Options for RDF Expression of Credibility Data

Discussion Draft - 7-11 Feb 2020

Sandro Hawke, editor

Previous version at Options for RDF Expression of "Date Website First Archived"

1. Introduction

The Credible Web Community Group aims to create an ecosystem where individuals and organizations share data which ends up reducing misinformation on the web. Briefly, the plan is that various parties can observe and encode knowledge which might help indicate the trustworthiness of web sites, web pages, etc.. Parties can then share that data, often in trusted relationships, to help produce better credibility assessments and reduce the spread of misinformation. For more on this, see the introductions to Credibility Tech 2018 and Credibility Signals.

To enable systems to accurately exchange credibility data, the syntax and semantics must both be defined. The Community Group is approaching these tasks separately, with one subgroup selecting and defining promising signals and another subgroup considering how to express that data in machine-readable form. For the second subgroup, expressing the data in RDF is one approach, which is the focus of this document.

The interface between the two groups is template sentences. These sentences express the semantics of a credibility observation in natural language. For example, two templates approved as promising signals are:

- Signal: Date Website First Archived : There was a website operational at URL [ ] as early as isodate [ ], as shown in the archive page at URL [ ].
- Signal: Corrections Policy : The news website with its main page at URL [ ] provides a corrections policy at URL [ ] and evidence of the policy being implemented is visible at URL [ ].

The rest of this document discusses issues and options in how to express these two signals, and signals in general, in RDF. This document assumes general familiarity with RDF 1.1 Concepts and Abstract Syntax and SPARQL Basic Graph Patterns (or TriG with variables)

2. Graph shapes for "Date Website First Archived"

For this section we consider Signal: Date Website First Archived, and only issues around overall RDF graph shape, not property or class names, datatypes, entity identification, or other possible issues.

Real world examples as template sentences:

- There was a website operational at URL https://news.mit.edu as early as isodate 2015-09-02, as shown in the archive page at URL https://web.archive.org/web/20150902023223/http://news.mit.edu/.
- There was a website operational at URL https://nytimes.com as early as isodate 1996-11-12, as shown in the archive page at URL https://web.archive.org/web/19961112181513/http://www.nytimes.com:80/

We can imagine using SPARQL-style variables in the template sentence, like this:

- There was a website operational at URL ?site as early as isodate ?date, as shown in the archive page at URL ?archive

Now the goal is to choose an RDF graph such that a SPARQL query would produce bindings like:

site	date	archive
<https://news.mit.edu>	“2015-09-02”^^xsd:date	<https://web.archive.org/web/20150902023223/http://news.mit.edu/>
<https://nytimes.com>	“1996-11-12”^^xsd:date	<https://web.archive.org/web/19961112181513/http://www.nytimes.com:80/>

We can thus express each shape option as a SPARQL pattern which would produce the above bindings.

For the JSON-LD and the diagrams, we actually use fictional bindings with shorter URLs, and the trickier case of two observations about the same website.

Shape 0 (bad)

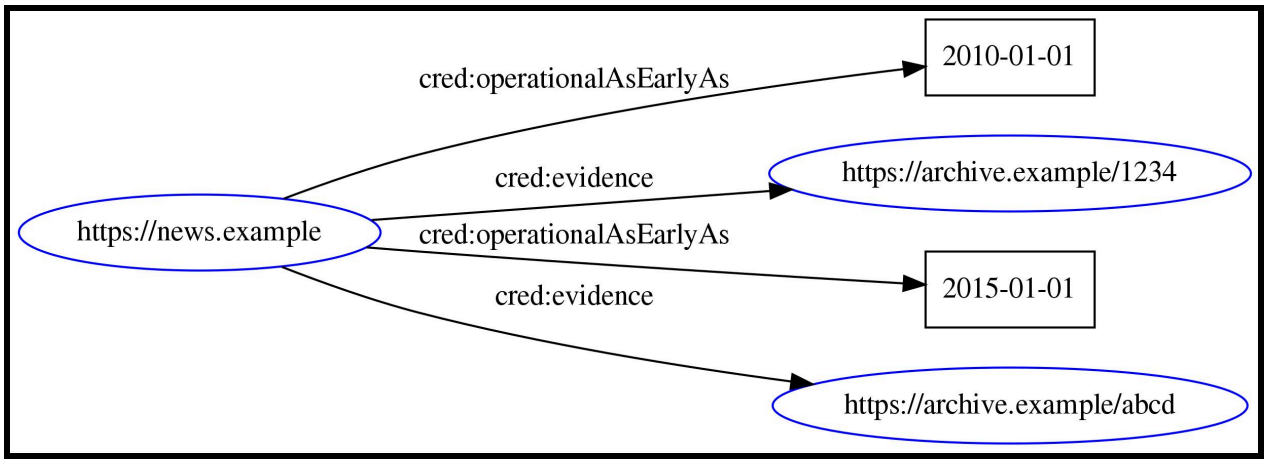
Perhaps the most obvious shape is this:

```
?site cred:operationalAsEarlyAs ?date;
      cred:evidence ?archive.
```

While this would work for the above data, it fails if there are two observations of the same site with different values for ?date and ?archive, like this:

```
<https://news.example>
```

```
        cred:operationalAsEarlyAs "2010-01-01";
        cred:evidence <https://archive.example/1234>;
        cred:operationalAsEarlyAs "2015-01-01";
        cred:evidence <https://archive.example/abcd>.
```



In such a situation, one cannot tell from the data which date goes with which archive page. A SPARQL query of such data would return all four possible pairings.

This highlights issues working with n-ary relations and with metadata in RDF. For more on this, see:

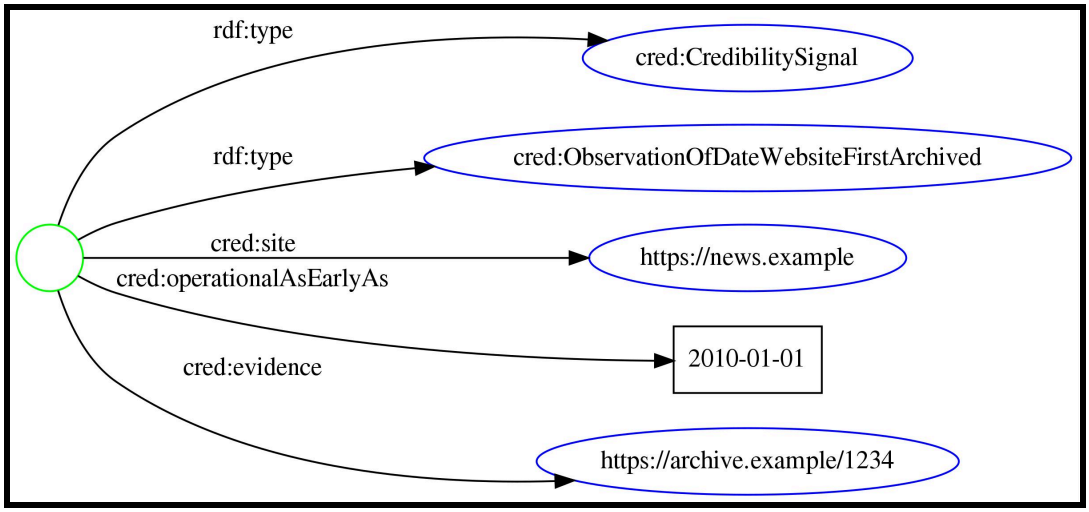
- Defining N-ary Relations on the Semantic Web (W3C Working Group Note, 2006)
- On Nary Relations

Addressing this issue generally requires introducing another node, typically a blank node. There are many different ways to do this, several of which appear quite reasonable, as shown below.

2.1. Shape 1: Star

Perhaps the simplest approach is to introduce a blank node representing the act of observation that results in the signal data. Attached to that node, we get all the relevant information, in what is topologically a star (although it looks more like a palm tree in a hurricane, in this left-to-right graph rendering):

```
?node1 a cred:ObservationOfDateWebsiteFirstArchived; #optional
       cred:site ?site;
       cred:operationalAsEarlyAs ?date;
       cred:evidence ?archive.
```



This JSON-LD is perhaps usable even for people unfamiliar with RDF:

```
{
  "@context": { ... },
  "@graph": [
      {
        "@type": "ObservationOfDateWebsiteFirstArchived",
        "site": "https://news.example",
        "operationAsEarlyAs": "2010-01-01",
        "evidence": "https://archive.example/1234"
      },
      {
        "@type": "ObservationOfDateWebsiteFirstArchived",
        "site": "https://news.example",
        "operationAsEarlyAs": "2015-01-01",
        "evidence": "https://archive.example/abcd"
      }
  ]
}
```

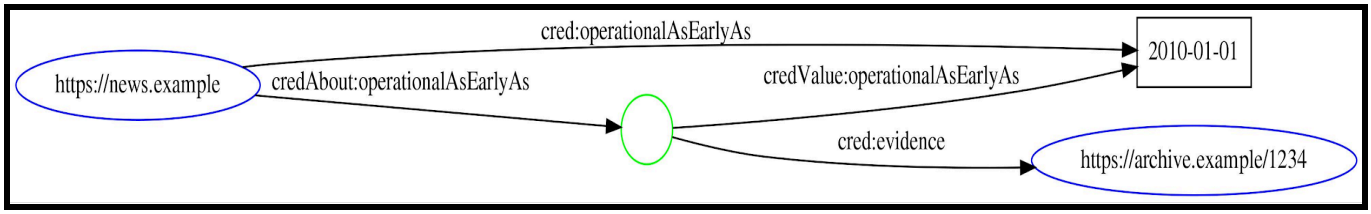
2.2. Shape 2: Interrupted Arc (used by WikiData)

This approach, used by WikiData, considers one arc to be the primary relation, like this:

```
?site cred:operationalAsEarlyAs ?date.
```

We then make another two versions of that predicate (in other namespaces), and route them through a new blank node, to which we can attach arbitrary metadata. The use of three predicates differing only in namespace is a bit unusual, and there might be a simpler alternative.

```
?site  credAbout:operationalAsEarlyAs ?node2.
?node2 credValue:operationalAsEarlyAs ?date;
       cred:evidence ?archive.
```



In JSON-LD:

```
{
  "@context": { ... }
  "@graph": [
      {
        "@id": "https://news.example",
        "operationAsEarlyAs": [ "2010-01-01", "2015-01-01" ],
        "aboutOperationAsEarlyAs": [ "_:n1", "_:n2" ]
      }
      {
        "@id": "_:n1",
        "valueOperationalAsEarlyAs": "2010-01-01"
        "evidence": "https://archive.example/1234",
      },
      {
        "@id": "_:n2",
        "valueOperationalAsEarlyAs": "2015-01-01"
        "evidence": "https://archive.example/abcd",
      },
  ]
}
```

One issue with this approach is the redundancy of including the primary statement along with the interrupted version. What should the status or meaning be if these are not in sync? Where wikidata is deployed on centralized servers with special software, the Credible Web use case involves data being provided from many sources, some of which will likely have flaws, so some handling for this kind of error will need to be specified.

Another issue, which does not arise with this signal but may with others, is how to apply this shape to a truly n-ary relation. Arguably, one element of such a relation can always be selected as primary. See, for example, how WikiData handles movie awards, with a “point in time” and “for work” treaded as qualifiers, as in Academy Award for Best Actor.

2.3. Shape 3: Named Graph

Another option is to use RDF Datasets, a mechanism for labeling subgraphs with a name and then using that name in the data. This seems quite intuitive for this use case, but is not without issues. Consider:

```
?site cred:operationalAsEarlyAs ?date.
GRAPH ?node3 { ?site cred:operationalAsEarlyAs ?date }
?node3 cred:evidence ?archive.
```

In JSON-LD :

```
{
  "@context": { ... },
  "@graph": [
      {
        "@id": "https://news.example",
        "operationAsEarlyAs": "2010-01-01"
      },
      {
        "@graph": [
          {
      "@id": "https://news.example",
      "operationAsEarlyAs": "2010-01-01"
          }
        ],
        "@id": "_:n3-2",
        "evidence": "https://archive.example/1234"
      },
  ]
 }
```

The first issue is the lack of standard semantics for named graphs. When named graphs were standardized in SPARQL and RDF 1.1, no semantics were assigned, because it was recognized that many were already in use. See RDF 1.1: On Semantics of RDF Datasets. For a discussion on how to potentially work around this lack of standardization, see Aligning with semantics of RDF Datasets · Issue #1 · w3c/N3.

To make this lack of standard semantics more concrete, consider that a website, https://alice.example/ publishes the news.mit.edu data (above) using this graph shape. In TriG (skipping namespace declarations), that might look like:

```
<https://news.mit.edu> cred:operationalAsEarlyAs "2015-09-02"^^xsd:date.
GRAPH _:g3 {
   <https://news.mit.edu> cred:operationalAsEarlyAs "2015-09-02"^^xsd:date
}
_:g3 cred:evidence <https://web.archive.org/web/20150902023223/http://news.mit.edu/>.
```

Hopefully this makes clear why we need to repeat the primary statement. In its first occurrence, we have Alice actually asserting the claim. In its second occurrence, we have Alice attaching a name (_:g3) to it, but not necessarily asserting it. Once that name is attached, the cred:evidence statement can be made. (Alternatively, instead of repeating the data, we could introduce a mechanism to assert a named graph. That is probably more fraught than simply repeating the data, both for the reasons discussed below in this section, and as discussed in 2.4. Shape 4: RDF Reification.) This repetition also introduces the risk of a mismatch in the data as in 2.2. Shape 2: Interrupted Arc (used by WikiData), and the behavior in the event of such a mismatch will need to be considered.

While this example dataset is fairly clear as graph data, its semantics are less so. How would one define cred:evidence? What is it a relation between? The standard RDF semantics only bind a name (an RDF term) to a graph, in the context of the dataset, while relations like cred:evidence are defined between entities denoted by RDF terms. The standard semantics notably do not say the name paired with the graph denotes the graph. It is unclear how exactly to bridge this gap, allowing cred:evidence to essentially reach backwards through the naming relation to refer to the graph associated with the name (or a name) of its subject. It is also unclear whether this will cause any problems in practice.

Meanwhile, a practical issue is that systems which gather RDF data from multiple web sources often use named graphs to keep that data separated. Triples fetched from https://alice.example/ might go in a graph named <https://alice.example/>, or perhaps in a new graph with a blank node name for each time a fetch is done, with some other data linking those blank nodes to https://alice.example/. That's fairly straightforward.

But what happens if Alice provides not triples (an RDF Graph) but quads (an RDF Dataset), as required with this shape? How does that get loaded into our quad store? We can't just load her quads in directly or Alice could add data to our store about what other, more trusted sources might have said.

We can imagine a rewrite/encoding mechanism, where quads are rewritten with different graph names, but it adds significant complexity on both input and output. What's more, this complexity is in a part of the system which might be a security boundary, given different trust levels for

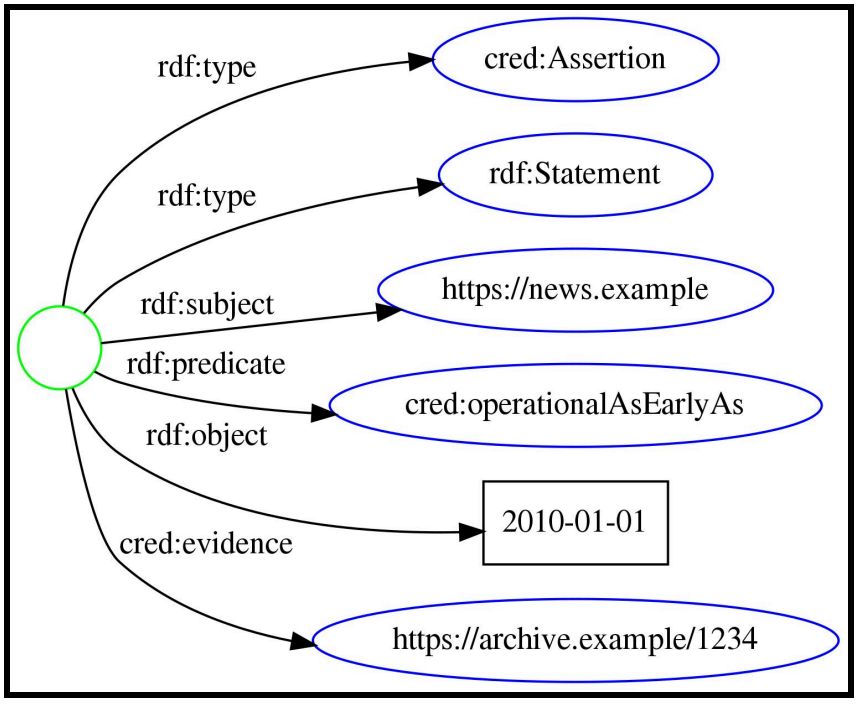
different sources. For security, a dataset from https://alice.example/ must never be able to inject data as if it came from https://bob.example. Complexity in this part of the system is therefore to be approached with extra caution.

2.4. Shape 4: RDF Reification

Finally, mostly for completeness, we might consider an option based on RDF Reification, a feature present in the 1999 standard but viewed by many as flawed. See the archived email under ISSUE-25: Should we deprecate (RDF 2004) reification of statements? - RDF Working Group Tracker.

This might look like:

```
?node4 a cred:Assertion; # a Statement being claimed to be true
       a rdf:Statement;  # optional
       rdf:subject ?site;
       rdf:predicate cred:operationalAsEarlyAs;
       rdf:object ?date;
       cred:evidence ?archive.
```



The JSON-LD for this would be very similar to 2.1 Star.

A significant issue lies in the definition of cred:Assertion. This is essentially a "truth predicate", which can lead to problematic semantics, especially when negated (such as by using OWL). This might not be a problem in practice, however.

2.5. Shape 5: Nanopub

This section (contributed by Davide Ceolin) shows how this might be done using nanopublications.

I have created two nanopublications, one per statement. Each statement has a different provenance, like those mentioned above. Also, the fact that http://alice.example is the creator of the observation is captured in the Pubinfo section of the nanopub.

```
@prefix nanopub: <http://www.nanopub.org/nschema#> .

:NanoPub_1_Head {
  : a nanopub:Nanopublication ;
    nanopub:hasAssertion :NanoPub_1_Assertion ;
    nanopub:hasProvenance :NanoPub_1_Provenance ;
    nanopub:hasPublicationInfo :NanoPub_1_Pubinfo .
}

:NanoPub_1_Assertion {
 <https://news.example> cred:operationAsEarlyAs: "2010-01-01"
}

:NanoPub_1_Provenance {
  :NanoPub_1_Assertion cred:evidence <https://archive.example/1234>;
}

:NanoPub_1_Pubinfo {
  : dcterms:creator <http://alice.example> ;
    dcterms:created "2020-02-08T12:11:30.758274Z"^^xsd:dateTime ;
    dcterms:rights <http://creativecommons.org/licenses/by/3.0/> .
}

:NanoPub_2_Head {
  : a nanopub:Nanopublication ;
    nanopub:hasAssertion :NanoPub_2_Assertion ;
    nanopub:hasProvenance :NanoPub_2_Provenance ;
    nanopub:hasPublicationInfo :NanoPub_2_Pubinfo .
}

:NanoPub_2_Assertion {
 <https://news.example> cred:operationAsEarlyAs: "2015-01-01"
```

```
}

:NanoPub_2_Provenance {
  :NanoPub_2_Assertion cred:evidence <https://archive.example/abcd>;
;
}

:NanoPub_2_Pubinfo {
  : dcterms:creator <http://alice.example> ;
    dcterms:created "2020-02-08T12:11:30.758274Z"^^xsd:dateTime ;
    dcterms:rights <http://creativecommons.org/licenses/by/3.0/> .
}
```