Talking About Occurrences

•••

Building on Reification and Named Graphs –
Tokens of Triples and Sets Thereof

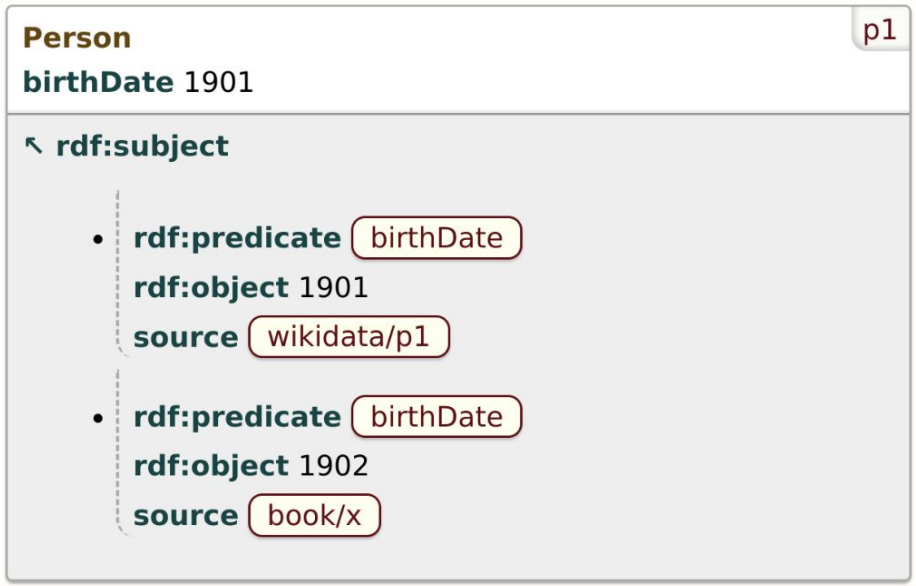
Description Resource
Description Description
Framework

the ability to concisely represent and query statements about statements

Extend RDF with the ability to concisely represent and query **statements about statements**.

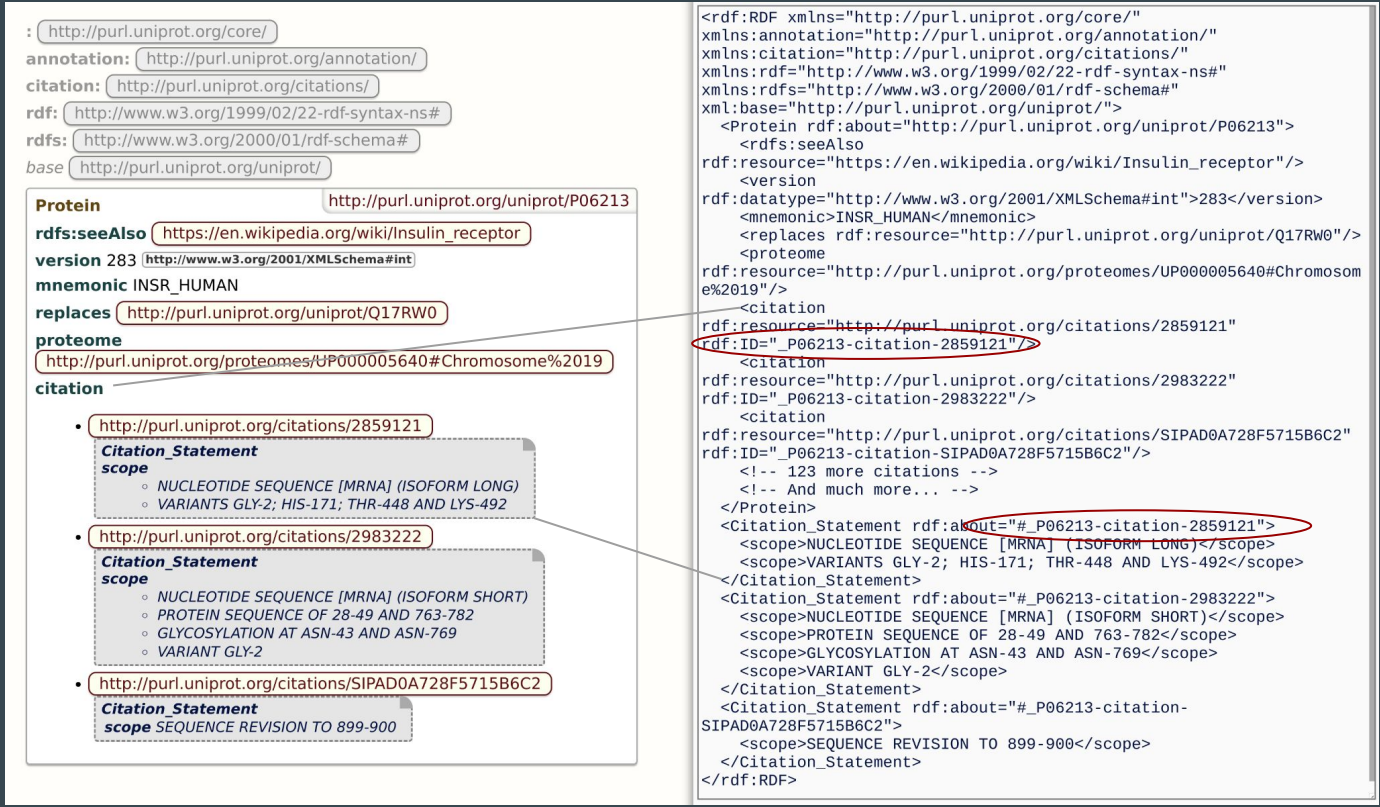
RDF-star WG Charter

Old-school Reification




```
{
  "@id": "p1",
  "@type": "Person",
  "birthDate": "1901",
  "@reverse": {
    "rdf:subject": [
      {
        "rdf:predicate": {"@id": "birthDate"},
        "rdf:object": "1901",
        "source": {"@id": "wikidata/p1"}
      },
      {
        "rdf:predicate": {"@id": "birthDate"},
        "rdf:object": "1902",
        "source": {"@id": "book/x"}
      }
    ]
  }
}
```

@rdf:ID on Arcs Is Heavily Used in UniProt




```
<rdf:RDF xmlns="http://purl.uniprot.org/core/"
xmlns:annotation="http://purl.uniprot.org/annotation/"
xmlns:citation="http://purl.uniprot.org/citations/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://purl.uniprot.org/uniprot/">
  <Protein rdf:about="http://purl.uniprot.org/uniprot/P06213">
    <rdfs:seeAlso
rdf:resource="https://en.wikipedia.org/wiki/Insulin_receptor"/>
    <version
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">283</version>
    <mnemonic>INSR_HUMAN</mnemonic>
    <replaces rdf:resource="http://purl.uniprot.org/uniprot/Q17RW0"/>
    <proteome
rdf:resource="http://purl.uniprot.org/proteomes/UP000005640#Chromosom
e%2019"/>
    <citation
rdf:resource="http://purl.uniprot.org/citations/2859121"
rdf:ID="_P06213-citation-2859121"/>
    <citation
rdf:resource="http://purl.uniprot.org/citations/2983222"
rdf:ID="_P06213-citation-2983222"/>
    <citation
rdf:resource="http://purl.uniprot.org/citations/SIPAD0A728F5715B6C2"
rdf:ID="_P06213-citation-SIPAD0A728F5715B6C2"/>
    <!-- 123 more citations -->
    <!-- And much more... -->
  </Protein>
  <Citation_Statement rdf:about="#_P06213-citation-2859121">
    <scope>NUCLEOTIDE SEQUENCE [MRNA] (ISOFORM LONG)</scope>
    <scope>VARIANTS GLY-2; HIS-171; THR-448 AND LYS-492</scope>
  </Citation_Statement>
  <Citation_Statement rdf:about="#_P06213-citation-2983222">
    <scope>NUCLEOTIDE SEQUENCE [MRNA] (ISOFORM SHORT)</scope>
    <scope>PROTEIN SEQUENCE OF 28-49 AND 763-782</scope>
    <scope>GLYCOSYLATION AT ASN-43 AND ASN-769</scope>
    <scope>VARIANT GLY-2</scope>
  </Citation_Statement>
  <Citation_Statement rdf:about="#_P06213-citation-
SIPAD0A728F5715B6C2">
    <scope>SEQUENCE REVISION TO 899-900</scope>
  </Citation_Statement>
</rdf:RDF>
```

RDF 1.1 Concepts

On Reification

The subject of a reification is intended to refer to a concrete realization of an RDF triple, such as a document in a surface syntax, **rather than a triple considered as an abstract object**.

This **supports use cases** where properties **such as dates of composition or provenance information** are applied to the reified triple, which are **meaningful only when thought of as referring to a particular instance or token of a triple**.

Named Graphs

Are Tokens Too

Pat Hayes, 2011:

It is quite sensible to have two RDF graphs (tokens) with different names which are the same RDF (abstract) graph.

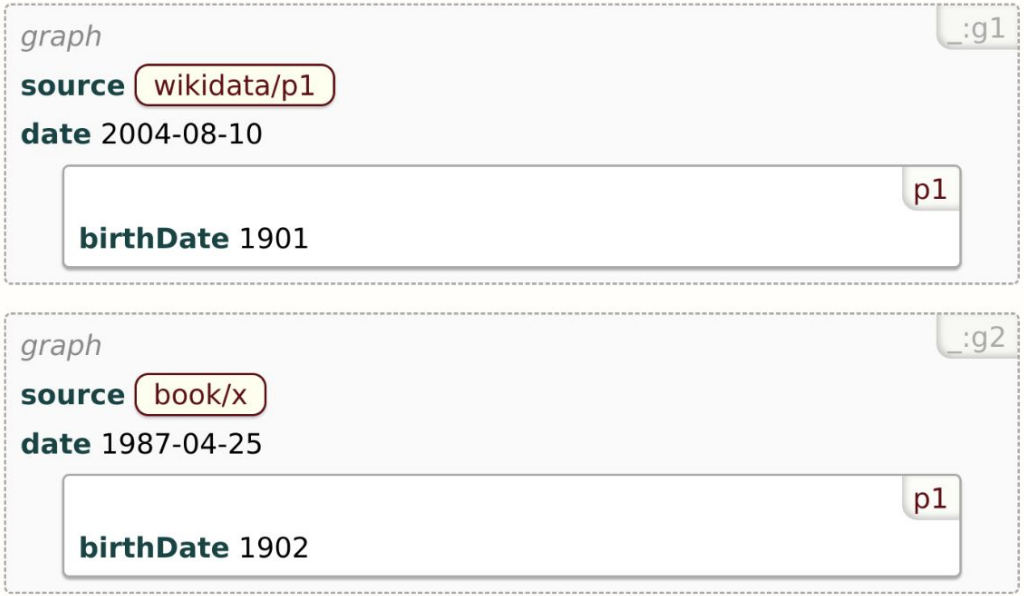
That is, two graph tokens which look like (i.e., when poked emit representations of) the same RDF abstract graph. This has always been an issue for the idea of 'named graphs': ***how can a name be attached*** to a particular RDF **abstract** graph (as opposed to some document or representation of that abstract graph)?

And OK, the answer is: ***it can't***, and this ***does not matter***, because **all we are ever needing to identify are graph tokens**, not abstract graphs. You name a graph by identifying a token of it. But that only gives you power over the token, not over the abstraction itself.

A blank node is a *mark* on a *surface*.

- What is missing in RDF concepts is something to capture the intuition that an RDF graph is like a node-arc *diagram*. (Not a 'mathematical' graph!)
- RDF graphs are drawn on *surfaces*. Blank nodes are *marks on the surface*. Intuitively, think of a surface as a piece of paper, or a screen, or a document.
- Surfaces provide the missing type/token distinction. Putting the same graph onto a new surface is like making a copy. But copying a graph onto a new surface always gets you new blank nodes, because ***a mark can only be on one surface***. Aha!

Named Graphs Are Useful For Provenance



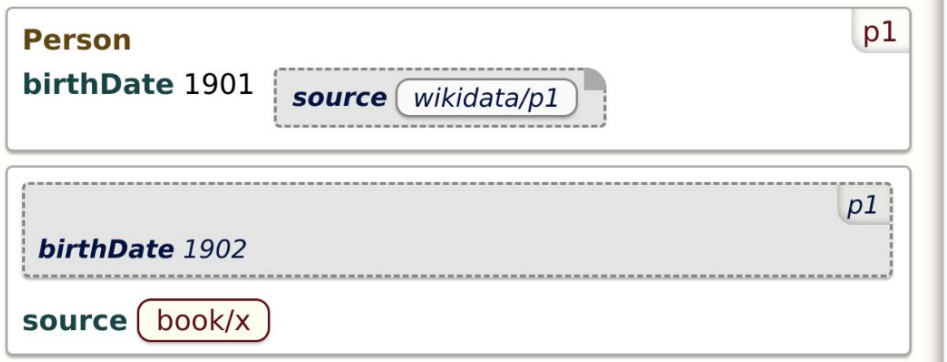

```
[
  {
    "@id": "_:g1",
    "source": {"@id": "wikidata/p1"},
    "date": "2004-08-10",
    "@graph": {
      "@id": "p1",
      "birthDate": "1901"
    }
  },
  {
    "@id": "_:g2",
    "source": {"@id": "book/x"},
    "date": "1987-04-25",
    "@graph": {
      "@id": "p1",
      "birthDate": "1902"
    }
  }
]
```

In The Wild

Verifiable Credentials already uses “blank graphs” for digital signatures.

RDF-star CG Report

Proposes: Quoted Triples as Terms, The Abstract Triples Themselves




```
# RDF-star

<p1> a :Person ;
  :birthDate "1901" {| :source <wikidata/p1> |} .

<<<p1> :birthDate "1902">> :source <book/x> .
```

About... What?

“However much this dragon tries to be spatial, he remains completely flat. Two incisions are made in the paper on which he is printed.

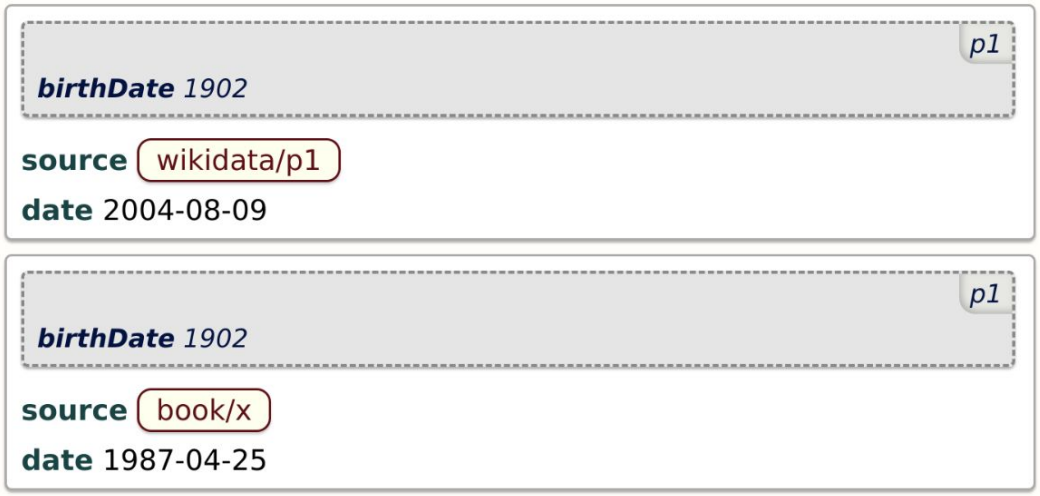
Then it is folded in such a way as to leave two square openings.

But this dragon is an obstinate beast, and in spite of his two dimensions he persists in assuming that he has three; so he sticks his head through one of the holes and his tail through the other.”

– M. C. Escher explains his painting Dragon (1952)

Dragon (1952) (https://en.wikipedia.org/wiki/Dragon_(M._C._Escher))

Problems Appear When You Talk About the Triples Themselves




```
<<<p1> :birthDate "1902">>
  :source <wikidata/p1> ;
  :date "2004-08-09" .

<<<p1> :birthDate "1902">>
  :source <book/x> ;
  :date "1987-04-25" .
```

The Triple Denotes Itself


```
<<<p1> :birthDate "1902">>
  :source <book/x>, <wikidata/p1>  ;
  :date "1987-04-25", "2004-08-09" .
```

Like Literals as Subjects

– How many parts in a triple?

– Three!

– What does that mean?

Old-school Reification Handles This By Design

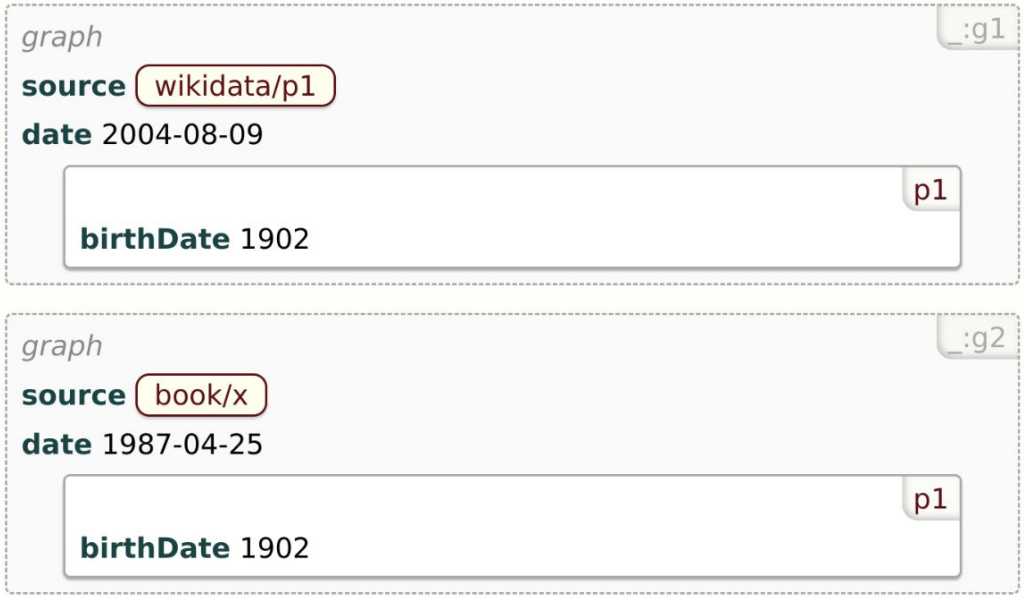
Person p1

↖ **rdf:subject**

- **rdf:predicate** birthDate
  rdf:object 1902
  source book/x
  date 1987-04-25
- **rdf:predicate** birthDate
  rdf:object 1902
  source wikidata/p1
  date 2004-08-09

```
{
  "@id": "p1",
  "@type": "Person",
  "@reverse": {
    "rdf:subject": [
      {
        "rdf:predicate": {"@id": "birthDate"},
        "rdf:object": "1902",
        "source": {"@id": "book/x"},
        "date": "1987-04-25"
      },
      {
        "rdf:predicate": {"@id": "birthDate"},
        "rdf:object": "1902",
        "source": {"@id": "wikidata/p1"},
        "date": "2004-08-09"
      }
    ]
  }
}
```

As Do Named Graphs: and more, they provide *Isolation of Worlds*




```
[
  {
    "@id": "_:g1",
    "source": {"@id": "wikidata/p1"},
    "date": "2004-08-09",
    "@graph": {
      "@id": "p1",
      "birthDate": "1902"
    }
  },
  {
    "@id": "_:g2",
    "source": {"@id": "book/x"},
    "date": "1987-04-25",
    "@graph": {
      "@id": "p1",
      "birthDate": "1902"
    }
  }
]
```

Adding a Separate Term is Not Necessary for The Use Cases

We don't appear to *need* a new term to solve the collected use cases:

- LPGs: tokens ("multisets")
- Wikidata: reification-like tokens
- UniProt attribution: reification = tokens
- CIDOC-CRM facts qualified as events (including interrelated statements) = tokens
- Detailed provenance and miscellaneous marginalia in libraries = tokens

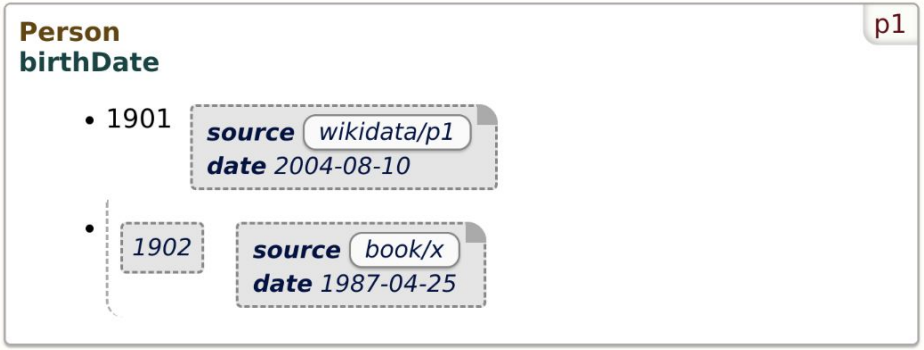
It *is* possible to add it and explicitly indirect from it for most cases. But as shown time and again, it is easy to trip up on this.

Named Graphs *may* provide what "triple opacity" (or partial versions thereof) attempts to solve:

- Isolation of beliefs

You can talk about a graph token without believing in it. Graphs must be *accepted* for their constituent triples to be used as assertions.

Ergonomic Shorthands are Asked For




```
# Using proposed shorthand -- a quotation dash

<p1> a :Person ;
   :birthDate "1901" {| :source <wikidata/p1> ;
                        :date "2004-08-10" |} ,
    -- "1902" {| :source <book/x> ;
                 :date "1987-04-25" |} .
```

Unfolding To Either The Old...

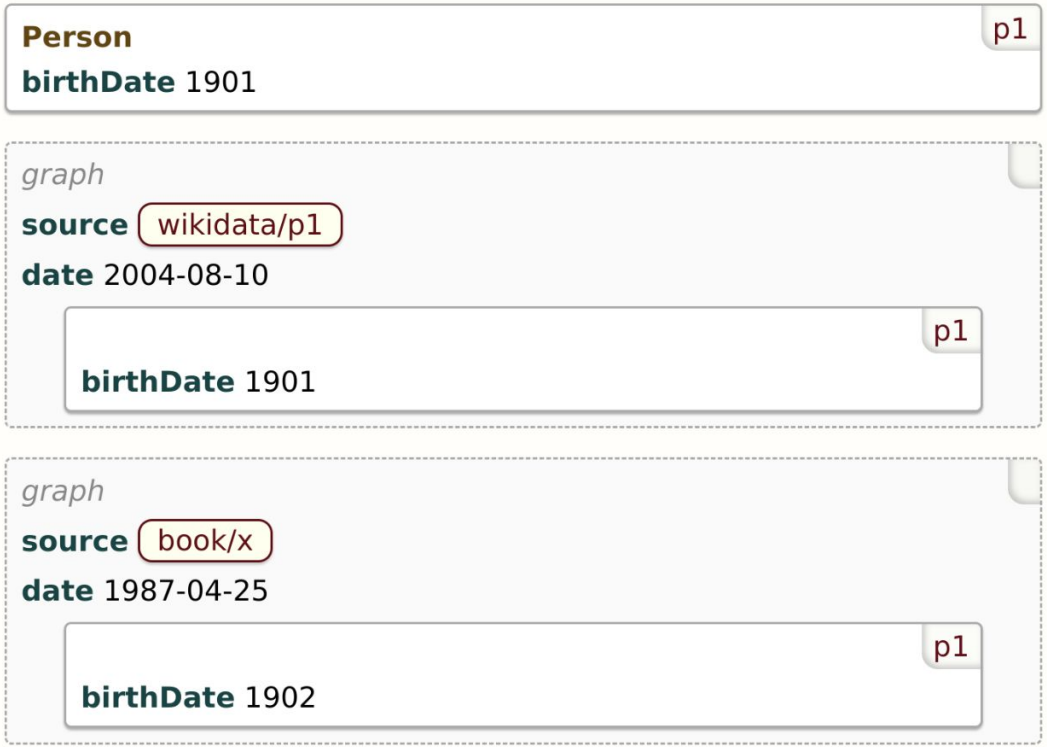
Person p1
birthDate 1901

↖ **rdf:subject**

- **rdf:predicate** birthDate
  rdf:object 1901
  source wikidata/p1
  date 2004-08-10
- **rdf:predicate** birthDate
  rdf:object 1902
  source book/x
  date 1987-04-25


```
{
  "@id": "p1",
  "@type": "Person",
  "birthDate": "1901",
  "@reverse": {
    "rdf:subject": [
      {
        "rdf:predicate": {"@id": "birthDate"},
        "rdf:object": "1901",
        "source": {"@id": "wikidata/p1"},
        "date": "2004-08-10"
      },
      {
        "rdf:predicate": {"@id": "birthDate"},
        "rdf:object": "1902",
        "source": {"@id": "book/x"},
        "date": "1987-04-25"
      }
    ]
  }
}
```

… Or The New Worlds




```
<p1> a :Person ;
  :birthDate "1901" .

[ :source <wikidata/p1> ;
  :date "2004-08-10"
] { <p1> :birthDate "1901" . }

[ :source <book/x> ;
  :date "1987-04-25"
] { <p1> :birthDate "1902" . }
```

Are They Equal?

A triple is identified with the singleton set containing it.

RDF 1.1 Semantics

Named Graphs, 2005

Named Graphs

Jeremy J. Carroll [a] Christian Bizer [b] Pat Hayes [c] Patrick Stickler [d]

[a] *Hewlett-Packard Labs, Bristol, UK*

[b] *Freie Universität Berlin, Germany*

[c] *IHMC, Florida,USA*

[d] *Nokia, Finland*

Thus, RDF reification fails to make this simple distinction.

Named triples solve this, for example as follows. The syntax for this example is TriG explained in section 5.3.

```
:t1 {
 eg:s1 eg:p eg:o .
}
:t2 {
 eg:s2 eg:p eg:o .
}
:g1 {
 :t1 dc:creator "Jeremy Carroll" .
 eg:s1 owl:sameAs eg:s2 .
}
:g2 {
 :t2 dc:creator "Jeremy Carroll" .
 eg:s1 owl:sameAs eg:s2 .
}
```

Named triples may be combined with RDF reification, noting the possibility expressed in RDF Sematics [3]:

> Semantic extensions MAY limit the interpretation of these so that a triple of the form
> `aaa rdf:type rdf:Statement .`
> is true in I just when $I(aaa)$ is a token of an RDF triple in some RDF document, and the three properties, when applied to such a denoted triple, have the same values as the respective components of that triple.

In this case, any interpretation conforming with a set of named graphs including a named triple, will satisfy the reification of that triple. For example, any interpretation conforming with the four named graphs above, with the above extension, would entail:

```
:t1 rdf:type rdf:Statement .
:t1 rdf:subject eg:s1 .
:t1 rdf:predicate eg:p .
:t1 rdf:object eg:o .
:t2 rdf:type rdf:Statement .
:t2 rdf:subject eg:s2 .
:t2 rdf:predicate eg:p .
:t2 rdf:object eg:o .
```

This includes interpretations that do not accept any of the graphs; and as before there are interpretations that accept `:g1` and do not accept `:g2`, so that the named triples preserve the syntactic intent of most use cases for reification.

Why Keep Alignment With rdf:Statement?

It allows for *informal*, messy, qualification.

A detailed token of *extra* information.
In the marginalia.

The simple triple is still the simple truth.

```
<x> :creator <book> {|
  :subject [ :comment
    “May have been his wife.”@en ];
  :predicate :author, :illustrator;
  :object [ :comment
    “First, unedited draft.”@en ]
 |} .
```

Talking About Occurrences

*Talking **about*** occurrences of triples and graphs (making *statements about statements*) requires *reifying* them (conceptually).

We *use* them all the time, that's just RDF.

And reifying graphs is what named graphs have been doing in practice all along.

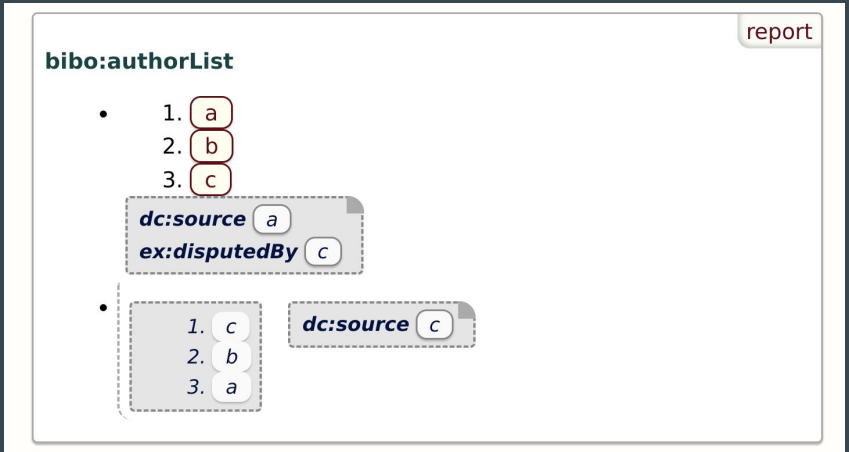
The <name, graph> pair is a *token* of its mathematical graph.

This token, which is denoted by this name, can be *many kinds of resources*:

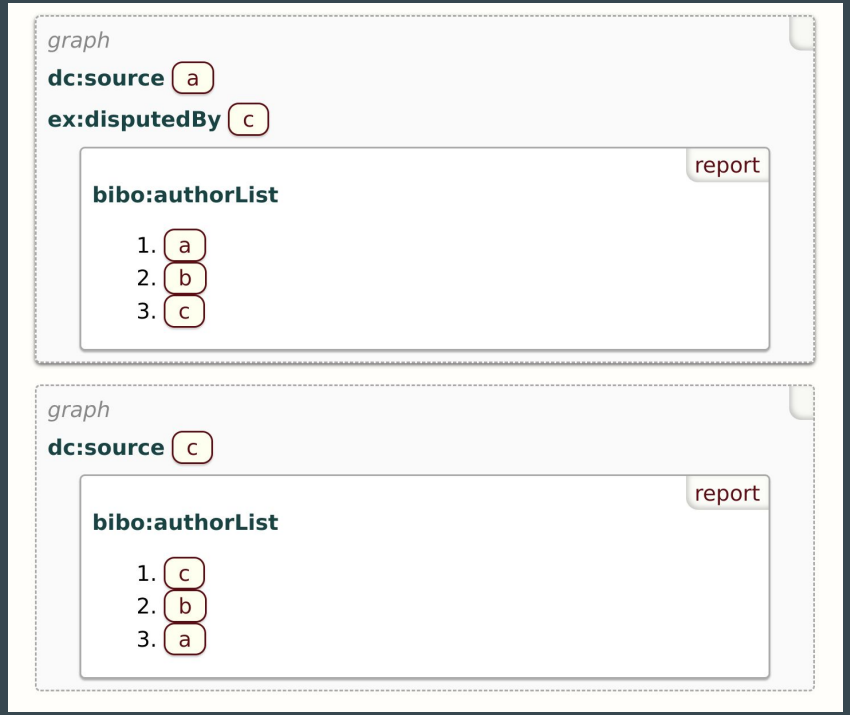
- Just a statement...
- An observed phenomenon.
- The beliefs of Lois Lane.
- Words in a book.
- A chunk of claims gleaned from a web page.

Those are *indirect* tokens of the graph, paired with the graph to make descriptions about it, and query for it.

Even Lists Can Be Contentious...




```
<report> bibo:authorList
  (<a> <b> <c>) {|
      dc:source <a> ;
      ex:disputedBy <c>
    |} ,
  -- (<c> <b> <a>) {| dc:source <c> |} .
```



Back To Work

We *could* add just syntax for Reification first.

No `<<  ...  >>` terms, only annotations `{|  ...  |}`.

Allowed to be repeated for the same triple (for talking about multiple occurrences thereof).

(Also supporting IRI fragment identifiers to be 1:1 with `@rdf:ID` on arcs in RDF/XML? That'd make UniProt work as is, but I'm not sure it's *required*.)

Then define the connection between that and named graphs. The token nature of named graphs provide for a natural equivalence (see Named Graphs, 2005, previous slides).

Or continue with named graphs (tokens) directly. This can allow statements to be entailed as the names of singleton sets, to be backwards-compatible with reification.

We need a way to say that a named graph (occurrence) is **from** or **of** a graph occurrence (or the default graph occurrence). An *appendix* of the graph. That's the missing piece.

That *may* require a new term. Or "protected, graph-local" blank nodes (or even IRIs). *Or just an important (system) relation.*

At least we need rules for Graph Store implementations. These "appendix" graphs must not be asserted. They are neutral.

Possible Approach for RDF 1.1 Systems

```
GRAPH <g1> {
 <x> :creator <o> {|:date "2023"|}.
}
```

```
# As N-Quads (RDF 1.1)

# In the Union Default Graph
<x> :creator <o> <g1> .
_:q1 :date "2023" <g1> .

# Queryable only using GRAPH ?g {…}
<x> :creator <o> _:q1 .

# Under the hood (not queryable)
_:q1 SYS:quoteFrom <g1> SYS:cfg .

# Determine entailment from type?
_:q1 SYS:entailment ent:D SYS:cfg .
```

Why Not...

... **nested graphs**? Appears closely related; but for assertion only. “Fragments” the graph when querying within it? Requires “graph literals” instead of conditional acceptance.

It is simple to have flat quads, asserted in asserted graphs, plus unasserted in “appendices”, whom we talk about. We can keep the relation to “appendix graphs” in the “margins” of a system (with a “protected” name or an explicit relation).

With graph “appendices” we allow for annotations to be excluded. (“Give me just simple asserted Turtle, please; no marginalia.” [This was an originally submitted use case,])

... **graph terms**? Same problem as for triple terms - these are abstract mathematical objects denoting themselves. This is not the realm RDF is *talking about* , it is *the logic substrate itself.*

Also, graphs are *sets*, so,

> *Within the framework of Zermelo–Fraenkel set theory, the axiom of regularity guarantees that no set is an element of itself. This implies that a singleton is necessarily distinct from the element it contains, thus 1 and {1} are not the same thing.*

a singleton set is not the triple it contains.

: https://schema.org/

base https://docs.google.com/presentation/d/e/

2PACX-1vT6luSkUUGrOgpl8vn_MZesCcE5c6TY2bNbLRGk_upB-yzTmM8BrnbYl8BMvqO2Qm2ZBNFcjwB9yuDZ/pub

PresentationDigitalDocument

creator https://neverspace.net/id#self

subject

name *Niklas Lindström*

worksFor *https://www.kb.se*

dateCreated 2023-10-25 xsd:date

```
prefix : <https://schema.org/>
base <https://docs.google.com/presentation/d/e/>

<2PACX-1vT6luSkUUGrOgpl8vn_MZesCcE5c6TY2bNbLRGk_upB-yzTmM8BrnbYl8BMvqO2Qm2ZBNFcjwB9yuDZ/pub>
  a :PresentationDigitalDocument ;
  :creator <https://neverspace.net/id#self> {|
      :subject [ :name "Niklas Lindström"; :worksFor <https://www.kb.se> ]
    |} ;
  :dateCreated "2023-10-25"^^xsd:date .
```