Named Graphs: an Approach to Provenance and Trust

Chris Bizer[1], Jeremy J. Carroll[2,⋆], Pat Hayes[3], and Patrick Stickler[4]

[1] Freie Universität Berlin, Germany
[2] Hewlett-Packard Labs,Bristol, UK
[3] IHMC, Florida, USA
[4] Nokia, Tampere, Finland

Abstract. Placeholder. **Todo:** *Decide on order of authors, the current order is alphabetically. I suggest that the final order should approximate to the amount of work each has put in, to be decided when the paper is nearly finished, perhaps decided by the first person who volunteers to go last*

1 Introduction

Todo:

2 Abstract Syntax

The abstract syntax of named graphs is based on that of RDF[3]. A set of named graphs is a partial function relating nodes (URIrefs and blank) to RDF graphs.

In more detail a set of named graphs $\mathbf{N}$ is a 5-tuple $\langle N, V, U, B, L\rangle$ where: U is a set of URIrefs; L is a set of literals (both plain and typed); B is a set of 'blank' nodes; $V = U \cup B \cup L$ is the set of *nodes* of $\mathbf{N}$; N is a partial function from $U \cup B$ to $V \times U \times V$. U, B and L are pairwise disjoint. $N(n)$ is hence an RDF graph[5] (a set of triples) which is *named* n. When $n \neq n'$ the blank nodes used in triples from $N(n)$ are all distinct from those used in triples from $N(n')$, i.e. blank nodes cannot be shared between different graphs named in N. For technical reasons, we require all nodes $n \in U \cup B \cup L$ to either be a name in the domain of N or to appear in a triple in some graph in the range of N.

Two sets of named graphs $\mathbf{N} = \langle N, V, U, B, L\rangle$ and $\mathbf{N}' = \langle N', V', U, B', L\rangle$ are isomorphic if there is a bijection $\phi : V \to V'$ such that ϕ is the identity on $U \cup L$ and:

$$\langle s, p, o\rangle \in N(n) \text{if and only if } \langle \phi(s), \phi(p), \phi(o)\rangle \in N'(\phi(n)) \tag{1}$$

In this case we see that the graphs named by $\mathbf{N}$ and the graphs named by $\mathbf{N}'$ are pairwise equivalent (in the sense of [3]).

[⋆] Jeremy Carroll is a visiting researcher at ISTI, CNR in Pisa, and thanks his host Oreste Signore.

[5] We have removed the legacy constraint that a literal cannot be the subject of a triple.

3 Concrete Syntax

Todo: *review in lite of Patrick's comments* We offer three concrete syntaxes for named graphs: RDF/XML[1] on the Web; TriX[2]; and a new informal syntax used in this paper.

The URL from which an RDF/XML file is retrieved can act as a name for the graph given by the RDF/XML file using the normal rules. This has some disadvantages:

- It is not clear where the boundary of a set of named graphs lies, the URL provides the name for a single graph, whereas the advantageouos of named graphs is the ability to consider a collection of graphs.
- It is not possible to use a blank node as the name of a graph, or a URIref which is not a URL.
- The known constraints and limitations of RDF/XML apply. For instance, it is not possible to serialize graphs which have predicates that do not end with a sequence matching the NCName production from XML Namespaces. Nor is it possible to use literals as subjects.
- It confuses the URL as a means of identifying the document, and the URL as a means of identifying the graph described by the document.

In balance, there is the major advantage of a deployed base, and current technology.

The TriX serialization of Carroll and Stickler is given by the following DTD

```
ToDo
```

In this paper we use an informal notation, TriG, derived from the informal notation used in the RDF and OWL recommendations. It is roughly N-triple[**?**] with qnames. We extend that notation by using '(' and ')' to group triples into multiple graphs, and to (optionally) precede each by the name of that graph. When the name is omitted it is a b-node that does not occur elsewhere.

4 Semantics

The meaning of a set of named graphs is built on the RDF Semantics [**?**]. We start by considering the meaning of any one graph in the set. This is as given by RDF Semantics, with an extension to cover graph naming. The extension is defined using a partial function *Gext* relating some resources in the domain of discourse *IR* with an RDF graph (as syntactic objects, as expressed by the RDF abstract syntax [3]). This partial function interacts with *extended interpretations* (shown as $I + A$ in section 1.5 of [**?**]). The conditions that must be satisfied by *Gext* and every $I + A$ is that:

$$\forall n \in \text{domain}(N), \mathit{Gext}(I + A(n)) = N(n) \tag{2}$$

This then permits some properties to describe relationships between the graph extensions of the resources, just like `rdf:subPropertyOf` is a relationship between the property extensions of a resource. Two such properties are built-in: `rdfg:equivalentGraph` and `rdfg:subGraphOf`. Their formal semantics are as follows:

$$\mathit{Iext}(I(\texttt{rdfg:equivalentGraph})) = \{(r_1, r_2) | \mathit{Gext}(r_1) \equiv \mathit{Gext}(r_2)\} \tag{3}$$

$$\mathit{Iext}(I(\texttt{rdfg:subGraphOf})) = \{(r_1, r_3) | \exists r_2 \text{with} \mathit{Gext}(r_1) \subset \mathit{Gext}(r_2) \text{and} \mathit{Gext}(r_2) \equiv \mathit{Gext}(r_3)\} \tag{4}$$

where equivalence between graphs is as defined by RDF Concepts.

The meaning of a set of named graphs depends on a separate decision about which of the graphs to accept. We represent this decision as a set A of nodes naming the accepted graphs. The meaning of a set of accepted named graphs $\langle A, \mathbf{N} \rangle$ is given by taking the graph merge $\bigcup_{a \in A} N(a)$, and then interpreting that graph using the semantics of RDF[**?**]. Any extension semantics of RDF can be used; in this paper we uniformly use those of OWL Full[4].

5 A Simple Query Language

Todo.

6 Provenance

Todo:

Todo: *This section should close by introducing the vocabulary to assert or affirm a graph*

7 Semantic Web Publishing

One application area for named graphs is publishing information on the Semantic Web. This scenario implies two basic roles embodied by humans or their agents: Information providers and information consumers. Information providers publish information together with meta-information about it's intended assertional status. Additionally, they might publish background information about themselves, e.g. their role in the application area. Information providers may decide to digitally sign the published information. Information providers have different levels of knowledge, and different intentions and different views of the world. Thus seen from the perspective of a information consumer, published graphs are claims by the information providers rather than facts. The information consumer has to decide which of these claims he wants to trust and use for a specific task.

7.1 The Information Provider

Named graphs allow information providers to annotate a graph with an indication of their intent in publishing that graph. This can be further augmented with a digital signature, when they wish to allow information consumers to have greater confidence in the information published.

We distinguish two different intents: a graph can be *asserted*, meaning that the information provider intends for it to be taken as logically valid according to the RDF Semantics [**?**], or it can be merely *quoted*, in which case the graph is being presented without any comment being made on its logically validity. The latter is particularly useful when republishing graphs as part of a syndication process, the original publisher may assert a news article, but the syndicator, acting as a common carrier, merely provides the graph as they found it, without making any commitment to it validity.

We hence introduce two properties `swp:assertedWith` and `swp:quotedWith` (where `swp:` is a namespace for Semantic Web publishing). Both of these take a graph as subject, and a `swp:Warrant` as object. A resource of class `swp:Warrant` abstracts the assertion or the quoting of a graph. Every warrant must have a single `swp:Authority`, related to it by the `swp:authority` property. The class `swp:Authority` is an abstraction over people, companies and agents that may publish a graph. A simple example is

```
_:g ( ...  RDF Graph
      ...
  _:g swp:assertedBy _:w .
  _:w rdf:type swp:Warrant .
  _:w swp:authority _:a .
  _:a rdf:type swp:Authority .
  _:a foaf:email mailto:chris@bizer.de . )
```

Todo: *foaf:email ??* This indicates that the person with the given e-mail asserts the graph, (or at least, that's what the graph says). The type information can be omitted since it follows from the domain and range of `swp:authority`.

These properties can be used within the graph being discussed, as above, or in a second graph. For instance, when republishing the above information, we might have:

```
_:g ( ...  RDF Graph
      ...
  _:g swp:assertedBy _:w .
  _:w swp:authority _:a .
  _:a foaf:email mailto:chris@bizer.de . )
_:h ( _:h swp:assertedBy _:w1 .
      _:w1 swp:authority _:s .
  _:s foaf:email mailto:patrick.stickler@nokia.com .
  _:g swp:quotedBy _:w2 .
      _:w2 swp:authority _:s .)
```

The second graph shows that the person with e-mail address patrick.stickler@nokia.com is quoting the first graph, and affirms the second graph. We take `swp:assertedBy` to be a subproperty of `swp:quotedBy`.

The reason for having a separate `swp:Warrant` for each graph is that signature information can be provided with the warrant. For instance, if Patrick has an X.509 certificate [**?**] and key pair, he can sign both graphs in this way:

```
_:g ( ...  RDF Graph
      ...
  _:g swp:assertedBy _:w .
  _:w swp:authority _:a .
  _:a foaf:email mailto:chris@bizer.de . )
_:h ( _:g swp:quotedBy _:w2 .
      _:w2 swp:method swp:std-method-A^^xsd:anyURI .
  _:w2 swp:x509Signature "..."^^xsd:base64Binary .
      _:w2 swp:authority _:s .
  _:s swp:x509Certificate "..."^^xsd:base64Binary .
  _:s foaf:email mailto:patrick.stickler@nokia.com .
  _:h swp:assertedBy _:w1 .
      _:w1 swp:method swp:std-method-A^^xsd:anyURI .
      _:w1 swp:authority _:s .
      _:w1 swp:x509Signature "..."^^xsd:base64Binary . )
```

The `swp:x509Signature` gives a binary signature of the graph related to the warrant. Some method of forming the signature has to be agreed. Such a method needs to specify, for example, a variation of the graph canonicalization algorithms provided in [**?**][6], and choosing one of the XML canonicalization methods and one of the signature methods supported by XML Signatures [**?**]. Rather than make a set of decisions about these methods, we permit the warrant to indicate the methods used by including the URL of a document that contains those decisions. The URL used by the publisher needs to be understood by the information consumer, so only a few well-known variations could be used. It may be beneficial to have a richer vocabulary for describing those methods in order to permit a more detailed statement to be included in the warrant. A different method, which does not depend on either RDF canonicalization or XML signatures, is that used by friend-of-a-friend [**?**], in which the original document needs to be included as part of the signature, and signature verification includes parsing the original document and checking that it does contain the correct graph, as well as verifying the signature of the original document as a byte sequence.

The signature can be verified using an X.509 certificate and the graph; the certificate is provided as a property of the `swp:Authority`. An authority could be named with a URIref node, in which case the certificate could be externally available and not included explicitly in the graph containing the `swp:Warrant`.

Todo: *This method stuff is badly explained*

Similarly, he could use a PGP certificate, by using properties `swp:pgpCertificate` and `swp:pgpSignature`.

The publisher may choose to do this to ensure that the maximum number of Semantic Web agents, believe the asserted graphs and act on the publication. Thus, it is the publishers responsibility to use the vocabulary for digital signatures provided above. Using this vocabulary does not modify the theoretical semantics of assertion, which is boolean; but it will modify the operational semantics, in that without signatures only the more trusting Semantic Web agents will act on any assertions. This is particularly important when the publisher's ideal scenario is that the agents engage in economic transactions with the publisher.

7.2 The Information Consumer

Different tasks require different levels of trust. Thus information consumers will use different trust policies in order to decide which graphs should be treated as trustworthy and used within specific applications. These trust policies depend on the application area, the subjective preferences and past experiences of the information consumer and

[6] It is necessary to exclude the last `swp:x509Signature` triple, from the graph before signing it: this step is included in the method.

the trust relevant information available. A naive information consumer might for example decide to trust all graphs which have been explicitly asserted. This trust policy will achieve a high recall rate but is also easily undermineable by information providers publishing false information. A more cautious consumer might require graphs to be signed and the signers to be known through a Web-of-Trust mechanism. This policy is hard to undermine, but also likely to exclude relevant information, which has been published by unknown information providers.

Trust policies can be based on different types of information:

1. First-hand information published by the actual information provider together with a graph, e.g. information about his role in the application domain or about the intended assertional status of the graph.
2. Information published by third parties about the graph (e.g. affirmations or denials) or about the information provider (e.g. ratings about his trustworthiness within a specific application domain).
3. Information created in the information gathering process, like the retrieval date and the retrieval URL of a graph or the information whether a warrant attached to a graph is verifiable or not.

Todo: *rework this para and next to link better with preceeding* We consider the use case in which an agent has read a set of named graphs off the Web. The first problem is to decide which of the graphs to assert. In terms of the semantics of named graphs, this amounts to determining the set A. We have embedded the provenance information for the graphs within the set of named graphs, hence most plausible trust policies require that we are able to provisionally understand the named graphs in order to determine, from their content, whether or not we wish to assert them. This is similar to reading a book, and believing it either because it says things you already believe, or because the author is someone you believe to be an authority: either of these steps require reading at least some of the book.

We will sketch an algorithm that allows the agent to implement a trust policy of trusting any RDF that is explicitly asserted. This is intended to be illustrative, in the sense that different agents should have different trust policies, and these will need different algorithms. We will then discuss variations of this policy, including a more cautious variation which requires digital signatures.

The agent has an RDF knowledge base, K, which may or may not be initially populated. The agent is presented with a set of named graphs $\mathbf{N}$, and augments the knowledge base with some of those graphs (implicitly determining the set A of accepted graphs).

1. Non-deterministically choose $n \in \text{domain}(N) - A$, terminate if no further choices possible.
2. Set $K' := K \cup N(n)$, provisionally assuming $N(n)$.
3.
4. If K' is inconsistent then backtrack to 1. If K' entails:
   `n swp:assertedBy _:w .`
   then set $K := K'$ and $A := A \cup \{n\}$, otherwise backtrack to 1.
5. Repeat from 1.

If initially K is empty, then the first graph added to K will be one that includes its own assertion, by an arbitrary warrant and authority. All such graphs will be added to K, as will any that are asserted as a consequence of the resulting K. The algorithm is equivalent to one that seeks to accept a graph by finding a statement of its assertion either within itself, or within some other accepted graph, or the initial knowledge base.

At step 4, a slightly more sophisticated query could implement a policy that, for example, only trusted a set of named individuals.

This algorithm is logically incomplete. Consider the pair of named graphs:

```
_:a ( _:b swp:assertedBy _:wa .
      _:wa swp:authority _:aa .
  _:aa foaf:email   <mailto:Jeremy.Carroll@hp.com> .
    )
_:b ( _:a swp:assertedBy _:wb .
      _:wb swp:authority _:ab .
  _:ab foaf:email   <mailto:Patrick.Stickler@nokia.com> .
    )
```

Each asserts the other, and so the goal of accepting any RDF that is explicitly asserted is not completely achieved. Publishers of RDF who wish to use this vocabulary to clarify its assertional status, should be aware of such bootstrapping problems and make it easy to process, by ensuring that at least some of their RDF does include its own assertion.

Using a Public Key Infrastructure The trust algorithm above would believe fraudulent claims of assertion. That is, any of the named graphs may suggest that anyone asserted any of the graphs, whether or not that is true, and the above algorithm has no means of detecting that.

We have earlier described how a publisher can sign their graphs and include such signatures in the published graphs. We will continue to explore the X.509 certified case; in general the PGP case is similar, and the approach taken does not assume a particular PKI.

The earlier example can be checked by modifying the query in step **??** to be:

```
SELECT ( ?certificate ?method ?sign )
( _:s swp:x509Certificate ?certificate .
  _:h swp:assertedBy _:w1 .
  _:w1 swp:method ?method .
  _:w1 swp:authority _:s .
  _:w1 swp:x509Signature ?sign . )
```

where this is understood as being over the interpretation of the graph, rather than a syntactic query over the graph. The signatures must be verified following the given method. If this verification fails then the graph is false and is rejected at step 4. If the verification succeeds then the certification chain should be considered by the agent. If the agent does not trust the certificate, then the graph is similarly rejected. A graph may have more than one warrant. The algorithm is nondeterminismic and hence should consider and valid warrant whose certification chain is trusted. However, any warrant that contains an incorrect signature is simply wrong, and indicates data or algorithmic corruption. A graph containing such a warrant is always rejected by the above algorithm. Where the information forming an invalid warrant is split over more than one of the graphs in the set of named graphs, the situation is difficult and a naive algorithm may fail to consider all possible cases, and hence reject more of the graphs than is strictly necessary.

Todo: *This point is dangling* The authority vouching for the naming relationship need not be the same as the one asserting the graph, thus the above can be further weakened to:

```
SELECT ( ?certificate ?method ?sign )
( _:s swp:x509Certificate ?certificate .
  _:h swp:quotedBy _:w1 .
  _:w1 swp:method ?method .
  _:w1 swp:authority _:s .
  _:w1 swp:x509Signature ?sign .
  _:h swp:assertedBy _:w2 . )
```

8 Formal Semantics of Publishing and Signing

This section provides an extension of RDF semantics [**?**] which: allows persons to be members of the domain of discourse; allows interpretations to be constrained by the identifying information in a digital certificate; allows the `swp:assertedBy` triple to have a *performative* semantics, in which the act of providing the triple *is* the act of assertion, making the triple true; and making `swp:x509Signature` triples true or false depending on whether the signature is valid or not. Together these extensions underpin the publishing framework of the previous section.

8.1 Persons in the Domain of Discourse

In RDF semantics quite what resources are, is left indeterminate, they are just things in the domain of the discourse. In contrast, the two frameworks of digital signatures we have considered both tie a certificate to a legal person (i.e. a human or a company), or, in the case of PGP, a software agent. In X.509, a certificate includes a distinguished name [**?**], which is chosen to adequately identify a legal person, and is verified as accurate by the certification authority. In PGP, a certificate contains identifying information, but it's exact form is unspecified, but it can be information "such as his or her name, user ID, photograph, and so on" [**?**]. **Todo:** *http://www.pgpi.org/doc/pgpintro/*.

The class extension of `swp:Authority` is constrained to be the set P of legal persons and software agents acting on behalf of legal persons.[7] This step, in itself, is not very interesting since we have not constrained which person in the real world corresponds to which URIrefs or blank nodes in the graph.

The second step, is to constrain the property extension of `swp:x509Certificate` to $\{(p, c)|p \in P, c$ a finite sequence of binary octets, with c being an X.509 certificate for $p\}$.

[7] A purist may prefer to leave the domain of discourse as an abstract mathematical object, and to have a second interpretation relating this mathematical object to the real world. This may be seen as clearer in that the philosophical difficulties with mixing the real world in with the mathematical world are then localized. Since making this mix is precisely the point of this section, we have not taken this two-level approach.

The binary octets can be represented in a graph using `xsd:base64Binary`, the interpretation of these sequences as X.509 is specified in [**?**], which gives a distinguished name from RFC @@@@, which identifies a person. We can similarly constrain the property extension of `swp:pgpCertificate`, but given the vagueness of the identifying information we should allow all pairs of in which the person matches the identifying information. For example, if the identifying information is only a GIF image, then all people who look like that image are paired with the certificate.[8]

8.2 Cardinality constraints on Warrants

`swp:quotedBy` is an `owl:InverseFunctionalProperty`; and `swp:authority` is an `owl:FunctionalProperty`. Moreover, every resource in the class extension of `swp:Warrant` is in the actual range of `swp:quotedBy` and the actual domain of `swp:authority`. These constraints are all be expressed using OWL restrictions, in the ontology we have constructed [**?**].

8.3 swp:assertedBy as a Performative

A known difficulty with RDF is that the semantics only discusses the meaning of asserted RDF, but no mechanism is provided for performing such an assertion. Having introduced the actual information providers (people and their agents) into the domain of discourse, we can now give `swp:assertedBy` a performative semantics similar to a person saying ”I solemnly swear that ...”. The act of saying a phrase makes it true (the swearing not necessarily what is being sworn as true).

Thus the formal semantics of `swp:assertedBy` is that (r, w) is in the property extension of `swp:assertedBy` if and only if there is (w, p) in the property extension of `swp:authority`, and the person p asserts the graph *Gext*(r). Moreover, if the person p provides this information, then that is an act of assertion. Assertion is in the sense of RDF semantics, with both the OWL extensions, and the extensions in this paper.

8.4 Signing Graphs

The final specialized vocabulary we consider is that for graph signatures. Strictly speaking this is not necessary for Semantic Web publishing, but just as a signed document has greater social force than an unsigned one, a signed `swp:assertedBy` triple is more credible than an unsigned one. Thus, this section is specifically intended to be used to sign graphs that are either the subject of, or include `swp:assertedBy` triples.

A pair (w, s) is in the property extension of `swp:x509signature`, if and only if,

1. s is a finite sequence of octets.
2. There is a pair (w, m) in the property extension of `swp:method`, and m is a URI which can be dereferenced to get a document.
3. There is a pair (w, a) in the property extension of `swp:authority` and a pair (a, c) in the property extension of `swp:x509Certificate`, and c is a finite sequence of octets.

[8] This shows why it is unwise to only provide an image in your PGP certificate.

4. There is a pair (g, w) in the property extension of `swp:quotedBy`, and g is in the domain of *Gext*.
5. And using the method described in the document retrieved from m to calculate a signature for the graph *Gext*(g) using c understood as an X.509 certificate, gives s.

Notice, that this definition does not depend upon verifying the certificate chain for c. We similarly can define the property extension of `swp:pgpSignature`.

8.5 Extensibility

The above approach to the publishing vocabulary relates the RDF semantics, which is at a very abstract level, to other specifications concerning Internet technology, which in turn connect to the real world. However, as is, we have only provided the ability to assert the formal truth of RDF graphs and with the extensions above, these can connect to the real world in as much as those graphs are about publishing of RDF graphs. So a possible untruth that one can assert is that someone else has asserted graph which they have not in fact asserted. However, if Patrick Stickler chose to use this vocabulary to assert a graph including the triple:

```
<http://www.w3.org/TR/rdf-mt> dc:creator "Patrick Stickler" .
```

while the informal meaning (that Patrick wrote RDF Semantics) is false, formally the graph is consistent, (there are possible interpretations, and possible domains of discourse, in which that triple is true).

Thus to permit Semantic Publishing to permit information providers to assert statements about the real-world, we need to provide an extensibility mechanism that allows various extensions to the semantics to be formally included in the graph being asserted.

We have already seen one such example, using `swp:method`. The formal semantics of `swp:x509signature` above, deferred to whatever method was described in the document available from the given URI on the web. This could be extended to arbitrary RDFS properties and classes by providing a further property `swp:isDefinedBy` which introduces semantic extensions, defining the formal semantics for properties and classes in documents. `swp:isDefinedBy` is thus a subproperty of `rdfs:isDefinedBy` with semantic force, rather than being merely an annotation. Some of these definitions could be OWL or RDFS documents; but the more interesting ones, like `swp:x509certificate` would need to defer to other standards in order to ground the formal interpretations in the real world, which is the intended ‘domain of discourse’.

While a full exploration of this lies beyond the scope of this paper, we note that any documents used as the formal definition of properties should be available from trusted organizations, typically standard bodies or other reputable third-parties. Moreover, the links to these formal definitions should be provided within the graph being signed (possibly using a mechanism like `owl:imports`) rather than relying on implicit knowledge about which properties have formal definitions, and which of those formal definitions the information provider is intending.

9 Trust

Todo: *Rework seem*

The previous section has described an approach to trust on the Semantic Web.

We think that the Semantic Web requires an open trust architecture without central trusted third parties. The trustworthiness of information should be subjectively evaluated by each application that processes the information. A trust architecture should not exclude information providers that have not been rated or do not publish trust relevant information in a specific way, e.g. sign their information. On the other hand, the system should be able to use all trust relevant information (signatures, context information, related information and ratings) published or generated during the information gathering process (source URL, crawling date). Users have different subjective preferences for specific trust mechanisms and - even in the same situation - different trust requirements. As a consequence an architecture should allow users to formulate subjective and task-specific trust policies combining different trust mechanisms. The key factor for building trust is the user's understanding of the information and the metrics used in trust evaluations. Thus an architecture should have the ability to justify its trust decisions and support something like Tim Berners-Lee's "Oh yeah?"-button [**?**], meaning that the user can click on every piece of information within an application and get explanations why she should trust the information.

Three general trust mechanism build on this information:

1. Reputation-Based Trust Mechanisms include rating systems like the one used by eBay and Web-Of-Trust mechanisms. All trust architectures proposed for the Semantic Web so far fall into this category [**?**,**?**,**?**]. The general problem with these approaches is that they require explicit and topic-specific trust ratings and that providing such ratings and keeping them up-to-date puts an unrealistically heavy burden on information consumers.
2. Context-Based Trust Mechanisms use metainformation about the circumstances in which information has been claimed, e.g. who said what, when and why. They include role-based trust mechanisms, using the author's role or his membership in a specific group, for trust decisions. Example policies from this category are: "Prefer product descriptions published by the manufacturer over descriptions published by a vendor" or "Distrust everything a vendor says about its competitor." An example policy using the statement context is "Distrust all product ratings that are older than a year."
3. Content-Based Trust Mechanisms: These approaches do not use metadata about information, but rules and axioms together with the information content itself and related information about the same topic published by other authors [4]. Example policies following this approach are "Believe information which has been stated by at least 5 independent sources." or "Distrust product prices that are more than 50

Context- and content-based trust mechanisms do not require explicit ratings, but rely on the availability of a dense mesh of background information. On the Semantic Web such a mesh will be available and therefore can be used for trust decisions.

9.1 Spare Text

Our architecture can be logically divided into four layers: The Information Integration Layer handles the aggregation of information from different sources and adds provenance metadata to the information. If information is digitally signed [**?**] and the signature can be verified, the information is marked as ”FromVerifiedOrigin” The Repository Layer stores the aggregated information. The Query and Trust Evaluation Layer handles the actual trust decisions using query-specific trust policies. The Application and Explanation Layer on which the retrieved information is used within an application context and which provides functionality to browse through explanations why data should be trusted.

10 Apparent Paradoxes

Todo: *refs: http://www.w3.org/2000/10/swap/log http://lists.w3.org/Archives/Public/www-rdf-rules/2002Dec/0003* Carroll and Stickler [2] noted that named graphs with N3’s logical vocabulary permits the creation of paradoxes, such as the liars paradox:

```
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix eg: <http://example.org/> .
{ eg:liar log:implies { eg:noone a owl:Nothing . } .
} owl:sameAs eg:liar .
eg:liar a log:Truth .
```

The ability to create paradox in RDF does not depend on named graphs but on abusing `rdf:comment`, for example:

```
<rdfs:Class rdf:ID="Russell" xml:lang="en">
  <rdfs:comment>A class is in the class-extension of the Russell class
  if and only if it is not in its own class-extension.</rdfs:comment>
</rdfs:Class>
```

When the comment is understood in English, the comment triple is necessarily false. There are no interpretations for which the comment describes any class.

Similarly, the logical vocabulary [**?**] of N3, is defined using `rdfs:comments` understood in English. For instance, the summary of the definition for `log:implies` is “Logical implication.” and for `log:Truth` we have “Something which is true:”. These definitions are simply false. There are no RDF interpretations that can make these be true, as shown by the existence of paradox if we take these definitions at face value. However, each of these definitions then continues with operational discussion of how CWM handles these symbols. This points to how the `log:` namespace could be rescued from incoherence by dropping all the model-theoretic concepts, and reworking it in a purely proof-theoretic manner.

11 Summary of New Vocabulary - slightly remodelled

We have introduced new vocabulary for named graphs using the `rdfg:` namespace. The classes are listed in table 1, the properties in table 2. **Todo:** *Currently intensional semantics - Patrick was using the terms intensionally*

Todo: *If we want to go fully intensional we can merge GraphAffirmation with Graph, and require multiple Graphs (related by `rdfg:equivalentGraph`)for multiple affirmations*

Class Name	Description
`rdfg:Graph`	Each resources of this class is associated with an RDF graph.
`swp:Authority`	An authority for, or an origin of, a graph, typically a person or company.
`swp:X509CertifiedAuthority`	An authority who holds an X.509 certificate, and key pair.
`swp:PGPCertifiedAuthority`	An authority who holds a PGP certificate, and key pair.
`swp:Warrant`	A relationship between an authority and a graph, in which the authority in some way, vouches for the graph. Warrants may include a digital signature of the graph by the authority.
`swp:AssertingWarrant`	A subclass of `rdfg:Warrant` in which the authority asserts the graph, in the sense of RDF Semantics [**?**].**Todo:** *nonassertions, timed assertions*.

Table 1. New Classes

Property	Domain	Range
Description		
`rdfg:equivalentGraph`	`rdfg:Graph`	`rdfg:Graph`
The graphs associated with the subject and object are equivalent.		
`rdfg:subGraphOf`	`rdfg:Graph`	`rdfg:Graph`
The graph associated with the subject is a subgraph of a graph equivalent to that associated with the object.		
`swp:withWarrant`	`rdfg:Graph`	`swp:Warrant`
`swp:signedBy`	`swp:Warrant`	`swp:Authority`
The object of the `swp:signedBy` statement vouches for the subject of the `rdfg:withWarrant` statement.		
`swp:assertedBy`	`swp:AssertingWarrant`	`swp:Authority`
A convenience sub-property of `swp:signedBy`, with different domain.		
`swp:signatureBytes`	`swp:Warrant`	xsd:base64Binary
`swp:pgpKey`	`swp:Warrant`	xsd:base64Binary
`swp:x509Key`	`swp:Warrant`	xsd:base64Binary
`rdfg:signatureMethod`	`swp:Warrant`	
The object identifies a well-known algorithm for signing RDF graphs.		

Table 2. New Properties

12 Conclusions

Todo.

Todo: *Update bib file*

References

1. D. Beckett. RDF/XML Syntax Specification (Revised). `http://www.w3.org/TR/rdf-syntax-grammar/`, 2003.
2. J. Carroll and P.Stickler. RDF Triples in XML. Submitted to WWW2004, 2003.
3. G. Klyne and J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. `http://www.w3.org/TR/rdf-concepts/`, 2003.
4. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. `http://www.w3.org/TR/owl-semantics/`, 2003.