Named Graphs, Provenance and Trust

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

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

Abstract. The Semantic Web consists of many RDF graphs named by URIs. This paper extends the syntax and semantics of RDF to cover such collections of named graphs. This enables RDF statements that describe graphs, which can be used in many Semantic Web applications. We explore in detail the important application of Semantic Web publishing: named graphs allow publishers to communicate assertional intent, and to sign their graphs; information consumers can evaluate specific graphs using task-specific trust policies, and act on the information from those named graphs that they accept. Graphs are trusted depending on: their content; information about the graph; and the task the user is performing. This provides a foundation for consumers to establish a personal 'web of trust' using a formally defined and consistently deployed framework.

1 Introduction

A simplified view of the Semantic Web is a collection of web retrievable RDF documents, each containing an RDF graph. The RDF Recommendation [14, 21, 3, 6], explains the meaning of any one document, and how to merge a set of documents into one, but does not explain how to decide which documents to trust, nor how to keep track of where you found anything when you merge documents together. Keeping track of who said what is crucial to being able to use a trust policy that trusts some people more than others, particularly when how much you trust someone depends on what they are talking about.

Reification has well-known problems in addressing these use cases as we have previously discussed [8].

This paper describes a simple variation on RDF, using sets of *named* RDF graphs. This makes it easier to record who said what, and to implement a variety of trust policies using this information.

2 Named Graphs and their Uses

A set of named graphs is a collection of RDF graphs, which may share URIrefs but not blank nodes, each one of which is associated with a *name*, which is either a URIref or a blank node. The name of a graph may occur either in the graph itself, or another one of the set of named graphs, or not at all. A graph can have multiple names, but a name cannot name two different graphs.

A more familiar representation of the same idea is quads like in [22, 10, 2, 15]; consisting of an RDF triple and a further URIref or blank node or ID, often called the 'context'. Such an ID corresponds to the graph name in our framework. Another similar concept is the notion of 'formula' in N3 [4].

Named graphs is a reformulation of quads in which the fourth element's distinct syntactic and semantic properties are clearly distinguished, and the relationship to RDF triples, abstract syntax and semantics is clearer.

It has been suggested, by [20] and others, that quads are relevant to a number of uses, such as:

data syndication Systems need to keep track of provenance information, and provenance chains.

signing RDF graphs As discussed in [19], it is necessary to keep a distinct idea of which graph has been signed, and the signature, and other metadata concerning the signing, may be kept in a second graph.

access control A triple store may wish to allow fine-grain access control, which appears as metadata concerning parts of the store [15].

expressing privacy preferences **Todo:** *Something here*

evaluating trust Fine-grained trust policies requires the combination of independently published information, while maintaining cogniscence of the independence [**?**].

expressing propositional attitudes such as modalities and beliefs **Todo:** *reference?*.

scoping assertions and logic See particularly [20, 4, **?**]. N3 formulae are used extensively to capture logical relationships between graphs.

We have concentrated on the use case of provenance information, looking particularly at how this helps enables Semantic Web publishing and how this interacts with the choices consumers of Semantic Web information make about which information to trust. Knowledge repository management and **Todo:** *others* are essentially special cases of the provenance use case, so the techniques we will show for provenance can be applied in those cases too.

The correct modelling and semantics of propositional attitudes is difficult **Todo:** *citation here*, and lies beyond the ambition of this paper.

We have explored the use of named graphs for logic, and have found severe problems which we briefly report in section 10.

3 Abstract Syntax and Semantics

RDF syntax is based on a mathematical abstraction: an RDF graph is defined as a set of triples. Naming however is best performed on concrete entities which can be transmitted and identified by their location as Web resources. We therefore distinguish between named graphs and the RDF graph that the named graph encodes or represents. Named graphs are a set of entities each of which has two functions *name* and *rdfgraph* defined on it which determine respectively its name, which is a URI , and the RDF graph that it encodes or represents. These functions assign a unique name and RDF graph to each named graph, but named graphs may have other properties.

@@@ Paragraph changed by JJC @@@ This definition actually begs a question, which is how exactly to determine identity between RDF graphs. Although our discussion in this paper does not depend on this critically, we follow the notion of graph equivalence defined in RDF [21]. We treat two RDF graphs which differ only in the identity of their blank nodes as being the same graph. The RDF model theory document [14] does this implicitly, an approach that we follow. A more explicit approach would take graph equivalence from [21] (i.e. a 1:1 mapping on blank nodes, a *renaming*), and say that a *nameblanked* RDF graph is an equivalence class under this equivalence relation of replacing blank nodes by other blank nodes under some renaming. Then the *rdfgraph* of a named graph is a *nameblanked* RDF graph. We will ignore this complication in what follows except to note where it may be relevant.

The only semantic constraint that we impose on named graphs as such is that the name should denote the graph it names in any satisfying interpretation. Using the notation and terminology of [14] this can be stated:

@@Todo formatting

For any named graph g, if I satisfies g then I(name(g)) = g

Note that the named graph itself, rather than the RDF graph it intuitively "names", is the denotation of the name. We consider the RDF graph to be related to the named graph in a way analogous to that in which a class extension is related to a class in RDFS. This 'intensional' (ref [14]) style of modelling allows for distinctions between several 'copies' of a single RDF graph and avoids pitfalls arising from accidental identification of similar named graphs.

Although the name is required to denote the named graph that it names, other URIrefs may also denote it. Thus for example it is quite consistent to assert

```
owl:sameAs <ex:graphName> <ex:URIref> .
```

In sections 5 and 9 we will extend the semantics to handle some applications of graph naming.

4 Concrete Syntax

A concrete syntax for named graphs has to exhibit the name, the graph and the association between them.

We offer three concrete syntaxes for named graphs: TriX[8]; RDF/XML[3] on the Web; and a new informal syntax used in this paper.

The TriX serialization of Carroll and Stickler is an XML format which corresponds fairly directly with the abstract syntax. It is given by the following DTD:

```
@@ TODO update xmlns, @@
@@ each graph must have a name??? @@
<!ELEMENT TriX         (graph*)>
<!ATTLIST TriX         xmlns    CDATA #FIXED "http://example.org/TriX/">
<!ELEMENT graph        (uri+, triple*)>
<!ELEMENT triple       ((id|uri|plainLiteral|typedLiteral),
                        uri, (id|uri|plainLiteral|typedLiteral))>
<!ELEMENT id           (#PCDATA)>
<!ELEMENT uri          (#PCDATA)>
```

```
<!ELEMENT plainLiteral (#PCDATA)>
<!ATTLIST plainLiteral xml:lang CDATA #IMPLIED>
<!ELEMENT typedLiteral (#PCDATA)>
<!ATTLIST typedLiteral datatype CDATA #REQUIRED>
```

A collection of RDF/XML documents on the Web also map naturally into the abstract syntax, by using the URL from which an RDF/XML file is retrieved as a name for the graph given by the RDF/XML file using the normal rules. This serialization of named graphs has some disadvantages:

- The set of named graphs is in many documents rather than one.
- Any particular information provide can only use certain URIs as names, specifically URLs from those Web servers on which they can publish.
- It is not possible to use a blank node as the name of a graph.
- 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.
- The URI at which an RDF/XML document is published is used for three different purposes: as a retrieval address, with an operation semantics typically specified by the URI; as a means of identifying the document; and as a means of identifying the graph described by the document. There is potential for confusion between these three uses.

None of these disadvantages is present in TriX. In balance, the major advantage of using RDF/XML is the deployed base, and current technology.

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[13] with qnames. We extend that notation by using '(' and ')' to group triples into multiple graphs, and to (**Todo:** *delete?* optionally) precede each by the names of that graph. The following TriG example contains two graphs, where the first graph is named using a URIref and the second using a bNode:

```
G1 (Monica ex:name "Monica Murphy".
    Monica rdf:type ex:Person .
    Monica ex:homepage <http://www.monicamurphy.org>)
_:G2 (G1 ex:author Chris.
    G1 ex:date "2/10/2003".)
```

5 Some Useful Vocabulary and its Semantics

Todo: *formatting* The intuitive meaning of a named graph G is **Todo:** *the standard RDF meaning [14] of* its associated RDF graph *rdfgraph*(G), which we will refer to as the *graph extension*. Any assertions in RDF about the graph structure of named graphs are understood to be referred to these graph extensions, just as the meanings of the RDFS class vocabulary are referred to relationships between the class extensions. In particular we propose two useful properties `rdfg:subGraphOf` and `rdfg:equivalentGraph`, with semantics defined as follows:

¡ f, g¿ in IEXT(I(rdfg:subGraphOf)) iff rdfgraph(f) is a subset of rdfgraph(g)

where the subset holds between nameblanked sets of triples, i.e. ignoring blank node identities, as discussed in section 3. Formally, the condition is that there is a renaming mapping m on the blank nodes of rdfgraph(f) such that the RDF graph m(rdfgraph(f)) is a subset of rdfgraph(g).

¡f, g¿ in IEXT(I(rdfg:equivalentGraph) iff rdfgraph(f) = rdfgraph(g)

where, again, identity is understood as holding between the nameblanked graphs: formally, in strict terms of RDF graphs as sets of triples, if some renaming mapping m is such that rdfgraph(f) = m(rdfgraph(g)).

5.1 RDF Reification

A 'reified statement' [14] is a single RDF statement described and identified by a URIreference. Within the framework of this paper, it is natural to think of this as a named graph containing a single triple, blurring the distinction between a (semantic) statement and a (syntactic) triple. With this convention, the subject of `rdfg:subGraphOf` can be a reified triple, and the property can be used to assert that a named graph contains a particular triple. This provides a useful connection with the traditional use of reification and a potential migration path. **Todo:** *Not sure we need to spell out the problems with reification: provides a more economical way to reify large graphs; a single named graph containing n triples packs into one URI the same content as 5.n triples describing the reified triples of the graph.*

5.2 Accepting Graphs

A set of named graphs is not given a single formal meaning. Instead, the formal meaning depends on an additional set $A \subset \text{domain}(N)$. A identifies some of the graphs in the set as *accepted*. Thus there are $2^{|\text{domain}(N)|}$ different formal meanings associated with a set of named graphs, depending on the choice of A. 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 as above.

The choice of A reflects that the individual graphs in the set may have been provided by different people, and that the information consumers who use the named graphs may make different choices as to which graphs to believe. Thus we do not provide one correct way to determine The 'correct' choice of A, but provide a vocabulary for the different information providers to express their intensions, and suggest techniques with which information consumers might come to their own choice of which graphs to accept.

6 A Simple Query Language

For querying named graphs we use TriQL a graph patterns based query language inspired by RDQL [**?**]. A graph pattern consists of an optional graph name and a set of triple patterns.

An example query is "Select all Persons together with the URL of their homepage, which have the skill programming, using only information which has been published after 1/1/2003".

```
SELECT ?x, ?y
WHERE
?a (?b rdf:type ex:Person .
    ?b rdf:hasName ?x .
    ?b ex:homepage ?y)
 (?a dc:date ?c)
AND ?c > "1/1/2003"
```

The example query uses two graph patterns. The first uses the variable $?a$ to refer to graph names and three triple patterns. In the second graph pattern the graph name is abbreviated because it doesn't matter. Queries are executed across all graphs in a document or in a repository.

Todo: *Patrick OK?* A further query language for named graphs is RDFQ [**?**] which uses an RDF vocabulary to structure queries, rather than a syntactically oriented query language.

7 Provenance

The example at the end of section 4 has already shown how simple provenance information can be encoded in named graphs. We can extend this to further to show a provenance chain:

```
G1 (Monica ex:name "Monica Murphy".
    Monica rdf:type ex:Person)
G2 (G1 ex:saidby Andy.
    G1 ex:SourceURL Doc1.trix.
    G1 dc:date "2/10/2004")
G3 (G2 ex:saidby Chris.
    G2 ex:SourceURL Doc2.trix.
    G2 dc:date "2/10/2004")
G4 (G1 dc:creator Peter.
    G2 dc:creator Peter.
    G3 dc:creator Peter)
G5 (G4 dc:creator Peter.
    G4 dc:date "2/10/2004".
G5 dc;creator Peter)
```

Todo: *Do we need this example? how do we link with the next section? should we move it into the next section?*

8 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 an 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.

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 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 the intended assertional status of the graph or about the role of the information provider in the application domain. Example policies using the information provider's role are: "Prefer product descriptions published by the manufacturer over descriptions published by a vendor" or "Distrust everything a vendor says about its competitor."
2. Information published by third parties about the graph (e.g. further assertions) or about the information provider (e.g. ratings about his trustworthiness within a specific application domain). Most trust architectures proposed for the Semantic Web so far fall into this category [12, 1, 7]. The general problem with these approaches is that they require explicit and domain-specific trust ratings and that providing such ratings and keeping them up-to-date puts an unrealistically heavy burden on information consumers.
3. The content of a graph together with rules, axioms and related content from graphs published by other information providers. 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% below the average price."
4. 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.

8.1 The Information Provider

Todo: *formatting* Named graphs allow one graph to refer to other graphs, or even to the same graph. This ability to self-refer provides a way to anchor assertions of a graph by an agent, or 'authority'.

An assertion, as opposed to merely a description of an assertion, is essentially an act that is performed by an agency of some kind: more particularly, it is a *performative*; that is, an act which is performed by saying that one is doing it. (Other examples of

performatives include promising, naming and, in some cultures, marrying: see **Todo:** *ref JL Austin "how to do things with words"* `http://www.sou.edu/English/Hedges/Sodashop/RCenter/Theory/People/austin.htm` `http://en.wikipedia.org/wiki/J._L._Austin`) The relevance of this for our purposes is a proposal to treat certain 'utterances' on Web pages as having performative force, so that by publishing an RDF graph with a certain form one is understood to be performing a performative act described by that graph. This can be stated by introducing a certain class of such 'acts' into the semantics and giving conditions for the truth of graphs which use this vocabulary in terms of these acts. Since acts are rather transient things to pin down, we will identify the act by a certain concordance between the agent publishing a graph and the content of the graph itself. Strictly, the act is the actual publication event, but we will instead use the graph which results from the act as the bearer of the appropriate meaning.[5]

The general technique applies to any 'web act', but we will illustrate it by the most useful one, which is the assertion by an agent of an RDF graph. Consider the property `swp:assertedBy` (where `swp:` is a namespace for Semantic Web publishing) which takes a named graph as a subject and an `swp:Warrant` as object. A warrant here is an entity which has a related *authority*, understood to be the agent of the act in question; the property relates warrants to agents. Each warrant must have a unique authority, so swp:authority is an OWL functional property. The class swp:Authority is an abstraction over people, companies and other agents that may assert a graph.

Now, consider an named graph G which says of itself that it is asserted by a warrant with an authority A, and suppose that the publisher of the warrant is in fact the authority A named in the graph. Then this combination amounts to a statement by A that A is asserting the content of the graph G; and if we agree that such uses of this vocabulary have performative force, then this can be loosely paraphrased as "I assert G" spoken by A, ie as (the result, or trace, of) an act of asserting. To make this precise, we need to assume that the notion of 'warrant published by agent' is given as a primitive in order to state the truth conditions for the vocabulary. The actual deployment then depends on this notion being given flesh in some concrete way, which we consider in the next section.

Todo: *Down to end of section delete???*

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 [14], 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.

[5] UK currency uses this technique, by having each twenty pound note bear the text :"I promise to pay the bearer on demand the sum of twenty pounds."

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 assert or quote 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:mbox mailto:chris@bizer.de . )
```

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,_:g1 ( ...  RDF Graph
      ...
      _:g swp:assertedBy _:w .
      _:w swp:authority _:a .
      _:a foaf:mbox mailto:chris@bizer.de . )
_:h ( _:h swp:assertedBy _:w1 .
      _:w1 swp:authority _:s .
      _:s foaf:mbox mailto:patrick.stickler@nokia.com .
      _:g1 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 asserts 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. In addition, other metadata such as an expiry date can be provided with the warrant. For instance, if Patrick has an X.509 certificate [18] and key pair, he can sign both graphs in this way:

```
_:g,_:g1 ( ...  RDF Graph
      ...
      _:g swp:assertedBy _:w .
      _:w swp:authority _:a .
      _:a foaf:mbox mailto:chris@bizer.de . )
_:h ( _:g1 swp:quotedBy _:w2 .
      _:w2 swp:signatureMethod swp:std-method-A^^xsd:anyURI .
      _:w2 swp:x509Signature "..."^^xsd:base64Binary .
      _:w2 swp:authority _:s .
      _:s swp:x509Certificate "..."^^xsd:base64Binary .
      _:s foaf:mbox mailto:patrick.stickler@nokia.com .
```

```
_:h swp:assertedBy _:w1 .
_:w1 swp:signatureMethod swp:std-method-A^^xsd:anyURI .
_:w1 swp:authority _:s .
_:w1 swp:x509Signature "..."^^xsd:base64Binary . )
```

Todo: *check XML Sig for RDF vocab for x509...* The `swp:x509Signature` gives a binary signature of the graph related to the warrant. Some method of forming the signature has to be agreed. This is indicated by the value of the `swp:signatureMethod` property on the warrant. In practice, there will be a small set of commonly implemented methods, so there will be only a few possible values for the object of this property. We require it to be a literal URI, which can be dereferenced on the Web to retrieve a document. The document describes the method of forming the signature in detail. Such a method could specify, for example, a variation of the graph canonicalization algorithms provided in [19][6], and choosing one of the XML canonicalization methods and one of the signature methods supported by XML Signatures [11]. 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 the 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`.

Similarly, Patrick 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, to provide verifiable information concerning the origins of any graph, it is the publishers responsibility to use a vocabulary for digital signatures, such as 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, any assertions made, will only be acted on by the more trusting Semantic Web information consumers, who do not need verifiable information concerning who is making them. This is particularly important when the publisher's ideal scenario is that the agents consumers in economic transactions with the publisher.

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

8.2 The Information Consumer

The information consumer needs to decide which graphs to accept. This decision may depend on information concerning who said what, and whether it is possible to verify such information. It may also depend on the content of what has been said.

We consider the use case in which an information consumer has read a set of named graphs off the Web. The first problem is to decide which of the graphs to accept. In terms of the semantics of named graphs, this amounts to determining the set A. Information about the graphs may be embedded 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 accept 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 (determining the set A of accepted graphs).

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

Note that step 4 cannot be executed as shown, and must be lazily evaluated. This is because we are using OWL Full, which has an undecidable theory. The position of step 4 indicates that when/if inconsistency is detected later, then a suggested truth maintenance policy is to recover as if this step failed. For a semantics with a complete and terminating consistency checker [9] (such as for OWL Lite), this step could be executed in a conventional non-lazy fashion.

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 5, 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:mbox   <mailto:Jeremy.Carroll@hp.com> .
    )
_:b ( _:a swp:assertedBy _:wb .
      _:wb swp:authority _:ab .
      _:ab foaf:mbox   <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 5 to be:

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

where this is understood as being over the interpretation of the graph, rather than as 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 information consumer. If the agent trusts anyone in the certificate chain[7], then the graph is accepted, otherwise not (more sophisticated algorithms would consider whether the person asserting the graph, who has now been verified, is rated in the topic which the graph discusses).

A graph may have more than one warrant. If any warrant contains an incorrect signature, the graph is simply wrong, and indicates data or algorithmic corruption. A graph containing such a warrant is rejected at step 4 in the above algorithm. The choice of which warrant to check is nondeterminismic and hence should consider any valid warrant whose certification chain is trusted. 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.

[7] For PGP, the specific method of determining whether the certificate is trusted is different.

Todo: *This point is dangling, and opens up a can of worms, delete?* 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:signatureMethod ?method .
  _:w1 swp:authority _:s .
  _:w1 swp:x509Signature ?sign .
  _:h swp:assertedBy _:w2 . )
```

9 Formal Semantics of Publishing and Signing

This section provides an extension of RDF semantics [14] 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 makes `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.

9.1 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 [24, 16, 17], 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" [23]; common practice is to use an e-mail address.

The class extension of `swp:Authority` is constrained to be a set P of legal persons and software agents acting on behalf of legal persons.[8] 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 $p\}$. The binary octets can be represented in a graph using `xsd:base64Binary`, the interpretation of these sequences as X.509 is specified in [18], which gives a distinguished

[8] 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.

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.[9]

This definition does *not* depend on whether the certificate is trusted or not. If the graph containing the `swp:x509Certificate` triple is accepted, using mechanisms such as those discussed in section 8.2, then the triple's meaning is as above. The certificate chain in the certificate is only checked as part of the process of deciding which graphs to accept.

9.2 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 [**?**].

9.3 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.

The algorithm for choosing which graphs to accept, presented in section 8.2, interacts with this performative semantics, by essentially assuming that a graph has been asserted, and then verifying that in that case the performative is true. As a consequence of this using `rdfs:subPropertyOf` or `owl:equivalentProperty` to introduce aliases of `swp:assertedBy` may be misleading and should be avoided. Information consumers should be suspicious of any graphs that attempt this, except when they are also asserted by the persons using the aliases so introduced.

9.4 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

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

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:signatureMethod`, 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.
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`.

9.5 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 Web 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:signatureMethod`. 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.

10 Using Named Graphs for Logic

Todo: *This section does not fit any more, possible strategy: e-mail to tbl and rdf-logic, then cite the e-mail thread from section 2, and delete this section* A further potential use case for named graphs for which we have completed our research, is using named graphs to represent logical relationships such as with N3's logical vocabulary [5]. **Todo:** *refs: http://www.w3.org/2000/10/swap/log http://lists.w3.org/Archives/Public/www-rdf-rules/2002Dec/0003* Carroll and Stickler [8] noted that this 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 `rdfs: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, as a definition, the comment triple is necessarily false. There are no interpretations for which the comment describes any class.

Similarly, the logical vocabulary [5] 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.

Todo: *Do we want to say this below?* We conclude that named graphs are not an appropriate syntax for logical formulae; and RDF semantics is not an appropriate vehicle for examining the validity of logical formulae.

11 Vocabulary Summary

Todo: *Turn this section into a picture, and move forward into semantic web publishing. Also turn this section into a RDF Schema and move onto web. Drop this section*

We have introduced new vocabulary for named graphs using the `rdfg:` and `swp:` namespaces. The classes are listed in table 1, the properties in table 2.

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: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.

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:quotedBy`	`rdfg:Graph`	`swp:Warrant`
`swp:authority`	`swp:Warrant`	`swp:Authority`
The object of the `swp:authority` statement vouches for the subject of the `rdfg:quotedBy` statement.		
`swp:assertedBy`	`swp:Warrant`	`swp:Authority`
A sub-property of `swp:quotedBy`, with performative semantics.		
`swp:x509signature`	`swp:Warrant`	xsd:base64Binary
`swp:pgpSignature`	`swp:Warrant`	xsd:base64Binary
`swp:x509Certificate`	`swp:Authority`	xsd:base64Binary
`swp:pgpCertificate`	`swp:Authority`	xsd:base64Binary
`swp:signatureMethod`	`swp:Warrant`	
The object identifies a well-known algorithm for signing RDF graphs.		

Table 2. New Properties

12 Conclusions

Todo: *This is a first sketch of the points, not the text*

Named graphs are better than RDF because ...

Named graphs are better than quads because ...

Our trust algorithm is better than ratings because ...

13 Acknowledgements

Thanks to the W3C Semantic Web Interest Group for their interest and comments on this work as it has developed. Chris Bizer is a visiting researcher at HP Labs in Bristol, and thanks his host Andy Seaborne. His visit is supported by the Leonardo Da Vinci grant no. D/2002/EX-22020. Jeremy Carroll is a visiting researcher at ISTI, CNR in Pisa, and thanks his host Oreste Signore.

References

1. R. Agrawal, P. Domingos, and M. Richardson. Trust Management for the Semantic Web. In *In Proceedings of the 2nd International Semantic Web Conference, ISWC2003*, 2003.
2. D. Beckett. Redland Notes - Contexts. `http://www.redland.opensource.ac.uk/notes/contexts.html`, 2003.
3. D. Beckett. RDF/XML Syntax Specification (Revised). `http://www.w3.org/TR/rdf-syntax-grammar/`, 2004.
4. T. Berners-Lee. Notation 3. `http://www.w3.org/DesignIssues/Notation3`, 1998.
5. T. Berners-Lee et al. RDF Schema for log: vocabulary. `http://www.w3.org/2000/10/swap/log`, 2003.
6. D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0. `http://www.w3.org/TR/rdf-schema/`, 2004.
7. C. Bizer. Semantic Web Trust and Security Resource Guide. `http://www.wiwiss.fu-berlin.de/suhl/bizer/SWTSGuide`, 2004.
8. J. Carroll and P.Stickler. RDF Triples in XML. Submitted to WWW2004, 2003.
9. J. J. Carroll and J. D. Roo. Web Ontology Language (OWL) Test Cases. `http://www.w3.org/TR/owl-test/`, 2004.
10. E. Dumbill. Tracking provenance of RDF data. Technical report, ISO/IEC, 2003.
11. D. Eastlake, J. Reagle, and D. Solo. XML-Signature Syntax and Processing, RFC 3275. `http://www.w3.org/TR/xmldsig-core/`, 2002.
12. J. Golbeck, B. Parsia, and J. Hendler. Trust Networks on the Semantic Web. In *In Proceedings of the 7th International Workshop on Cooperative Intelligent Agents, CIA2003*, 2003.
13. J. Grant and D. Beckett. RDF Test Cases. `http://www.w3.org/TR/rdf-testcases/`, 2004.
14. P. Hayes. RDF Semantics. `http://www.w3.org/TR/rdf-mt/`, 2004.
15. Intellidimension. Intellidimension: RDF Gateway - Database Fundamentals. `http://www.intellidimension.com/default.rsp?topic=/pages/rdfgateway/dev%-guide/db/db.rsp`, 2003. This proof is known to be flawed, the conjecture is open.
16. ITU-T. The Directory – Models X.501, 1993.
17. ITU-T. The Directory – overview of concepts, models and services. X.500, 1993.
18. ITU-T. Information Technology - Open Systems Interconnection - The Directory Authentication Framework. X.509, 1997.
19. J.J.Carroll. Signing RDF Graphs. In *2nd International Semantic Web Conference, ISWC*, volume 2870 of *Lecture Notes in Computer Science*, Sanibel Island, Florida, October 2003. Springer.
20. G. Klyne. Circumstance, provenance and partial knowledge. `http://www.ninebynine.org/RDFNotes/UsingContextsWithRDF.html`, 2002.

21. G. Klyne and J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. `http://www.w3.org/TR/rdf-concepts/`, 2004.
22. R. MacGregor and I.-Y. Ko. Representing Contextualized Data using Semantic Web Tools. In @@@*todo*@@, 2003.
23. Phil Zimmerman and Network Associates Inc. An Introduction to Cryptography. `ftp://ftp.pgpi.org/pub/pgp/6.5/docs/english/IntroToCrypto.pdf`, 1999.
24. M. Wahl, S. Kille, and T. Howes. Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names, RFC 2253, 1997.

14 Issues

1. Abstract OK - included Patrick's punchline and Jeremy's first line
2. Warrant , PublishingEvent?
3. Method needs to have literal anyURI as value or can we tie it in with a web document in a different way.
4. Do we want to talk about stable model semantics?
5. Second para abstrac syntax.
6. *Every* graph now needs a URI name?
7. reified triple text: proposed path, modified jjc
8. Not clear whether Pat wanted to delete text on accepting subset or to include it unchanged.
9. **Todo:** *Change TriG () to* {}
10. **Todo:** *Get rid of any bnode graph names*