Named Graphs, Provenance and Trust

Jeremy J. Carroll[1], Christian Bizer[2], Patrick 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 is beneficial in many Semantic Web application areas. As a case study, we explore the application area 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 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. The extension of RDF to Named Graphs provides a formally defined framework which could be used as a foundation for the Semantic Web trust layer.

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 [17, 25, 4, 9], explains the meaning of any one graph, and how to merge a set of graphs into one, but does not provide suitable mechanisms for talking about graphs or relations between graphs. The ability to express metainformation about graphs is required in many application areas where RDF is used for:

Data syndication systems need to keep track of provenance information, and provenance chains.

Restricting information usage Information providers might want to attach information about intellectual property rights or their privacy preferences to graphs in order to restrict the usage of published information [13, 27].

Access control A triple store may wish to allow fine-grain access control, which appears as metadata concerning the graphs in the store [19].

Signing RDF graphs As discussed in [23], 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.

Expressing propositional attitudes such as modalities and beliefs [18].

Scoping assertions and logic where logical relationships between graphs have to be captured [24, 6, 29].

RDF reification has well-known problems in addressing these use cases as previously discussed in [11]. To avoid these problems several authors have proposed the

usage of quads [26, 14, 3, 19]; consisting of an RDF triple and a further URIref or blank node or ID. The proposals vary widely in the semantic of the forth element, using it to refer to information sources, to model IDs or statement IDs or more general to 'contexts'.

We propose a general and simple variation on RDF, using sets of *named* RDF graphs. A set of named graphs is a collection of RDF graphs, each one of which is named with a URIref. The name of a graph may occur either in the graph itself, in other graphs, or not at all. Graphs may share URIrefs but not blank nodes.

Named Graphs can be seen as 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.

In the second part of this paper we describe how Named Graphs can be used for Semantic Web publishing, looking in particular on provenance tracking and how it interacts with the choices consumers of Semantic Web information make about which information to trust. **Todo:** *Extend description of part 2*

2 Abstract Syntax and Semantics

RDF syntax is based on a mathematical abstraction: an RDF graph is defined as a set of triples. These graphs are stored in documents which can be retrieved from URIs on the Web. Often these URIs are also used as a name for the graph, for example with an `owl:imports`. To avoid confusion between these two usages we 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.

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 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 $N(n) \neq N(n')$ then 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[6] graphs named in N.

The only semantic constraint that we impose on named graphs as such is that the name should denote the named graph it names in any satisfying interpretation. Using the notation and terminology of [17] this can be stated:

For any named graph g, if I satisfies g then $I(\mathit{name}(g)) = g$

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' (c.f. [17]) style of modelling allows for distinctions between several

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

[6] Equivalent, but non-identical, graphs are different.

'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

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

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 [25]. 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 [17] does this implicitly, an approach that we follow. A more explicit approach would take graph equivalence from [25] (i.e. a 1:1 mapping on blank nodes, a *renaming***Todo:** *What kind of renaming? renaming function?*), 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 intuitive meaning of a named graph G is the standard RDF meaning [17] 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:

$\langle f, g \rangle$ 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 above. 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).

$\langle f, g \rangle$ 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*$(g))$.

2.1 RDF Reification

A 'reified statement' [17] 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.

2.2 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.

In section 6 we will extend the semantics to handle some applications of graph naming.

3 Concrete Syntaxes

A concrete syntax for Named Graphs has to exhibit the name, the graph and the association between them. We offer three concrete syntaxes: TriX and RDF/XML both based on XML; and TriG as a compact plain text format.

The TriX[11] serialization is an XML format which corresponds fairly directly with the abstract syntax, allowing the effective use of generic XML tools such as XSLT, XQuery, while providing syntax extensibility using XSLT. TriX is given by the following DTD:

```
<!ELEMENT TriX         (graph*)>
<!ATTLIST TriX         xmlns CDATA #FIXED
                       "http://www.w3.org/2004/03/trix/trix-1/">
<!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>
```

In this paper we use TriG as a compact and readable alternative to TriX. TriG is a variation of Turtle [5] which extends that notation by using '{' and '}' to group triples into multiple graphs, and to precede each by the name of that graph. The following TriG document contains two graphs. The first graph contains information about itself. The second graph refers to the first one.

```
@prefix ex: <http://www.example.org/exampleVocabulary/> .
@prefix pr: <http://www.example.org/privacyVocabulary/> .
```

```
@prefix : <http://www.example.org/document/> .

:G1 { _:Monica ex:name "Monica Murphy" .
      _:Monica ex:email <mailto:monica@murphy.org> .
      :G1 pr:disallowedUsage pr:Marketing }

:G2 { :G1 ex:author :Chris .
      :G1 ex:date "2003-09-03"^^xsd:date }
```

Named Graphs are downward compatible with RDF. A collection of RDF/XML[4] documents on the Web map naturally into the abstract syntax, by using the first xml:base declaration in the document or the URL from which an RDF/XML file is retrieved as a name for the graph given by the RDF/XML file. This serialization of named graphs has some disadvantages:

- The set of named graphs is in many documents rather than one.
- Any particular information provider can only use certain URIs as names, specifically URLs from those Web servers on which they can publish.
- 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 and TriG. In balance, the major advantage of using RDF/XML is the deployed base, and current technology.

4 Query Languages

There are currently two query languages for Named Graphs: RDFQ [31] uses an RDF vocabulary to structure queries. Queries can be constrained to Named Graphs matching one or more graph templates.

TriQL [7] is a graph patterns based query language inspired by RDQL [30]. A graph pattern consists of a set of triple patterns and an optional graph name.

The following TriQL query selects persons together with their email addresses, using only information which has been stated by Chris after January 2003.

```
SELECT ?person ?email
WHERE  ?graph ( ?person ex:email ?email )
       ( ?graph ex:author doc:Chris .
         ?graph ex:date ?date )
AND    ?date > "2003-01-31"^^xsd:date
USING  ex FOR <http://www.example.org/exampleVocabulary/>
       doc FOR <http://www.example.org/document/>
```

The example query uses two graph patterns. The variable `?graph` is bound to the names of all graphs that contain information about email addresses. The second pattern restricts `?graph` to graphs fulfilling both triple patterns.

5 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 its intended assertional status. Additionally, they might publish background information about themselves, e.g. their role in the application area. They may also 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.

Different tasks require different levels of trust. Thus information consumers will use different trust policies in order to decide which graphs should be accepted and used within the specific application. 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 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 harder to undermine, but also likely to exclude relevant information, which has been published by unknown information providers. In general, trust policies can be based on the following types of information [8]:

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."

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 fall into this category [16, 1, 10]. The general problem with these approaches is that they assume 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.

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."

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.

5.1 WebActs as Performatives

The ability of self-reference 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 [2]. 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.[7]

5.2 The Information Provider

The general technique applies to any 'web act', but we will illustrate it by the most useful one, which is the assertion of an RDF graph by an information provider. The key notion here is that of a 'warrant' which is a resource that:

1. says of itself that it asserts a named graph;
2. says of itself that it is authorized by an authority;
3. is *in fact authorized by the authority it mentions.* ***Todo:*** Is this necessary? Shouldn't the trust policy describe how to figure this out?

This requires two vocabulary items: a property `swp:assertedBy` (where `swp:` is a namespace for Semantic Web publishing) relating warrants to authorites, and another, `swp:authority`, relating warrants to asserted graphs. The property `swp:assertedBy` takes a named graph as a subject and an `swp:Warrant` as object; `swp:authority` takes a warrant as a subject and an `swp:Authority` as an object. 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. The separation of authority from assertion allows a single warrant to refer to several graphs, and for a warrant to record other properties such as publication or expiry date.

For web acts other than assertions, the warrant simply uses a different property to relate itself to the graph, expressing different intentions of the authority towards the

[7] The Bank of England 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."

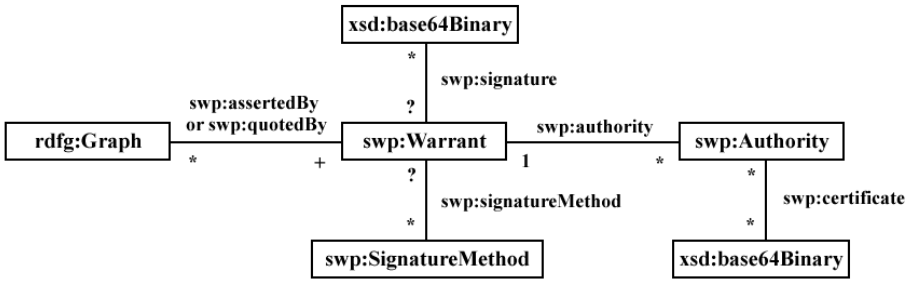

Fig. 1. The Semantic Web Publishing Vocabulary

graph. In particular, `swp:quotedBy` means that the graph is being presented without any comment being made on its truth. This 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 as to its truth. Whatever the intent of a warrant is, it must be explicitly indicated in order for the warrant to have the required performative force. A simple example is:

```
:G1 { :Monica ex:name "Monica Murphy" .
      :G1 swp:assertedBy :G1 .
      :G1 rdf:type swp:Warrant .
      :G1 swp:authority _:a .
      _:a rdf:type swp:Authority .
      _:a foaf:mbox <mailto:chris@bizer.de> }
:G2 { :G2 swp:assertedBy _:w1 .
      _:w1 swp:authority _:s .
      _:s <foaf:mbox mailto:patrick.stickler@nokia.com> .
      :G1 swp:quotedBy _:w2 .
      _:w2 swp:authority _:s }
```

The first graph claims that the person with the given e-mail asserts the graph. The second graph says that the person with e-mail address patrick.stickler@nokia.com is quoting the first graph, and asserts the second graph.

Todo: *Paragraph logically inconsistent, fix interferes with comment above*The asserted graph may be the same graph as the warrant. In this case, the graph is `swp:assertedBy` itself and cites an `swp:Authority` as its `swp:authority` value. Note that the self-assertion triple is required, and that in this case the graph itself must be authorized by the named authority in order to be a correct warrant.

A formal model-theoretic semantics specifies truth conditions. To fully capture the meaning of a performative, however, we need to go beyond truth-conditions, since the very same form of words may be true whoever uses them, but will only count as a performative if used by the authority it mentions. For example “Patrick promises...” uttered by Patrick is a promise - a performative act - but uttered by Alex is merely a description of the act; yet it may well still be true, and for the same reasons. We will

deal with this by considering a warrant to be 'genuine' as a performative just when it describes its authority accurately, and to be true in any interpretation under which a genuine warrant actually exists. For applications in which questions of what constitutes a performative act are important, warrants can be signed and the performance of the act can be checked by examining warrants and their signatures. The actual deployment then depends on this notion being given flesh in some concrete way, which we consider in the next section.

5.3 Publishing with Signatures

Information providers may decide to digitally sign graphs, when they wish to allow information consumers to have greater confidence in the information published. For instance, if Patrick has an X.509 certificate [22] and key pair, he can sign both graphs in this way:

```
:G1 { :Monica ex:name "Monica Murphy" .
      :G1 swp:assertedBy _:w .
      _:w swp:authority _:a .
      _:a foaf:mbox <mailto:chris@bizer.de> }
:G2 { :G1 swp:quotedBy _:w2 .
      _:w2 swp:signatureMethod swp:std-method-A^^xsd:anyURI .
      _:w2 swp:signature "..."^^xsd:base64Binary .
      _:w2 swp:authority _:s .
      _:s swp:certificate "..."^^xsd:base64Binary .
      _:s foaf:mbox <mailto:patrick.stickler@nokia.com> .
      :G2 swp:assertedBy _:w1 .
      _:w1 swp:signatureMethod swp:std-method-A^^xsd:anyURI .
      _:w1 swp:authority _:s .
      _:w1 swp:signature "..."^^xsd:base64Binary }
```

The `swp:signature` 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. **Todo:** *Is dereferening really necessary? Maybe use a vaguer formulation*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 [23][8], and choosing one of the XML canonicalization methods and one of the signature methods supported by XML Signatures [15]. 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. 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

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

checking that it does contain the correct graph, as well as verifying the signature of the original document as a byte sequence.

The publisher may choose to sign graphs to ensure that the maximum number of Semantic Web agents believe them and act on the publication. Using signatures 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.

5.4 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, while maintaining a consistent knowledge base. 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 [12] (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.

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
WHERE ( doc:G1 swp:assertedBy ?w1 .
        ?w1 swp:authority ?s .
        ?w1 swp:signatureMethod ?method .
        ?w1 swp:signature ?sign )
      ( ?s swp:x509Certificate ?certificate )
```

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[9], then the graph is accepted, otherwise not. More sophisticated algorithms would consider whether the person asserting the graph, who has now been verified, is in the group of persons which the information consumer trusts on the topic 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 **Todo:** *Why ANY? Otherwise I can publish lots of incorrect warrents and invalidate third party graphs*. 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.

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

6 Formal Semantics of Publishing and Signing

This section provides an extension of RDF semantics [17] 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:signature` triples true or false depending on whether the signature is valid or not. Together these extensions underpin the publishing framework of the previous section.

6.1 Persons in the Domain of Discourse

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 [32, 20, 21], 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" [28]; common practice is to use an e-mail address.

Since a warrant describes such legal persons as authorities referred to by the object nodes of RDF triples, our formal semantics requires the universe of discourse to contain resources which are actual legal persons or software agents. Such a requirement goes beyond the usual 'logical' bounds of model-theoretic semantics, but these have already been breached in any case by the RDF semantic requirements for such things as data types. We expect that Web languages will further extend their semantics into the real world of agents, acts and things as they become applied in real-world applications. We will write *person(g)* to indicate the legal person who is the authorized authority of a signed graph g.

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. 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 [22], 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.[10]

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

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

such as those discussed in section 5.4, 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.

6.2 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).

Let us say that a named graph g is a *warrant graph* if it has the correct syntactic form for a warrant of an assertion, i.e. if it contains triples of the form

```
g1  swp:assertedBy  g2 .
g2  swp:authority a .
```

where g1, and g2 are graph names; and say that an interpretation I is a *warranting interpretation* of g when it makes g into a warrant, i.e. when the following conditions are satisfied:

I(g2) = g I(a) = *person(g)*

The truth-conditions on `swp:assertedBy` and `swp:authority` can then be stated as follows in terms of axiomatic conditions on the property extensions:

$\langle w, a \rangle$ is in IEXT(I(`swp:authority`)) just when w is a warrant graph, I is a warranting interpretation of w and *person*(w) = I(a).

$\langle g, w \rangle$ is in IEXT(I(`swp:assertedBy`)) just when w is a warrant graph containing a triple

```
g1  swp:assertedBy  g2 .
```

and I(g1) = g and I satisfies g.

Taken together with the basic semantic condition on named graphs:

I(name(g)) = g

these conditions mean that the characteristic triples of a warrant graph are automatically true in any warranted interpretation of the graph; and therefore, more significantly, that such a graph constitutes a warrant whenever the object of the `swp:authority` triple is interpreted to be the actual person authorizing the graph. Notice also that a warrant graph which refers to itself by name can *only* be satisfied by a warranting interpretation, since it fails to be a warrant under any other interpretation, rendering itself false.

These truth conditions may satisfy a graph that does not itself constitute a warrant, provided that the warrant referred to is indeed itself a correct warrant. This allows graphs to describe warranting relationships between other graphs. For example,

```
:G4 { :G1 swp:assertedBy :G3 .

      :G3 swp:authority _:s .
```

```
_:s <foaf:mbox mailto:phayes@ihmc.us> .
```

may be true in an interpretation I by virtue of G3 denoting a warrant for G1 under that interpretation, even though G4 itself, lacking any signature or authorization, cannot be determined to be a warrant.

The typical warranting situation will be a named graph W which uses its own name explicitly as the object of `swp:assertedBy` and the subject of `swp:authority` . Under these conditions, the semantic conditions can be simply stated: a warranting interpretation of a warrant is one where the object of the `swp:authority` triple denotes the authorizing person identified by by the graph's signature

Almost all of the above semantics applies to any other expressed intention towards a graph, with adjusted truth conditions for the other properties. For example, the truth conditions for `swp:quotedBy` are simply:

¡g, w¿ is in IEXT(I(`swp:quotedBy`)) just when w is a warrant graph containing a triple

```
g1  swp:quotedBy  g2 .
```

and I(g1) = g . Notice that I is not required to satisfy g in this case. A (hypothetical) 'swp:deniedBy' property would have truth conditions:

¡g, w¿ is in IEXT(I(`swp:deniedBy`)) just when w is a warrant graph containing a triple

```
g1  swp:deniedBy  g2 .
```

and I(g1) = g and I(g) = false. The conditions which fix the referent of the subject of the triple expressing the intent are identical in each case: the only change is to the required relationship between I and the graph itself.

The algorithm for choosing which graphs to accept, presented in section 5.4, 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.

6.3 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: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`.

7 Conclusions

Having a clearly defined abstract syntax and formal semantics Named Graphs provide greater precision and potential interoperablity than the variety of *ad hoc* RDF extensions currently used. Combined with specific further vocabulary, this will be beneficial in a wide range of application areas and will allow the usage of a common software infrastructure spanning these areas.

The ability of self-reference combined with the Semantic Web Publishing vocabulary addresses the problem of differentiating asserted and non-asserted forms of RDF and allows information providers to express different degrees of commitment towards published information.

Linking information to authorities and optionally assuring these links with digital signatures gives information consumers the basis for using different task-specific trust-policies. We have shown how operational trust can depend on what is being said, rather than simply depending on who said it, and the trust-rating of the author.

Todo: *Should we mention that Named Graphs will be implemented as a Jena extension?* **Todo:** *We need at least one additional nice sounding last sentence!*

8 Acknowledgements

Thanks to the W3C Semantic Web Interest Group for their interest and comments on this work as it has developed. Christian 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. J. Austin. @@Todo: Check source@@How to do things with words. `http://www.sou.edu/English/Hedges/Sodashop/RCenter/Theory/People/austin.htmhttp://en.wikipedia.org/wiki/J._L._Austin`, 2003.
3. D. Beckett. Redland Notes - Contexts. `http://www.redland.opensource.ac.uk/notes/contexts.html`, 2003.

4. D. Beckett. RDF/XML Syntax Specification (Revised). http://www.w3.org/TR/rdf-syntax-grammar/, 2004.
5. D. Beckett. Turtle - terse rdf triple language. http://www.ilrt.bris.ac.uk/discovery/2004/01/turtle/, 2004.
6. T. Berners-Lee. Notation 3. http://www.w3.org/DesignIssues/Notation3, 1998.
7. C. Bizer. TriQL - A Query Language for Named Graphs. http://www.wiwiss.fu-berlin.de/suhl/bizer/TriQL/, 2004.
8. C. Bizer and R. Oldakowski. Using Context- and Content-Based Trust Policies on the Semantic Web. In *@@Todo: How to cite poster@@ Proceedings of the 13th World Wide Web Conference, WWW2004*, 2004.
9. D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0. http://www.w3.org/TR/rdf-schema/, 2004.
10. C. Bizer. Semantic Web Trust and Security Resource Guide. http://www.wiwiss.fu-berlin.de/suhl/bizer/SWTSGuide, 2004.
11. J. Carroll and P.Stickler. RDF Triples in XML. Submitted to WWW2004, 2003.
12. J. J. Carroll and J. D. Roo. Web Ontology Language (OWL) Test Cases. http://www.w3.org/TR/owl-test/, 2004.
13. Creative Commons Website. http://creativecommons.org/, 2003.
14. E. Dumbill. Tracking provenance of RDF data. Technical report, ISO/IEC, 2003.
15. D. Eastlake, J. Reagle, and D. Solo. XML-Signature Syntax and Processing, RFC 3275. http://www.w3.org/TR/xmldsig-core/, 2002.
16. 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.
17. P. Hayes. RDF Semantics. http://www.w3.org/TR/rdf-mt/, 2004.
18. A. Ibrahim. Agent communication languages (acl). http://www.engr.uconn.edu/~ibrahim/publications/acl.htm, 2002.
19. Intellidimension. Intellidimension: RDF Gateway - DatabaseFundamentals. http://www.intellidimension.com/default.rsp?topic=/pages/rdfgateway/dev-guide/db/db.rsp, 2003.
20. ITU-T. The Directory – Models X.501, 1993.
21. ITU-T. The Directory – overview of concepts, models and services. X.500, 1993.
22. ITU-T. Information Technology - Open Systems Interconnection - The Directory Authentication Framework. X.509, 1997.
23. 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.
24. G. Klyne. Circumstance, provenance and partialknowledge. http://www.ninebynine.org/RDFNotes/UsingContextsWithRDF.html, 2002.
25. G. Klyne and J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. http://www.w3.org/TR/rdf-concepts/, 2004.
26. R. MacGregor and I.-Y. Ko. Representing ContextualizedData using Semantic Web Tools. In @@@*todo*@@, 2003.
27. M. Marchiori. http://www.w3.org/TR/P3P/, 2002.
28. Phil Zimmerman and Network Associates Inc. An Introduction to Cryptography. ftp://ftp.pgpi.org/pub/pgp/6.5/docs/english/IntroToCrypto.pdf, 1999.
29. R.V.Guha. *Contexts: A Formalization ad Some Applications*. PhD thesis, Stanford, 1995.
30. A. Seaborne. RDQL - A Query Language for RDF. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109, 2004.
31. P. Strickler. RDFQ. http://sw.nokia.com/rdfq/RDFQ.html, 2004.
32. M. Wahl, S. Kille, and T. Howes. Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names, RFC 2253, 1997.

9 Issues

1. Abstract update with Patrick's comments
2. Update DTD
3. Do we want to talk about stable model semantics?
4. **Todo:** *vocabulary as diagram*
5. sections 5 and 6 to update
6. section 6.3 is weak and needs work; I need to talk to Pat
7. conclusion is weak and needs work (any volunteers?)