a Web agent wants to consult these preferences and adds a negated query at the end:

```
( ) log:onNegativeSurface {
    :ABC a :DepartmentPreference .
    :ABC a :ResearcherPreference .
} .
```

The negation:

$$\neg(\langle \texttt{:ABC a :ResearcherPreference}\rangle \wedge \langle \texttt{:ABC a :DepartmentPreference}\rangle)$$

leads to a contradiction. Therefore, the Web agent can conclude that journal ABC is a researcher and department preference.

Three key observations can be drawn from the scholarly communication use case. First, our scholarly communication scenario demonstrates that sharing preference information on the Web is feasible using a Web language that extends RDF —- specifically, RDF triples augmented with Hayes triples. Preferences expressed in N3S not only retain the expressivity of RDF but also enable reasoning over these triples. This capability would not be readily achievable if preference information were expressed as SPARQL queries. While SPARQL queries can be executed against a knowledge base, in the absence of a knowledge base, it is not possible to query the SPARQL query itself and infer information from it. For instance, given Equation (7) and the hypothesis that some venue is a journal but not a preference of department Y, it should logically follow that this journal is not indexed in WOS.

Second, the predicate `:prefers` and its negation could embody a form of permission and prohibition. If no conflicts arise, the suggested venue is 'permitted' and does not violate the researcher preferences. Conversely, if conflicts exist, the suggested venue is 'prohibited' and violates the research preferences. Given a concept of 'permitted', we are not required to introduce a separate predicate for 'not-permitted'.

Third, when authoring an N3S document to represent a researcher's preferences, one might envision comparing it with departmental documents to identify inconsistencies. This process would enhance conflict resolution capabilities through a direct translation from researcher and departmental preferences to FOL.

These three features of RDF Surfaces for modeling policy languages beyond the scope of scholarly communication were highly influential in shaping our research. An example of such an approach is from the Flemish SHARCS project[9] where a demonstrator was created to make a subset of ODRL policies actionable by compiling them to RDF Surfaces.[10] A similar approach, applying RDF Surfaces to policy languages, was taken by Zhao and Zhao (2024).

5.2. Medicine Prescription

In the healthcare domain, we envision knowledge systems that include both positive and negative information about medications and policies on how these medications can be prescribed to patients with pre-existing conditions. These systems would consider that for example certain medications may cause allergic reactions when prescribed to patients. Consider a collection of medicines for a medical prescription use case: low-dosage aspirin, high-dosage aspirin, and beta-blockers. A low dosage of aspirin is an effective treatment for a fever. In turn, a high dosage of aspirin or a prescription of beta-blockers are both considered to be an effective treatment for acute myocardial infarction. However, both high and low dosages of aspirin may only be prescribed when a patient is known not to be allergic to aspirin. Aspirin can also not be prescribed when a patient has active peptic ulcer disease. Beta-blockers, on the other hand, cannot be prescribed when a patient suffers from severe asthma.

We assume the following expert policies in terms of medicine prescription:

- A low aspirin dosage can be prescribed for a patient with a fever and with neither an allergy to aspirin nor active peptic ulcer disease.
- For a patient with acute myocardial infarction but without an allergy to aspirin or active peptic ulcer disease, a high dosage of aspirin can be prescribed.
- For a patient with acute myocardial infarction but without severe asthma or chronic obstructive pulmonary disease, beta-blockers can be prescribed.

The first policy indicates that if a given patient has a fever (A), one of the following must hold: the patient has an aspirin allergy (B), has active peptic ulcer disease (C), or is prescribed a low dosage of aspirin (D). This can be inferred from the logical equivalence $\neg(A \wedge \neg B \wedge \neg C \wedge \neg D) \equiv (A \wedge \neg B \wedge \neg C) \rightarrow D \equiv A \rightarrow (B \vee C \vee D)$. Any of these three forms could be used to translate the policy in a FOL format. In Equation (9), we use a symbolic form that closely matches the method applied in Equation (6) for researcher preferences.

$$\begin{aligned} \forall \texttt{x} : \Big(& \langle \texttt{x :has :Fever} \rangle \wedge \\ & \neg \langle \texttt{x :has :AllergyForAspirin} \rangle \wedge \\ & \neg \langle \texttt{x :has :ActivePepticUlcerDisease} \rangle \Big) \\ & \longrightarrow \langle \texttt{x :isPrescribed :aspirinLowDose} \rangle \end{aligned} \tag{9}$$

The translation for the other two pieces of expert knowledge can be constructed analogously. The N3S version of these policies is provided in Appendix E.

With the expert knowledge encoded in RDF Surfaces, we are able to infer the correct prescription for a patient, given their medical condition as well as allergies (or lack thereof). We consider the following two

[9] https://www.imec-int.com/en/research-portfolio/sharcs

[10] https://github.com/eyereasoner/Notation3-By-Example/tree/main/examples/sharcs-odrl

patients, Ann and Joe. Ann has a fever and does not have an allergy to aspirin or an active peptic ulcer disease. Joe suffers from acute myocardial infarction and is allergic to aspirin. However, he does not have active peptic ulcer disease, severe asthma, or chronic obstructive pulmonary disease. These facts can be translated into symbolic form as demonstrated by Equation (10) and Equation (11) with a translation in RDF Surfaces in Appendices F and G.

$$\begin{aligned} &\langle \texttt{:Ann :has :Fever} \rangle \wedge \\ &\neg\langle \texttt{:Ann :has :AllergyForAspirin} \rangle \wedge \\ &\neg\langle \texttt{:Ann :has :ActivePepticUlcerDisease} \rangle \end{aligned} \tag{10}$$

$$\begin{aligned} &\langle \texttt{:Joe :has :AcuteMyocardialInfarction} \rangle \wedge \\ &\langle \texttt{:Joe :has :AllergyForAspirin} \rangle \wedge \\ &\neg\langle \texttt{:Joe :has :ActivePepticUlcerDisease} \rangle \wedge \\ &\neg\langle \texttt{:Joe :has :SevereAsthma} \rangle \wedge \\ &\neg\langle \texttt{:Joe :has :ChronicObstructivePulmonaryDisease} \rangle \end{aligned} \tag{11}$$

The RDF Surfaces from Appendices E, F, and G can be consulted as one resource at https://w3c-cg.github.io/rdfsurfaces/examples/medication_prescription.n3. Using the experimental RDF Surfaces implementation of Section 5.1, this resource can be consulted to test if we can prescribe a high-dose aspirin for patient Ann and a beta blocker for patient Joe. The procedure is similar to the one in the previous section. We can add to the stated RDF Surfaces a negative query and check for a contradiction:

```
() log:onNegativeSurface {
    :Ann :isPrescribed :aspirinLowDose .
    :Joe :isPrescribed :betaBlocker .
} .
```

If a contradiction is found, then we can conclude that patient Ann can be prescribed a low aspirin dosage, and patient Joe can be prescribed a beta-blocker.

6. Discussion and future road map

This paper discussed the need for an expressive Semantic Web Logic extending RDF. We especially identified classical negation as crucial for our use cases in scholarly communication and healthcare. Therefore, we proposed RDF Surfaces, provided an abstract and a concrete syntax and the semantics for it, and showed how our use cases could benefit from it. The authors are currently experimenting with implementing RDF Surfaces in several reasoners . The first results in this direction will be discussed below. In 2022, a W3C Community group was formed to define the semantics and discuss the general requirements for implementations and Web logic.[11,12] Of course, this new logic comes with many open challenges, and this vision paper is only the starting point of a longer endeavor. Below, we list the most important challenges ahead and briefly discuss possible ways to solve them.

[11] https://www.w3.org/community/rdfsurfaces/
[12] https://w3c-cg.github.io/rdfsurfaces/

6.1. Negation on the Web

With RDF Surfaces, we provided logic and syntax to express negative information on the Web using RDF. To implement negation, we extended the RDF model with surfaces that describe a (possibly empty) set of negated triples. These triples are quantified by adding graffiti nodes that act as existentially quantified variables. Combining these features with the default conjunction of triples under simple entailment semantics provides the full expressivity of FOL in RDF. Ambiguities in the quantification can be avoided by providing an explicit scope for graffiti nodes embedded in a surface. We demonstrated how to solve real-world use cases in scholarly communication and healthcare that contain shared negative information, disjunctions, and implications in a decentralized setting. Using RDF Surfaces, Hayes's BLOGIC vision can be realized in concrete implementations .

The Semantic Web is an endeavor to express the combined human knowledge irrespective of the computability of this knowledge. Consequently, we do not anticipate that any realistic Web reasoner will be capable of delivering a fully portable solution (P) accepting any FOL input –without applying restrictions to the expressivity of the language nor the entailments performed on it–, consistently providing complete (C) responses to any query, and invariably terminating (H) within a finite time frame. Demanding these (P)+(C)+(H) attributes simultaneously is infeasible related to the undecidability of FOL and forms an iron triangle: at most, only two of these features can be selected for any implementation. If we want a portable Web logic, feature (P) is unavoidable. Any portable Web reasoner can at best only choose between being always complete (C) or always halt (H) in a finite time. In practice, we do not assume that the worst-case scenarios will be the norm on the Web when applying RDF Surfaces to real-world use cases. In general, reasoners will not be able to provide answers for every possible input and every possible query. They will always be restricted in the type of problem sets that can be handled. In our experience, however, non-trivial applications of FOL can be built using our implementations in domains of scholarly communication and policy enforcement. The TPTP world demonstrates that the undecidability of FOL does not stop innovation and the availability of implementations that solve a wide variety of FOL (and even higher-order and modal logic) problems.

An interesting application of RDF Surfaces would be publishing the proof of every derivation. Rather than publishing what is a (derived) fact on the Web, Web agents could publish the proof of how they concluded these facts. For instance, in a scholarly communication setting, one would like to know why a journal was selected as a valid preference for a researcher. In a healthcare setting, one would trace why or why not a medication was provided to a patient.

We do not anticipate that every Web author should use FOL terminology to express knowledge and create logical formulas. We regard RDF Surfaces as a low-level language that can transport logic on the Web. Higher-level concepts are still required to create a compact serialization for human consumption (human editors). For instance, the meaning RDF `rdfs:subClassOf` predicate can be encoded in FOL terminology using a rule as specified in Equation (4). This is possible in RDF Surfaces because we build up RDF entailment from the ground up. RDF Surfaces does not rely on concepts such as classes. The meaning of the predicate `rdfs:subClassOf` can be explicitly defined as a rule in RDF Surfaces, which manifests as subsumption. Human editors would still write `rdfs:subClassOf` in their documents and refer to a library document containing the RDF Surfaces rules to handle concepts such as `rdfs:subClassOf` using FOL terminology. In this way, an editor can select a minimal set of RDFS or OWL constructs relevant to a given use case, or create a new FOL construct. A related use case is to translate RDF semantics directly to FOL, providing an interoperability layer between RDF and FOL reasoning systems that consume standard formats such as TPTP and SMT-LIB, an approach we discuss further in Section 6.4.[13]

Extensions to RDF Surfaces would certainly be helpful when expressing non-monotonic negation. For instance, in the scholarly communication use case, expressing which topics are not part of a database is not always feasible. Notation3 provides the capabilities for scoped negation as failure that might need

[13] https://smt-lib.org

to be combined with RDF Surfaces. Real-world use cases also need to provide input on which extended predicates are required for things that are not easily expressable in FOL, such as calculations, string manipulation, and list manipulation. This requires a delicate balance between the expressiveness of the language for general computation use cases and the formal semantics of classic FOL.

6.2. Syntax

A first attempt at a hosting language for RDF Surfaces led us to use the Notation3 syntax in the form of "RDF Surfaces in N3" (N3S), which only requires a reduced set of features. Notation3 was only used for its syntactical features – graph terms for creating a scope around negated triples and list terms for declaring graffiti nodes – but not its semantic features. Some authors of this paper are active participants in both the Notation3 and RDF Surfaces groups, where overlap and differences between both approaches are debated. One point of argument is the scoping of blank nodes that in Notation3 semantics is local to a graph term but in RDF Surfaces explicit within a negative surface. Another point is Peter Patel-Schneider's argument that a same-syntax extension of RDF to FOL necessarily leads to paradoxes due to self-referential statements [48]. RDF Surfaces does not rely on potential self-referential statements. By using N3 list terms in the subject position, negative surfaces inside an N3S document do not have an identifier and can not be referred to. But, this does not absolve us from logical paradoxes because these are very hard to avoid in highly expressive languages.[14]

6.3. Formal semantics

As RDF Surfaces is introduced as a logic, the obvious next step is to define it formally. We have already provided the syntax in this paper, but the semantics are only discussed briefly. The model-theoretic approach presented here aims to serve as a starting point towards a more robust formalization of the semantics. In order to obtain such a formalization, There are several possibilities that can be taken into consideration; Given that Pat Hayes' original idea of a BLOGIC was inspired by Peirce's existential graphs, we could base RDF Surfaces semantics on existential graphs and map them to this framework. This would make it easy to define a calculus for RDF Surfaces by following existing literature, e.g., Zeman [60] and Dau [11]. A difficulty, however, is that existential graphs do not have native support for constants. To formalize this, one would need to address this limitation, for example, by mimicking constants using relations in combination with existential quantification. Another possibility would be to base RDF Surfaces semantics on FOL, which is close to the approach we followed in this paper . An advantage of this approach would be that FOL is well understood regarding expressiveness and limitations. As RDF Surfaces extends RDF, RDF formalizations based on existential graphs or FOL would have the burden of proof that the semantics is in line with the one of RDF. In the case of a FOL formalization, this could be done following the work of De Bruijn et al. [12]. This last aspect – the fact that RDF Surfaces aim to extend RDF– makes a formalization extending RDF semantics [24] an interesting choice. It would make sense only to consider simple entailment and D-entailment as the semantics of RDFS could be modeled through RDF Surfaces axioms. A demonstration of such formal mapping of RDF Surfaces to FOL was provided by the authors in 2024 [5]. By mapping RDF Surfaces into FOL, RDF Surfaces is able to "borrow" the semantics of FOL. Entailment under these semantics coincides with RDF simple entailment and behaves as intended, as shown in [5]. Although RDF Surfaces can be mapped into FOL, its restriction is one of signature, not of logic: RDF Surfaces can express all FOL operators, but is limited in what it can apply them to. As plain RDF expresses information in the form of triples $\langle \boldsymbol{s}\ \boldsymbol{p}\ \boldsymbol{o} \rangle$, RDF Surfaces supports only one relation, the ternary relation $tr(\boldsymbol{s}, \boldsymbol{p}, \boldsymbol{o})$, and no functions. This includes the (lack of the) equality relation =. These compromises were necessary to construct RDF Surfaces on the foundation of RDF simple entailment, which lacks a built-in concept of equality and does not support predicates with higher arity. OWL DL offers `owl:sameAs`, and N3 offers `log:equalTo`. One way to incorporate equality into RDF Surfaces requires explicitly specifying the

[14] See Russell's paradox for an example https://en.wikipedia.org/wiki/Barber_paradox

meaning of equality in terms of added FOL formulas. Such formulas for N3 were provided by Tomaszuk [53] and can readily be translated into RDF Surfaces. Researching possible extensions of RDF Surfaces to FOL with equality and functions can be an area for further research.

6.4. Implementations

To be of use for the Web, RDF Surfaces should not only be a theoretical framework, but should be applicable in practice. We need implementations that can check and exchange each other's findings to achieve this. We thus need to agree on the explicit syntax and the N3S-based framework we introduced in this paper could be a starting point for that. Reasoners should use this as input and apply a calculus that is proven correct and – if possible – also complete according to the semantics. Currently, there are five experimental implementations based on our N3S-based syntax: *rs2fol* [43][5], which translates the N3S format to the TPTP format and uses the Vampire FOL theorem prover[15], *Latar* [25], whose calculus is inspired by existential graph reasoning, *EYE* [13], which is originally a Notation3 reasoner extended to support RDF Surfaces, *retina* [14] based on a rewrite of EYE for RDF Surfaces in Trealla and Scryer Prolog, and *Tension.js* [55] a Typescript implementation. The calculus applied is close to FOL reasoning. To ensure interoperability between these reasoners and encourage more implementors to support RDF Surfaces, shared test cases are needed to express the expected behavior of reasoning systems. A first repository providing such test cases is already available[16]. Still, the test infrastructure needs to become more fine-grained to check for different types of problems like simple parsing errors or the correct application of single inference steps. Use cases like the ones presented above will help generate more test cases and better understand the specific needs the reasoning should satisfy. They could help to decide on various design and implementation choices, such as optimizations for data storage and inferencing.

Employing different implementations is also a strategic approach to addressing ex falso quodlibet in RDF Surfaces documents. Our repository of test cases includes various consistency checks that these implementations must satisfy. Many more benchmarks for consistency checks can be found in the CADE ATP System Competition (CASC) repositories. There is no universal strategy for performing these checks, as there are many calculi for first-order logic FOL.

Addressing inconsistency repair is an entirely separate challenge. In our view, resolving inconsistencies will largely depend on human intervention and cannot be fully automated. One possible approach is to incorporate provenance information for RDF sources on the Web. For example, if RDF source A does not cause an inconsistency, but the combination of A and B does, we could classify RDF source A as more reliable than B.

The five implementations provided above offer a strong starting point for community-wide discussion on the desirable implementations of the proposed RDF Surfaces theory; rs2fol and Tension.js are implementations that adhere closer to classical FOL reasoning, the former translating RDF Surfaces N3S syntax to TPTP format commonly used in FOL theorem provers. By contrast, EYE reasons directly upon the Notation3 syntax, as proposed in Section 4.2. retina caters to Prolog-oriented use cases, whereas Latar follows the philosophy of existential graphs, in line with the original BLOGIC vision. These implementations showcase the strength and diversity of RDF Surfaces.[17]

6.5. Relation to other Web logics and standards

Alongside the formalization of the logic, its relationship to existing frameworks should be investigated. Since RDF Surfaces supports classical negation, existential quantification, and conjunction, it is likely that it can express FOL, though this claim must be proven. It is particularly interesting to explore how we can

[15] https://vprover.github.io

[16] https://github.com/eyereasoner/rdfsurfaces-tests

[17] Since this paper was first submitted in 2024, a sixth implementation has appeared, based on SMT-LIB: https://patrickhochstenbach.net/git/phochste/juliett.

express OWL DL and its profiles [45] in RDF Surfaces, but other common Web frameworks should also be considered.

The relationship to RDF Surfaces is clear for RDF simple entailment, as the latter is defined as an extension of the former. However, the relationship is less clear for RDFS or rule formats like Notation3 Logic [59] or SWRL [29]. And it becomes even more difficult regarding WC3 recommendations like SHACL [36] or SPARQL [19], which both support non-monotonic features. In its current form, as presented in this paper, RDF Surfaces cannot cope with non-monotonicity. Still, other aspects, like querying with recursive property paths, can be covered in RDF Surfaces.

6.6. Extensions

The last aspect mentioned, the current inability to support non-monotonic behavior, illustrates another direction for future research: it could be possible and maybe even necessary to support concepts, such as negation as failure or closed predicates [46], to be suitable to fully support use cases like the ones introduced in this paper. Notation3 and SPARQL come with built-in or filter predicates, which makes it easy to, e.g., perform operations on strings or sum up two integers. As RDF Surfaces is introduced as a logic facilitating practical use cases, it would be beneficial to include such predicates. As a third, but – given the variety of use cases in the Web – certainly not least extension, we believe that support for unasserted triples like it is proposed in RDF-star [20] and unasserted graphs as they exist in N3 would be beneficial for many use cases, like for example the exchange of provenance knowledge and the reasoning about it. To also support the latter, this support for unasserted knowledge should come with a mechanism that could assert it in the case of provenance, for example, if we decide to trust a source.

Of course, this section only provides a few examples of future development, and we hope our readers already have their own visions here. In that sense, we are very curious to see what the future brings, and we plan to work on these and other research to further RDF actively.

7. Conclusion

RDF Surfaces is a concrete realization of Pat Hayes' BLOGIC vision of portable Web logic based on FOL. In this vision paper, we provide a translation of BLOGIC into a concrete RDF syntax and semantics with FOL expressivity . RDF Surfaces, with its N3S syntax, deviates from Hayes' original "annotated Turtle" and is a sub-language of Notation3 with a reduced set of features: N3 list terms to represent a set of graffiti blank nodes and graph terms to represent a (possibly empty) set of triples. Using these extensions to RDF and adding a surface type as a predicate, the full expressivity of FOL is available using simple entailment. The applicability of RDF Surfaces was demonstrated in two use cases, one from scholarly communication and one from healthcare.

The benefits of RDF Surfaces as Web logic becomes evident in decentralized networks that need to exchange data and the logic behind the data. In our scholarly communication use-case, no facts were published by the researcher and institute, but conditions on future facts in the form of preferences. In the healthcare use case, the conditions on which medication would be applicable for (future) a patient were exchanged. In a decentralized network, Web agents should agree on a Web logic to analyze these preferences and conditions for possible triples that can become facts.

Another use case that could benefit from RDF Surfaces is rights policies that need to exchange information about the permissions and prohibitions to access and use information. With RDF Surfaces, policy documents could not only make this information machine actionable but, in addition, in a decentralized setting, contradictions can be discovered between policies at the authoring time (and not at run time). The computer can say "no" to a policy author who does not have to wait for "the customer at the door" to find out something is wrong.

Publishing negative information is part of human activity; it is a part of commercial activity. The soft drink industry's "This drink contains no sugar" marketing statement is an explicit negative fact. For a

consumer, this explicit negative information could be the reason to buy this drink. One option could be to add many negative predicates or classes to Web ontologies to publish this negative information. We believe this would obfuscate that classical negation is intended but not expressed and only adds to Hayes' "diamond of confusion."

Computation complexity is an issue, and as Web logic can be used for different purposes, many levels of complexity can be imagined. Adding the undecidability of FOL queries can potentially require vast computer resources or even run forever without providing an answer. Using Sowa argumentation [50], we do not assume that worst-case scenarios will be the norm on the Web. The theorems of Whitehead and Russell's *Principia Mathematica* could already be proven in the 1960s by vacuum-tube machines. We advocate with RDF Surfaces to choose for sharing information and the logic behind this information using an RDF syntax, which makes as few assumptions as possible to create FOL expressivity. With FOL, we have semantics that is well understood.

Our research is still young and requires follow-up research to strengthen our claims. Implementations must demonstrate that values can be added to the Semantic Web using FOL expressivity for real-world use cases. Machines must cooperate and demonstrate, using proofs, why particular derivations were made and their reasoning for providing a conclusion. As Web citizens, we can clearly state what is and what is not using RDF Surfaces.

8. Acknowledgements

This work is partly funded by the Andrew W. Mellon Foundation (grant number: 1903-06675), SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10), and the FWO Project FRACTION (Nr. G086822N). The authors would like to thank the W3C RDF Surfaces community group for joint discussions on the requirements for Web logic. The authors would also like to thank Jesse Wright and Ieben Smessaert for creating a Web version of the EYE reasoner and RDF Surfaces as a Web application.

References

[1] S. Abiteboul, R. Hull and V. Vianu, *Foundations of databases*, Vol. 8, Addison-Wesley Reading, 1995.
[2] R. Angles and C. Gutierrez, Negation in SPARQL, *arXiv preprint arXiv:1603.06053* (2016).
[3] D. Arndt, T. Schrijvers, J. De Roo and R. Verborgh, Implicit quantification made explicit: How to interpret blank nodes and universal variables in Notation3 Logic, *Journal of Web Semantics* **58** (2019), 100501. doi:10.1016/j.websem.2019.04.001.
[4] D. Arndt and P.-A. Champin, Notation3 Semantics, Technical Report, W3C Commuity Group. https://w3c.github.io/N3/reports/20230703/semantics.html.
[5] D. Arndt, J. De Roo, P. Hochstenbach, R. Martens, F. Ongenae and M. van Noort, RDF Surfaces as a First-Order Language for the Semantic Web, in: *Rules and Reasoning*, S. Kirrane, M. Šimkus, A. Soylu and D. Roman, eds, Springer Nature Switzerland, pp. 200–216. ISBN 978-3-031-72407-7. doi:10.1007/978-3-031-72407-7$_1$5.
[6] S. Battle, A. Bernstein, B. Grosof, M. Gruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, D. McGuiness, J. Su and S. Tabet, Semantic Web Services Language (SWSL), 2005.
[7] D. Beckett, T. Berners-Lee, E. Prud'hommeaux and G. Carothers, RDF 1.1 Turtle, 2014.
[8] T. Berners-Lee, D. Karger, L.A. Stein, R. Swick and D. Weitzner, SWELL - Semantic Web Logic Language, 2000.
[9] T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf and J. Hendler, N3Logic: A logical framework for the World Wide Web, *Theory and Practice of Logic Programming* **8**(3) (2008), 249–269. doi:10.1017/S1471068407003213.
[10] C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler and M. Smith, OWL 2 Web Ontology Language, 2012, http://www.w3.org/TR/owl2-syntax/.
[11] F. Dau, *The logic system of concept graphs with negation: And its relationship to predicate logic*, Vol. 2892, Springer Science & Business Media, 2003. doi:10.1007/b94030.
[12] J. de Bruijn and S. Heymans, Logical Foundations of (e) RDF (S): Complexity and Reasoning, *Lecture Notes in Computer Science* **4825** (2007), 86. doi:10.1007/978-3-540-76298-0_7.
[13] J. De Roo, Euler Yet another proof Engine, GitHub. https://github.com/eyereasoner/eye.
[14] J. De Roo, retina, GitHub. https://github.com/KNowledgeOnWebScale/retina.

[15] T. Eiter, G. Ianni and T. Krennwallner, *Answer set programming: A primer*, Springer, 2009. doi:10.1007/978-3-642-03754-2_2.
[16] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer and H. Tompits, Combining answer set programming with description logics for the Semantic Web, *Artificial intelligence* **172**(12–13) (2008), 1495–1539. doi:10.1016/j.artint.2008.04.002.
[17] B. Esteves, H.J. Pandit and V. Rodríguez-Doncel, ODRL profile for expressing consent through granular access control policies in SOLID, in: *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, IEEE, 2021, pp. 298–306. doi:10.1109/EuroSPW54576.2021.00038.
[18] M. Genesereth and R.E. Fikes, Knowledge Interchange Format Version 3.0, 1992.
[19] S. Harris and A. Seaborne, SPARQL 1.1 Query Language, W3C Recommendation, W3C, 2013, https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.
[20] O. Hartig, P.-A. Champin, G. Kellogg and A. Seaborne, RDF-star and SPARQL-star, 2023, https://w3c.github.io/rdf-star/cg-spec/editors_draft.html.
[21] S. Hawke and A. Polleres, RIF In RDF (Second Edition), 2009. https://www.w3.org/TR/rif-in-rdf/.
[22] P. Hayes, Re: [EXT] Re: Upper ontologies from phayes@ihmc.us on 2021-01-20 (semantic-web@w3.org from January 2021).
[23] P. Hayes, BLOGIC. (ISWC 2009 Invited Talk), 2009. https://www.slideshare.net/PatHayes/blogic-iswc-2009-invited-talk.
[24] P. Hayes and P.F. Patel-Schneider, RDF 1.1 Semantics, 2014.
[25] P. Hochstenbach, Latar, GitHub. https://github.com/phochste/Latar.
[26] A. Hogan, Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora. https://aidanhogan.com/docs/thesis/thesis-centred.pdf.
[27] A. Hogan, M. Arenas, A. Mallea and A. Polleres, Everything you always wanted to know about blank nodes, *Journal of Web Semantics* **27** (2014), 42–69.
[28] I. Horrocks, P.F. Patel-Schneider and F. Van Harmelen, From SHIQ and RDF to OWL: the making of a Web Ontology Language, *Journal of Web Semantics* **1**(1) (2003), 7–26. doi:10.1016/j.websem.2003.07.001. https://linkinghub.elsevier.com/retrieve/pii/S1570826803000027.
[29] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean, SWRL: A Semantic Web Rule Language Combining OWL and RuleML (2004).
[30] R. Iannella and S. Villata, ODRL Information Model 2.2, 2018.
[31] Information technology — Common Logic (CL) — A framework for a family of logic-based languages, Standard, International Organization for Standardization, Geneva, CH, 2018. https://www.iso.org/standard/66249.html.
[32] M.G. Kebede, G. Sileno and T. Van Engers, A critical reflection on ODRL, in: *International Workshop on AI Approaches to the Complexity of Legal Systems*, Springer, 2018, pp. 48–61.
[33] G. Kellogg, rdf-n3. https://github.com/ruby-rdf/rdf-n3.
[34] B. Ketsman and C. Koch, Datalog with Negation and Monotonicity, in: *23rd International Conference on Database Theory (ICDT 2020)*, Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2020.
[35] M. Kifer, Rule Interchange Format: The Framework, in: *Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems*, 2008, pp. 1–11.
[36] H. Knublauch and D. Kontokostas, Shapes Constraint Language (SHACL), Recommendation, World Wide Web Consortium (W3C), 2017. https://www.w3.org/TR/shacl/.
[37] H. Knublauch, J. Hendler and I. Kingsley, SPIN - SPARQL Inferencing Notation, 2013.
[38] M.-R. Koivunen and E. Miller, W3C Semantic Web Activity, 2001.
[39] D. Krach, RDFLib. https://github.com/RDFLib/rdflib.
[40] O. Lassila, Web metadata: A matter of semantics, *IEEE Internet Computing* **2**(4) (1998), 30–37.
[41] O. Lassila and R. Swick, Resource Description Framework (RDF) Model and Syntax Specification.
[42] C. Lefèvre and P. Nicolas, A First Order Forward Chaining Approach for Answer Set Computing, in: *Logic Programming and Nonmonotonic Reasoning*, E. Erdem, F. Lin and T. Schaub, eds, Springer, pp. 196–208. ISBN 978-3-642-04238-6. doi:10.1007/978-3-642-04238-6_18.
[43] R. Martens, rs2fol, GitHub. https://github.com/RebekkaMa/rs2fol.
[44] J. Mei and H. Boley, Interpreting SWRL Rules in RDF Graphs **151**(2) (2006), 53–69. doi:10.1016/j.entcs.2005.07.036. https://www.sciencedirect.com/science/article/pii/S1571066106003367.
[45] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue and C. Lutz, OWL 2 Web Ontology Language Profiles (Second Edition), 2012, https://www.w3.org/TR/owl2-profiles/.
[46] N. Ngo, M. Ortiz and M. Simkus, Closed predicates in description logics: Results on combined complexity, in: *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2016. doi:10.5555/3032027.3032056.
[47] G. O'Regan, Computability and Decidability, in: *Mathematical Foundations of Software Engineering: A Practical Guide to Essentials*, G. O'Regan, ed., Springer Nature Switzerland, pp. 229–239. ISBN 978-3-031-26212-8. doi:10.1007/978-3-031-26212-8_14.
[48] P.F. Patel-Schneider, RDF: Back to the Graph, 2009.

[49] J. Sowa, *Knowledge Representation: Logical, Philosophical and Computational Foundations*, Brooks/Cole, 2000. ISBN 0-534-94965-7.
[50] J.F. Sowa, Fads and fallacies about logic, *IEEE Intelligent Systems* **22**(2) (2007), 84–87.
[51] U. Straccia and G. Casini, A Minimal Deductive System for RDFS with Negative Statements, arXiv, 2022. http://arxiv.org/abs/2202.13750.
[52] T. Tammet and G. Sutcliffe, Combining JSON-LD with First Order Logic, in: *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, pp. 256–261, ISSN: 2325-6516. doi:10.1109/ICSC50631.2021.00051.
[53] D. Tomaszuk, Inference rules for RDF(S) and OWL in N3Logic, arXiv. http://arxiv.org/abs/1601.02650.
[54] D. Tsarkov, A. Riazanov, S. Bechhofer and I. Horrocks, Using Vampire to Reason with OWL, *Lecture notes in Computer Science* **3298** (2004), 471–485. doi:10.1007/978-3-540-30475-3_33.
[55] J. Van Herwegen, Tension.js, GitHub. https://github.com/joachimvh/tension.js.
[56] R. Verborgh, n3. https://www.npmjs.com/package/n3.
[57] G. Wagner, Web Rules Need Two Kinds of Negation, in: *Principles and Practice of Semantic Web Reasoning*, Vol. 2901, 2003, pp. 33–50. doi:10.1007/978-3-540-24572-8_3.
[58] G. Wagner, C.V. Damasio and G. Antoniou, Towards a general Web rule language, *International Journal of Web Engineering and Technology* **2**(2) (2005), 181–206. doi:10.1504/IJWET.2005.008483.
[59] W.V. Woensel, D. Arndt, P.-A. Champin, D. Tomaszuk and G. Kellogg, Notation3 Language, 2023, https://w3c.github.io/N3/reports/20230703/.
[60] J.J. Zeman, The graphical logic of CS Peirce, PhD thesis, The University of Chicago, 1964.
[61] FIPA RDF Content Language Specification, 2001. http://www.fipa.org/specs/fipa00011/XC00011B.html.

Appendix A EBNF Grammar of N3S

```
n3sDoc ::= statement* EOF
statement
        ::= directive
          | triples '.'
directive
        ::= prefixID
          | base
          | sparqlPrefix
          | sparqlBase
prefixID ::= '@prefix' PNAME_NS IRIREF '.'
base ::= '@base' IRIREF '.'
sparqlBase
        ::= BASE IRIREF
sparqlPrefix
        ::= PREFIX PNAME_NS IRIREF
triples ::= subject predicateObjectList
          | blankNodePropertyList predicateObjectList?
          | hayestriple
hayestriple ::= graffiti surfacetype hayesgraph
graffiti ::= '(' BlankNode* ')'
surfacetype ::= predicate
hayesgraph ::= formula
predicateObjectList
        ::= verb objectList ( ';' ( verb objectList )? )*
objectList
        ::= object ( ',' object )*
verb ::= predicate
          | 'a'
```

```
subject ::= iri
          | BlankNode
          | collection
predicate
        ::= iri
object ::= iri
          | BlankNode
          | collection
          | blankNodePropertyList
          | literal
literal ::= rdfLiteral
          | NumericLiteral
          | BooleanLiteral
blankNodePropertyList
        ::= '[' predicateObjectList ']'
collection
        ::= '(' object* ')'
formula ::= '{' ( triples ( '.' triples )* '.'? )? '}'
rdfLiteral
        ::= String ( LANGTAG | '^^' iri )?
iri ::= IRIREF
          | prefixedName
prefixedName
        ::= PNAME_NS
          | PNAME_LN
_ ::= COMMENT
          | WS
         /* ws: definition */

<?TOKENS?>

COMMENT ::= '#' [^#xa#xd]*
NumericLiteral
        ::= INTEGER
          | DECIMAL
          | DOUBLE
BooleanLiteral
        ::= 'true'
          | 'false'
String ::= STRING_LITERAL_QUOTE
          | STRING_LITERAL_SINGLE_QUOTE
          | STRING_LITERAL_LONG_SINGLE_QUOTE
          | STRING_LITERAL_LONG_QUOTE
BlankNode
        ::= BLANK_NODE_LABEL
          | ANON
IRIREF ::= '<' ( [^<>"{}|^`\#x0000-#x0020] | UCHAR )* '>'
PNAME_NS ::= PN_PREFIX? ':'
PNAME_LN ::= PNAME_NS PN_LOCAL
BLANK_NODE_LABEL
        ::= '_:' ( PN_CHARS_U | [0-9] ) ( ( PN_CHARS | '.' )* PN_CHARS )?
```

```
LANGTAG ::= '@' [a-zA-Z]+ ( '-' [a-zA-Z0-9]+ )*
INTEGER ::= [+#x2D]? [0-9]+
DECIMAL ::= [+#x2D]? [0-9]* '.' [0-9]+
DOUBLE ::= [+#x2D]? ( [0-9]+ ( '.' [0-9]* )? | '.' [0-9]+ ) EXPONENT
EXPONENT ::= [eE] [+#x2D]? [0-9]+
STRING_LITERAL_LONG_SINGLE_QUOTE
        ::= "'''" ( ( "'" | "''" )? ( [^'\] | ECHAR | UCHAR ) )* "'''"
STRING_LITERAL_LONG_QUOTE
        ::= '"""' ( ( '"' | '""' )? ( [^"\] | ECHAR | UCHAR ) )* '"""'
STRING_LITERAL_QUOTE
        ::= '"' ( [^#x0022#x005C#x000A#x000D] | ECHAR | UCHAR )* '"'
STRING_LITERAL_SINGLE_QUOTE
        ::= "'" ( [^#x0027#x005C#x000A#x000D] | ECHAR | UCHAR )* "'"
UCHAR ::= ( '\u' | '\U' HEX HEX HEX HEX ) HEX HEX HEX HEX
ECHAR ::= '\' [tbnrf"'\]
WS ::= [#x0020#x0009#x000D#x000A]
ANON ::= '[' WS* ']'
PN_CHARS_BASE
        ::= [A-Za-z#x00C0-#x00D6#x00D8-#x00F6#x00F8-#x02FF#x0370-
        #x037D#x037F-#x1FFF#x200C-#x200D#x2070-#x218F#x2C00-
        #x2FEF#x3001-#xD7FF#xF900-#xFDCF#xFDF0-#xFFFD]
PN_CHARS_U
        ::= PN_CHARS_BASE
          | '_'
PN_CHARS ::= PN_CHARS_U
          | [-0-9#xB7#x0300-#x036F#x203F-#x2040]
BASE ::= ( 'B' | 'b' ) ( 'A' | 'a' ) ( 'S' | 's' ) ( 'E' | 'e' )
PREFIX ::= ( 'P' | 'p' ) ( 'R' | 'r' ) ( 'E' | 'e' ) ( 'F' | 'f' )
        ( 'I' | 'i' ) ( 'X' | 'x' )
PN_PREFIX
        ::= PN_CHARS_BASE ( ( PN_CHARS | '.' )* PN_CHARS )?
PN_LOCAL ::= ( PN_CHARS_U | [:0-9] | PLX ) ( ( PN_CHARS | '.' | ':' | PLX )*
        ( PN_CHARS | ':' | PLX ) )?
PLX ::= PERCENT
          | PN_LOCAL_ESC
PERCENT ::= '%' HEX HEX
HEX ::= [0-9A-Fa-f]
PN_LOCAL_ESC
        ::= '\' ( '_' | '~' | '.' | '-' | '!' | '$' | '&' | "'" | '(' | ')'
        | '*' | '+' | ',' | ';' | '=' | '/' | '?' | '#' | '@' | '%' )
EOF ::= $
```

Appendix B Scholarly communication - researcher X preferences in N3S

```
@prefix : <https://example.org/ns#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .

# Pref 1 . Publications in a subject repo
```

```
(_:V) log:onNegativeSurface {
  _:V a :SubjectRepository .

  ( ) log:onNegativeSurface {
       :ResearcherX :prefers _:V .
  } .
} .

# Pref 2 . Publications by a journal that doesn't charge APC costs
(_:V) log:onNegativeSurface {
    _:V a :Journal .

    ( ) log:onNegativeSurface {
        _:V :charges :APC .
    } .

    ( ) log:onNegativeSurface {
        :ResearcherX :prefers _:V .
    } .
} .

# Pref 3 . Publications by a publisher that is in WOS
(_:V) log:onNegativeSurface {
   _:V a :Journal .
   :WOS :indexed _:V .

   ( ) log:onNegativeSurface {
       :ResearcherX :prefers _:V .
   } .
} .
```

Appendix C Scholarly communication - departmental Y preferences in N3S

```
@prefix : <https://example.org/ns#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .

# Pref 1 . Only journals that are indexed in WOS
(_:V) log:onNegativeSurface {
   _:V a :Journal .
   :WOS :indexed _:V .

   ( ) log:onNegativeSurface {
      :DepartmentY :prefers _:V .
   } .
} .
```

Appendix D Scholarly communication - journal facts in N3S

```
@prefix : <https://example.org/ns#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .

## Journal facts
:ABC a :Journal .
:DEF a :Journal .
:GHI a :Journal .

# APC facts
:ABC :charges :APC .

## GHI is a journal that does not require APC costs
( ) log:onNegativeSurface {
    :GHI :charges :APC .
} .

# WOS Facts
:WOS :indexed :ABC , :DEF .

( ) log:onNegativeSurface {
    :WOS :indexed :GHI .
} .

## JKL is a subject repository
:JKL a :SubjectRepository .
```

Appendix E Medicine prescription - healthcare policies in N3S

```
@prefix : <https://example.org/ns#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .

(_:WHO) log:onNegativeSurface {

    _:WHO :has :Fever .

    ( ) log:onNegativeSurface {
        _:WHO :has :AllergyForAspirin .
    } .

    ( ) log:onNegativeSurface {
        _:WHO :has :ActivePepticUlcerDisease .
    } .

     ) log:onNegativeSurface {
        _:WHO :isPrescribed :aspirinLowDose .
```

```
        } .
    } .

    (_:WHO) log:onNegativeSurface {

        _:WHO :has :AcuteMyocardialInfarction .

        ( ) log:onNegativeSurface {
            _:WHO :has :AllergyForAspirin .
        } .

        ( ) log:onNegativeSurface {
            _:WHO :has :ActivePepticUlcerDisease .
        } .

        ( ) log:onNegativeSurface {
            _:WHO :isPrescribed :aspirinHighDose .
        } .

    } .

    (_:WHO) log:onNegativeSurface {

        _:WHO :has :AcuteMyocardialInfarction .

        ( ) log:onNegativeSurface {
            _:WHO :has :SevereAsthma .
        } .

        ( ) log:onNegativeSurface {
            _:WHO :has :ChronicObstructivePulmonaryDisease .
        } .

        ( ) log:onNegativeSurface {
            _:WHO :isPrescribed :betaBlocker .
        } .
    } .
```

Appendix F Medicine prescription - patient Ann data in N3S

```
    @prefix : <https://example.org/ns#> .
    @prefix log: <http://www.w3.org/2000/10/swap/log#> .

    # patient Ann
    :Ann :has :Fever .

    ( ) log:onNegativeSurface {
        :Ann :has :AllergyForAspirin .
```

```
}.

( ) log:onNegativeSurface {
    :Ann :has :ActivePepticUlcerDisease .
} .
```

Appendix G Medicine prescription - patient Joe data in N3S

```
@prefix : <https://example.org/ns#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .

# patient Joe
:Joe :has :AcuteMyocardialInfarction.
:Joe :has :AllergyForAspirin.

( ) log:onNegativeSurface {
    :Joe :has :ActivePepticUlcerDisease .
} .

( ) log:onNegativeSurface {
    :Joe :has :SevereAsthma .
} .

( ) log:onNegativeSurface {
    :Joe :has :ChronicObstructivePulmonaryDisease .
} .
```