RDF Surfaces: Computer Says No*

Patrick Hochstenbach[1,2,*], Jos De Roo[2] and Ruben Verborgh[2]

[1]Ghent University Library, Rozier 9, Ghent, 9000, Belgium
[2]Ghent University IDLab - ELIS, Technologiepark-Zwijnaarde 122, Zwijnaarde, 9052, Belgium

Abstract

Logic can define how agents are provided or denied access to resources, how to interlink resources using mining processes and provide users with choices for possible next steps in a workflow. These decisions are for the most part hidden, internal to machines processing data. In order to exchange this internal logic a portable Web logic is required which the Semantic Web could provide. Combining logic and data provides insights into the reasoning process and creates a new level of trust on the Semantic Web. Current Web logics carries only a fragment of first-order logic (FOL) to keep exchange languages decidable or easily processable. But, this is at a cost: the portability of logic. Machines require implicit agreements to know which fragment of logic is being exchanged and need a strategy for how to cope with the different fragments. These choices could obscure insights into the reasoning process. We created RDF Surfaces in order to express the full expressivity of FOL including saying explicitly 'no'. This vision paper provides basic principles and compares existing work. Even though support for FOL is semi-decidable, we argue these problems are surmountable. RDF Surfaces span many use cases, including describing misuse of information, adding explainability and trust to reasoning, and providing scope for reasoning over streams of data and queries. RDF Surfaces provide the direct translation of FOL for the Semantic Web. We hope this vision paper attracts new implementers and opens the discussion to its formal specification.

Keywords

Semantic Web, First-order Logic, Logical Reasoning

1. Introduction

RDF [1] is the standard for modeling data on the Semantic Web. From a syntactic viewpoint, RDF is a simple data model for expressing relations between triples where each triple is a first-class web object. From a semantic viewpoint, a triple is an assertion expressing what is believed to be *true*. Combinations of triples form a logical conjunction (AND). Any combination of resources on the Semantic Web creates a universal AND statement [2]. This is not unusual as the majority of database systems assert truth similarly.

Human and software agents interpret data and use internal logic to turn these assertions into decisions regarding what data to *trust* or not (negation), provide *options* to select data to ingest

ESWC '23: Trusting Decentralised Knowledge Graphs and Web Data Workshop – TrusDeKW, May 28–June 1, 2023, Hersonissos, Greece

*Corresponding author.
Patrick.Hochstenbach@UGent.be (P. Hochstenbach); Jos.DeRoo@UGent.be (J. D. Roo); Ruben.Verborgh@UGent.be (R. Verborgh)
https://patrickhochstenbach.net/ (P. Hochstenbach); https://ruben.verborgh.org/ (R. Verborgh)
0000-0001-8390-6171 (P. Hochstenbach); 0000-0001-8862-0666 (J. D. Roo); 0000-0002-8596-222X (R. Verborgh)

or compute (disjunction) and make *inferences* from this data (implication). Current Web logics in the form of rule and ontology languages provides insights into the reasoning process but only carries a fragment of first-order logic (FOL). These fragments of FOL are tuned to make the processing of Web logic decidable or easily processable, at the cost of portability. *Portability* is the ability to represent data and logic independently irrespective of the processing environment. RDF is portable, any RDF triple expresses the same data and meaning irrespective of the source. Adding Web logic, this becomes much harder: machines must first agree on an entailment regime before they know what each triple represents. In this position paper, we present our case for a portable Web logic language extending RDF semantics called *RDF Surfaces*, which is as powerful as FOL. RDF Surfaces is able to group RDF data within surfaces and provide semantics to say 'no': expressing an explicit scope and classical negation.

2. Background

We started our research into portable logic in relation to our involvement in the SolidLab Flanders [3] and Mellon-funded project "Scholarly Communications in the Decentralized Web" [4]. In both projects, a decentralized network of *data pods* exist on which individuals store (personal) data. In the case of SolidLab, data are documents Flemish citizens share with the government or businesses. In the case of the Mellon project, data are research artifacts such as scholarly articles and research data that are shared within a research community. Both projects are investigating how automated systems can assist in managing these types of data, interlink data with existing (external) resources, and provide enforcement of data policies using logic-based rules. In each use case, automated systems can't assume that only one actor creates and manages these rules. In a decentralized world, many agents can set their own requirements of what should or shouldn't happen with data. For instance, in the case of data policies, different actors on a personal, institutional, and governmental level can define the permissions and prohibitions (re)using data. These policies can overlap and possibly contradict themselves. Such clashes can benefit from being detected at an early stage and not at run time. Another challenge is the single language problem. Machines should be capable to implement potentially heterogeneous collection policy languages with many possible dialects. The strength of RDF is that the data model for these variations is portable, there is a common format that can express the overlap between the different policy languages. But, it is not the case that the logic – what all RDF triples entail – is portable. Without knowing which kind of inferences are possible (using entailment regimes such as RDFS, OWL-DL, OWL2, Notation3,..), it is possible that designers of policies can reach other conclusions than the consumer (the policy enforcer on the pod). For these use cases, we need a common portable logic, with a common syntax and semantics.

3. RDF Surfaces

Only two extensions are required to the semantics of RDF to make it expressable as FOL: a notion of a (possible nested) *surface* to group zero or more RDF graphs with a truth-functional type, and a notion of *graffiti* as marks on such as a surface with the function of existentially quantified variables.

Surfaces can be regarded as nestable virtual sheets of paper on which RDF graphs are written. A *positive surface* asserts all triples written on it; as with RDF, multiple triples form a logical conjunction (AND). The default surface is positive. All existing RDF graphs are regarded to be on the default positive surface. A *negative surface* negates all triples written on it, and multiple triples form a negation (NOT) of a conjunction (AND).[1] With logical connectives AND and NOT, any truth-functional statement can be created by composition, similar to how logical gates on a computer chip can be created by combining NAND gates.

Graffiti are marks on a surface representing quantified logical variables. Graffiti marks on a positive surface are interpreted as *existentially* quantified variables, this is how blank nodes are currently interpreted in RDF. The difference between blank nodes and graffiti marks is that blank nodes act as *coreferences* to graffiti marks, and that every surface contains its own unique set of graffiti marks. Transporting marks to a new surface requires creating new graffiti marks (not relabeling or merging graffiti marks, as is the case with blank nodes). Using De Morgan's duality[2], graffiti marks on a negative surface are interpreted as *universally* quantified variables.

Given the notion of a surface and graffiti, we define an **H-graph** as the combination of a (typed) surface S, graffiti Gr, and a graph H which is again an H-graph. Each RDF graph is an H-graph on the default positive typed surface. The empty H-graph is regarded as a tautology on a positive-typed surface and a contradiction on a negative-typed surface. By combining and nesting H-graphs any truth-functional statement can be created. H-graphs have similar semantics as the alpha and beta Existential graphs of Pierce[3] which are as expressive as FOL[5][6]. Our position is that these two added primitives (surfaces and graffiti marks) are a much more economical way to express FOL than in OWL Full, plus they are an immediate natural extension to RDF semantics.

We need a notation to transport the H-graphs over the Web and require an RDF syntax for that. This syntax should provide a way to scope RDF graphs and codify the graffiti marks. Our first choice was to use Notation3 [7], because it provides syntactic support for nested quoted graphs (surfaces), built-ins (to reason with surface types), and lists (codifying the graffiti marks). Using Notation3, an H-graph is expressed using graffiti marks as the subject list, a typed surface as a predicate, and an H-graph as the object. In our implementations, the surface predicate is implemented as a built-in to allow for reasoning with H-graphs.

Listing 1 provides an example of an H-graph with semantics $\forall s\colon learns(s, Physics) \Rightarrow reads(s, Newton) \vee reads(s, Einstein)$ by stating it as $\forall s\colon \neg(learns(s, Physics) \wedge \neg reads(s, Newton) \wedge \neg reads(s, Einstein))$ which means "Anyone that learns physics reads Newton or Einstein (or both)".

For a detailed introduction to RDF Surfaces we refer to our RDF Surface Primer [8].

4. Related Work

The case for portable Web logic is not new. Already in the 2000s Berners-Lee [9] called for the development of a language on top of RDF in his Semantic Web Logic Language (SWeLL) proposal.

[1] Individual triples are not necessarily negated; only their conjunctions.
[2] $\neg\exists x\colon P(X) \equiv \forall x\colon \neg P(x)$
[3] https://plato.stanford.edu/entries/peirce-logic/#AlphSyst

Listing 1: The RDF Surfaces semantics of a material implication with a disjunction using the Notation3 syntax.

```
(_:S) log:onNegativeSurface {
    _:S :learns :Physics .
    () log:onNegativeSurface { _:S :reads :Newton } .
    () log:onNegativeSurface { _:S :reads :Einstein } .
} .
```

SWeLL was imagined as a unifying language acting as a logical bus to "allow any web software to read and manipulate data published by any other web software". For all logical relations to be expressed, SWeLL had to extend RDF by including negation and explicit quantification.

A similar portability argument was given in 2009 by Hayes in his ISWC invited talk [10]. According to his argumentation, the idea of Web logic portability has not been achieved due to the layering of logic in the Semantic Web stack. Different layers of the stack cannot guess which entailment regimes are used when receiving data. Without this information, two independent machines cannot arrive at the same conclusions given a model. As a first step towards a solution for this conundrum, Hayes presented RDF Redux which provides a syntax for expressing Web logic with FOL semantics. His ideas were very influential for our work and led to our current research into RDF Surfaces.

There are two main reasons why classical negation has thus far been avoided. The first reason is negation opens the door to create *paradoxes* on the Semantic Web. One of the main motivations of Berners-Lee et al. for N3Logic [11] was to introduce the ability to compare Web documents and make inferences about assertions in each document. For this reason, a quoting mechanism was introduced. Quoting in combination with classical negation can lead to paradoxes when assertions in resources are self-referential. This self-referential problem resembles the liar paradox in the philosophy of language: "This sentence is false". To avoid paradoxes, Scoped Negation As Failure (SNAF) was introduced to N3Logic and acts as a monotonic version of Negation As Failure (NAF). SNAF disallows self-referential negative statements about resources and defines a scope for negation by the absence of information (Scope + NAF = SNAF). However, SNAF cannot avoid semantic inconsistencies nor has a syntactic mechanism for expressing them. With RDF Surfaces we introduce a formalism requiring negative facts to be explicitly expressed as negative surfaces, without using (S)NAF. Inconsistencies can be expressed as an assertion plus a negative surface thereof. These inconsistencies still need to be detected which leads to our second reason.

Classical negation is also avoided due to combining the expressive power of logics with negation and *decidability* problems. *Satisfiability* is a decidability problem to find inconsistencies. *Completeness* is a decidability problem, related to satisfiability, to find all valid statements from a knowledge base. *Halting* is a decidability problem requiring a machine to stop in a finite time. Logics containing logical variables and negation can be as powerful as first-order logic (FOL) for which it is proven [12] solving any of these problems is undecidable. Three options are available for dealing with undecidability for which, alas, only two can be chosen: (P) portable, allow any ontology input, (C) be complete, (H) always halt [13]. OWL 2 Full is an example of a Web logic

that can express any ontology but is undecidable. OWL 2 Description Logics drops option (P) and creates decidable fragments of FOL and reasoners can always return all answers and always halt, i.e. (C+H). This choice raises the question of what fragment of FOL to choose. This is at the cost of Web portability. There are at least as many choices as there are OWL Description Logics. Reasoners that cannot guess which fragment to choose to have to implement them all to provide portability. Reasoners should be as expressive as FOL in order to do so. In general, if we want portable Web logic, option (P) seems to be unavoidable. We argue: any of the (P+C), (P+H), and (C+H) axis are valuable depending on the use case. A reasoner as powerful as (P+C) or (P+H) is required to implement a portable Web logic, but the logic could be 'tunable' for specific use cases. A 'tuned down' (C+H) reasoner could implement OWL Description Logics as syntactic sugar. (P+C) and (P+H) can provide a portable logic for reasoning over decentralized sources. Undecidability is a hard problem, but unavoidable in the latter case. The logic provided by RDF Surfaces, we argue, can be created with fewer primitives than OWL 2 Full and is in line with a large body of research on FOL. Undecidability itself does not stop the Web from evolving as can be seen from the popularity of Web programming languages such as JavaScript.

5. Conclusion and future work

We believe that RDF Surfaces provides an expressivity comparable to Peirce's alpha and beta graphs and FOL. We still need proof that a complete mapping from RDF Surfaces to FOL (or Peirce's graphs) exists. This is part of our current research. Existential rules have shown already to be sufficient to implement the inference rules of RDFS, OWL-RL, and OWL-EL [14, 15, 16]. These existential rules were implemented in the Notation3 language, but we are porting them to RDF Surfaces. We started to port the Notation3 language itself in RDF Surfaces as ongoing work.[4]

The expressivity of RDF Surfaces goes beyond what is provided by OWL Description Logics. It could be impractical to express every inference rule solely in the language of first-order logic. The Notation3 language provides built-in functions and relations for arbitrary operations on RDF graphs of practical assistance to the programmer. A pure subset of these functions (deterministic and without side effects) could form the basis for an assembly language of the web. In addition to Web portability, use cases for RDF Surfaces can be found in implementing policy languages where one would explicitly want to express what is regarded as misuse of information and reason over that. RDF Surfaces can provide the semantics to limit the scope of reasoning over RDF data streams or limit the scope for queries and provide alternatives to query whatever is in reach [17].

The authors are currently experimenting with reasoners implementing RDF Surfaces along the (P+H) [18] and (C+H) [19] axis and formed a W3C Community Group [5] to define the semantics and find new implementers.

[4] https://github.com/eyereasoner/Notation3-By-Example/tree/main/examples/n3s
[5] https://www.w3.org/community/rdfsurfaces/

Acknowledgments

This work is funded by the Andrew W. Mellon Foundation (grant number: 1903-06675) and supported by SolidLab Vlaanderen (Flemish Government, EWI, and RRF project VV023/10). The authors thank Dörthe Arndt and Ruben De Decker for several insightful discussions about logic and applications of RDF Surfaces.

References

[1] R. Cyganiak, D. Wood, M. Lanthaler, RDF 1.1 concepts and abstract syntax, 2014. URL: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#section-skolemization.

[2] P. Hayes, P. F. Patel-Schneider, RDF 1.1 semantics, 2014. URL: https://www.w3.org/TR/rdf11-mt/.

[3] SolidLab flanders – SolidLab flanders, 2023. URL: https://solidlab.be/.

[4] M. Foundation, Scholarly communications in the decentralized web : Ghent university, 2019. URL: https://mellon.org/grants/grants-database/grants/ghent-university/1903-06675/.

[5] J. J. Zeman, The graphical logic of c.s. peirce, 1964.

[6] D. D. Roberts, The Existential Graphs of Charles S. Peirce, Approaches to Semiotics, 1973.

[7] D. Arndt, W. Van Woensel, D. Tomaszuk, G. Kellogg, Notation3, 2023. URL: https://w3c.github.io/N3/spec/.

[8] P. Hochstenbach, J. De Roo, RDF surfaces primer, 2023. URL: https://w3c-cg.github.io/rdfsurfaces/.

[9] T. Berners-Lee, SWELL - semantic web logic language, 2000. URL: https://www.w3.org/2000/01/sw/DevelopmentProposal.

[10] P. Hayes, BLOGIC. (ISWC 2009 invited talk), 2009. URL: https://www.slideshare.net/PatHayes/blogic-iswc-2009-invited-talk.

[11] T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf, J. Hendler, N3logic: A logical framework for the world wide web (2007). URL: http://arxiv.org/abs/0711.1533. arXiv:0711.1533.

[12] Trakhtenbrot, On recursive separability, in: DOKL AKAD NAUK SSSR, volume 88, 1953, pp. 953–956.

[13] A. Hogan, Exploiting RDFS and OWL for integrating heterogeneous, large-scale, linked data corpora (2011).

[14] D. Tomaszuk, Inference rules for RDF(s) and OWL in n3logic, 2016. URL: http://arxiv.org/abs/1601.02650. arXiv:1601.02650 [cs].

[15] D. Arndt, B. De Meester, P. Bonte, J. Schaballie, J. Bhatti, W. Dereuddre, R. Verborgh, F. Ongenae, F. De Turck, R. Van de Walle, E. Mannens, Ontology reasoning using rules in an eHealth context, in: N. Bassiliades, G. Gottlob, F. Sadri, A. Paschke, D. Roman (Eds.), Rule Technologies: Foundations, Tools, and Applications, Lecture Notes in Computer Science, Springer International Publishing, 2015, pp. 465–472. doi:10.1007/978-3-319-21542-6_31.

[16] M. Kroetzsch, Efficient rule-based inferencing for OWL EL (2011).

[17] O. Hartig, SPARQL for a web of linked data: Semantics and computability, in: E. Simperl, P. Cimiano, A. Polleres, O. Corcho, V. Presutti (Eds.), The Semantic Web: Research and Applications, volume 7295, Springer Berlin Heidelberg, 2012, pp. 8–23. URL: http://

link.springer.com/10.1007/978-3-642-30284-8_8. doi:10.1007/978-3-642-30284-8_8, series Title: Lecture Notes in Computer Science.
[18] J. De Roo, Euler yet another proof engine, 2023. URL: https://github.com/eyereasoner/eye.
[19] P. Hochstenbach, Latar, 2023. URL: https://github.com/MellonScholarlyCommunication/Latar.