Towards Supporting Multiple Semantics of Named Graphs Using N3 Rules

Dörthe Arndt[1] and William Van Woensel[2]

[1] IDLab, Department of Electronics and Information Systems, Ghent University – imec, Belgium
[2] NICHE, Dalhousie University, Canada
doerthe.arndt@ugent.be, william.van.woensel@dal.ca

Abstract. Semantic Web applications often require the partitioning of triples into subgraphs, and then associating them with useful metadata (e.g., provenance). This led to the introduction of RDF datasets, with each RDF dataset comprising a default graph and zero or more named graphs. However, due to differences in RDF implementations, no consensus could be reached on a standard semantics; and a range of different dataset semantics are currently assumed. For an RDF system not be limited to only a subset of online RDF datasets, the system would need to be extended to support different dataset semantics—exactly the problem that eluded consensus before. In this paper, we transpose this problem to Notation3 Logic, an RDF-based rule language that similarly allows citing graphs within RDF documents. We propose a solution where an N3 author can directly indicate the intended semantics of a cited graph—possibly, combining multiple semantics within a single document. We supply an initial set of companion N3 rules, which implement a number of RDF dataset semantics, which allow an N3-compliant system to easily support multiple different semantics.

Keywords: N3, TriG, Named graphs, Semantics

1 Introduction

For provenance, versioning or contextualizing purposes, Semantic Web applications often require the partitioning of triples into (named) graphs and then associating metadata with them. For this purpose, Version 1.1 of the Resource Description Framework (RDF) [7] introduced the concept of RDF datasets, which comprise a (possibly empty) default graph and zero or more named graphs. While the different syntaxes of RDF have clear definitions how to formulate such named graphs—for RDF Turtle [2] there is TriG [5], the latest draft for JSON-LD [14] also covers this concept—there is no consensus about their semantics. The W3C Working Group (WG), formed to define the semantics of named graphs, could not agree on one single interpretation—since a standard semantics had been lacking until then, existing implementations had been making assumptions about their meaning. Many of these assumptions were found to

differ fundamentally from each other. The WG collected a set of currently assumed named graph semantics, and used these as a basis to define eight different model-theoretic semantics, which in some cases are overlapping [21]. They furthermore discussed the possibility that everyone who publishes an RDF dataset also gives an indication about its intended semantics (although no standard was defined for doing that). Given that the Semantic Web relies on interoperability, this situation is rather problematic: to fully support named graphs, an application should implement up to eight different interpretations, and then support the various non-standard ways of indicating this semantics.

In this paper, we show how this problem can be addressed in Notation3 Logic (N3) using rule-based reasoning. N3 is an RDF-based rule language which allows the user to directly embed graphs (called cited formulas/graphs) in RDF triples. Our solution allows an author to directly indicate the intended semantics of a cited graph using a custom predicate. In order for any N3-compliant system to support these indicated semantics, we supply an initial companion set of N3 rules that covers a number of useful named graph semantics. We note that, in contrast to existing methods, our solution can be used in contexts where different intended meanings are needed within the same document. Furthermore, we provide compelling use cases for named graphs from the healthcare domain.

The remainder of this paper is structured as follows. In Section 2, we give a general introduction to N3 Logic and elaborate on the elements needed to support our solution. Afterwards, Section 3 introduces RDF datasets and named graphs, together with their possible semantics as presented by the W3C Working Group. We detail our solution in Section 4, presenting a concrete ruleset for implementing these semantics. In Section 5, we discuss related work, and we conclude the paper in Section 6.

2 Notation3 Logic

First proposed over a decade ago by Tim Berners-Lee et al. [3,4], Notation3 Logic (N3) forms a superset of RDF Turtle [2], extending this framework by supporting – inter alia – references to graphs within triples, rules which can act on simple triples but also on constructs involving graphs, and various built-in predicates implementing for example mathematical operations or manipulations of lists. Below, we introduce the aspects of this logic which are relevant for the application presented in this paper. For a more detailed introduction, we refer to the sources mentioned above as well as to our previous work [1].

2.1 Cited graphs

Notation 3 extends RDF Turtle. All triples and conjunctions of triples which are valid in Turtle are hence also valid in N3. In addition, N3 supports the use of graphs within a triple. This is exemplified in the following statement[3]:

[3] In this paper, the empty prefix refers to the example namespace `http://example.org/ex#`, other prefixes are conform with `https://prefix.cc`.

```
:LoisLane :believes {:Superman :can :fly}.          (1)
```

Here, the graph `{:Superman :can :fly}` is used in the object position. The semantics of such cited graphs is rather simple: According to the paper introducing N3Logic, cited graphs (or cited formulas, as they are called in N3) are not *referentially transparent* [4, p.7]. This means that, even if two terms denote the same resource in the domain of discourse, two cited formulas which only differ in the use of these two terms should be considered as different. From Formula 1 we can thus not directly conclude that:

```
:LoisLane :believes {:ClarkKent :can :fly}.          (2)
```

even if we know that `:Superman` and `:ClarkKent` denote the same resource. In the given example, it could for example be that `:LoisLane` does not know that `:ClarkKent` is `:Superman`[4]. Notation3 understands the cited graph as a resource on its own[5].

2.2 Rules on cited graphs

As explained, the semantics of Notation3 does not inherently take into account the possible relations of resources (such as equivalence) within a cited graph. But, since the logic supports rules, and these rules can also act on graphs, we can customize the entailment behaviour towards any given context. We illustrate this idea below. Rules in N3 are written using the graph notation `{ }` and an implication arrow `=>`. Universally quantified variables are indicated using a question mark `?`. Let us assume that Formula 1 and the following triple hold:

```
:Superman owl:sameAs :ClarkKent.                    (3)
```

In order to customize the entailment behaviour towards referential transparency in the Superman example , i.e. to make sure that Formula 2 is indeed a logical consequence of Formula 1, we can write the following rule:

```
{?x owl:sameAs ?y. :LoisLane :believes {?x :can :fly}}=>
                        {:LoisLane :believes {?y :can :fly}}.   (4)
```

The variable `?x` in the antecedent of the rule can be unified with the instance `:Superman` in Formulas 3 and 1, and the variable `?y` can be unified with `:ClarkKent`. Having found evidence for the two triples in the antecedent of the rule, this rule can be applied, and its instantiated consequent – namely Formula 2 – follows. Note that the unification for variable `?x` relied on triples within cited

[4] The example is often called the "superman-problem" and has been broadly discussed in the context of RDF reification. See for example `https://www.w3.org/2001/12/attributions/#superman`.

[5] For the associated model theoretic semantics, we refer to our previous work [1].

graphs (among others). It is further possible to unify universal quantifiers with a whole cited graph. If we are for example sure that everything `:LoisLane` believes is true, we could write the following rule to represent that entailment behavior:

```
{:LoisLane :believes ?x}=>?x.
```
(5)

This rule allows us for example to conclude the triple `:Superman :can :fly.` from Formula 1.

The examples we showed so far were quite basic in the sense that our rules only act on specific patterns, such as the Superman problem or Lois Lane's superb journalism qualities. Nevertheless, they illustrate the advantage of having a rule-based logic where cited graphs do not carry a lot of meaning on their own: By declaring the right rules, it is possible to support different entailment behaviours for cited graphs, depending on one's needs. We illustrate this in Section 4, where we present rules for a number of more generic entailment behaviors that support named graph semantics.

2.3 Built-ins

In addition to their ability to reference, and even look inside, cited graphs, the expressivity of N3 rules is bolstered by a set of built-in functions[6]. In the context of our paper, i.e., multiple different semantics for cited graphs, we focus here on built-in functions which produce or operate on cited graphs. Such builtins include `log:equalTo`, `log:semantics`, `log:conjunction` and `log:conclusion`. Using these built-ins, one can go beyond entailment behaviors for specific cases (e.g., Superman problem) and define generic entailments for implementing generic cited graph semantics. We explain the meaning of these built-in functions below.

We start with `log:equalTo`. This built-in function, which is not limited to graphs, checks whether two IRIs, blank nodes, literals or graphs are identical. Even if `:ClarkKent` and `:Superman` represent the same resource in the domain of discourse, the triple:

```
:ClarkKent log:equalTo :Superman.
```
(6)

Is false, because different URIs are used. The predicate directly operates on the representation level and not on the resources in the domain of discourse. It can also be used in combinations with graphs—here the order of the triples within the graph is ignored—and on graph patterns containing variables.

While a predicate operating only on the representation of resources is already special, at least from a logical point of view, there are many more built-ins which are rather technical in the sense that they cover extra-logical operations. One example for such an operation is reading-in data from local files or the Web.

[6] It is currently under discussion which exact built-ins should be part of N3 (see `https://github.com/w3c/N3/issues/11`) but in practice most N3 reasoners support the different built-ins implemented by Cwm (see `https://www.w3.org/2000/10/swap/doc/CwmBuiltins`).

Using the built-in `log:semantics`, we are able to do exactly that: the predicate reads data from a source and produces its corresponding RDF graph. If we for example know that a file `sameAs.n3` consists of only Formula 3, the rule

```
{<sameAs.n3> log:semantics ?x}=>{:we :parsed ?x}.        (7)
```

Results in

```
:we :parsed {:Superman owl:sameAs :ClarkKent.}.          (8)
```

This built-in is quite useful in contexts where a URI needs to be dereferencable and/or where existing Web knowledge is used for reasoning.

Another built-in which acts on cited graphs is `log:conjunction`. Given a list of different graphs, this built-in can be used in a rule to produce the conjunction of these graphs. For instance, the rule:

```
{({:a :b :c}{:s :p :o}) log:conjunction ?x}
                                    =>{:we :conclude ?x}.  (9)
```

Results in:

```
:we :conclude {:a :b :c. :s :p :o}.                      (10)
```

This built-in is interesting in situations where the triples used in different graphs need to be connected in order to—for example—construct new graphs or draw further conclusions. In our implementation, we currently use it to connect graphs and entailment rules and produce their deductive closure (Section 4.3).

To support the action we just described, i.e., apply rules on a single cited graph that combines data and rules, we require another built-in. Provided with a graph that includes rules in its subject position, `log:conclusion` produces the deductive closure of this graph and returns it as a graph in the object position. For example, the rule:

```
{{:a :b :c. {:a :b :c}=>{:s :p :o}} log:conclusion ?x}
                                    =>{:we :conclude ?x}.  (11)
```

Produces the triple:

```
:we :conclude {:a :b :c. {:a :b :c}=>{:s :p :o}. :s :p :o.}  (12)
```

Every possible triple which can be produced by applying these rules on the data within the subject graph, together with the original triples and rules within this graph, is contained in the output graph of the triple. As said above, this powerful built-in can be used to apply reasoning on a single cited graph.

3 RDF Named graphs

On the Semantic Web, the need often arises to attach metadata to triples and sets of triples, for instance, to describe the provenance, version or context of particular knowledge on the Web. In other cases, the ability to partition triples into separate graphs is already sufficient. Whereas RDF reification could be utilized to attach metadata with individual triples, RDF 1.1 introduced the concept of RDF Datasets [20] where triples may be partitioned into named graphs that can have associated metadata. An RDF dataset constitutes a set of RDF graphs, comprising one default graph (which may be empty) and zero or more named graphs [21]. Each named graph is a pair, consisting of a *graph name*, which may be an IRI or a blank node, and an *RDF graph*.

At the time, the W3C Working Group, chartered with defining a standard semantics for RDF datasets, found that very different assumptions existed among practitioners on (**a**) what graph names denote (or, similarly, constraints on what these names can denote); and (**b**) how triples from named graphs influence the meaning of the associated RDF dataset. Instead of a concrete standard, the Working Group defined a set of potential RDF dataset semantics that were discussed during their standardization effort [21]. Below, we discuss and exemplify four of these RDF dataset semantics. In the next section, we illustrate how we implemented these four semantics in N3 (Section 4).

3.1 Partitioning of triples

In this case, named graphs are utilized as a convenient way of sorting triples within a knowledge base (i.e., RDF dataset). Here, the dataset semantics represent a union/merge of the default and named graphs—a dataset is true if the union/merge of all triples in the dataset's graphs are true. This semantics does not impact the meaning of graph names. A distinction is made depending on the intended meaning of blank nodes; when intentionally shared across named graphs, a union operation suffices; else, a merge operation is needed that forces blank nodes within disparate graphs to be globally distinct [13].

This semantics is problematic when named graphs are used to keep disparate RDF data found "in the wild"—while mostly consistent internally, online RDF graphs could conflict with other graphs. Since named graphs contribute to a shared dataset meaning, such cases will result in inconsistencies.

As an example, this semantics can be utilized to represent an Electronic Medical Record (EMR) that is shared across multiple points-of-care. Named graphs are used to represent consultations, and datasets group all named graphs belonging to the same patient. A system hooked up to the EMR can submit a named graph with a new consultation, which will then simply be added to the dataset, facilitating data management. For instance (using SNOMED-CT [18]):

```
1 { :emr1021 dc:topic :doerthe_arndt ; a sct:Patient . }
2
3 :cons_320 {_:gc a sct:FindingOfBloodGlucoseLevel ;
4   sct:findingMethod [ a sct:UrineDipstickForGlucose ] ;
```

```
5    sct:findingInformer :B ;
6    :value "6.8mmol/L " ; :time "2019-07-30"^^xsd:date }
7
8  :cons_002 {_:ac a sct:AllergyToClam; :findingInformer :C ;
9    :time "2015-01-01"^^xsd:date }
```

Listing 1. Example RDF dataset representing an EMR snippet.

This example shows an EMR dataset on the patient shown in the default graph. Since the EMR is organized per consultation and shared across points-of-care, a clinician can easily check for prior diagnostic tests (e.g., blood glucose) or allergies (e.g., clam), and which clinicians (*findingInformer*) were responsible.

In this case, one can actually argue that conflicts between named graphs *should* be flagged—for instance, when contradictory diagnoses are made by two different clinicians:

```
1  :cons_321 { _:dia a sct:UpperRespiratoryInfection ;
2    sct:interprets [ a sct:Fever ], [ a sct:Headache ],
3      [ a sct:SwallowingPainful ] ;
4    sct:findingInformer :A ; time "2019-08-15" }
5
6  :cons_322 { _:kna :findingContext [ a sct:KnownAbsent ] ;
7    associatedFinding [ a sct:UpperRespiratoryInfection ;
8      sct:interprets [ a sct: ThroatSwabCultureNegative ] ;
9      sct:findingInformer :B ; :time "2019-08-17" ] }
```

Listing 2. Example RDF dataset representing an EMR (2).

Here, during consultation 321, clinician A (*findingInformer*) diagnosed an upper respiratory infection, based on their interpretation (*interprets*) of the patient's symptoms. However, during a later consultation, clinician B discounted an upper respiratory infection (*findingContext*, *KnownAbsent*; *associatedFinding*) based on a negative throat swab culture (*interprets*). Domain-specific rules can be in place here to flag conflicting findings within a given time span.

3.2 Quoted sets of triples

This semantics assumes named graphs as occurrences of RDF graphs—one can say that a named graph comprises a quoting of an RDF graph. Moreover, a graph name is defined as denoting its named graph pair within the dataset. This allows one to describe a particular graph occurrence using the graph name as subject or object in RDF statements; indicating the graph provenance, such as retrieval date, author and source, version, etc.

Since named graphs comprise quoted RDF graphs, graphs will exhibit a lack of referential transparency [4], similar to the semantics of cited formulas in N3 (Section 2.1). Referential opaqueness can nevertheless be useful to keep exact statements replicas. In the Superman example, Lois Lane likely to say that "Superman" and not "Clark Kent" can fly, although knowledge on their equality may be available to whoever (likely Batman) who constructed the dataset.

This semantics could for instance be applied to represent patient charts. Similar to before (Section 3.1), one can use named graphs to keep single patient charts, whereas datasets will comprise all named graphs belonging to the same patient (we refer to listings 1 and 2 for examples). When using electronic charting systems, one generally expects an exact transcript of the encounter—no words are being put in the clinician's mouth (personal understanding of medical terms often differs); and one knows which precise medical terms are being used, which is useful for educational purposes or even training speech recognition systems.

3.3 Isolated triple contexts

This semantics defines each named graph as an isolated context wherein the pair's RDF graph triples are true. An RDF dataset is true when the default graph and all constituent named graphs are individually true. Since this semantics involve drawing conclusions at the named graph level, they can be seen as interpreting, as opposed to quoting, RDF graphs. Similar to before (Section 3.2), one may use graph names to encode metadata about the named graphs.

As opposed to the first dataset semantics (i.e., partitioning of triples), this semantics allow reasoning over possibly conflicting graphs, for instance, supplying contradictory viewpoints or represent the evolution of knowledge over time.

Compared to the second semantics (i.e., quoted sets of triples), this semantics offers referential transparency; e.g., when it is known that "Superman" and "Clark Kent" denote the same object, the statement "Superman can fly" would imply "Clark Kent can fly". But this also means there is no guarantee that a named graph's original triples are preserved—the comprised RDF graphs may be replaced by equivalent graphs, or any inferences may be added, without impacting their truth value.

These dataset semantics can be useful for computerizing clinical recommendations, which often differ over time and across geographic regions. For instance, we consider the case of Reye's syndrome, a currently rare but severe illness that occurs mainly in children and adolescents [17]. Given conflicting medical opinions on a link between salicylate therapy (e.g., aspirin) and Reye's syndrome [9,16], some countries, such as the US and UK, excluded its use for suspected viral disease in children from 1980s on [17], whereas other countries, such as France and Belgium, continued to use it for children into the 1990s [16].

In such cases, it can be useful to isolate the guideline knowledge inside named graphs within an RDF dataset (using SNOMED-CT [18]):

```
1  { :srd dc:topic [ a sct:SalicylateProphylaxis ] . }
2
3  :rd1 { _:p :ageUnder "19"^^xsd:int ;
4     :hasContext _:avd .
5    _:avd sct:associatedFinding [ a :ViralDisease ] ;
6      sct:findingContext sct:Suspected ;
7      sct:associatedProcedure [ a :SalicylateProphylaxis ] ;
8      sct:procedureContext sct:Contraindicated }
9
10 :rd2 { _:p :hasContext _:vdf .
```

```
11    _:vdf sct:associatedFinding [ a :ViralDisease ] ;
12      sct:findingContext sct:Suspected ;
13      sct:associatedProcedure [a sct:SalicylateProphylaxis] ;
14      sct:procedureContext sct:Indicated }
```

Listing 3. Example RDF dataset including two conflicting medical guidelines.

The first named graph states that, for a patient with an age under 19, where a finding (*associatedFinding*) of *ViralDisease* is suspected (*findingContext*), the procedure (*associatedProcedure*) *SalicylateProphylaxis*, i.e, treatment with salicylates such as aspirin, is contraindicated (*procedureContext*). The second named graph states that, given the same suspected finding, treatment with salicylates is indicated, which constitutes a clear conflict.

By assuming this semantics, the guidelines are cleanly separated within isolated named graph contexts, contained within a dataset that pertains to the same clinical concept. In addition, this semantics allow attaching metadata to the named graphs (in this case, within the default graph of the dataset):

```
1 { :srd dc:topic [ a sct:SalicylateProphylaxis ] .
2   :re1 :country :US, :UK ; :from "1970" .
3   :re2 :country :France, :Belgium ; :from "1950" . }
```

Listing 4. Example RDF dataset including metadata for named graphs.

3.4 Online RDF graphs

In these semantics, a graph name is assumed to dereference to an online RDF graph, which is in a particular relation (e.g., equals, is subgraph of, entails / is entailed) with the RDF graph from the pair. An RDF dataset is true when the default graph is true, and when RDF graphs from all named graph pairs adhere to the specified relations.

This interpretation thus depends on the current state of Web resources, and different entailments may take place at different times. For instance, one can assume this semantics when verifying local content with online, externally maintained and up-to-date knowledge. Nevertheless, this semantics would be unsuitable when one needs consistent output even when the system is offline.

For instance, this semantics are useful when targeting conformance with computerized clinical guidelines. Clinical guidelines are evidence-based recommendations for a specific illness, and are frequently updated by expert panels [6]. Here, we consider the scenario where local recommendations (e.g., formulated by a clinician) must be validated against up-to-date online computerized guidelines. Assuming these dataset semantics, we can implement this scenario by formulating graph names that dereference to online guidelines. For instance, an online RDF document, found at `http://example.org/ex#htn_gln1`, comprises the following hypertensive disorder (HTN) guideline snippet:

```
1
2 sct:findingContext sct:KnownPresent ;
3 sct:associatedFinding [ a sct:HypertensiveDisorder ] ;
```

```
4 sct:associatedProcedure [ a sct:DrugTherapy ;
5   sct:drugUsed [ a sct:ThiazideDiuretic ]] ;
6 sct:procedureContext sct:ToBeDone .
```

Listing 5. Example RDF document including a HTN medication guideline.

I.e., in case a finding (*associatedFinding*) of *HypertensiveDisorder* is known to be present (*findingContext*), then a drug therapy (*associatedProcedure*) involving a type of *Thiazide Diuretic* should be done (*procedureContext*).

A local process generates the following named graph:

```
1 :htn_gln1 {
2   sct:associatedFinding [
3     a sct:SustainedDiastolicHypertension ] ;
4   sct:associatedProcedure [
5     sct:drugUsed [ a sct:Methylclothiazide ] ] ;
6   sct:procedureContext sct:ToBeDone }
```

Listing 6. Example RDF dataset comprising a locally generated recommendation.

I.e., prescribing *Methylclothiazide*, a type of Thiazide Diuretics, for a finding of *SustainedDiastolicHypertension*, a type of HTN. With the prefixed name `:htn_gln1` resolving to the online RDF guideline, the system can check whether the local recommendation is entailed by the online HTN guideline.

4 Named Graphs in N3

We represent an RDF named graph pair as an N3 triple, with the graph name from the pair as subject, and an N3 cited graph, standing in for the RDF graph from the named graph pair, as object. We then make use of the predicate position to indicate the intended meaning of the named graph. For example, in case the second named graph in Listing 1 (Lines 8-9) should follow the merge partition semantics (i.e., assuming a merge of all named graphs with the default graph; Section 3.1), we indicate that using the predicate `sem:mergedWithDefault`:

```
:cons_002 sem:mergedWithDefault
      { _:ac a sct:AllergyToClam ; :findingInformer :C ;
                              :time "2015-01-01"8sd:date . }   (13)
```

We present a series of predicates (discussed below) to indicate the named graph semantics discussed in Section 3. For each of these predicates, we define rules to support their associated named graph semantics. These rules can be executed with existing N3 reasoners[7] which will produce valid conclusions, if any. All our rules were tested on EYE [19]. The code can be accessed at: `https://github.com/IDLabResearch/TriGRules`.

[7] For example, the ones listed at: `https://github.com/w3c/N3/blob/master/implementations.md`.

4.1 Partitioning of triples: Rules

A first interpretation of named graphs assumes them as a convenient way to partition the dataset (Section 3.1). In this semantics, a dataset is true if the union/merge of all named graph and default graph triples are true. Predicates `sem:unionWithDefault` and `sem:mergeWithDefault` indicate the union and merge partition semantics, respectively. Both these predicates have a common super-predicate, namely `sem:combinedWithDefault`. To support N3 named graphs with general partition semantics, we may use the following simple rule:

```
1 { ?x sem:combinedWithDefault ?y } => ?y.
```

Listing 7. N3 rule supporting the partition semantics.

In this way, an N3 reasoner directly adds the graphs to the knowledge base, stating them as true. Depending on whether a merge or union semantics is needed, blank nodes will need to be treated in different ways: they can be understood as local to the cited graph (merge); or blank nodes from different graphs sharing the same name will be considered equal (union).

By default, N3 applies the merge semantics—i.e., blank nodes are scoped on (cited) graph level [3]. Hence, to support the union semantics as well, an additional step is needed that skolemizes blank nodes within the named graph triples [7, Sec. 3.5]. Then, after applying the rule in Listing 7, the Skolem IRIs can be replaced again by blank nodes. The EYE reasoner [19] supports this step by command-line arguments[8]. Alternatively, a custom N3 built-in could be created (and perhaps standardized by the current W3C Community Group[9]) for implementing this step directly within a rule.

4.2 Quoted sets of triples: Rules

The second interpretation assumes that named graphs construct a kind of quoting of RDF graphs (Section 3.2). This is indicated using the `sem:quotedGraph` predicate. As discussed previously (Section 2.1), this interpretation is very similar to the semantics of cited formulas in N3. For example, if the named graph from Formula 13 should be interpreted using this semantics, i.e., using the `sem:quote` predicate, we do not need extra rules to support this interpretation.

4.3 Isolated triple contexts: Rules

In this semantics, each named graph defines the context in which its triples hold (Section 3.3). It is indicated by the predicate `sem:isolatedGraph`.

We discuss the case where each graph comes which its own entailment rules—the case where all graphs follow the same entailment regime can be seen as a special case of that. Using the predicate `sem:entailment`, one can indicate a

[8] More concretely, options `--no-qvars` and `--no-skolem`, see [8] for more details.
[9] https://www.w3.org/community/n3-dev/

source containing N3 rules that implement the desired entailment behavior. This can be a known entailment regime, like OWL-RL[10], or a custom rule set.

To illustrate the use of these predicates, we show the N3 version of the first recommendation (i.e., named graph) given in Listing 3:

```
:rd1 sem:isolatedGraph { p :age "11"^^xsd:int .
{ _:avd a :Recommendation ;
   :ageRange [:op math:lessThan ; :limit "19"^^xsd:int];
   sct:associatedFinding [ a :ViralDisease ] ;
   sct:findingContext sct:Suspected ;
   sct:associatedProcedure[a sct:SalicylateProphylaxis];
   sct:procedureContext sct:Contraindicated .
  :p0 :age "11"^^xsd:int ;
    :prescription [ a sct:SalicylateProphylaxis ] ;
    :hasContext _:pc0 .
  _:pc0 sct:associatedFinding [ a :ViralDisease ] ;
    sct:findingContext sct:Suspected };
 sem:entailment <rule2.n3> .
```

Listing 8. Extended N3 example assuming isolated graph semantics.

(We refer to the description of Listing 3 for details.) In this use case, the named graph is further extended with triples representing a concrete patient case (i.e., age, prescribed treatment and context). Further, we refer to a file that defines an entailment rule that warning clinicians about (age-related) contraindicated treatments:

```
{ ?rec a :Recommendation ; :ageRange ?ar .
  ?ar :op ?op ; :limit ?lim . p :age ?a . ?a ?op ?lim .
  ?p :hasContext ?c .
  ?c sct:associatedFinding [ a ?fin ] ;
    sct:findingContext sct:Suspected .
  ?rec sct:associatedFinding [ a ?fin ] ;
    sct:findingContext sct:Suspected .
  ?p :prescription [ a ?pre ] ;
  ?rec sct:associatedProcedure [ a ?pre ] ;
    sct:procedureContext sct:Contraindicated
} => {
  _:x a :ContraindicationWarning ; causedBy ?pre } .
```

Listing 9. Content of the file rule2.n3.

When the patient matches the recommendation, i.e., its age range (lines 1-2) and finding (e.g., *ViralDisease*) (lines 3-7), and their prescription is contraindicated by the recommendation (lines 8-10), a warning is issued to the clinician.

To support this named graphs semantics, we present the following rule:

```
{
  ?x sem:isolatedGraph ?y .
  ?x sem:entailment ?regime .
  ?regime log:semantics ?graph .
```

[10] N3 rules implementing OWL2-RL are provided at `http://eulersharp.sourceforge.net/2003/03swap/eye-owl2.html`.

```
5    (?y ?graph) log:conjunction ?all .
6    ?all log:conclusion ?out
7  } => {
8    ?x :isolatedGraph ?out }.
```

Listing 10. N3 rule supporting the `sem:isolatedGraph` semantics.

In this rule, we make use of the different built-ins we discussed earlier (Section 2.3). Provided with the pointer to the entailment behavior source file, we use the built-in `log:semantics` to read the rules from the source and produce an RDF graph. This entailment behavior graph is then combined with the graph from the named graph pair, using the built-in `log:conjunction`. As a last step, `log:conclusion` is used to produce the deductive closure of our combined graph.

Applying this rule on the graph from Listing 8, referencing `rule2.n3` (Listing 9), we get a warning as the prescription is contraindicated for the patient.

Note that conflicts would occur if these graphs did not have an isolated context. For instance, assuming a partition semantics (Section 3.1), the patient case, the two recommendations and entailment regimes become part of the default graph—leading to both an indication and contraindication for this patient.

4.4 Online RDF graphs: Rules

In this semantics, the graph name is dereferencable to an online graph, which stands in a particular relation with the graph from the named graph pair (Section 3.4). The general predicate utilized here is `sem:onlineRelationTo`, which can be sub-predicated depending on the intended relationship—we illustrate the case here where the content of the source equals (i.e., is identical to) the local graph, as indicated by `sem:onlineEquals`. For instance, Listing 6 can be easily written as an "N3 named graph triple" using this predicate.

We provide the following rule which checks whether the content of the graph and the source are identical:

```
1 {
2   ?x sem:onlineEquals ?d.
3   ?x log:semantics ?s.
4   ?s log:equalTo ?d.
5 } => {
6   ?x :confirmedEqual ?d. }.
```

Listing 11. N3 rule supporting the online-equals semantics.

The rule utilizes the built-in `log:semantics` to access the online source indicated by the graph name, and the built-in `log:equalTo` to check whether the graph and content of the source file are syntactically identical. In that case, a triple is derived that confirms the intended relation between the online graph and named graph. Similarly, rules could be devised that check for entailment (in either direction); or, using the builtin's negated form, namely `log:notEqualTo`, we could write a rule that detects violations of the `sem:onlineEquals` relation.

5 Related work

The RDF 1.1 Working Group [21] borrowed the notion of RDF datasets from SPARQL. A SPARQL query [11] is executed on an RDF Dataset, which comprises a default graph and zero or more named graphs (identified by an IRI). While the SPARQL specification defines query evaluation behavior on these RDF datasets in detail, it does not define their model-theoretic semantics. We note that the authors of the proposed RDF 1.1 dataset semantics make the following analogy with SPARQL entailment [10]: when a non-variable SPARQL ASK query with an entailment regime returns true on an RDF graph, then this graph entails the query under the regime. They observe that the semantics of isolated triple contexts (Section 3.3) are compatible with this SPARQL ASK case [21].

Hartig et al. presented an extension of RDF semantics called RDF*, together with a SPARQL* extension, that tackles the shortcomings of a conventional use of RDF reification [12]—i.e., where one describes an RDF statement using four statements about a statement token (e.g., blank node), respectively listing the subject, predicate and object of the original triple, and the *Statement* type of the token [15]. This statement token can then be utilized to represent associated metadata. Instead, RDF* proposes the embedding of a triple directly within the s/o position of other triples, which represent metadata about the embedded triple. RDF* is limited to embedding single triples in a s/o position (note that embeddings may be nested). Moreover, its semantics are defined via a transformation to RDF reification, which is known to lack semantics.

6 Conclusion and future work

In this paper, we illustrated the power of N3 rules and built-ins, by formulating a set of rules that implement generic entailment behaviors for RDF named graph semantics. We introduced relevant aspects of N3 and discussed a set of semantics proposed for RDF datasets. We showed the real-world applicability of these named graph semantics through various healthcare use cases.

This is an initial effort towards supporting a variety of semantics for cited graphs within N3. Future work involves extending support for the currently considered semantics: e.g., multiple graph relations for the named graph semantics with dereferencable graph names; and, in cooperation with the N3 Community Group, studying the option of a Skolem built-in for supporting union partition semantics. Moreover, an important avenue of future work involves supporting additional semantics discussed by the W3C Working Group; possibly, others as well as that were not covered by this group.

References

1. Arndt, D., Schrijvers, T., De Roo, J., Verborgh, R.: Implicit quantification made explicit: How to interpret blank nodes and universal variables in Notation3 Logic. Journal of Web Semantics 58, 100501 (oct 2019), https://www.sciencedirect.com/science/article/pii/S1570826819300241

2. Beckett, D., Berners-Lee, T., Prud'hommeaux, E., Carothers, G.: Turtle - Terse RDF Triple Language. W3C Recommendation (Feb 2014), http://www.w3.org/TR/turtle/
3. Berners-Lee, T., Connolly, D.: Notation3 (N3): A readable RDF syntax. W3C Team Submission (Mar 2011), http://www.w3.org/TeamSubmission/n3/
4. Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., Hendler, J.: N3Logic: A logical framework for the World Wide Web. Theory and Practice of Logic Programming 8(3), 249–269 (2008)
5. Bizer, C., Cygniak, R.: RDF 1.1 TriG. W3C Recommendation (Feb 2014), http://www.w3.org/TR/trig/
6. Brush, J.E., Radford, M.J., Krumholz, H.M.: Integrating Clinical Practice Guidelines Into the Routine of Everyday Practice. Critical Pathways in Cardiology: A Journal of Evidence-Based Medicine 4(3), 161–167 (2005)
7. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1: Concepts and Abstract Syntax. W3C Recommendation (Feb 2014), http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
8. De Roo, J.: Euler yet another proof engine (1999–2014), http://eulersharp.sourceforge.net/
9. Glasgow, J.F.T.: Reye's Syndrome: the case for a causal link with aspirin. Drug Safety 29(12), 1111–1121 (2006), http://www.ncbi.nlm.nih.gov/pubmed/17147458http://www.ncbi.nlm.nih.gov/pubmed/17147458
10. Glimm, B., Ogbuji, C.: SPARQL 1.1 Entailment Regimes (2013), https://www.w3.org/TR/sparql11-entailment/
11. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language - RDF Dataset (2013), https://www.w3.org/TR/sparql11-query/#rdfDataset
12. Hartig, O., Thompson, B.: Foundations of an Alternative Approach to Reification in RDF (jun 2014), http://arxiv.org/abs/1406.3399
13. Hayes, P.J., Patel-Schneider, P.F.: RDF 1.1 Semantics - merging (2014), https://www.w3.org/TR/rdf11-mt/#dfn-merge
14. Kellogg, G., Champin, P.A., Longley, D., Sporny, M., Lanthler, M., Lindström, N.: Json-ld 1.1. W3C Working Draft (Aug 2019), https://www.w3.org/TR/json-ld11/
15. Manola, F., Miller, E.: RDF Primer - RDF Reification (2004), https://www.w3.org/TR/rdf-primer/#reification
16. Orlowski, J.P., Hanhan, U.A., Fiallos, M.R.: Is Aspirin a Cause of Reye'ss Syndrome? A case against. Drug Safety 25(4), 225–231 (2002), http://www.ncbi.nlm.nih.gov/pubmed/11994026
17. Pugliese, A., Beltramo, T., Torre, D.: Reye's and Reye's-like syndromes. Cell Biochemistry and Function 26(7), 741–746 (oct 2008), http://doi.wiley.com/10.1002/cbf.1465
18. U.S. National Library of Medicine: SNOMED CT, https://www.nlm.nih.gov/healthit/snomedct/index.html
19. Verborgh, R., De Roo, J.: Drawing conclusions from linked data on the web. IEEE Software 32(5) (May 2015), http://online.qmags.com/ISW0515?cid=3244717&eid=19361&pg=25
20. Wood, D.: What's New in RDF 1.1 - Datasets (2014), https://www.w3.org/TR/rdf11-new/#datasets
21. Zimmermann, A.: RDF 1.1: On Semantics of RDF Datasets. W3C Working Group Note (Feb 2014), http://www.w3.org/TR/2014/NOTE-rdf11-datasets-20140225/